Oracle® Database
高可用性概要およびベスト・プラクティス

F29505-11(原本部品番号:F23691-22)

2023年11月

タイトルおよび著作権情報

Oracle Database高可用性概要およびベスト・プラクティス

F29505-11

目次

はじめに

このマニュアルでは、可用性の高いデータベース環境をデプロイするためのOracleベスト・プラクティスを紹介し、Oracle MAAリファレンス・アーキテクチャを構成するためのベスト・プラクティスを示します。

第1部では、高可用性について概説し、高可用性要件の特定を支援します。高可用性をサポートするよう設計されたOracle Databaseの製品と機能、およびビジネスの高可用性実現に役立つ主要なデータベース・アーキテクチャについても説明します。

第2部では、Oracle Databaseで提供される機能を使用して、可用性の高いOracleデータベースを構成するためのベスト・プラクティスについて説明します。これにより、MAA Bronzeリファレンス・アーキテクチャ・サービス・レベルを実現できます

第3部では、レプリケーションおよびデータ保護のためにOracle Data Guardを使用して可用性の高いOracleデータベースを構成するためのベスト・プラクティスについて説明します。これにより、MAA Goldリファレンス・アーキテクチャ・サービス・レベルを実現できます。

「はじめに」では、次の項目について説明します。

対象読者

このマニュアルは、次の仕事を遂行する最高技術責任者、情報テクノロジ・アーキテクト、データベース管理者、システム管理者、ネットワーク管理者およびアプリケーション管理者を対象としています。

- データ・センターの計画
- データ・センター・ポリシーの実装
- 高可用性システムのメンテナンス
- 高可用性ソリューションの計画および構築

ドキュメントのアクセシビリティ

オラクル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

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。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

このマニュアルで説明されている構成と実装の詳細を理解するには、Oracle Database、Oracle RACおよびData Guardの概念および用語に関する知識が必要です。詳細は、Oracle Databaseのドキュメント・セットを参照してください。特に関連

性の高いドキュメントを次に示します。

- Oracle Database管理者ガイド
- Oracle Clusterware管理およびデプロイメント・ガイド
- Oracle Real Application Clusters管理およびデプロイメント・ガイド
- Oracle Automatic Storage Management管理者ガイド
- Oracle Data Guard概要および管理
- 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

ドキュメント・セットの多くのマニュアルでは、Oracle Databaseとともにデフォルトでインストールされる、シード・データベースのサンプル・スキーマを使用しています。これらのスキーマの詳細は、『Oracle Databaseサンプル・スキーマ』を参照してください。

また、Oracle MAAベスト・プラクティス・ホワイト・ペーパーはhttp://www.oracle.com/goto/maaでダウンロードできます。

表記規則

このドキュメントでは、次のテキスト表記規則が使用されます:

規則	意味
boldface	太字体は、アクションに関連付けられたグラフィカル・ユーザー・インタフェース要素や、本文または用語集で定義されている用語を示します。
italic	イタリック体は、ブック・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ー変数を示します。
`monospace`	等幅体は、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、またはユーザーが入力するテキストを示します。

第I部 Oracle Database高可用性概要

1 高可用性の概要

高可用性およびそれが重要な理由を学習するには、次のトピックを参照してください。その後、ロードマップに従って最大可用性アーキテクチャを実装します。

高可用性とは

高可用性とは、アプリケーションおよびデータベース・サービスがどの程度使用可能であるかということです。

可用性は、アプリケーションのユーザーの感覚を基準に評価されます。データが使用できない場合やコンピュータ・システムが予想どおりに機能しない場合、ユーザーは不満を感じます。ユーザーがソリューション全体の複雑なコンポーネントを理解したり、区別しようと考えたりすることはありません。予想を超える使用状況が原因で発生したパフォーマンス障害は、アーキテクチャの重要なコンポーネントに障害が発生した場合と同じ混乱をもたらします。ユーザーがアプリケーションまたはデータベース・サービスにアクセスできない場合は、使用不可と呼ばれる状態です。一般的に、システムの使用不可期間は停止時間と呼ばれます。

システムをいつでも使用できる状態にしておくには、高可用性が必要です。可用性の高いシステムは、主要期間、年間を通じて毎日毎時間、割込みのない計算処理を提供するように設計されています。これは24x365と表されます。このようなシステムにも、システムのハードウェアまたはソフトウェアのアップグレードなど、計画メンテナンス操作のための高可用性ソリューションが必要です。

信頼性、リカバリ可能性、タイムリなエラー検出および継続的稼働は、高可用性ソリューションの主な特徴です。

- 信頼性: 信頼できるハードウェアは高可用性ソリューションの1つの要素です。信頼できるソフトウェア(データベース、Webサーバー、アプリケーションなど)も同様に、高可用性ソリューションを実現する上で重要です。関連する特性はリジリエンスです。たとえば、低コストの汎用ハードウェアをOracle Real Application Clusters(Oracle RAC)などのソフトウェアを組み合せることで、非常に信頼性の高いシステムの実装に使用できます。Oracle RACデータベースのリジリエンスが高ければ、個々のサーバーに障害が発生しても、処理は続行できます。たとえば、Oracle RACデータベースでは、個々のサーバーに障害が発生しても、処理は続行できます。
- リカバリ可能性: 障害からのリカバリを行う場合は多くの方法がありますが、高可用性の環境でどのようなタイプの障害が発生する可能性があるかを判断し、ビジネスの要件を満たすために迅速にそれらの障害からリカバリする方法を決めることが重要です。たとえば、重要な表が誤ってデータベースから削除された場合、その表をリカバリするにはどのような処置を取ればよいでしょうか。アーキテクチャには、品質保証契約(SLA)で規定された時間内にリカバリする能力があるでしょうか。
- タイムリなエラー検出: アーキテクチャのコンポーネントに障害が発生した場合、予期しない障害からリカバリするには素早い検出が重要です。停止から素早くリカバリできても、問題の検出にさらに90分かかったとしたら、SLAを遵守できない可能性があります。環境の状態を監視するには、状態を素早く表示するための信頼できるソフトウェア、およびデータベース管理者に問題を通知する機能が必要です。
- 継続的稼働: データへの継続的なアクセスを提供することが、メンテナンス・アクティビティを実行するための停止時間をほとんど、あるいはまったく許容できないという場合に不可欠です。表をデータベース内の別の場所に移動したり、ハードウェアにCPUを追加したりといったアクティビティは、高可用性アーキテクチャではユーザーに対して透過的に行う必要があります。

より具体的に言えば、高可用性アーキテクチャは次の特徴を備えている必要があります。

- 中断を最小限またはゼロにして、処理が継続するように障害を許容すること

- システム、データまたはアプリケーションの変更に対して透過的である(または耐性がある)こと
- 予防措置が組み込まれていること
- 障害のアクティブな監視と素早い検出が可能であること
- 素早いリカバリが可能であること
- 検出操作およびリカバリ操作が自動化されていること
- データの損失および破損を最小限に抑えるか防止するためにデータを保護すること
- 運用面でのベスト・プラクティスを実施して環境を管理していること
- できるかぎり安価な総所有コストでSLA(リカバリ時間目標(RTO)やリカバリ・ポイント目標(RPO)など)で設定した目標を達成すること

可用性の重要性

高可用性がどの程度重要であるかはアプリケーションによって異なります。データベースとインターネットは、組織やコミュニティの隅々にまでデータベース・アプリケーションを行き渡らせ、世界的なコラボレーションと情報共有を可能にしました。

こうしたデータベース・アプリケーションの浸透により、データ管理ソリューションにおける高可用性の重要性が浮き彫りになっています。中小企業もグローバルな企業も、1日24時間データへのアクセスを必要とするユーザーを世界中に抱えています。このデータ・アクセスが行えなければ、業務は停止し、収益が失われます。これらのソリューションに依存する傾向が強まったことを反映して、ユーザーは情報技術(IT)部門やソリューション提供者の品質保証契約を求めるようになりました。可用性は、時間と利便性のみでなく、ドル、ユーロおよび円で評価されることがますます増えています。

企業は、そのITインフラストラクチャを利用して、競争力を付け、生産性を高め、より多くの情報に基づいてより迅速に決定を下す力をユーザーに与えてきました。しかし、こうした利点により、インフラストラクチャへの依存度はますます高まりました。重要なアプリケーションが使用できなくなった場合、ビジネスが危機に瀕します。ビジネスの収益が減少し、違約金を被り、顧客や会社の株価に対する影響が続く悪評を呼ぶ可能性があります。

データの保護方法を決定する要因について検討し、ユーザーへの可用性を最大限に高めることが非常に重要です。

停止時間のコスト

企業がソリューションを再構築して競争力を強めるにつれて、より高いレベルの可用性を提供する必要性は高まる一方です。ほとんどの場合、こうした新しいソリューションは重要なビジネス・データへの即時アクセスに依存しています。

データが使用できない場合、業務が停止する可能性もあります。停止時間は、生産性の低下、収益の損失、カスタマ・リレーションシップの悪化、悪評および訴訟につながりかねません。

停止時間の直接のコストを試算するのは必ずしも容易ではありません。顧客の怒り、従業員の空き時間、悪評などはすべてコストに結び付きますが、通貨に直接換算することはできません。その一方で、SLAの目標が達成されないことによる収益の損失や法的処罰は容易に定量化可能です。ソリューションに依存してサービスを提供している業界では、停止時間のコストが急速に増大する可能性があります。

停止時間のコストという点で考慮が必要なその他の要因は、次のとおりです。

- 1回の計画外停止の最大許容時間

  イベントの継続時間が30秒未満の場合、イベントが与える影響は非常に小さく、ユーザーがイベントを感じ取ることはほとんどありません。停止時間が長くなるにつれて、その影響は著しく大きくなり、ビジネスにも悪影響を及ぼすことになり

ます。

- 許容できるインシデントの最大頻度

  停止の頻度が高くなると、停止が短時間であったとしても、同様に業務が中断する可能性があります。

ソリューションを設計する場合、停止時間の実際のコストを把握して、可用性が向上することでどのように利益が得られるのかを認識することが重要です。

オラクル社では、規模に関係なくあらゆる組織に適する幅広い高可用性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とインターネットがあれば、いつでもどこからでも、アプリケーションおよびデータへの信頼性の高いアクセスが可能になります。

停止時間の原因

高可用性ソリューションを設計する上での課題の1つは、停止時間のあらゆる原因を検討し、対処することです。

フォルト・トレラントおよびリジリエンスがあるITインフラストラクチャを設計する際、計画外停止時間と計画停止時間の両方の原因を考慮することは重要です。複数のタイムゾーンのユーザーをサポートするグローバル企業では特に、計画停止時間も計画外停止時間と同じく業務に悪影響を及ぼします。

次の表は、計画外停止のタイプと説明、および各タイプの例を示しています。

表1-1 計画外停止時間の原因

タイプ	説明	例
サイト障害	サイト障害は、データ・センターのすべての処理、またはデータ・センターによってサポートされているアプリケーション・サブセットに影響を及ぼす場合があります。 サイトの定義は、オンプレミスとクラウドのコンテキストによって異なります。 ● サイト障害 - リージョン全体の障害 ● データセンター - データセンターの場所全体 ● 可用性ドメイン - リージョン内の独立したデータ・センター(他の多くの可用性ドメインがある可能性があります) ● フォルト・ドメイン - 可用性ドメインまたはデータ・センター内の分離された一連のシステム・リソース 通常、各サイト、データ・センター、可用性ドメインおよびフォルト・ドメインには、分離されたハードウェア、DB コンピュート、ネットワーク、ストレージおよび電源の固有のセットがあります。	● サイト全体に及ぶ停電 ● サイト全体のネットワーク障害 ● データ・センターを操業不可能にする自然災害 ● 業務またはサイトへのテロまたは悪意のある攻撃
クラスタ全体の障害	Oracle RAC データベースのホストであるクラスタ全体が使用不可能になるか、故障します。次のものが含まれます。	● Oracle RAC クラスタ上で最後まで動作しているノードに障害が発生し、ノードまたはデー

タイプ	説明	例
	● クラスタ内のノードの障害 ● クラスタが使用不可能になる原因となった他のコンポーネントの障害および使用不可能なサイト上の Oracle データベースやインスタンスの障害	タベースが再起動不能 ● 冗長クラスタのインターコネクトが両方故障またはクラスタウェアの障害 ● データベースの破損が、現行データベース・サーバーでの継続が不可能になるほど深刻 ● 可用性や安定性を損なう、Clusterware およびハードウェア・ソフトウェアの不具合
コンピュータ障害	コンピュータ障害による停止は、データベースを実行しているシステムが停止または利用不可能になり、使用できない状態になったときに発生します。データベースが Oracle RAC を使用している場合、コンピュータ障害はシステムのサブセットを表します(データへの完全なアクセスを維持)。	● データベース・システム・ハードウェア障害 ● オペレーティング・システム障害 ● Oracle インスタンス障害
ネットワーク障害	ネットワーク障害停止は、ネットワーク・デバイスが停止するか、アプリケーション・サービスの処理に不可欠な、アプリケーションからデータベース、データベースからストレージ、またはシステムからシステムのネットワーク・トラフィックや通信が減少したときに発生します。	● ネットワーク切替え障害 ● ネットワーク・インタフェース障害 ● ネットワーク・ケーブル障害
ストレージ障害	ストレージ障害による停止は、データベース・コンテンツの一部またはすべてが格納されているストレージが停止または利用不可能になり、使用できない状態になったときに発生します。	● ディスクまたはフラッシュ・ドライブの障害 ● ディスク・コントローラ障害 ● ストレージ・アレイ障害
データ破損	破損ブロックは変更されたブロックです。そのため、Oracle Database が検出するものとは異なります。ブロック破損は、物理的破損と論理的破損に分類されます。 ● 物理ブロック破損はメディア破損とも呼ばれ、ブロックがデータベースにまったく認識されません。チェックサムが無効になるか、ブロック内容がすべてゼロになります。より複雑なブロック破損の例として、ブロックのヘッダーとフッター	● オペレーティング・システムまたはストレージ・デバイス・ドライバ障害 ● 障害のあるホスト・バス・アダプタ ● ディスク・コントローラ障害

タイプ	説明	例
	が一致しない場合があげられます。 ● 論理ブロック破損では、ブロックの内容は物理的には正常で、物理ブロック・チェックを通過します。ただし、ブロックが論理的に一貫性を欠いている場合があります。論理ブロック破損の例には、不正なブロック・タイプ、不正なデータまたは REDO ブロック順序番号、行ピースや索引エントリの破損、あるいはデータ・ディクショナリの破損があります。 ブロック破損は、さらにインターブロック破損とイントラブロック破損に分類できます。 ● イントラブロック破損の場合、破損はブロック内部で発生し、物理ブロック破損または論理ブロック破損のいずれかになります。 ● インターブロック破損の場合、破損はブロック間で発生し、論理ブロック破損になります。 データ破損による停止は、ハードウェア、ソフトウェアまたはネットワークのコンポーネントで、読取りまたは書込み対象のデータが破損したときに発生します。データ破損による停止から受けるサービス・レベルの影響は、アプリケーションまたはデータベースの一部(単一データベース・ブロック・レベル)から大部分(基本的に使用不可にする)まで、大きく異なります。	● ディスクの不正な読取りまたは書込みの原因となるボリューム・マネージャ・エラー ● ソフトウェアまたはハードウェアの不具合
人的エラー	人的エラーによる停止は、データベース内のデータを破損または使用不可能の状態にする意図的でない操作やその他の操作が行われた場合に発生します。人的エラーによる停止から受けるサービス・レベルの影響は、影響を受けたデータの量および重大性によって大きく異なります。	● (ファイル・システム・レベルの)ファイルの削除 ● 削除されたデータベース・オブジェクト ● 故意でないデータ変更 ● 悪意のあるデータ変更
書込み欠落または宛先違いの書込み	データ破損の別の形である書込み欠落や宛先違いの書込みは、素早く検出し修復することがさらに困難です。データ・ブロックの宛先違いの書込みまたは書込み欠落は次の場合に発生します。 ● 書込み欠落は、書込み I/O が永続記憶域で発生し	● オペレーティング・システムまたはストレージ・デバイス・ドライバ障害 ● 障害のあるホスト・バス・アダプ

タイプ	説明	例
	なくても、I/O サブシステムがブロック書込みの完了を確認する場合。プライマリ・データベースでの後続ブロック読取りのときに、I/O サブシステムが、データベースの他のブロックの更新に使用される古いバージョンのデータ・ブロックを返すため、ブロックが破損します。 ● 宛先違いの書込みは、書込み I/O は完了したが、他の場所に書き込まれ、後続の読取り操作で古い値が返される場合。 ● Oracle RAC システムでは、1 つのクラスタ・ノードからの読取り I/O で、別のノードからの書込み I/O 完了後に古いデータが返される場合(書込みの欠落)。たとえば、属性キャッシュを無効にせずに(noac オプションを使用しないなど)ネットワーク・ファイル・システム(NFS)が Oracle RAC でマウントされている場合に発生します。この場合、1 つのノードからの書込み I/O は、キャッシュされるために別のノードに即時表示されません。 宛先違いの書込みや書込み欠落が原因のブロック破損は、データベースの可用性を大きく損ねる可能性があります。データ・ブロックが物理的または論理的に正しくても、後続のディスク読取りで、古いブロックや Oracle Database のブロック・アドレスが正しくないブロックが表示されます。	タ ● ディスク・コントローラ障害 ● ボリューム・マネージャ・エラー ● その他のアプリケーション・ソフトウェア ● クラスタ全体で見えないネットワーク・ファイル・システム(NFS)書込み ● ソフトウェアまたはハードウェアの不具合
遅延、減速またはハング	遅延または減速が発生するのは、リソースやロックの競合が原因でデータベースまたはアプリケーションがトランザクションを処理できなくなったときです。認識される遅延は、システム・リソースの不足が原因で引き起こされます。	● データベースまたはアプリケーションのデッドロック ● システム・リソースを消費する暴走プロセス ● ログオン・ストームまたはシステム・フォルト ● アプリケーションのピークとシステムまたはデータベースのリソース不足の同時発生。リソースが適切に管理されていない統合データベース環境では、1 つのアプリケーションで起こることも、多数のアプリケーションで起こる

タイプ	説明	例
		こともあります。 ● アーカイブ REDO ログの宛先または高速リカバリ領域の宛先がいっぱいになった場合 ● 過剰にサブスクライブされた、または大規模に統合されたデータベース・システム

次の表は、計画停止のタイプと説明、および各タイプの例を示しています。

表1-2 計画停止時間の原因

タイプ	説明	例
ソフトウェアの変更	● 安定性とセキュリティのために軽微な修正を適用するための計画された定期的なソフトウェアの変更 ● 新機能を導入するための計画された年次または半年ごとのメジャー・アップグレード	● ソフトウェア更新(オペレーティング・システム、クラスタウェアまたはデータベースに対するセキュリティ更新を含む) ● オペレーティング・システム、クラスタウェアまたはデータベースのメジャー・アップグレード ● アプリケーション・ソフトウェアの更新またはアップグレード
システム変更およびデータベース変更	● 障害が発生したハードウェアを交換するための計画されたシステム変更 ● システム・リソースの拡張または縮小のための計画されたシステム変更 ● パラメータの変更を適用するための計画されたデータベースの変更 ● 新しいハードウェアまたはアーキテクチャに移行するための計画された変更	● サーバーへのプロセッサまたはメモリーの追加または削除 ● クラスタへのノードの追加、またはクラスタからのノードの削除 ● ディスク・ドライブまたはストレージ・アレイの追加および削除 ● 任意のフィールド交換可能ユニット(FRU)の交換 ● 構成パラメータの変更 ● システム・プラットフォームの移行 ● クラスタ・アーキテクチャへの移行 ● 新しい記憶域への移行

タイプ	説明	例
データ変更	Oracle Database オブジェクトの論理構造または物理的な編成に対する計画的なデータ変更。これらの変更は、主にパフォーマンスまたは管理性を向上させる目的で行われます。	● 表定義の変更 ● 表のパーティション化の追加 ● 索引の作成および再構築
アプリケーション変更	計画されたアプリケーション変更には、データ変更やスキーマおよびプログラムの変更などがあります。これらの変更は、主にパフォーマンス、管理性、および機能性を向上させる目的で行われます。	アプリケーション・アップグレード

オラクル社では、計画外停止時間と計画停止時間の回避、および障害からのリカバリに役立つ高可用性ソリューションを提供しています。「計画外停止時間用Oracle Database高可用性ソリューション」および「計画停止時間用Oracle Database高可用性ソリューション」では、これらの高可用性ソリューションの詳細を説明しています。

カオス・エンジニアリング

最大可用性アーキテクチャでは、テストおよび開発ライフ・サイクルを通してカオス・エンジニアリングを活用することで、障害やメンテナンス・イベントの際にアプリケーションとデータベースのエンドツーエンドの可用性を確保したり、最適なレベルに維持することを保証します。

カオス・エンジニアリングは、システムが本番環境で乱流状態に耐えることができるよう信頼性を高めるためにシステムで実験を行う分野です。具体的には、MAAでは様々な障害や計画メンテナンス・イベントを注入して、開発、ストレスおよびテスト・サイクルを通してアプリケーションとデータベースに対する影響を評価します。この実験により、ベスト・プラクティス、不具合および知見が導出され、その知識を応用してMAAソリューションを進化および向上させます。

最大可用性アーキテクチャ実装のロードマップ

Oracleの高可用性ソリューションと正常な操作実行が、ITインフラストラクチャの実装を成功させるキーです。ただし、テクノロジのみでは十分ではありません。

可用性要件に最も適したアーキテクチャを選択し、実装するのは、きわめて困難な作業です。Oracle Maximum Availability Architecture (MAA)は、次の事項を考慮したビジネス要件に適した高可用性アーキテクチャを選択し実装するプロセスを簡略化します。

- すべてのコンポーネントに冗長性を持たせること
- コンピュータ障害、ストレージ障害、人的エラー、データ破損、書込み欠落、システムの遅延または減速、およびサイト障害からの保護および許容が可能であること
- できるだけ素早く透過的に停止からリカバリすること
- 計画停止時間をなくすか短縮するソリューションを提供すること
- 一貫した高パフォーマンスと堅牢なセキュリティを実現すること
- デプロイメントおよび管理を簡単に実行し、最大のスケーラビリティ、パフォーマンスおよび可用性を達成するための

Oracle Engineered Systemおよびクラウドのオプションを提供すること

- できるかぎり安価な総所有コストでSLAを遵守すること
- オンプレミス、Oracle Public Cloud、およびオンプレミスとクラウドの一部で構成されるハイブリッド・アーキテクチャに適用すること
- コンテナまたはOracle Multitenant、Oracle Database In-MemoryおよびOracle Shardingアーキテクチャに特別な考慮事項を提供すること

この種のアーキテクチャを構築、実装および維持するには、次のことが必要です。

1. ITシステムおよびビジネス・プロセスの技術面および運営面の両方を含め、固有の高可用性の要件を分析します(「高可用性およびデータ保護 - 要件からのアーキテクチャの取得」を参照してください)。
2. Oracle MAAリファレンス・アーキテクチャの説明に従って、様々な高可用性アーキテクチャとその利点やオプションを評価します。
3. 各MAAリファレンス・アーキテクチャや、業務とアプリケーションに関する様々な高可用性機能に対する可用性の影響を理解します(「計画停止時間用Oracle Database高可用性ソリューション」および「計画外停止時間用Oracle Database高可用性ソリューション」を参照してください)。
4. Oracle高可用性機能を理解します。「可用性を最大化するための機能」を参照してください。
5. 運用のベスト・プラクティスを使用してMAAを正常に実装します。「可用性を最大化するための運用前提条件」を参照してください。
6. Oracle MAAのリソースを使用して高可用性アーキテクチャを実装します。Oracle MAAのリソースでは、様々なOracle MAA高可用性テクノロジに関する技術的な詳細が提供され、そのようなテクノロジを構成して使用するためのベスト・プラクティスの推奨事項も提供されます(Oracle MAAベスト・プラクティスのホワイト・ペーパー、概念実証に関する顧客の資料、顧客事例、記録されたWebキャスト、デモ、プレゼンテーションなど)。

   その他のOracle MAAリソースは、http://www.oracle.com/goto/maaで入手できます。

2 高可用性およびデータ保護 - 要件からのアーキテクチャの取得

Oracle Maximum Availability Architectureによって企業の高可用性要件を効果的に評価するフレームワークが提供される仕組みを学習するには、次のトピックを参照してください。

高可用性要件

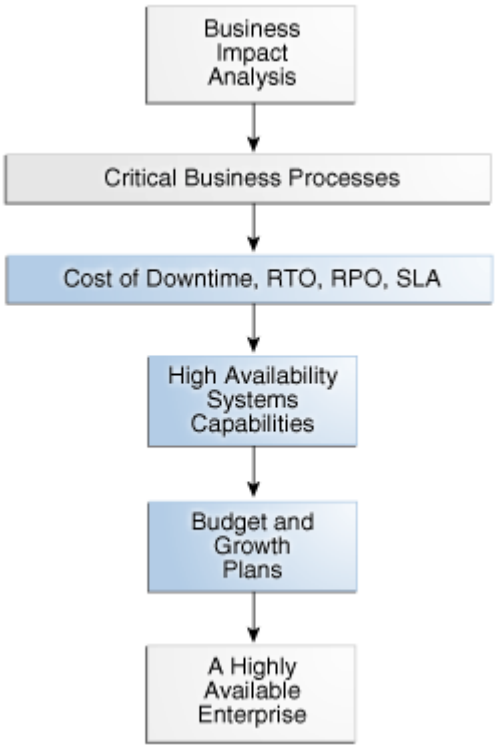
Oracle Databaseに高可用性戦略を設計して実装する作業は、ビジネス影響分析を徹底的に実行し、原因が計画外停止であるか計画停止であるかによって、停止時間やデータ損失でエンタープライズにどのような影響があるかを確認することから始めます。

エンタープライズが営利企業であるか、政府機関であるか、あるいは非営利団体であるかを意識しなくてよいように、ビジネス影響という言葉を使用しています。いずれの場合も、データ損失や停止時間は、エンタープライズの業務遂行能力に深刻な影響を及ぼします。高可用性の実現は、次のような重要な課題を伴います。

- レガシー・システムの廃止
- より高度で堅牢なシステムや設備への投資
- この高可用性モデルに合せたITアーキテクチャと操作全体の再設計
- 高可用性インフラストラクチャを最大限に利用するための既存アプリケーションの変更
- ビジネス・プロセスの再設計
- 人員の雇用およびトレーニング
- アプリケーション/データベースの一部また全体のOracle Public Cloudへの移動
- 連結、柔軟性、分離の適切なレベルでのバランス調整
- 既存のシステムおよびネットワーク・インフラストラクチャの機能と制限の把握

ビジネス要件を分析した上で、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を理解することで、ビジネス面と技術面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。

図2-1 可用性の高い企業の計画および実現



高可用性要件を文書化する方法

この分析フレームワークは次の要素で構成されます。

- ビジネス影響分析
- 停止時間のコスト
- リカバリ時間目標
- リカバリ・ポイント目標
- 管理性目標
- 総所有コストおよび投資利益率

ビジネス影響分析

ビジネス影響分析では、IT関連の停止の影響の重大度に基づいてビジネス・プロセスを分類します。

厳密なビジネス影響分析では、次のことが行われます。

- 組織内の重要なビジネス・プロセスの特定
- 各ビジネス・プロセスに影響を与える計画外IT停止および計画IT停止の定量化可能な損失リスクの算出
- 停止が与える影響の概略のまとめ
- 基本的なビジネス機能、人的リソースおよびシステム・リソース、政府規制、内部および外部のビジネス依存関係の考慮
- 豊富な知識を持つ経験豊かな人員とのインタビューで収集した客観的、主観的データの使用
- ビジネス手法の履歴や財務報告書、ITシステム・ログなどの調査

たとえば、世界各地にチップ製造工場を持つ半導体製造業者について考えてみましょう。半導体製造業は、高生産量で償却

される、巨額な財政投資を必要とする非常に競争の激しいビジネスです。工場管理で使用される人的リソース・アプリケーションは、工場の製造プロセスを制御するアプリケーションほどミッションクリティカルであるとみなされない傾向があります。製造をサポートするアプリケーションに障害が発生すると、その影響は生産レベルに及び、企業の決算報告に直接の影響が出ます。

別の例として、経営コンサルティング企業では内部ナレッジ管理システムがミッションクリティカルであるとみなされる傾向があります。これは、クライアントを重視する企業のビジネスにとって、コンサルタントやナレッジ・ワーカーが内部調査にアクセスできることが基本であるためです。このようなシステムの停止時間のコストは、このビジネスにとってきわめて高くつきます。

同様に、E-Commerce会社は、Webサイトへの顧客のトラフィックに大きく依存して収益をあげています。サービスの中断や可用性の喪失が発生すると、顧客の操作性が損なわれ、競合他社に顧客を奪われてしまいます。このため、顧客のトラフィックの急増に対して既存のインフラストラクチャをスケーリングして対応できることを確認する必要があります。オンプレミスのハードウェアを使用したり、クラウドを移動することでは、システムが常に稼働し続けるようにできない場合があります。

停止時間のコスト

ビジネス影響分析を完全に実施すれば、計画外停止時間および計画停止時間のコストの定量化に必要な見通しが得られます。

このコストは、高可用性の投資の優先付けに役立ち、停止時間のリスクを最小限に抑えるために選択する高可用性テクノロジに直接影響を与えるため、必ず理解しておく必要があります。

様々な業界の停止時間のコストをまとめた各種レポートが発行されています。コストは、仲買業務やクレジット・カード販売の1時間当たり数百万ドルから、荷物輸送サービスの1時間当たり数万ドルまで、多岐にわたります。

まさに驚くべき数字です。インターネットおよびクラウドでは、ビジネスと何百万もの顧客を直接結び付けることができます。アプリケーションの停止時間によってこの結び付きは分断され、ビジネスと顧客が切り離されてしまいます。停止時間は収益の損失を生むだけでなく、カスタマ・リレーションシップ、競争力、法的義務、業界での評価、株主の信頼などにも悪影響を及ぼす可能性があります。

リカバリ時間目標

ビジネス影響分析では、リカバリ時間目標(RTO)とも呼ばれる、停止時間の許容範囲を決定します。

RTOは、組織が受け入れがたい結果(経済的損失、顧客不満足、評価など)を被り始める前に、ITベースのビジネス・プロセスを停止できる最大時間として定義されます。RTOは、一般的に、ビジネス・プロセスまたは組織の許容停止時間に相当します。

RTO要件はビジネスのミッションクリティカル性によって左右されます。したがって、証券取引所を運営するシステムの場合、RTOはゼロまたはゼロに近い数値になります。

組織には、各種ビジネス・プロセスにまたがる様々なRTO要件があるものです。大規模なE-Commerce Webサイトの場合、高速のレスポンスが期待され、顧客のスイッチング・コストが非常に小さいため、E-Commerceの売上を押し上げるWebベースの顧客対応システムのRTOはゼロまたはゼロに近くなります。しかし、出荷や請求などのバックエンド業務をサポートするシステムのRTOはそれより長くてもかまいません。このようなバックエンド・システムが停止した場合は、一時的に手動操作に切り替えて、重大な影響が出るのを防ぐことができます。

障害の発生確率に基づいてRTOを変えている組織もあります。簡単なクラスの分離としては、ローカルな障害(単一データベースの計算、ディスク/フラッシュ、ネットワーク障害など)、およびそれと対照的な大規模な障害(完全なクラスタ、データベース、データの破損、サイトの障害など)があります。通常、ビジネスで重要なシステムを運用する顧客の場合は、ローカルな障害に対しては1分より短いRTO、大規模な障害に対しては1時間より短いRTOを設定することがあります。ミッション・クリティカルなアプリケーションの場合、RTOはすべての計画外停止で同じである可能性があります。

リカバリ・ポイント目標

ビジネス影響分析では、リカバリ・ポイント目標(RPO)とも呼ばれる、データ損失の許容範囲も決定します。

RPOは、組織に損害を与えずに、ITベースのビジネス・プロセスで失うことのできる最大データ量です。RPOは、一般的に、ビジネス・プロセスまたは組織の許容データ損失量を計測します。このデータ損失は、通常、データ損失ゼロ、何秒分、何時間分または何日分のデータ損失というように、時間に換算して測定されます。

毎分数百万ドルに相当するトランザクションが発生する証券取引所では、データを失うわけにはいきません。したがって、RPOをゼロにする必要があります。E-Commerceの例について言えば、Webベースの販売システムではRPOがゼロであることが求められるわけではありませんが、顧客満足のためにはRPOを抑えることが不可欠です。ただし、バックエンドのマーチャンダイジング・システムおよび在庫更新システムでは、RPOが大きくなることもあります。失ったデータの再入力が可能であるためです。

障害に対してRPOをゼロにすることは困難ですが、データベースを保護する各種のOracleテクノロジ(特にZero Data Loss Recovery Appliance)を使用して達成することができます。

管理性目標

管理性目標は、RPOやRTOよりも主観的です。組織で使用できるスキル・セット、管理リソースおよびツールの客観的な評価、および組織が高可用性アーキテクチャのすべての要素を正常に管理できる程度を決定する必要があります。

RPOおよびRTOが停止時間やデータ損失に対する組織の許容差を測定するのと同様に、管理性目標はIT環境の複雑さに対する組織の許容差を測定します。複雑さがそれほど要求されない場合、高可用性を実現するには、管理が複雑な方法よりも単純な方法が好まれ、複雑な方法の方が積極的なRTOおよびRPO目標を達成できる場合でも当てはまります。管理性目標を理解すれば、組織で実現が可能と思われる事項と実現が実用的である事項とを区別する際に役立ちます。

OracleデータベースをOracle Cloudに移動すると、管理性のコストと複雑さを大幅に軽減できます。Oracle Cloudを使用すると、最大可用性アーキテクチャの各種のアーキテクチャを組込みの構成やライフ・サイクル操作で選択できるからです。Autonomous Database Cloudを使用すると、バックアップとリストア、ソフトウェア更新、キーの修復操作などのデータベースのライフ・サイクル操作が自動化されます。

総所有コストおよび投資利益率

総所有コスト(TCO)と投資利益率(ROI)の目標を理解することは、組織のビジネス目標も達成する高可用性アーキテクチャを選択する際に不可欠です。

TCOには、選択する高可用性ソリューションの耐用年数におけるすべてのコスト(取得、実装、システム、ネットワーク、設備、スタッフ、トレーニング、サポートなど)が含まれます。同様に、ROIの計算では、指定の高可用性アーキテクチャで発生するすべての金銭的利益が算出されます。

たとえば、リモートのスタンバイ・サイトのITシステムおよびストレージがアイドル状態の高可用性アーキテクチャがあり、そのスタンバイ・システムで処理できるビジネス用途が他にないとします。このスタンバイ・サイトの投資利益率に含まれるのは、フェイルオーバー・シナリオで使用された場合に回避される停止時間関連のコストのみです。これを、スタンバイ・サイトのITシステムおよびストレージがスタンバイ・ロール(レポート作成や、プライマリ・システムからユーザー問合せのオーバーヘッドをオフロードする場合または読取り/書込みワークロードを分散する場合など)のときに、生産的に使用できる別の高可用性アーキテクチャと比較します。そのようなアーキテクチャの投資利益率には、回避された停止時間のコストと、高可用性やデータ保護を可能にすると同時に、生産的に使用した場合に発生する金銭的利益の両方が含まれます。

企業は、新しいデータ・センターを構築するために事前の資本投資を行うのではなく、ワークロードをクラウドに移動することによって、インフラストラクチャのニーズの増大によるTCOを減らすこともできます。経済的な面での主なアピールとしては、設備投資を

営業のための支出に変換することによって、ROIがより高くなることが挙げられます。

アーキテクチャへの要件のマッピング

ビジネス影響分析は、すでに認識されている事項をドキュメント化するために役立ちます。ビジネス影響分析の結果により、類似したRTOおよびRPO目標を持つデータベースをグループ化するために必要な見通しが提供されます。

様々なアプリケーションやそれらをサポートするいくつものデータベースがありますが、エンタープライズにとっての重要度は、それぞれに異なります。停止したとしても、すぐさまエンタープライズに影響するわけではないアプリケーションにとっては、高可用性インフラストラクチャへの高額の投資も意味を成しません。では、何から始めたらよいのでしょうか。

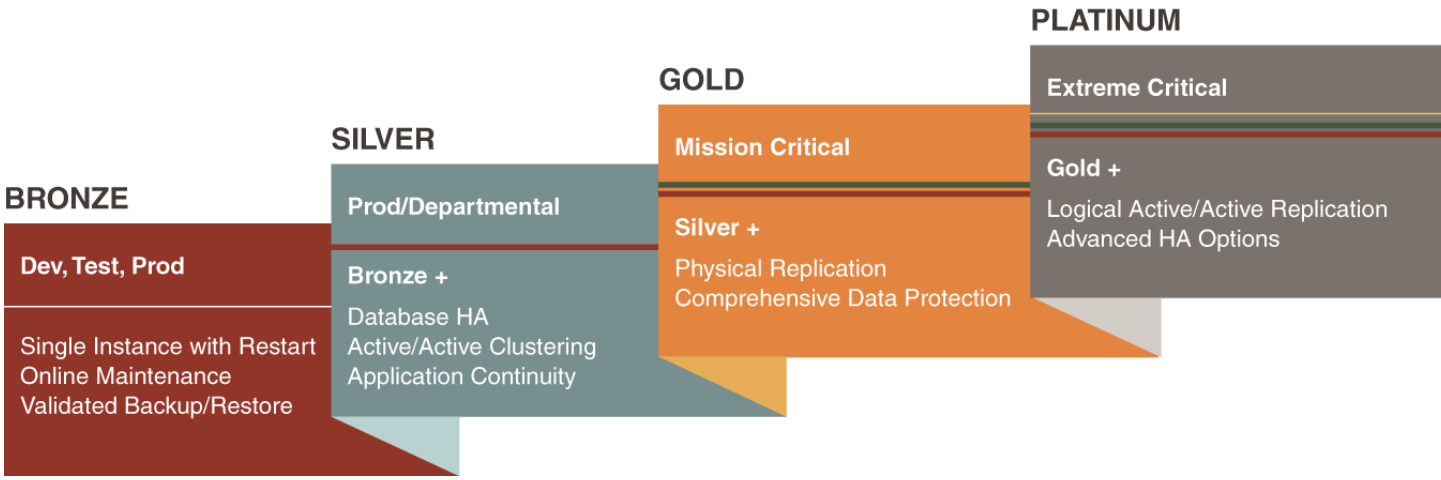
類似するRTOおよびRPOごとのデータベースのグループを、必要なサービス・レベルに最も近い処理が行われる高可用性リファレンス・アーキテクチャの制御されたセットにマップできます。データベース間に依存性がある場合は、高可用性要件が最も厳しいデータベースとグループ化されることに注意してください。

Oracle MAAリファレンス・アーキテクチャ

Oracle MAAのベスト・プラクティスでは、高可用性リファレンス・アーキテクチャを定義し、あらゆる規模や業種のエンタープライズで必要とされる可用性とデータ保護のすべての範囲に対応しています。

Platinum、Gold、Silver、BronzeのMAAリファレンス・アーキテクチャ(層)は、オンプレミス、プライベート・クラウドとパブリック・クラウド構成、およびハイブリッド・クラウドに適用できます。これらは、次の図に記載されているサービス・レベルに対応します。

図2-2 Oracle MAAリファレンス・アーキテクチャ



各層では異なるMAAリファレンス・アーキテクチャを使用して、Oracle高可用性機能の最適なセットがデプロイされ、指定されたサービス・レベルが、最小限のコストと複雑さで確実に達成されます。これらの層は、メンテナンス、移行またはその他の目的による計画停止のみでなく、データ破損、コンポーネント障害、システムやサイトの停止など、あらゆるタイプの計画外停止にも明示的に対応します。

Oracle Multitenantを使用するコンテナ・データベース(CDB)は、BronzeからPlatinumまでどの層にも存在でき、統合密度とTCOを高めることができます。通常、統合密度はBronzeおよびSilver層で高く、Platinum層をデプロイする場合、統合密度は低くなるか、ゼロです。

また、Oracle Database In-MemoryをどのMAA層でも利用できます。インメモリー列ストアがOracle Databaseにシームレスに統合されているため、MAA層から取得される高可用性のメリットは、Oracle Database In-Memoryの実装時に継承されます。

Oracle Engineered Systemはどの層にも存在できます。Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)をデータ・センター全体のOracle Databaseバックアップおよびリカバリ・ソリューションとして統合すると、バックアップから復元する際のRPOおよびRTOが削減されます。Oracle Exadata Database MachineをMAAリファレンス・アーキテクチャのデータベース・プラットフォームとして利用すると、最小のRTOおよび低下で最適なデータベース・プラットフォーム・ソリューションがExadataのMAAのサービス品質とともに提供されます。

関連項目:

高可用性リファレンス・アーキテクチャ

Oracle Exadata Database Machine: 最大可用性アーキテクチャおよびOracle Exadata Database MachineのMAAベスト・プラクティス

MAAホワイト・ペーパー『Oracle Database In-Memoryの高可用性のベスト・プラクティス』(http://www.oracle.com/goto/maa)

Bronzeリファレンス・アーキテクチャ

Bronze層は、障害が発生したコンポーネント(リスナー、データベース・インスタンス、データベースなど)の単純な再起動、またはバックアップからのリストアが十分なHAとDRで行われるデータベースに適しています。

Bronzeリファレンス・アーキテクチャは、すべてのOracle Enterprise Editionのライセンスに含まれているデータ保護と高可用性のための多くの機能を実装するMAAベスト・プラクティスを使用した、単一インスタンスのOracle Databaseに基づいています。Oracle Recovery Manager (RMAN)を使用してOracleに最適化されたバックアップによりデータが保護され、停止によってデータベースの再起動が妨げられる場合に、可用性をリストアするために使用されます。Bronzeアーキテクチャでは、Oracleのテクノロジ(Oracle Restart、Recovery Manager (RMAN)、Zero Data Loss Recovery Appliance、フラッシュバック・テクノロジ、オンライン再定義、オンライン・パッチ適用、自動ストレージ管理(ASM)、Oracle Multitenantなど)で拡張された冗長なシステム・インフラストラクチャを使用します。

Silverリファレンス・アーキテクチャ

Silver層では、ほとんどの一般的な計画メンテナンス・イベント(ハードウェアおよびソフトウェアのアップデートなど)のみでなく、データベース・インスタンスやサーバー障害の際に、停止時間を最小限またはゼロにする必要があるデータベースの高可用性のレベルが高くなります。

Silverリファレンス・アーキテクチャでは、Oracle RACまたはOracle RAC One Nodeを使用したクラスタリング・テクノロジを含む、エンタープライズ機能と利点の充実したセットが追加されます。また、アプリケーション・コンティニュイティは、実行中のトランザクションの信頼できるリプレイを提供し、ユーザーが停止を認識しないようにして、アプリケーションのフェイルオーバーを単純化します。

Goldリファレンス・アーキテクチャ

Gold層は、データベース、クラスタ、破損またはサイト障害などの障害に対して高いRTOおよびRPOを許容できないビジネスで重要なアプリケーションに対して、多大な役割を果たします。また、データベースのメジャー・アップグレードまたはサイトの移行を秒単位で実行できます。

Gold層では、すべてのレプリカが常時、積極的に使用されるため、投資利益率が向上すると同時に、コストも削減されます。

Goldリファレンス・アーキテクチャでは、データベース対応のレプリケーション・テクノロジ(Oracle Data GuardおよびOracle

Active Data Guard)が追加され、本番データベースのレプリカが1つ以上同期されて、リアル・タイムのデータ保護と可用性が実現されます。データベース対応のレプリケーションにより、ストレージ・レプリケーション・テクノロジで可能な範囲を超えて、高可用性とデータ保護(破損保護)が大幅に強化されます。Oracle Active Data Guardの遠隔同期は、距離に関係なくデータ損失なしの保護に使用されます。

従来のソリューションより優れているOracle Data Guardの利点も参照してください。

Platinumリファレンス・アーキテクチャ

Platinum層では、停止時間なしのアップグレードおよび移行のためのOracle GoldenGateを含めて、いくつかの新しいOracle Database機能が導入されています。

エディション・ベースの再定義により、アプリケーション開発者は停止時間なしのアプリケーションのアップグレードを設計できます。Oracle Sharding用のアプリケーションを別の方法で設計できます。この方法では、データベースのサブセットを可用性の高いシャードに分散することによって最大の可用性が提供され、アプリケーションは1つの論理データベースとしてデータベース全体にアクセスできます。

これらのテクノロジのいずれも、実装に追加の作業が必要ですが、停止時間が許されない、最も重要なアプリケーションにとっては十分な価値があります。

層ごとの高可用性およびデータ保護の属性

各MAAリファレンス・アーキテクチャにより、テストされた既知のレベルの停止時間およびデータ保護が提供されます。

次の表に、各アーキテクチャに固有の高可用性およびデータ保護の属性をまとめます。それぞれのアーキテクチャには、その前のアーキテクチャの機能がすべて含まれ、そのアーキテクチャ上に構築することで、対応範囲が広がった一連の停止に対処しています。各アーキテクチャに含まれる様々なコンポーネントや、達成されるサービス・レベルについては、別のトピックで説明します。

表2-1 MAAリファレンス・アーキテクチャ別の高可用性およびデータ保護の属性

MAAリファレンス・アーキテクチャ	計画外停止(ローカル・サイト)	計画メンテナンス	データ保護	リカバリ不能なローカル停止および障害時リカバリ
Bronze	単一インスタンス、リカバリ可能なインスタンスおよびサーバー障害での自動再起動。単一のコンポーネント(ディスク、フラッシュ、ネットワークなど)の障害によって停止時間が発生しないようにするための、システム・インフラストラクチャの冗長性。	一部オンライン、大部分オフライン	基本的なランタイム検証と手動チェックの組合せ	バックアップからのリストア、最後のバックアップ以降に生成されたデータを失う可能性。Zero Data Loss Recovery Appliance を使用すると、データ損失の可能性がゼロまたはほとんどゼロになります。
Silver	インスタンスおよびサーバーの障害に自動フェイルオーバーを使用する	ほとんどがローリング、一部がオンライン、ごく一部がオフライン	基本的なランタイム検証と手動チェックの	バックアップからのリストア、最後のバックアップ以降に生成されたデータを失う可能性。Zero Data

MAAリファレンス・アーキテクチャ	計画外停止(ローカル・サイト)	計画メンテナンス	データ保護	リカバリ不能なローカル停止および障害時リカバリ
	HA		組合せ	Loss Recovery Appliance を使用すると、データ損失の可能性がゼロまたはほとんどゼロになります。実行中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を使用して維持されます。
Gold	包括的な高可用性および障害時リカバリ	すべてがローリングまたはオンライン	包括的なランタイム検証と手動チェックの組合せ	リアルタイム・フェイルオーバー、ゼロまたはゼロに近いデータ損失
Platinum	Platinum 対応アプリケーションではアプリケーションの停止なし	アプリケーションの停止なし	包括的なランタイム検証と手動チェックの組合せ	Platinum 対応アプリケーションでは、アプリケーションの停止はゼロになります(データ損失もゼロ)。Oracle RAC、Oracle Active Data Guard および Oracle GoldenGate が相互に補完し合い、計画外停止でのデータベース・サービスの停止時間をゼロにするための広範なソリューションが提供されます。または、サイト障害の保護のために Oracle Sharding を使用すると、アプリケーションに対する影響がデータベース全体ではなく障害の発生したサイトのシャードのみになります。Platinum 対応アプリケーションの場合、各シャードは、リアルタイムのフェイルオーバー、ゼロまたはゼロに近いデータ損失、またはアプリケーションの停止なしで構成できます。実行中のトランザクションは維持され、データ損失はありません。

関連項目:

http://www.oracle.com/goto/maa

3 可用性を最大化するための機能

MAAソリューションで使用される次のOracle Database高可用性機能について理解します。

Oracle Data Guard

Oracle Data Guardは、企業データの高可用性、データ保護および障害時リカバリを保証します。

Data Guardは、Oracleデータベースが自然災害やデータ破損などのあらゆる種類の停止に耐えられるよう、1つ以上のスタンバイ・データベースの作成、メンテナンス、管理および監視を行うサービスの包括的なセットを提供します。Data Guardスタンバイ・データベースは本番データベースの正確なレプリカなので、従来のバックアップ、リストア、フラッシュバックおよびクラスタ技術と組み合せて透過的に利用し、可能なかぎり高いレベルのデータ保護、データ可用性および障害時リカバリを提供できます。Data GuardはOracle Enterprise Editionに含まれています。

Data Guard構成は、1つのプライマリ・データベースと1つ以上のスタンバイ・データベースからなります。プライマリ・データベースは、単一インスタンスOracleデータベースまたはOracle RACデータベースのいずれかになります。プライマリ・データベースと同様に、スタンバイ・データベースは、単一インスタンスOracleデータベースまたはOracle RACデータベースのいずれかになります。プライマリ・データベースのバックアップ・コピーを使用してプライマリ・データベースからREDOを直接受信するスタンバイ・データベースを最大で30作成できます。オプションで、カスケード・スタンバイを使用して、プライマリがREDOを単一のリモート宛先に転送し、その宛先がREDOを複数のスタンバイ・データベースに転送する、Data Guard構成を作成することができます。これにより、プライマリ・データベースは必要に応じて、30よりも多くのスタンバイ・データベースと効率的に同期することができます。

ノート:

Oracle Active Data Guardは基本Data Guardの拡張で、様々なタイプの処理を本番データベースからオフロードする高度な機能を提供し、データ損失ゼロの保護をどんな距離にでも拡張して、高可用性を強化します。Oracle Active Data GuardはOracle Database Enterprise Editionとは別のライセンスです。

スタンバイ・データベースにはいくつかのタイプがあります。Data Guardフィジカル・スタンバイ・データベースはデータ保護および障害時リカバリについてのMAAベスト・プラクティスで、最も一般的に使用されるスタンバイ・データベースのタイプです。フィジカル・スタンバイ・データベースはREDO適用(Oracleメディア・リカバリの拡張)を使用して、本番データベースの正確なフィジカル・レプリカを保持します。MAAベスト・プラクティスを使用して構成する場合、REDO適用は複数のOracle対応検証チェックを使用して、プライマリに影響を与える可能性のある破損がスタンバイに影響を与えることを防ぎます。その他のタイプのData Guardスタンバイ・データベースには、スナップショット・スタンバイ(テストまたはその他の目的で読取り/書込みでオープンするスタンバイ)、ロジカル・スタンバイ(計画停止時間の短縮のために使用)があります。

Data Guard使用の利点

- 更新がスタンバイ・データベースに適用される前に、Oracleデータ・ブロックおよびREDO内の構造の物理的および論理的整合性に対する複数のチェックを使用した、すべての変更のOracle対応の継続的な検証。これにより、スタンバイ・データベースが分離され、プライマリ・システムで発生する可能性のあるデータ破損による影響を受けることを防ぎます。
- 透過的な操作 - データ保護のためのData Guardフィジカル・スタンバイの使用には制約がありません。REDO適用はすべてのデータおよびストレージタイプ、すべてのDDL操作、およびすべてのアプリケーション(カスタムおよびパッケージ・アプリケーション)をサポートし、プライマリとスタンバイ・データベース間の整合性を保証します。
- 最高のパフォーマンス: 最高のリカバリ・ポイントの目標のための高速REDO転送、最高のリカバリ時間の目標のための高速適用パフォーマンス。マルチインスタンスREDO適用により、REDO適用にOracle RACのスケーラビリティが提供

され、単一のデータベース・サーバーのボトルネックが排除されます。REDO適用は、基本的にOracle RACクラスタで使用可能なCPU、I/Oおよびネットワークまでスケールアップされます。8ノードRAC Exadataでは、3500 MB/秒(12 TB/時)のRedo適用レートが確認されています。

- なんらかの理由でプライマリ・データベースに障害が発生した場合の、可用性を維持するためのスタンバイ・データベースへの高速フェイルオーバー。フェイルオーバーはData Guardの構成方法により、手動または自動の操作になります。
- フェイルオーバーの発生後、アプリケーション・クライアントが新しいプライマリ・データベースに接続できるようにする、統合されたクライアント通知フレームワーク。
- フェイルオーバーの発生後、障害の発生したプライマリ・データベースの自動または自動化された(構成による)再同期化、同期化されたスタンバイ・データベースへの迅速な変換。
- すべてのネットワーク構成、可用性およびパフォーマンスSLA、ビジネス要件をサポートする、柔軟なデータ保護レベルの選択。
- Data Guard Brokerコマンド・ライン・インタフェースまたはOracle Enterprise Manager Cloud Controlのいずれかを使用して、プライマリおよびそのスタンバイ・データベースのすべてを単一の構成として管理し、管理と監視を簡略化。
- Data Guard Brokerは次の追加機能により大幅に管理容易性が向上しています。どの時点でも、構成によるリカバリ・ポイントの目標およびリカバリ時間の目標のSLAのサポートを自動的に監視するための、包括的な構成のヘルス・チェック、再開可能スイッチオーバー操作、合理化されたロール・トランジション、カスケード・スタンバイ構成のサポート、転送および適用ラグに対するユーザー構成可能なしきい値。
- プライマリ本番データベースから送信されてカスケード・スタンバイ・データベースにより転送される単一のREDOストリームを使用した、複数のリモート宛先への効果的な転送。
- スナップショット・スタンバイにより、フィジカル・スタンバイ・データベースをテストおよび本番データのレプリカの読取り/書込みが必要なアクティビティ用に読取り/書込みでオープン可能。スナップショット・スタンバイは受信は続行しますが、プライマリにより生成された更新の適用は行いません。テストが完了すると、読取り/書込みでオープンしている間の変更を最初に破棄し、次にプライマリ・データベースから受信したREDOを適用することで、スナップショット・スタンバイは同期されたフィジカル・スタンバイ・データベースへと変換されて戻されます。プライマリ・データは常に保護されます。スナップショット・スタンバイはOracle Real Application Testingと一緒に使用される場合特に有用です(スタンバイ・データベース(本番の正確なレプリカ)でのリプレイおよび後続のパフォーマンス分析のためにワークロードが本番データベースで取得されます)。
- スタンバイ・データベースを使用してローリング方式でメンテナンスを実行することにより、計画停止時間を短縮。停止時間はData Guardスイッチオーバーの実行に必要な時間だけです。つまりアプリケーションはメンテナンスの実行中も使用可能なままです。
- プライマリ・システムとスタンバイ・システムで異なるCPUアーキテクチャまたはオペレーティング・システムが使用されているData Guard構成の柔軟性の向上は、My Oracle Supportのノート413484.1に定義されている制限が前提です。
- コンテナ・データベース(CDB)の効率的な障害時リカバリ。Data Guardフェイルオーバーおよびスイッチオーバーは、CDBに統合されたプラガブル・データベース(PDB)の数にかかわらず、CDBレベルで1つのコマンドを使用して実行されます。
- 特定の管理特権であるSYSDGで、Data Guardに対する標準の管理業務を処理できるようになります。この新しい権限は、特定の機能を実行するのに必要な権限のみユーザーに付与され、それ以上は付与されない、最低限の権

限の原則に基づいています。SYSDBA権限は以前のリリースでも引き続き動作します。

- Active Data Guard環境のスタンバイ・データベースで、Oracle Databaseインメモリー列ストアがサポートされます。
- 新しいRMANコマンドRECOVER DATABASE NOLOGGINGで修復できるようにNOLOGGING操作からの情報をトラッキングしてData Guard構成でのデータ・ウェアハウスのパフォーマンスと可用性をさらに改善します。
- 新しいパラメータDATA_GUARD_SYNC_LATENCYを使用して、複数のSYNCトランスポート宛先がプライマリ・データベースに与える影響を改善します。このパラメータは、少なくとも1つの同期スタンバイがREDOの受信を確認した後、後続の宛先を切断するまで、プライマリ・データベースが待機する最大時間(秒単位)を定義します。
- Data Guard Brokerは、代替の宛先の管理を拡張するのみでなく、プライマリとは異なるエンディアンの宛先をサポートすることで管理性を改善します。
- Data Guardは、次を含む複数の機能により、保護およびリカバリ時間目標(RTO)とリカバリ・ポイント目標(RPO)を改善します。
    - マルチインスタンスREDO適用(MIRA)は、Oracle RACインスタンスでスケーラブルなREDO適用パフォーマンスを実現し、本番環境のOLTPやバッチ・ワークロードが高い場合でもRTOを削減します
    - 新しいDBMS_DBCOMPパッケージを使用してプライマリ・データベースとスタンバイ・データベース・ブロックを比較し、書込み欠落を特定してそれらを効率よく解決できるようにできます。
    - Fast Start Failover (FSFO)は、最大保護モードのサポートによる高可用ゼロ・データ損失構成の堅牢性を備えると同時に、構成の高可用性のために複数のオブザーバーと複数のフェイルオーバーの柔軟性を提供します。FSFOは、プライマリで書込み欠落を検出してスタンバイに自動的にフェイルオーバーするよう設定することもできます。
    - 非同期構成での保管失敗後のデータ損失なしフェイルオーバーおよびアプリケーション・コンティニュイティに対するData Guard Brokerサポートにより RPOが改善されて、Data Guardロール・トランジション時のユーザー・エクスペリエンスが向上します。
- Oracle Data Guard Brokerは、プライマリ宛先が使用できない場合の代替アーカイブ先管理の拡張に加えて、プライマリとは異なるendianessの宛先をサポートして管理性を改善します。
- Oracle Data Guardデータベース比較ツールでは、Oracle Data Guardプライマリ・データベースとそのフィジカル・スタンバイ・データベースに格納されたデータ・ブロックを比較します。DBVERIFYユーティリティなどの他のツールでは検出できないディスク・エラー(書込み欠落など)を検出するには、このツールを使用します(Oracle Database 12cリリース2の新機能)。
- Oracle Data Guard Brokerは、ファスト・スタート・フェイルオーバー構成で複数の自動フェイルオーバー・ターゲットをサポートします。複数のフェイルオーバー・ターゲットを指定すると、必要なときに自動フェイルオーバーに適したスタンバイが常に存在する可能性が大幅に向上します(Oracle Database 12cリリース2の新機能)。
- Oracle Data Guard Brokerのファスト・スタート・フェイルオーバーのターゲットを動的に変更します。ファスト・スタート・フェイルオーバーを無効化せずに、ファスト・スタート・フェイルオーバーのターゲット・スタンバイ・データベースをターゲット・リスト内の別のスタンバイ・データベースに動的に変更できます(Oracle Database 19cの新機能)。
- プライマリからスタンバイ・サイトにリストア・ポイントを伝播します。プライマリ・データベースで作成されたリストア・ポイントは、フェイルオーバー操作後も使用できるようにスタンバイ・サイトに伝播されます(Oracle Database 19cの新機能)。
- Oracle Data Guardの自動停止解決は、特定のニーズにあわせてチューニングできます。Oracle Data Guardには、ハングしたプロセスを検出して停止するための内部メカニズムがあるため、通常の停止解決を実行できます

(Oracle Database 19cの新機能)。

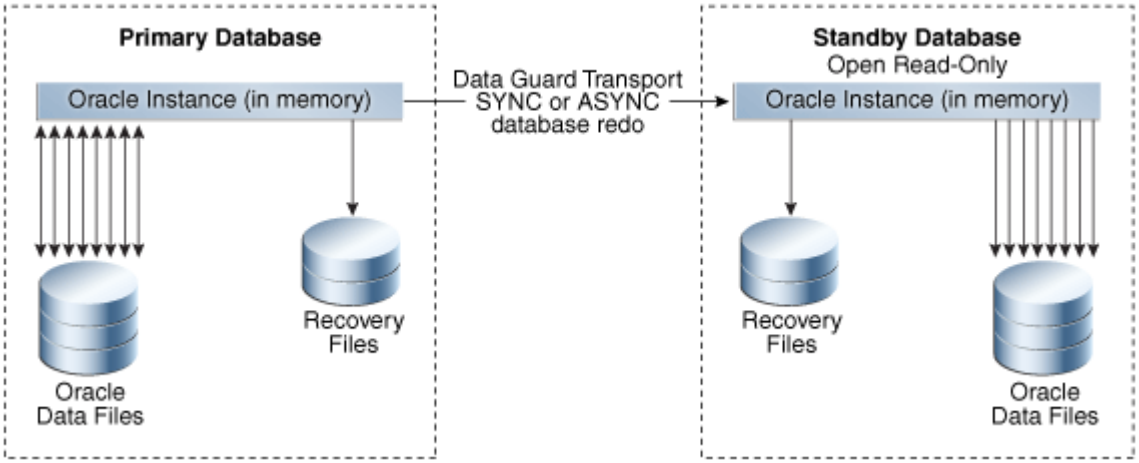
- Active Data GuardのDMLリダイレクションは、プライマリ・データベースとスタンバイ・データベース間のロード・バランシングに役立ちます。偶発的なデータ操作言語(DML)操作をActive Data Guardスタンバイ・データベースで実行できます。これにより、なんらかの書込みが必要なときにActive Data Guardスタンバイ・データベースの使用からメリットを得られるアプリケーションが多くなります。Active Data Guardのスタンバイ・データベースで偶発的なDMLが発行されると、更新は処理されるプライマリ・データベースに渡されます。トランザクションの結果のREDOによってスタンバイ・データベースが更新されてから、制御がアプリケーションに戻されます(Oracle Database 19cの新機能)。

Oracle Active Data Guard

Oracle Active Data Guardはフィジカル・レプリケーション・プロセスを使用した、Oracleデータベースのリアルタイムのデータ保護および障害時リカバリに対するOracleの戦略的ソリューションです。

Oracle Active Data Guardはまた、プライマリ・データベースから受信した変更を適用する間もスタンバイ・システムを読取り専用でオープンできることにより、障害時リカバリ・システムで高い投資利益率を提供します。Oracle Active Data Guardは、Oracle Enterprise Editionに含まれるData Guard機能を大幅に拡張する高度な機能を提供する、別個のライセンス製品です。

図3-1 Oracle Active Data Guardアーキテクチャ



Oracle Active Data Guardにより、プライマリ・データベースから受信した変更を適用する間も読取り専用でオープンできるフィジカル・スタンバイ・システムに、プライマリ・データベースから処理をオフロードすることにより、管理者はパフォーマンスを向上できます。Oracle Active Data Guardのオフロード機能には、次のものが含まれています。読取り専用レポート作成およびグローバル一時表および一意のグローバルまたはセッション順序へのDMLを含む非定型問合せ、データの抽出、高速増分バックアップ、REDO転送圧縮、複数のリモート宛先の効果的な処理、およびプライマリ・データベースのパフォーマンスに影響を与えずにデータ損失ゼロの保護をリモート・スタンバイ・データベースへ拡張する機能。また、Oracle Active Data Guardは、自動ブロック修復を実行し、高可用性アップグレードの自動化を有効にすることによって、高可用性を向上させます。

ノート:

Oracle Active Data GuardはOracle Database Enterprise Editionのデータベース・オプション・ライセンスとして、別個にライセンスされます。すべてのOracle Active Data Guard機能も、Oracle Enterprise EditionのOracle Golden Gateライセンスに含まれます。これにより顧客は、Oracle Active Data Guardのスタンドアロン・ライセンス、またはすべての高度なOracleレプリケーション機能へのアクセスを取得するためのOracle GoldenGateのライセンスを選択できるようになります。

Oracle Active Data Guardの利点

Oracle Active Data Guardは前述のData Guardの利点のすべてと、次の内容を継承します。

- プライマリ・データベースのパフォーマンスの向上: 読取り専用アプリケーション、レポート作成および非定型問合せのOracle Active Data Guardスタンバイ・データベースへの本番オフロード。読取り専用データベースと互換性のあるアプリケーションはOracle Active Data Guardスタンバイで実行できます。Oracleは多くのOracle E-Business Suite Reports、PeopleToolsのレポート作成、Oracle Business Intelligence Enterprise Edition (OBIEE)およびOracle TopLinkアプリケーションのOracle Active Data Guardスタンバイ・データベースへのオフロードを可能にする統合も提供します。
- DMLグローバル一時表およびスタンバイ・データベースでの順序の使用により、本番データベースからOracle Active Data Guardスタンバイ・データベースにオフロードできる読取り専用アプリケーションの数が大幅に増加します。
- 複数のOracle Active Data Guardスタンバイ・データベースを使用して、読取りパフォーマンスを容易に高めることのできるユニークな機能は、リーダー・ファームとも呼ばれます。
- Oracle Data Pumpまたはその他のソース・データベースから直接読み取る手法を使用した、データの抽出の本番オフロード。
- プライマリとスタンバイ・データベースが数百または数千マイルも離れている、同期したデータ損失ゼロの構成で、ネットワーク待機時間からのパフォーマンスの影響の本番オフロード。遠隔同期はプライマリ・データベースとは独立したシステムにデプロイされた軽量インスタンス(制御ファイルおよびアーカイブ・ログ・ファイルはあるが、リカバリ・ファイルおよびデータ・ファイルはない)を使用します。遠隔同期インスタンスは、アプリケーションが同期転送のパフォーマンスの影響に耐えて最適な保護を提供できる、プライマリ・システムから最も離れた場所に配置するのが理想的です。Data GuardはREDOを遠隔同期インスタンスに同期して転送し、遠隔同期はそのREDOを最終フェイルオーバー・ターゲットであるリモート・スタンバイ・データベースに非同期に転送します。プライマリ・データベースに障害が発生すると、Data Guard構成で使用される同じフェイルオーバー・コマンドまたはOracle Enterprise Manager Cloud Controlを使用したマウス・クリック、あるいはData Guardファスト・スタート・フェイルオーバーを使用した自動フェイルオーバーにより、リモート宛先へのゼロデータ損失フェイルオーバーが開始されます。この透過性は、同期構成でのWANネットワーク待機時間のプライマリ・データベースのパフォーマンスの影響を回避しながら、REDOをプライマリ・データベースから直接受け取っているかのように、データ損失ゼロの保護をリモート・スタンバイ・データベースに拡張します。
- 遠隔同期を使用した複数のリモート・スタンバイ宛先にサービスするオーバーヘッドの本番オフロード。遠隔同期構成では、プライマリ・データベースは同期または非同期転送を使用して、REDOの単一のストリームを遠隔同期インスタンスへ送信します。遠隔同期インスタンスは、ソース・データベースの増分オーバーヘッドなしで、REDOを29のリモート宛先に非同期で転送できます。
- Data Guardの最大可用性では、次の使用がサポートされます

  ```
  NOAFFIRM
  ```

  REDO転送属性。REDOがメモリーで受信されるとすぐに、スタンバイ・データベースは受領通知をプライマリ・データベースに返します。スタンバイ・データベースでは、リモート・ファイル・サーバー(RFS)によるスタンバイREDOログ・ファイルへの書込みを待っていません。

  この機能により、最大可用性とSYNCREDO転送を使用するData Guard構成でのプライマリ・データベースのパフォーマンスが向上します。高速同期により、最大可用性構成内のプライマリ・データベースが、スタンバイ・データベースでの遅いI/Oによるパフォーマンスの影響から隔離されます。この新しいFAST SYNC機能は、フィジカル・スタンバイ・ターゲットとともに使用することも、遠隔同期構成内で使用することも可能です。
- REDO転送圧縮の実行に必要なCPUサイクルの本番オフロード。Data Guard構成がOracle Advanced

Compression用にライセンスされている場合、REDO転送圧縮は遠隔同期インスタンスにより実行できます。これはプライマリ・データベースの増分オーバーヘッドなしでバンド幅を保存します。

- RMANブロック変更のトラッキングのためにOracle Active Data Guardサポートを使用することで、高速増分バックアップをプライマリ・データベースからスタンバイ・データベースへ移動することによる、本番オフロードおよびバックアップ・パフォーマンスの向上。
- プライマリまたはスタンバイのいずれかで検出された、ファイル・ヘッダー破損を含むブロック破損を、アプリケーションおよびユーザーに透過的に修復するための、Oracle Active Data Guard自動ブロック修復を使用した高可用性が向上。
- 高可用性アップグレードにより提供される追加の自動化を使用して、新しいOracle Databaseパッチ・セットおよびデータベース・リリースへのアップグレードの計画停止時間を短縮することにより、高可用性を向上。
- ロールの変更によりActive Data Guardスタンバイで接続を維持することで、レポート作成が容易になり、ユーザー・エクスペリエンスが向上。データベース・ロールがプライマリ・データベースに変更され、再開している間接続は一時接続するため、ユーザー・エクスペリエンスが向上します。
- Oracle Enterprise Manager診断ツールをActive Data Guardとともに使用してパフォーマンス・データをキャプチャし、自動ワークロード・リポジトリに送信できると同時に、SQLチューニング・アドバイザを使用してプライマリ・データベースのSQL文のチューニングをスタンバイ・データベースにオフロード可能。
- Oracle Database In-Memoryオプションに対するActive Data Guardのサポートにより、レポート作成をスタンバイ・データベースにオフロードできると同時に、スタンバイ・データベースのワークロード用に調整された列ストアなどのIn-Memoryオプションを利用可能。

従来のソリューションより優れているOracle Data Guardの利点

Oracle Data Guardには、従来のソリューションよりも優れた多くの利点があります。

- データ破損、書込み欠落、データベースおよびサイト障害に対するデータベース・フェイルオーバーは、高速かつ自動で実行され、従来のソリューションを使用した場合は数時間かかるリカバリが、Data Guardでは数秒になります
- Oracle Active Data Guard遠隔同期の使用時、ワイド・エリア・ネットワーク経由でデータ損失ゼロ
- Active Data Guard遠隔同期を使用したREDO転送圧縮処理のオフロードおよび最大29のリモート宛先へのREDO送信
- 破損自動修復の機能により、プライマリ・データベースまたはフィジカル・スタンバイ・データベースの物理ブロックの破損を、フィジカル・スタンバイ・データベースまたはプライマリ・データベースの良好なブロックをコピーして自動的に置換
- プライマリ・データベースでのデータ破損と書込み欠落に対する最も包括的な保護
- ストレージ、Oracle ASM、Oracle RAC、システムの移行と一部のプラットフォームの移行、およびData Guardスイッチオーバーを使用した変更による停止時間の短縮
- Data Guardのローリング・アップグレード機能による、データベースのアップグレードのための停止時間の短縮
- Database In-Memory列のサポートを含め、リアルタイム問合せの適用ラグ機能を使用して、スタンバイ・データベースを読取り専用リソースとして使用するRTOおよびRPOの機能を犠牲にせずに、プライマリ・データベースのアクティビティ(バックアップ、問合せまたはレポート作成など)のオフロードが可能
- サイト全体のフェイルオーバー操作の一部としてOracle Database File System (DBFS)またはOracle Automatic Storage Managementクラスタ・ファイル・システム(Oracle ACFS)を使用したデータベース以外のファイルの統合機能

- サイト障害の後で、インスタンスの再起動、ストレージの再マスタリングまたはアプリケーションの再接続が不要
- アプリケーションに対する透過性
- アプリケーション・フェイルオーバーに対する透過的な統合サポート(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)
- ネットワーク使用の効率化
- Database In-Memoryのサポート
- アプリケーション全体のRTOを短縮する統合されたサービスとクライアント・フェイルオーバー
- Oracle RAC、RMAN、Oracle GoldenGate、Enterprise Manager、ヘルス・チェック(orachk)、DBCA、フリート・パッチ適用およびプロビジョニングなどの既存のOracleテクノロジで拡張および統合されるData Guard認識

Oracle Databaseに常駐するデータの場合、データ損失ゼロ機能が組み込まれているData Guardは、データ保護および障害時リカバリに関して従来のリモート・ミラー化ソリューションよりも効率的かつ安価でより最適化されています。Data Guardには、障害時リカバリとデータ保護のテクノロジを選択する際に、従来のリモート・ミラー化ソリューションよりもData Guardを採用する方が正しいとする、技術上およびビジネス上の理由があります。

Data Guardおよび計画メンテナンス

Data Guardスタンバイ・データベースを使用してローリング方式でメンテナンスを実行することにより、計画停止時間を短縮できます。スタンバイ・データベースで変更が最初に実装されます。新しいバージョンが本番用に確実に準備できるまで、古いバージョンでプライマリを実行し、新しいバージョンでスタンバイを実行する構成が可能です。Data Guardスイッチオーバーを実行して、本番を新しいバージョンに移行させたり、同じ変更をローリング方式で本番に適用することができます。可能性のあるデータベースの停止時間は、スイッチオーバーを実行するのに必要な時間のみです。

Data Guardスタンバイを使用してローリング方式でメンテナンスを実行するには、いくつかの方法があります。顧客の要件および好みにより、どの方法を使用するかが決まります。

Data Guard REDO適用およびStandby-First Patchの適用

Oracle Database 10gから、Data Guard REDO適用を使用してクロス・プラットフォームのサポートの柔軟性が向上しました。

特定のData Guard構成で、プライマリおよびスタンバイ・データベースは、異なるオペレーティング・システム(WindowsとLinuxなど)、ワード・サイズ(32ビット/64ビット)、異なるストレージ、異なるExadataハードウェアとソフトウェア・バージョンまたは異なるハードウェア・アーキテクチャのシステムで実行できます。REDO適用を使用して、Oracle Automatic Storage Management (ASM)への移行、単一インスタンスOracle DatabasesからOracle RACへの移行、テクノロジ・リフレッシュの実行、またはあるデータ・センターから次のデータ・センターへの移動を実行することもできます。

Oracle Database 11gリリース2 (11.2)から、Standby-First Patchの適用(REDO適用を使用したフィジカル・スタンバイ)は、ローリング方式でOraclepパッチを適用および検証する目的で、プライマリ・データベースとそのフィジカル・スタンバイ・データベース間で異なるデータベース・ソフトウェア・パッチ・レベルをサポートします。Standby-First Patchの適用が可能なパッチには次のものがあります。

- データベースのリリース更新(RU)またはリリース更新のリビジョン(RUR)
- データベースのパッチ・セット更新(PSU)
- データベースのクリティカル・パッチ・アップデート(CPU)

- データベースのバンドル・パッチ

Standby-First Patchの適用は、Oracle Database Enterprise Edition 11gリリース2 (11.2)以降の認定データベース・ソフトウェア・パッチでサポートされます。

前述の計画メンテナンスの各タイプで、構成はプライマリおよびフィジカル・スタンバイ・データベースから始まります(新しいプラットフォームまたはASM、Oracle RACへの移行の場合、スタンバイは新しいプラットフォームに作成されます)。すべての変更がフィジカル・スタンバイ・データベースで実装されると、REDO適用(フィジカル・レプリケーション)が使用されてスタンバイとプライマリが同期します。Data Guardスイッチオーバーが使用され、本番がスタンバイ(新しい環境)に転送されます。

関連項目:

Data Guard構成でサポートされる混在プラットフォームの組合せの詳細は、My Oracle Supportのノート 413484.1を参照してください。

Standby First Patchの適用の詳細は、My Oracle Supportのノート1265700.1、ターゲット・パッチがStandby-First Patchとして認定されているかどうか判断するには、各パッチのREADMEを参照してください。

Data Guard一時ロジカル・ローリング・アップグレード

データベースの元のバージョンと変更後またはアップグレードされたバージョンを同期するのに、REDO適用(フィジカル・レプリケーション)を使用できない、多くのタイプのメンテナンス・タスクがあります。次のようなタスクがあります。

- Standby-First Patchの適用に適格ではないデータベース・パッチまたはアップグレード。これにはデータベース・パッチ・セット(11.2.0.2から11.2.0.4)および新しいOracle Databaseのリリース(18cから19c)へのアップグレードが含まれます。
- 停止時間を必要とするデータベースの物理構造を変更するメンテナンスが実行される必要があります(パーティション化されていない表へのパーティションの追加、Basicfile LOBからSecurefile LOBへの変更、XML-CLOBからBinary XMLへの変更、表のOLTP圧縮への変更など)。

Data Guard SQL Apply (ロジカル・レプリケーション)を使用して、以前のタイプのすべてのメンテナンスを、Data Guardスタンバイ・データベースを使用したローリング方式で実行し、古いバージョンのデータベースと新しいバージョンのデータベースを同期できます。Oracle Database 11g以前、これにはロジカル・スタンバイ・データベースの作成、ロジカル・スタンバイでのメンテナンスの実行、スタンバイとプライマリの再同期、およびスイッチオーバーが必要でした。さらに、フィジカル・スタンバイが障害時リカバリに使用された場合、新しいフィジカル・スタンバイ・データベースは新しいバージョンで本番データベースのバックアップから作成される必要があります。これは複数のTBデータベースのアップグレード時の、多くのロジスティックおよびコスト上の課題を表しています。

Oracle Database 11g以降、データベースのローリング・アップグレードでは、一時ロジカルと呼ばれる、フィジカル・スタンバイ・データベースで始まりフィジカル・スタンバイ・データベースで終わる、新しいプロシージャを使用できます。SQL Applyは、Data Guardが古いバージョンと新しいバージョンで同期している場合のフェーズ中でのみ使用されます。フィジカル・スタンバイがすでに所定の場所にある場合、新しいロジカル・スタンバイ・データベースを作成する必要はありません。メンテナンスが完了した後に、新しいフィジカル・スタンバイ・データベースを、新しいバージョンで本番データベースのバックアップから作成する必要はありません。インプレース・フィジカル・スタンバイを持つData Guard構成のアップグレードの従来のプロセスと同様に、元のプライマリは新しいプライマリ・データベースからのREDOおよびREDO適用を使用してアップグレードまたは変更されます(単一カタログのアップグレードではプライマリおよびスタンバイ・データベースの両方を新しいOracleリリースに移行します)。

一時ロジカル・アップグレードでは、プライマリ・データベースがOracle Database 11g リリース1 (11.1)以降で、データベースがSQL Applyの前提条件を満たしていることが必要です。

一時ロジカル・ローリング・アップグレード・プロセスに必要ないくつかの手動のステップを自動化するBourneシェル・スクリプトが用意されています。

Oracle Database Vaultを使用するデータベースは、Oracle Data Guardデータベース・ローリング・アップグレードを使用して新しいOracle Databaseリリースおよびパッチ・セットにアップグレードできます(一時ロジカル・スタンバイのみ)。

関連項目:

Oracle MAAホワイト・ペーパー『Oracle Databaseローリング・アップグレード: Data Guardフィジカル・スタンバイ・データベースの使用』(http://www.oracle.com/goto/maa)

Oracle Active Data Guardを使用したローリング・アップグレード

Oracle Active Data Guardを使用したデータベースのローリング・アップグレードでは、手動の一時ロジカル・ローリング・アップグレードで表される方法よりも、より簡単で、自動化された、容易に繰返し可能な、計画停止時間を短縮するための方法が提供されます。

Oracle Active Data Guardを使用したローリング・アップグレードでは、手動プロシージャで必要な42以上のステップが、いくつかの使いやすいDBMS_ROLLING PL/SQLパッケージに変換されます。DBMS_ROLLING PL/SQLパッケージを使用して実行するローリング・アップグレードが、マルチテナント・コンテナ・データベース(CDB)でサポートされます。

Oracle Active Data Guardを使用したローリング・アップグレードでは、次の操作が行われます。

- アップグレード・プロセスをガイドするための、構成に固有の指示セットを含むアップグレード・プランを生成
- ローリング・アップグレードのパラメータを変更
- アップグレードに関連するプライマリおよびスタンバイ・データベースを構成
- 本番データベースから新しいバージョンへのスイッチオーバーの実行。スイッチオーバーは必要な停止時間のみ
- Data Guard構成で古いプライマリと追加のスタンバイ・データベースのアップグレードを完了し、新しいプライマリと同期

Oracle Active Data Guardを使用したローリング・アップグレードには、次の利点もあります。

- 簡単な指定-コンパイル-実行プロトコルを提供
    - コンパイル・ステップで構成エラーを検出
    - ランタイム・エラーを処理中に検出
- 状態をデータベースに保持
    - 信頼できる、繰返し可能なプロセスが可能
- 含まれるデータベースの数に関係なく、実行時のステップは一定
- 元のプライマリ・データベースでエラーを処理
- アップグレードされたプライマリに対するデータ保護がいつでも可能

関連項目:

Oracle MAAホワイト・ペーパー『Oracle Databaseローリング・アップグレード: Data Guardフィジカル・スタンバイ・データベースの使用』(http://www.oracle.com/goto/maa)

Oracle GoldenGate

Oracle GoldenGateはデータ分散およびデータ統合のためのOracleの戦略的ロジカル・レプリケーション・ソリューションです。

Oracle GoldenGateはリアルタイムのログ・ベースの変更データの取得およびレプリケーション・ソフトウェア・プラットフォームを提供します。このソフトウェアでは、異種のデータベース間で、トランザクション・データのキャプチャ、ルーティング、変換および配信をリアルタイムに行います。

他のベンダーのレプリケーション・ソリューションとは違い、Oracle GoldenGateはOracle Databaseとより密接に統合されながらも、異機種データベース管理システム間のレプリケーションに最適な、オープンなモジュラー・アーキテクチャも提供します。これらの性質の組合せにより妥協が排されるため、Oracle GoldenGateはOracle Database環境と非Oracle Database環境にまたがる要件に対処するための優先の論理レプリケーション・ソリューションとなります。

典型的な環境として、取得、ポンプおよび配信プロセスがあります。これらの各プロセスは、Oracle Databaseを含む、ほとんどの一般的なオペレーティング・システムおよびデータベースで実行できます。データのすべてまたは一部をレプリケートできます。またこれらのどのプロセス内のデータも異種機間環境だけでなく、異なるデータベース・スキーマ、表名または表構造で操作できます。Oracle GoldenGateは、事前設定された競合検出と解決ハンドラにより双方向のレプリケーションをサポートし、データ競合を解決を支援します。

Oracle GoldenGate論理レプリケーションにより、Oracle GoldenGate構成のすべてのデータベース(ソースおよびターゲット・データベースの両方)を読取り/書込みで開くことができます。これにより、停止時間ゼロのメンテナンス、クロス・プラットフォームの移行を使用したデータの連続可用性に対する広範囲の挑戦に対処するための、MAAの主要なコンポーネントになっています。具体的には次のとおりです。

- ゼロまたはゼロに近い停止時間でのメンテナンス。このアーキテクチャにおいて、Oracle GoldenGateでは、Data Guardで提供される基本機能よりも高い柔軟性が提供されます。Oracle GoldenGateのソースおよびターゲット・データベースは、異なるフィジカルおよびロジカル構造を持つことが可能で、異なるハードウェアおよびオペレーティング・システム・アーキテクチャに存在でき、Oracle Databaseリリースが大きく異なっていること(12.2と19cなど)、またはOracleと非Oracleシステムの混在が可能です。これにより、24x7サーバーの近代化が可能になり、データベースの可用性に影響を与えずに新しいOracle機能を実装できます。メンテナンスは本番がソースで実行している間に、まずターゲット・データベースで実行されます。メンテナンスが完了すると、Data Guardスイッチオーバーのように、本番を一度にすべてソースに移動できます。オプションで、双方向レプリケーションを使用して、停止時間ゼロと認識させて、ユーザーを新しいシステムへ徐々に移動できます。いずれの場合も、Oracle GoldenGateレプリケーションは、遷移中、逆方向に元のソース・システム・データベースの同期化を保持することが可能で、これにより、必要な場合、最小限の停止時間でデータの損失なしで、簡単に前のバージョンへの計画フォールバックが実行できます。
- Data Guardソリューションが適用できない場合にゼロまたはゼロに近い停止時間での移行。プラットフォームまたはデータベースの移行は、古いシステムと新しいシステム間でのデータ同期方法としてOracle GoldenGateを使用して実行できます。データベースが別のホストでインスタンス化されると、Oracle GoldenGateは本番データベースから変更をレプリケートするよう構成されます。保証付きリストア・ポイントを移行したデータベースに作成して、ユーザー・テストの後、データベースをフラッシュ・バックできるようにします。また、Oracle GoldenGateは、スナップショット・スタンバイ・データベースと同様に、新しいデータベースにアプリケーション・ユーザーを移行する前に、本番データベースからの未処理のデータ変更を適用できます。必要に応じて、双方向のレプリケーションを移行したデータベースから本番データベースに戻して構成し、フォールバック・ソリューションとして使用することもできます。
- ゼロまたはゼロに近い停止時間でのアプリケーションのアップグレード。バックエンド・データベース・オブジェクトを変更する

アプリケーション・アップグレードでは、通常、メンテナンスの実行中、大幅な計画停止時間が発生します。Oracle GoldenGateレプリケーションでは、アプリケーションの前のバージョンにより使用されたデータベース・オブジェクトを、アプリケーションの新しいバージョンで変更されたオブジェクトにマップするデータ変換が可能です。これにより、アプリケーションの可用性に影響を与えずに、本番データベースの別のコピーで、データベース・メンテナンスの実行が可能になります。メンテナンスが完了し、Oracle GoldenGateの古いバージョンと新しいバージョンの同期が終了すると、ユーザーはアプリケーションの新しいバージョンにスイッチできます。

- ソース・データベースと同期化されている間のレプリカ・データベースへの読取り/書込みアクセス。これは、レポート・アプリケーションが動作のためにデータベースへの読取り/書込みを必要とする場合の、本番データベースのコピーへのレポート作成のオフロードに最も頻繁に使用されます。これは、アプリケーション層に使用されるテクノロジの性質が、リカバリ時間の目標を満たすために常にDRデータベースへのアクティブな読取り/書込み接続を必要とする、障害時リカバリ環境にも関係します。
- アクティブ-アクティブ・レプリケーション。Oracle GoldenGateは、アクティブ-アクティブのマルチ・ディメンショナル構成をサポートします。この構成では、いずれかのシステムのアプリケーション・ユーザーによって変更できる同じデータ・セットを持つシステムが2つ以上存在します。Oracle GoldenGateは、トランザクション・データの変更を各データベースから他のデータベースへレプリケートして、すべてのデータ・セットを最新の状態に維持します。
- データベース・インスタンスで障害が発生した場合、または適切なメンテナンス操作中に、Oracle Real Application Clusters (RAC)ノード間をシームレスに移動。この機能はOracle GoldenGateに高可用性を提供し、Oracle GoldenGateが現在実行されているノードに影響を与えずに、クラスタ内の1つ以上のノードでOracle GoldenGateソフトウェアにパッチを適用したり、アップグレードできます。その後、事前に決定した時間で、Oracle GoldenGateをアップグレードするいずれかのノードに切り替えることができます。構成情報はOracle RACクラスタで共有されるため、Oracle GoldenGateを再構成することなく切替えが実行されます。

関連項目:

Oracle GoldenGateのドキュメント

Oracle MAAのOracle GoldenGateのホワイト・ペーパー(http://www.oracle.com/goto/maa)

ベスト・プラクティス: Oracle Active Data GuardおよびOracle GoldenGate

Oracle Active Data GuardおよびOracle GoldenGateはそれぞれOracle Databaseの同期化されたコピーを保持できますが、要件に応じて、1つのテクノロジまたは別のテクノロジ、あるいは同時に両方を使用できる高可用性アーキテクチャになる、ユニークな特徴があります。

MAAベスト・プラクティスのガイドラインの例を次に示します。

Oracle Active Data Guardを使用する場合

簡易性、データ保護および可用性が重要視される場合は、Oracle Active Data Guardを使用します。

- Oracle Database全体の最も簡単で高速な一方向レプリケーション。
- 制約なし: Data Guard REDO適用はすべてのデータ型と記憶域のタイプおよびOracle機能、DDLの透過的レプリケーションをサポートします。

- データ保護に最適化された機能: ソースまたはターゲットに発生する可能性のあるサイレントな破損を検出し、破損ブロックを自動的に修復
- 同期されたスタンバイの読取り専用でのオープンにより、最大ROIのための簡単な読取り専用オフロードが提供
- バックアップの透過性: Data Guardのプライマリおよびスタンバイはお互いに物理的に正確なコピーであり、RMANバックアップは完全に代替可能
- データベースのパフォーマンスに影響を与えず、どんな距離でもデータ損失なしの保護
- Standby-First Patchの適用、データベースのローリング・アップグレード、プラットフォームの移行の選択を使用した計画停止時間およびリスクの最小化
- Data Guardスナップショット・スタンバイを使用したテスト用の二重目的のDRシステムによる変更の導入のリスクの減少
- 統合されたデータベースおよびクライアントの自動フェイルオーバー
- 構成全体の統合された管理: Data Guard Brokerコマンド・ラインまたはOracle Enterprise Manager Cloud Control

Oracle GoldenGateを使用する場合

Oracle Active Data Guardで対処されない高度なレプリケーション要件が重要視される場合は、Oracle GoldenGateを使用します。

- プライマリ・データベースと同期しながら、レプリカ・データベースを読取り/書込みでオープンする必要がある要件
- マルチマスターおよび双方向レプリケーション、サブセットのレプリケーション、多対1レプリケーションおよびデータ統合などのデータ・レプリケーション要件。
- エンディアン形式のプラットフォーム間またはデータベース全体のメジャー・バージョン間で、データ・レプリケーションが必要な場合。
- ゼロまたはゼロに近い停止時間が必要なメンテナンスまたは移行。Oracle GoldenGateを使用して、停止時間なしで、Application 1.0からApplication 2.0へなど、アプリケーションのバージョン間で移行できます。
- 高速フォールバックの目的で、新しいバージョンから古いバージョンにレプリケートする場合のデータベースのローリング・アップグレードは、うまくいきません。
- 双方向レプリケーションを使用して、停止時間ゼロを認識させて、ユーザーを新しいシステムへ徐々に移動する場合の、停止時間ゼロの計画メンテナンス。双方向レプリケーションでは異種のデータベースで発生する可能性のある更新の競合を回避または解決する必要があります。

Oracle Active Data GuardおよびOracle GoldenGateを一緒に使用する場合

Oracle Active Data GuardおよびOracle GoldenGateは相互に排他的ではありません。次に示すのは、Oracle Active Data GuardとOracle GoldenGateの同時使用を含む、高可用性アーキテクチャのユースケースです。

- Oracle Active Data Guardスタンバイは、ミッション・クリティカルなOLTPデータベースに対する障害保護およびデータベースのローリング・アップグレード用に使用されます。同時に、Oracle GoldenGateは、エンタープライズ・データ・ウェアハウスのETL更新に対し、Data Guardプライマリ・データベースから(またはOracle GoldenGate ALOモードを使用してスタンバイ・データベースから)データをレプリケートするのに使用されます。

- Oracle GoldenGateサブセットのレプリケーションを使用して、多数のデータ・ストアからデータを抽出、変換および集計する、操作データ・ストア(ODS)を作成します。ODSは企業に多くの収益をもたらす、ミッション・クリティカルなアプリケーション・システムをサポートします。Oracle Active Data Guardスタンバイ・データベースを使用して、ODSを保護し、最適なデータ保護と可用性を提供します。
- Oracle GoldenGateの双方向レプリケーションは、数千マイル離れた2つのデータベースを同期させるために使用されます。ユーザー・ワークロードは、グローバル・データ・サービス(GDS)を使用して、地理、ワークロードおよびサービス・レベルに基づき各データベース間で分散されます。Oracle GoldenGateのコピーは、停止が発生した場合にデータ損失ゼロのフェイルオーバーを可能にする、固有のローカル同期Data Guardスタンバイ・データベースを持ちます。Oracle GoldenGateのキャプチャおよび適用プロセスは、プライマリとスタンバイがお互いに正確で最新のレプリカであるので、フェイルオーバー後の新しいプライマリ・データベースで容易に再起動されます。
- 障害保護用に使用されているOracle Active Data Guardスタンバイ・データベースは、Data Guardによりサポートされない計画メンテナンスの実行のために、一時的にOracle GoldenGateターゲットに変換されます。たとえば、バックエンド・データベース・オブジェクトの変更を必要とするSiebelアプリケーションのアップグレードで、ユーザーを新しいシステムにスイッチオーバーする前に包括的なテストが必要な場合です。
- Oracle Active Data Guardは、停止時間がゼロまたはゼロに近い状態で、データベースのメジャー・バージョンのアップグレード(Oracle 18cから19cなど)が必要な場合に、本番環境の保護に使用されます。新しいバージョンのデータベースを使用して、プライマリ/スタンバイ環境がもう1つ作成され、Oracle GoldenGateは、本番環境からコピーの環境へ、一方向または双方向のレプリケーションでデータをレプリケートするために使用されます。Oracle GoldenGateで古い環境と新しい環境の同期が完了すると、本番は新しい環境に切り替わり、古い環境は廃止されます。これによって、構成により停止時間はゼロまたは最小となり、古い環境と新しい環境が完全に分離されることでリスクが軽減され、アップグレード・プロセス中に問題が発生しても、データ保護および可用性SLAへの影響が回避されます。

関連項目:

Oracle MAAベスト・プラクティス・ホワイト・ペーパー『Oracle Data GuardおよびOracle GoldenGateによる透過的なロール・トランジション』(http://www.oracle.com/goto/maa)

Recovery Manager

Recovery Manager (RMAN)は、データベースを効率的にバックアップおよびリカバリするための包括的な基盤を提供します。RMANは、操作の複雑さを排除するとともに、データベースの優れたパフォーマンスおよび可用性をもたらします。

RMANは、リクエストされたバックアップ、リストアまたはリカバリの操作を実行する最も効率的な方法を決定し、Oracle Databaseサーバーでの処理のために、これらの操作を発行します。RMANおよびサーバーは、データベース構造に加えられた変更を自動的に識別し、変更に適応するために必要な操作を動的に調整します。

RMANは、リカバリ・アプライアンス、ローカル・ディスク(ZFSストレージ)、テープ、クラウド・オブジェクト・ストアからバックアップおよびリストアする標準インターフェイスです。

RMANには次のような利点があります。

- Oracle Shardingのサポート - すべての独立したデータベース(シャード)に対するRMANのサポート
- スパース・データベースの拡張機能 - 次に対してバックアップおよびリストアを実行できます

  ```
  SPARSE
  ```

バックアップ・セットまたはイメージ・コピー(あるいはその両方)

- ネットワーク経由のスタンバイ・データベースの修復での

```
NONLOGGED
```

操作 - スタンバイでの検証および修復用の新しい構文 -

```
VALIDATE/RECOVER .. NONLOGGED BLOCK;
```

- `RMAN DUPLICATE`

機能。プライマリおよびバックアップからの遠隔同期の作成をサポートするために機能拡張されました

- `RMAN DUPLICATE`

暗号化バックアップの使用 - RMANは、次を使用した暗号化バックアップに基づく非自動ログイン・ウォレットがサポートするように機能拡張されます。新しい

```
SET
```

コマンド - 中断されることなくクローニングを実行できます

- ネットワーク経由でのクロス・プラットフォームのバックアップおよびリストアのサポート
- ネットワーク対応のリストアにより、次が可能になります

```
RESTORE
```

操作により、データベースから別のデータベースにネットワーク経由でデータ・ファイルを直接コピー

- 次を使用した簡略化された表のリストア

```
RECOVER TABLE
```

コマンド

- 個々のプラガブル・データベースのバックアップとリストアを含む、Oracle Multitenantのサポート
- 個々のPDBのバックアップとリストアを含む、クロスプラットフォームのOracle Multitenantのサポート
- バックアップおよびリストア操作での自動チャネル・フェイルオーバー
- リストア操作で欠落した、または破損したバックアップが検出された場合に、前回のバックアップへの自動フェイルオーバー
- リカバリ中の一時ファイルおよび新規データベースの自動作成
- 前回のポイント・イン・タイム・リカバリを使用した自動リカバリ(リセットログによるリカバリ)
- ブロック・メディア・リカバリでは、データ・ファイルをオンラインに保ったまま、ブロックの破損を修復
- ブロック・チェンジ・トラッキングを使用した高速増分バックアップ
- イントラファイルおよびインターファイルの並列処理による高速バックアップ操作とリストア操作
- 仮想プライベート・リカバリ・カタログによるセキュリティの強化
- 増分バックアップがイメージ・コピーにマージされ、最新の状態にリカバリ可能
- 必要なファイルのみ、バックアップおよびリストアを最適化
- 保存方針により、関連のバックアップを確実に保存

- 失敗した場合に、操作のバックアップおよびリストアを再開可能
- 制御ファイルおよびサーバー・パラメータ・ファイルの自動バックアップを行うことにより、データベース構造の変化や、メディア障害や災害の発生時に、バックアップ・メタデータの使用が可能
- 既存のバックアップから、または次を使用して本番データベースから直接(この場合ステージング領域は不要)、新しいデータベースを簡単に再インスタンス化

  ```
  DUPLICATE
  ```

  コマンド。

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』

Oracle Real Application ClustersおよびOracle Clusterware

Oracle RACとOracle Clusterwareを使用すると、Oracle Databaseはクラスタ化された一連のサーバー全体にパッケージまたはカスタム・アプリケーションを実行できます。

この機能により、高いレベルの可用性および最も柔軟的なスケーラビリティが得られます。クラスタ化されたサーバーに障害がある場合でも、Oracle Databaseは残りのサーバー上で稼働し続けます。より高い処理能力が必要なときは、データへのユーザーのアクセスを中断することなく、他のサーバーを追加できます。

Oracle RACを使用すると、インターコネクトでリンクされた複数のインスタンスによるOracleデータベースへのアクセスの共有が可能になります。Oracle RAC環境では、Oracle Databaseは単一の共有データベースに同時にアクセスしながら、クラスタ内の2つ以上のシステム上で稼働します。その結果、複数のハードウェア・システムにまたがる単一データベース・システムとなり、Oracle RACではクラスタ内での障害時に高可用性と冗長性を提供できます。Oracle RACは、読取り専用のデータ・ウェアハウス・システムから更新頻度の高いオンライン・トランザクション処理(OLTP)システムまで、あらゆるタイプのシステムに対応しています。

Oracle Clusterwareは、同じオペレーティング・システムを実行するサーバーにインストールされる場合、これらのサーバーが連係して単一サーバーとして機能できるようにして、ユーザー・アプリケーションとOracleデータベースの可用性を管理するソフトウェアです。また、ノードのメンバーシップ、グループ・サービス、グローバルなリソース管理、高可用性機能といったクラスタ管理に必要な機能もすべて提供します。

- 高可用性については、Oracleデータベース(単一インスタンスまたはOracle RACデータベース)とユーザー・アプリケーション(OracleおよびOracle以外)をOracle Clusterwareの管理と保護の下に置いて、プロセス障害時にはデータベースとアプリケーションが再起動し、ノード障害後には別のノードへのフェイルオーバーが行われるようにすることができます。
- クラスタ管理については、複数の独立したサーバーがまるで単一システムのイメージまたは単一の仮想サーバーであるかのように示されます。この単一の仮想サーバーは、すべての管理操作に対してクラスタ全体で維持されるため、管理者はインストール、構成、バックアップ、アップグレードおよび監視の機能を実行できます。その後、Oracle Clusterwareが、これらの管理機能の処理を、クラスタ内の該当するノードに自動的に分散します。

Oracle Clusterwareは、Oracle RACを使用するための要件です。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。Oracle Databaseでは引き続きサード・パーティのクラスタウェア製品を指定されたプラットフォームでサポートしますが、Oracle Clusterwareを使用すれば、次の主な利点があります。

- ベンダー独自開発のクラスタウェアが不要
- ローカルまたはリモートのOracle Automatic Storage Management(Oracle Flex ASM)によるディスク管理を提供するOracleから、Oracle DatabaseおよびOracle RACによるデータ管理まで、統合ソフトウェア・スタックを使用
- Oracle Flex Clusterと呼ばれる大規模クラスタに構成可能

さらに、OracleサービスなどのOracle Database機能では、Oracle Clusterwareの基本的なメカニズムを使用して、それらの機能を提供します。

Oracle Clusterwareには2つのクラスタウェア・コンポーネントが必要です。1つはノード・メンバーシップ情報を記録する投票ディスク、もう1つはクラスタ構成情報を記録するOracle Cluster Registry(OCR)です。投票ディスクとOCRは共有ストレージに存在する必要があります。Oracle Clusterwareでは、各ノードがプライベート・インターコネクト経由でプライベート・ネットワークに接続されている必要があります。

Oracle Clusterware使用の利点

Oracle Clusterwareには次のような利点があります。

- コンピュータおよびインスタンス障害を許容し、迅速にリカバリします。
- Oracle ClusterwareとOracle Databaseの併用により、管理およびサポートを単純に行えます。少数のベンダーとすべてのOracleスタックを使用することで、サード・パーティ・クラスタウェアを使用した場合より優れた統合を得られます。
- システム変更およびハードウェア変更のローリング・アップグレードを実行します。たとえば、ローリング方式で、Oracle Clusterwareのアップグレード、パッチ・セットおよび個別パッチを適用できます。

  Oracle Database 12cにアップグレードすると、Oracle ClusterwareおよびOracle ASMバイナリがOracle Grid Infrastructureという1つのバイナリとしてインストールされます。Oracle Clusterwareは、Oracle Clusterware 10gおよびOracle Clusterware 11gからローリング方式でアップグレードできますが、Oracle ASMはOracle Database 11gリリース1(11.1)からの場合のみローリング方式でアップグレードできます。
- エラーが発生したOracleプロセスを自動的に再開。
- 仮想IP(VIP)アドレスを自動的に管理。ノードに障害がある場合、そのノードのVIPアドレスは、VIPアドレスが接続を受け入れられる他のノードにフェイルオーバーされます。
- 障害が発生したノードのリソースを残りのノードで自動的に再開。
- Oracleプロセスを次のように制御します。
  - Oracle RACデータベースの場合、すべてのOracleプロセスはデフォルトでOracle Clusterwareによって制御されます。
  - 単一インスタンスのOracleデータベースの場合、Oracle Clusterwareで制御されるリソース・グループへのOracleプロセスを構成できます。
- OracleアプリケーションおよびOracle以外のアプリケーションの場合、Application Programming Interface(API)が提供され、Oracle Clusterwareによるその他のOracleプロセス(再開、または障害や特定のルールへの応答など)を制御できます。
- ノードのメンバーシップを管理し、2つ以上のインスタンスがデータベースを制御する際のスプリット・ブレイン・シンドロームを防止。

- サーバーの重みベースのノード削除を使用すると、ビジネス要件を含むクラスタで特定の失敗が発生した場合に削除されるノードの選択をあわせて、最も重要なワークロードが必ずできるだけ長く維持されるようにし、サーバー間で同じ選択をしているとみなすことができます。
- アプリケーションの停止時間ゼロで、Oracle Clusterwareのローリング・リリース・アップグレードの実行が可能。

Oracle Real Application ClustersおよびOracle Clusterware使用の利点

Oracle RACとOracle Clusterwareの併用には、Oracle Clusterwareのすべての利点に加え、次のような利点があります。

- すべてのOracleソフトウェア・スタックを使用することで、サード・パーティ・クラスタウェアを使用した場合より優れたOracle Databaseの統合およびサポートを提供します。
- Oracle Serviceの自動的再配置。さらに、高速アプリケーション通知(FAN)とクライアント構成を追加で実行する場合は、高速かつ自動のインテリジェントな接続とフェイルオーバーを実現するために、アプリケーションが素早く反応できるようなFANイベントの配信。
- 接続障害を高速に自動検出し、Oracle Universal Connection Pool(Oracle UCP)、高速接続フェイルオーバーおよびFANイベントを使用するJavaアプリケーションの終了済接続を削除します。
- Oracle UCPランタイム接続ロード・バランシングを使用して作業リクエストを均等に分散します。
- Oracle UCP、Oracle Call Interface(OCI)およびOracle Data Provider for .NET(ODP.NET)でランタイム接続ロード・バランシングを使用します。
- ロード・バランシング・アドバイザを使用して、使用可能なすべてのインスタンスに作業を分散します。
- Oracle Clusterwareが特定のデータベースに対するCPU要件と制限を認識するようデータベースを構成できます。Oracle Clusterwareはこの情報を使用して、CPU数、メモリー量またはその両方が十分備わっているサーバーにのみデータベース・リソースを配置します。
- 停止時間またはアプリケーションへの変更なしで、汎用ハードウェアを使用して処理能力を向上できる柔軟性を提供します。
- データベースおよびクラスタの機能を統合する包括的な管理性を提供します。
- データベース・インスタンス全体にわたるスケーラビリティを提供します。
- プールされていない接続の高速接続フェイルオーバーを実装します。

従来のコールド・クラスタ・ソリューションより優れているOracle RACの利点

Oracle RACには、従来のコールド・クラスタ・ソリューションよりも多くの利点があり、次のような利点があります。

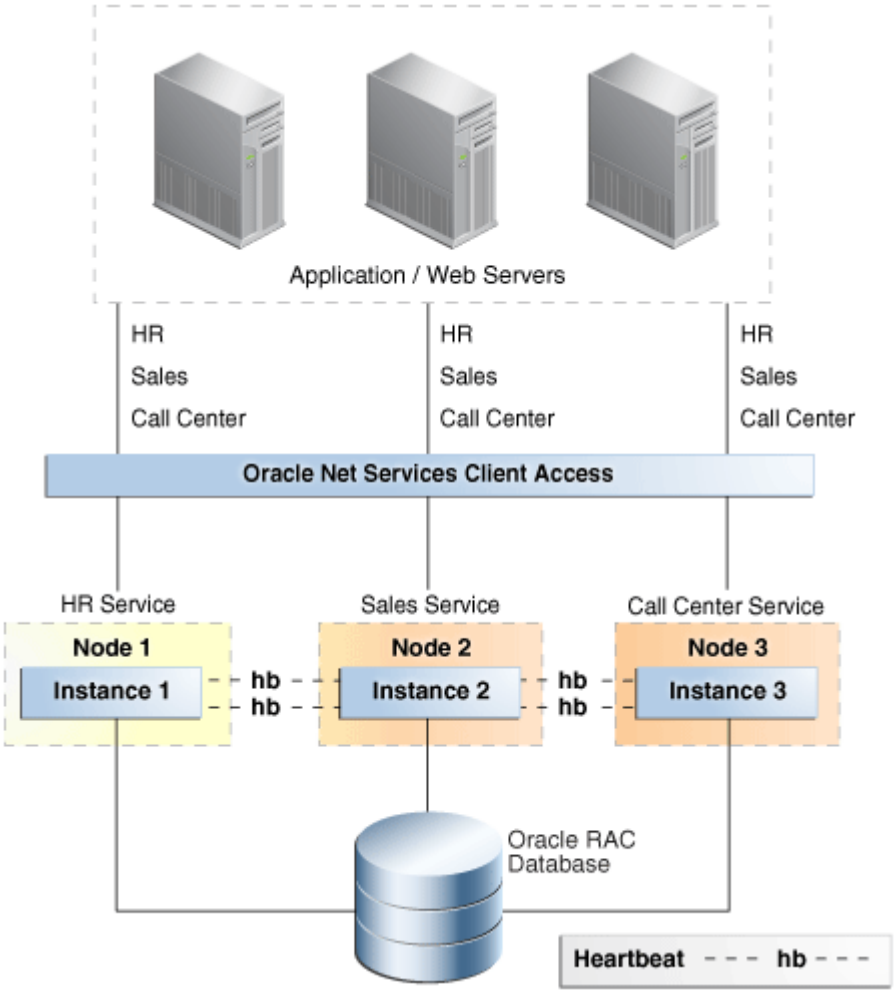
- データベース・インスタンス全体にわたるスケーラビリティ
- 停止時間またはアプリケーションへの変更なしで、汎用ハードウェアを使用して処理能力を向上できる柔軟性
- コンピュータおよびインスタンスの障害の許容および障害からの迅速なリカバリが可能(秒単位)
- アプリケーションの低下を、コールド・クラスタ・ソリューションの場合の数分から数時間と比べて、ゼロまたは数秒にすることが可能
- 結合またはその他のテクノロジを使用せずに、冗長ネットワーク・インタフェースを使用したクラスタでの通信を最適化

Oracle Grid InfrastructureおよびOracle RACによる、ネットワーク・トラフィックを分散し、クラスタ内の最適な通信を保証する冗長インターコネクトの使用。この機能は、Oracle Database 11gリリース2(11.2.0.2)以上で使用できます。それまでのリリースでは、結合またはトランキングのようなテクノロジを使用してインターコネクト用の冗長ネットワークを使用していました。

- システム変更およびハードウェア変更のローリング・アップグレード
- 個別パッチ、セキュリティ・パッチ、CPUおよびクラスタ・ソフトウェアのローリング・パッチ・アップグレード
- 高速、自動およびインテリジェントな接続とサービスの再配置ならびにフェイルオーバー
- データベースおよびクラスタの機能を統合する包括的な管理性(グリッド・プラグ・アンド・プレイ、およびポリシーに基づくクラスタ管理と容量管理を使用)
- 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・バランシングによる、適切なリソース全体での作業のリダイレクトおよび均等な分散
- データベース・ワークロードへのリソース割当てをポリシーベースでランタイム管理するためのOracle Quality of Service(QoS)管理(動的状況下でのビジネス・ニーズの順にサービス・レベルが満たされます)。これはデータベースが実行しているサーバー・プールにサービスを割り当てることにより行われます。プールからのリソースを使用して、必要な容量が使用可能になるようにします。
- Oracle Enterprise Managementによる、Oracle ASMおよびOracle ACFS、グリッド・プラグ・アンド・プレイ、クラスタ・リソース管理、Oracle ClusterwareとOracle RACのプロビジョニングとパッチ適用のサポート。
- Oracle RACに接続しているクライアントへの単一の名前としての、SCAN(単一クライアント・アクセス名)サポート。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。

次の図は、Oracle RACアーキテクチャを使用したOracle Databaseを示しています。この図は、パーティション化された3つのノードからなるデータベースの、Oracle RACを使用したOracle Databaseアーキテクチャを示しています。Oracle RACデータベースは、異なるノード上の3つのインスタンスに接続されています。各インスタンスは、サービス(人事、販売およびコール・センター)に関連付けられています。インスタンスはハートビートを確認することでお互いを監視します。Oracle Net Servicesは、図の最上部にあるアプリケーション/Webサーバー層へのクライアント・アクセスを提供します。

図3-2 Oracle RAC使用のOracle Databaseアーキテクチャ



ノート:

Oracle リリース 11.2 以降では、Oracle Clusterware (コールド・クラスタ・フェイルオーバー)を使用するよりも、より完全で、機能豊富なソリューションである Oracle RAC One Node または Oracle RAC をお薦めします。

関連項目:

Oracle RAC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 RAC One Node

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 RACデータベースのシングル・インスタンスです。

この機能により、最小限のオーバーヘッドで多数のデータベースを1つのクラスタに統合し、計画済および計画外の両方の停止から保護することができます。統合されたデータベースはフェイルオーバー保護、アプリケーションのオンライン・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・アップグレードという、高可用性の利点を得ます。

Oracle RAC One Nodeでは、オンライン・データベース再配置と呼ばれるOracleテクノロジにより、単一インスタンス・データベースのコールド・フェイルオーバーよりも高い可用性が可能になります。このテクノロジは高可用性とロード・バランシングのために

データベース・インスタンスと接続を別のクラスタ・ノードへとインテリジェントに移行します。オンライン・データベース再配置はServer Control Utility(SRVCTL)を使用して実行されます。

Oracle RAC One Nodeでは次のことが提供されます。

- 単一インスタンス・データベース・サービスが常に使用可能
- 高可用性のための組込みクラスタ・フェイルオーバー
- サーバーをまたがるインスタンスのライブ・マイグレーション
- 単一インスタンス・データベースに対するオンライン・ローリング・パッチ適用およびローリング・アップグレード
- 単一インスタンスから複数インスタンスOracle RACへのオンライン・アップグレード
- データベース・サーバーの優れた統合
- 強化されたサーバーの仮想化
- 完全Oracle RACの開発およびテスト・プラットフォームのコストの低減
- Data Guardとともに構成されているOracle RACプライマリおよびスタンバイ・データベースの再配置。この機能は、Oracle Database 11gリリース2(11.2.0.2)以上で使用できます。

また、Oracle RAC One Nodeはデータベース記憶域の統合を促進し、データベース環境を標準化し、必要な場合は、完全な複数インスタンスOracle RACデータベースに停止時間または中断なしに移行することができます。

Oracle Automatic Storage Management。

Oracle ASMは、垂直方向に統合されたファイル・システムおよびボリューム・マネージャを直接Oracle Databaseカーネル内に提供します。

この設計には、次のようないくつかの利点があります。

- データベース・ストレージのプロビジョニングに必要な作業量を大幅に削減
- より高いレベルの可用性
- 特殊なストレージ製品のコスト、インストールおよびメンテナンスを排除
- データベース・アプリケーションの固有の機能

最適なパフォーマンスを実現するため、Oracle ASMは、ファイルをすべての使用可能なストレージ間で分散させます。また、データ損失から保護するために、Oracle ASMは、SAME(Stripe and Mirror Everything)の概念を拡大して、ディスク全体のレベルではなくデータベース・ファイル・レベルでのミラー化が可能となるように、SAMEにより柔軟性を持たせます。

さらに重要なことに、Oracle ASMは、ミラー化の設定、ディスクの追加および削除プロセスを簡素化します。数百または数千にもなりうる(大規模なデータ・ウェアハウスの場合)多数のファイルを管理するかわりに、データベース管理者は、Oracle ASMを使用してディスク・グループと呼ばれるさらに大きなオブジェクトを作成および管理します。ディスク・グループは、論理ユニットとして管理されるディスク・セットを識別します。ファイルの名前付け、および基礎となるデータベース・ファイルの配置を自動化することにより、データベース管理者の時間を節約でき、標準的なベスト・プラクティスの適用を保証できます。

Oracle ASMネイティブ・ミラー化メカニズム(2方向または3方向)がストレージ障害から保護します。Oracle ASMミラー化を使用して、障害グループを使用する場合により高いレベルのデータ保護を実現できます。障害グループとは、障害が許容される、共通のリソース(ディスク・コントローラまたはディスク・アレイ全体)を共有するディスク・セットのことです。Oracle ASMの障害グループを定義することで、データの冗長コピーが個別の障害グループにインテリジェントに配置されます。これにより、ストレージの

サブシステム内のいずれかのコンポーネントで障害が発生した場合でも、このデータを使用でき、また、透過的に保護されるようになります。

Oracle ASMを使用すると、次のことが可能です。:

- ドライブおよびストレージ・アレイ間でのミラー化とストライプ化
- 障害が発生したドライブから残りのドライブへの自動再ミラー化
- データベースがオンラインのまま、ディスクの追加または削除時に格納されたデータの自動リバランスが可能
- Oracle Automatic Storage Managementクラスタ・ファイル・システム(Oracle ACFS)を使用したOracleデータベース・ファイルおよびデータベース以外のファイルのサポート
- データベース・ストレージ管理の操作の簡素化が可能
- Oracle Cluster Registry(OCR)および投票ディスクの管理
- インスタンスのローカル・ディスク上の優先読取り機能による、拡張クラスタのパフォーマンスの向上
- 大規模データベースのサポート
- Oracle ASMローリング・アップグレードのサポート
- ミラー・ディスクを使用したロジカル・データ破損の検出および修復のためのOracle ASMディスクのスクラブ・プロセスを使用した可用性と信頼性の向上
- チューニングおよびセキュリティのきめ細かな粒度のサポート
- ミラーのいずれかに適切なコピーが存在する場合、Oracle ASMの高速ミラー再同期およびブロック破損自動修復の機能を使用した、一時的なディスク障害発生後の高速修復を提供
- Oracle ACFSのネットワーク経由でのリモート・サイトへのレプリケーションを可能にすることにより、ファイル・システムの障害時リカバリ機能を提供
- Oracle Flex ASMを使用して、サービス中のクライアントに影響を与えることなく、Oracle ASMインスタンスのパッチを適用します。接続されている現在のOracle ASMインスタンスが計画メンテナンスでオフラインになっている間、データベース・インスタンスは、別の場所からOracle ASMメタデータにアクセスするように指定できます。
- Oracle ASMディスク再同期化およびリバランス操作の速度と状態の監視および管理
- Oracle ASM再同期化の指数制限を使用して再同期化の並列度を制御することにより、複数のディスクを同時にオンラインにし、パフォーマンスをより適切に管理します。セルまたはディスクの障害、および再同期を実行中のインスタンスのエラーの後の高速リカバリ。再同期化を最初から始めるかわりに中断または停止した時点から再開することができるディスク再同期チェックポイントを使用することでこれが可能です。
- Oracle Flex ASMを使用したデータベース・インスタンスの別のOracle ASMインスタンスへの自動的な接続計画外停止によりOracle ASMインスタンスに障害が発生した場合、ローカル・データベース・インスタンスは必要なメタデータおよびデータに引き続きアクセスできます。
- フレックス・ディスクグループを使用して、同じディスクグループを使用する複数のデータベースでの高可用性のメリットに優先順位をつけます。主なHAの利点には、ファイル・エクステントの冗長性、指数制限のリバランス、優先順位のリバランスなどがあります。フレックス・ディスクグループを使用して、データベースごとに前述の機能の異なる値を設定できます。その結果、1つのディスクグループ内の複数のデータベースで優先順位を付けることができます。
- フレックス・ディスクグループを使用して、1つのディスクグループを共有する複数のデータベースにquoto_groupsを実

装し、領域管理と保護に利用できます。

- フレックス・ディスクグループを使用して、ASM分割ミラー機能でポイント・イン・タイム・データベース・クローンを作成します。
- ストレッチ・クラスタで優先読取りを使用し、読取りを1つのサイトに関連付けてパフォーマンスを改善します。

関連項目:

Oracle Automatic Storage Management管理者ガイド

高速リカバリ領域

高速リカバリ領域は、Oracle Database内のすべてのリカバリ関連ファイルおよびアクティビティを対象とした一元的な格納場所です。

この機能を有効にすると、すべてのRMANバックアップ、アーカイブREDOログ・ファイル、制御ファイルの自動バックアップ、フラッシュバック・ログおよびデータ・ファイルのコピーが自動的に特定のファイル・システムまたはOracle ASMディスク・グループに書き込まれ、このディスク領域はRMANとデータベース・サーバーによって管理されます。

高速リカバリ領域を使用するとテープへの書込みのボトルネックが解消されるため、ディスクへのバックアップ実行が高速化されます。さらに重要なことに、データベースのメディア・リカバリを行う必要がある場合は、データ・ファイルのバックアップがすぐに使用できます。必要なデータ・ファイルおよびアーカイブREDOログ・ファイルをリストアするためのテープおよび空きテープ・デバイスを見つける必要がないため、リストアおよびリカバリの時間が短縮されます。

高速リカバリ領域には、次の利点があります。

- 関連リカバリ・ファイルの格納場所の一元化
- リカバリ・ファイルに割り当てられたディスク領域を管理し、データベース管理タスクを簡素化
- 高速で信頼性の高いディスクベースのバックアップおよびリストア

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』

破損の予防、検出および修復

データ・ブロックの破損は、非常に破壊的で修復が困難な場合があります。破損は、発生した場合に重大なアプリケーションやデータベースの停止時間およびデータ損失が発生することがあり、さらに数時間、数日および数週間検出されなかった場合は、検出時にアプリケーションの停止時間がより長くなります。残念ながら、データベース内のデータ破損を包括的に回避、検出および修復する1つの方法はありません。破損の発生源および原因は、メモリー、ハードウェア、ファームウェア、ストレージ、オペレーティング・システム、ソフトウェア、ユーザー・エラーなどの多岐に渡るからです。さらに悪いことには、Oracleデータ・ブロックのセマンティクスやOracleのデータ・ブロックの変更方法を理解していないサード・パーティのソリューションでは、データ・ブロック破損の防止および検出ができません。サード・パーティのリモート・ミラーリング・テクノロジは、データ破損をデータベース・レプリカ(スタンバイ)に伝播する可能性があり、これは二重障害、データ損失および停止時間の長期化につながります。サード・パーティのバックアップおよびリストア・ソリューションでは、リストアまたは検証操作が実行されるまで破損したバックアップまたは無効なセクターを検出できないため、リストア時間が長くなり、データが失われる可能性があります。

Oracle MAAには、物理ブロック破損、論理ブロック破損、逸脱した書込みおよび書込みの欠落を含む、あらゆる形のデータ・ブロック破損を防止、検出および修復する包括的な計画があります。これらの追加の保護手段は、最も包括的なOracleデータ・ブロック破損の防止、検出および修復ソリューションを提供します。この計画の詳細は、My Oracle Supportのノート「Data Guard構成における破損の検出、防止および自動修復のベスト・プラクティス」(Doc ID 1302539.1)に記載されています。

様々な手動操作チェック、およびランタイムやバックグラウンドの破損チェックに対するブロック破損チェックを次にまとめます。データベース管理者と作業チームは、Oracle Recovery Manager (RMAN)バックアップの実行、RMANのCHECK LOGICAL検証、または重要なオブジェクトに対するANALYZE VALIDATE STRUCTUREコマンドの実行など、手動チェックを組み込むことができます。手動チェックは、更新や問合せがあまり行われないデータを検証する際に、特に重要です。

ランタイム・チェックは、問合せや更新が頻繁に行われるデータの破損を、即座にまたはランタイムに検出する点で非常に優れています。ランタイム・チェックにより、破損が回避されたり、自動的に修正されたりするため、より強力にデータを保護し、アプリケーションの可用性をさらに高めることが可能です。Exadataに新しいバックグラウンド・チェックが導入され、アプリケーション・オーバーヘッドなしで、ディスクのスキャンとスクラブが自動的かつインテリジェントに行われ、物理的に破損したブロックは自動的に修正されます。

表3-1 ブロック破損チェックの概要

チェック	機能	物理ブロック破損	論理ブロック破損
手動チェック	Dbverify、Analyze	物理的ブロック・チェック	ブロック内およびオブジェクト間の論理的な一貫性チェック
手動チェック	RMAN	バックアップおよびリストア操作中の物理的ブロック・チェック	ブロック内論理チェック
手動チェック	ASM スクラブ	物理的ブロック・チェック	ブロック内の複数の論理チェック
ランタイム・チェック	Oracle Active Data Guard	1. 転送および適用中に、物理的ブロック・チェックをスタンバイにおいて継続的に実行 2. 強力なデータベース分離により、データベースの単一点障害を排除 3. Oracle Database 12c リリース 2 のファイル・ブロック・ヘッダーを含む、ブロック破損の自動修正 4. データベースの自動フェイ	1. DB_LOST_WRITE_PROTECT が有効化されている場合、書込み欠落を検出(11.2 以降)。11.2.0.4 および Data Guard Broker では、プライマリ・データベースで書込み欠落が検出された場合に、プライマリを停止可能。 2. スタンバイで DB_BLOCK_CHECKING が有効化されている場合、ブロック内の追加の論理チェックが可能

チェック	機能	物理ブロック破損	論理ブロック破損
		ルオーバー	
ランタイム・チェック	データベース	DB_BLOCK_CHECKSUM により、インメモリー・データ・ブロックおよび REDO チェックサムを検証	DB_BLOCK_CHECKING により、インメモリーのブロック内チェックを検証 Oracle Database 18c 以降では、シャドウ消失書込み保護を有効にすると、追跡対象のデータ・ファイルのシステム変更番号(SCN)が Oracle で追跡され、書込み欠落が早期に検出されます。書込み欠落が検出されると、エラーが即時に返されます。 この表の後にあるシャドウ消失書込み保護についての説明を参照してください。
ランタイム・チェック	ASM および ASM ソフトウェアのミラー化 (Exadata、Supercluster および Zero Data Loss Recovery Appliance に固有)	書込み中に、正常な ASM エクステント・ブロックのペアが存在する場合、読取りおよび書込みの暗黙的なデータ破損検出と自動修復を実行	.
ランタイム・チェック	DIX + T10 DIF	オペレーティング・システムから HBA コントローラ、ディスク(ファームウェア)に至るまでのチェックサム検証。認定済の Linux、HBA およびディスクに対する読取りと書込みを検証。	.
ランタイム・チェック	ハードウェアおよびストレージ	Oracle 統合が行われていないため、限定的なチェック。チェックサムが最も一般的。	Oracle 統合が行われていないため、限定的なチェック。チェックサムが最も一般的

チェック	機能	物理ブロック破損	論理ブロック破損
ランタイム・チェック	Exadata	書込みに対する包括的な HARD チェック	書込みに対する HARD チェック
バックグラウンド・チェック	Exadata	自動ハード・ディスク・スクラブおよび修復。不良セクターの検出および修正。	.

シャドウ消失書込み保護

Oracle Database 18cの新機能のシャドウ消失書込み保護では、データが大きく破損する前に書込みが失われたことを検出します。Oracle Data Guardスタンバイ・データベースを必要とせずに、データベース、表領域またはデータファイルのシャドウ消失書込み保護を有効化できます。シャドウ消失書込み保護は、消失書込みを迅速に検出して即時に対応し、データベースでデータ破損が原因で発生する可能性のあるデータ損失を最小限に抑えます。

関連項目:

これらのビューと初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

My Oracle Supportノート1302539.1

データ・リカバリ・アドバイザ

データ・リカバリ・アドバイザは、永続的な(ディスク上の)データ障害を自動的に診断し、適切な修復オプションを示し、リクエストに応じて修復操作を実行します。

データ・リカバリ・アドバイザを使用して、プライマリ・データベース、ロジカル・スタンバイ・データベース、フィジカル・スタンバイ・データベースおよびスナップショット・スタンバイ・データベースをトラブルシューティングできます。

データ・リカバリ・アドバイザには、次の機能があります。

- 障害診断

  通常、データベース障害の最初の兆候は、エラー・メッセージ、アラーム、トレース・ファイルおよびダンプ、ヘルス・チェックの失敗などです。これらの兆候を評価する作業はたいてい複雑で間違いやすく、時間がかかります。データ・リカバリ・アドバイザを使用すると、データ障害が自動的に診断され、これらの詳細が通知されます。

- 障害の影響のアセスメント

  障害の診断後、修復戦略について検討する前に、障害の範囲を把握し、アプリケーションに与える影響を評価する必要があります。データ・リカバリ・アドバイザを使用すると、障害の影響が自動的に評価され、わかりやすい形式でその結果が表示されます。

- 修復の生成

  障害が正しく診断されたとしても、正しい修復方法の選択は容易ではなく、負担になります。さらに、判断を誤ると、停止時間の増加やデータ損失という点で大きな不利益を被ることになります。データ・リカバリ・アドバイザは、自動的に一連の障害に関する最善の修復方法を判断して提示します。

- 修復の実行可能性チェック

データ・リカバリ・アドバイザでは、修復オプションが示される前に、特定の環境や提案される修復処理を完了するために必要なメディア・コンポーネントの可用性に関して、ファイルをプライマリまたはスタンバイ・データベースから直接リストアして提案された修復を完了するなどの、これらの修復オプションが検証されます。

- 修復の自動化

  提示された修復オプションを使用すると、この修復オプションが自動的に実行され、修復が成功したかどうかが検証されて、該当する障害が処置済となります。

- データの整合性およびデータベースのリカバリ可能性の検証

  データ・リカバリ・アドバイザでは、選択すればいつでも、データ、バックアップおよびREDOストリームの整合性を検証できます。

- 破損の早期検出

  状態モニターを使用して、データ・リカバリ・アドバイザによる診断チェックを定期的に実行するようスケジュール設定できます。これにより、トランザクションを実行しているデータベース・プロセスによって破損が検出されてエラーが通知される前に、データ障害を検出できます。早期の警告により、破損による損害を制限できます。

- データの検証および修復の統合

  データ・リカバリ・アドバイザは、データの検証および修復用の単一のツールです。

ノート:

データ・リカバリ・アドバイザでは、単一インスタンス・データベースのみがサポートされています。Oracle RAC データベースはサポートされていません。

関連項目:

データ・リカバリ・アドバイザでサポートされているデータベース構成の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください。

Oracle Flashback Technology

Oracle Flashback Technologyは、Oracle Database機能のグループで、データベース、データベース・オブジェクト、トランザクションまたは行の過去の状態を表示し、データベース、データベース・オブジェクト、トランザクションまたは行を、ポイント・イン・タイム・メディア・リカバリを使用せずに前の状態に戻すことができます。

フラッシュバック機能を使用すると、次のことができます。

- 過去のある時点でのデータの様子を表示する問合せの実行
- データベースに対する変更の詳細履歴を示したメタデータを戻す問合せを実行します。
- 表または行を前の時点にリカバリします。
- トランザクション・データの変更を自動的に追跡およびアーカイブします。
- データベースがオンラインである間にトランザクションおよびその依存トランザクションをロールバックします。
- 表の削除取消

- リストア操作なしでのポイント・イン・タイムへのデータベースのリカバリ

フラッシュバック・データベース機能以外に、ほとんどのOracle Flashback機能では、自動UNDO管理(AUM)システムにより、トランザクションに関するメタデータおよび履歴データが取得されます。フラッシュバック機能はUNDOデータに依存します。UNDOデータは、個々のトランザクションの結果のレコードです。たとえば、ユーザーが給与を1000から1100に変更するUPDATE文を実行すると、Oracle DatabaseによってUNDOデータに値1000が格納されます。

UNDOデータは永続的であり、データベース停止時にも失われません。フラッシュバック機能を使用すると、UNDOデータを使用して過去のデータを問い合せたり、論理的な損害をリカバリしたりすることができます。UNDOデータは、フラッシュバック機能以外でも、Oracle Databaseによって次の処理に使用されます。

- アクティブなトランザクションのロールバック
- データベースまたはプロセス・リカバリを使用した終了済トランザクションのリカバリ
- SQL問合せに対する読取り一貫性の提供

Oracle Flashbackは、不注意または悪意でデータを変更し、不正なインストールおよびアップグレードの原因となり、アプリケーションでのロジカル・エラーを招く様々な人的エラーまたはオペレータによるエラーにより損なわれたデータに対処して巻き戻すことができます。これらの問題では、フラッシュバック・トランザクション、フラッシュバック・ドロップ、表のフラッシュバック、データベースのフラッシュバックなどの機能が使用されます。

関連項目:

Oracle Database開発ガイド

Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイドのフラッシュバックおよびデータベースのポイント・イン・タイム・リカバリの実行、フラッシュバック・データベースおよびリストア・ポイントの使用およびブロック・メディア・リカバリの実行

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 Recovery Managerリファレンス』

Oracle Flashback Query

Oracle Flashback Query (フラッシュバック問合せ)では、自動UNDO管理システムを利用してトランザクションのメタデータおよび履歴データを取得することで、過去に存在したデータを表示できます。

UNDOデータは永続的で、データベースの異常や停止の際にも失われません。フラッシュバック問合せの独自の機能により、表の以前のバージョンの問合せが可能になり、誤った操作からリカバリするための強力なメカニズムも提供されます。

フラッシュバック問合せの用途は次のとおりです。

- 失われたデータのリカバリや、誤ったコミット済の変更の取消しを行います。たとえば、削除または更新された行を、コミット後でもただちに修復できます。
- 現行のデータと過去のある時点の対応するデータとを比較します。たとえば、前日のデータの変更を示す日次レポートを使用すると、表データの個々の行を比較したり、一連の行の共通部分または和集合を検索したりできます。
- 特定の日の勘定残高を確認するなど、ある時点におけるトランザクション・データの状態をチェックします。
- 特定のタイプの一時データを格納する必要をなくすことで、アプリケーションの設計を簡素化します。フラッシュバック問合せを使用すると、過去のデータをデータベースから直接取得できます。

- レポート生成ツールなどのパッケージ・アプリケーションを過去のデータに適用します。
- アプリケーションのセルフサービス・エラー修正を実現し、ユーザーが自分のエラーの取消しおよび修正を行えるようにします。

Oracle Flashback Version Query

Oracle Flashback Version QueryはSQLの拡張機能で、特定の表から特定の期間に存在した行のバージョンを取得するために使用できます。

Oracle Flashback Version Queryでは、指定期間に存在した行のバージョンごとに1行が返されます。どの表についても、COMMIT文が発行されるたびに行のバージョンが新たに作成されます。

Oracle Flashback Version Queryは、データベース管理者が問題の原因特定の分析を実行するために使用できる強力なツールです。さらに、アプリケーション開発者はOracle Flashback Version Queryを使用して、監査目的のカスタム・アプリケーションを作成できます。

Oracle Flashback Transaction

Oracle Flashback Transactionは、トランザクションとその依存トランザクションをバックアウトします。

```
DBMS_FLASHBACK.TRANSACTION_BACKOUT()
```

プロシージャは、データベースがオンラインのまま、トランザクションとその依存トランザクションをロール・バックします。このリカバリ操作では、UNDOデータを使用して補正トランザクションを作成および実行します。このトランザクションによって影響のあったデータが元の状態に戻ります。次の問合せができます

```
DBA_FLASHBACK_TRANSACTION_STATE
```

ビュー: トランザクションが依存性ルールを使用してバックアウトされたか、次のいずれかを実行することによって強制的に破棄されたかを確認できます。

- 競合していない行のバックアウト
- UNDO SQLの適用

Oracle Flashback Transactionでは、特定のトランザクション、またはトランザクションのセットとその依存トランザクションを迅速にバックアウトすることにより、論理リカバリ時の可用性が向上します。データベースがオンラインのまま、1つのコマンドを使用してトランザクションをバックアウトします。

Oracle Flashback Transaction Query

Oracle Flashback Transaction Queryは、データベースに加えられたすべての変更をトランザクション・レベルで表示するためのメカニズムを提供します。

Oracle Flashback Version Queryと併用した場合、人的エラーまたはアプリケーションのエラーからリカバリする高速で効率的な手段となります。Oracle Flashback Transaction Queryでは、行を変更したデータベース・ユーザーが返されるため、データベースの問題のオンライン診断を実行する能力が強化されます。また、トランザクションの分析および監査も実行されます。

Oracle Flashback Table

Oracle Flashback Tableを使用すると、過去のある時点の状態に表をリカバリできます。

人的エラーまたはアプリケーションのエラーによって変更された1つの表または一連の表をリカバリするための高速なオンライン・ソ

リューションを提供します。ほとんどの場合、Oracle Flashback Tableを使用することで、管理者がより複雑なポイント・イン・タイム・リカバリ操作を実行する必要性は軽減されます。元の表のデータは、Oracle Flashback Tableを使用すると表を元の状態に戻せるため失われません。

Oracle Flashback Drop

削除された表、索引、制約またはトリガーをリカバリする簡単な方法はありませんが、Oracle Flashback Dropでは、オブジェクトを削除する際の安全策が提供されます。

表を削除すると、その表は自動的にごみ箱に入ります。ごみ箱は、すべての削除済オブジェクトが入っている仮想コンテナです。削除した表のデータは、引き続き問い合せることができます。

リストア・ポイント

Oracle Flashbackのリカバリ操作をデータベースで実行するとき、後でデータをフラッシュバックできるポイント(システム変更番号(SCN)またはタイムスタンプで識別)を決定する必要があります。

Oracle Flashbackのリストア・ポイントとは、フラッシュバック・データベース、フラッシュバック表およびOracle Recovery Manager(RMAN)操作で使用されるSCNまたはトランザクション時間に置き換えるためにユーザーが定義できるラベルです。さらに、データベースは、前回のデータベース・リカバリによりフラッシュバックでき、次を使用してオープンできます。

```
OPEN RESETLOGS
```

コマンド(保証されたリストア・ポイントを使用)。保証されたリストア・ポイントを使用して、データベースの巻戻しに必要なUNDOが保存されるようにすることで、データベースのバッチ・ジョブ、アップグレードまたはパッチなどの主なデータベース変更を素早く元に戻すことができます。

リストア・ポイント機能を使用すると、次の利点があります。

- 一貫性のある状態、つまり失敗した計画操作(バッチ・ジョブ、Oracleソフトウェア・アップグレードまたはアプリケーション・アップグレードの失敗など)より前の適切なポイントに素早くリストアすることが可能
- スナップショット・スタンバイ・データベースをプライマリ・データベースと再同期させる機能
- テスト・データベースまたはクローン化されたデータベースを元の状態に戻す迅速なメカニズム

Oracle Flashback Database

Oracle Flashback Databaseは高速巻戻しボタンのようなもので、データベースを過去の時点にすばやく巻き戻します。バックアップやアーカイブ・ログを使用したリストアやロール・フォワードに時間をかける必要がありません。

データベースのサイズが大きくなるほど、Oracle Flashback Databaseを使用する高速なポイント・イン・タイム・リカバリの利点も大きくなります。

Oracle Flashback Databaseを有効にすると、次の利点があります。

- ポイント・イン・タイム・リカバリで論理破損(管理エラーを原因とする破損など)を高速に修復。
- Oracleリストア・ポイントと組み合わせた反復テストに便利。リストア・ポイントの設定、データベースの変更の実装、影響を評価するためのワークロードのテスト実行が可能。Oracle Flashback Databaseを使用して、変更を破棄して元の起点に戻り、別の変更を行ってから、同じワークロードでもう一度テスト実行することで、正当な基盤に基づいて異なる構成変更の影響を比較することができます。
- Data GuardはOracle Flashback Databaseを使用して、障害の発生したプライマリ・データベースを、(フェイル

オーバー後に)新しいスタンバイとしてすばやく再インスタンス化します(障害の発生したプライマリをバックアップからリストアする必要はありません)。

- Flashback DatabaseはCDBレベルまたはPDBレベルで動作します。

プラガブル・データベースのフラッシュバック

PDBを前のSCNまで巻き戻しできます。SQLまたはRecovery Managerから使用可能なFLASHBACK PLUGGABLE DATABASEコマンドは、非CDBでのFLASHBACK DATABASEに類似しています。

フラッシュバックPDBはデータ破損、広範囲のユーザー・エラー、およびREDO破損から個々のPDBを保護します。操作ではCDB内の他のPDBのデータは巻き戻しされません。

次を使用して

```
CREATE RESTORE POINT ... FOR PLUGGABLE
        DATABASE
```

PDBのリストア・ポイントを作成できます。これは、指定されたPDB内でのみ使用できます。CDBリストア・ポイントと同様に、PDBリストア・ポイントも通常または保証付きのいずれかです。保証付きリストア・ポイントは、制御ファイルからエージ・アウトされず、明示的に削除する必要があります。ルートに接続していて、次を指定しない場合、

```
FOR PLUGGABLE
        DATABASE
```

句。すべてのPDBで使用できるCDBリストア・ポイントが作成されます。

特殊なPDBリストア・ポイントのタイプはクリーン・リストア・ポイントで、これはPDBが閉じている場合のみ作成できます。共有UNDOを使用するPDBの場合、クリーン・リストア・ポイントまでのPDBの巻き戻しは他の操作よりも高速に行われます。これは、バックアップのリストアまたは一時データベース・インスタンスの作成が不要なためです。

フラッシュバック・ログまたはフィジカル・スタンバイ・データベースを使用したブロック・メディア・リカバリ

自動的な破損ブロック修復の試行後、ブロック・メディア・リカバリ機能で、フラッシュバック・ログからデータ・ブロックの最新コピーを必要に応じて取得して、リカバリ時間を短縮できます。

自動ブロック修復を使用すると、検出されたプライマリ・データベースの破損ブロックをフィジカル・スタンバイ・データベースの良好なブロックを使用してただちに自動修復できます。

その上、インスタンスのリカバリで発生した破損ブロックによってインスタンスのリカバリが失敗することはありません。そのブロックは自動的に破損とマークされ、次のRMAN破損リストに追加されます

```
V$DATABASE_BLOCK_CORRUPTION
```

表。その後、RMANの次を発行します

```
RECOVER BLOCK
```

コマンド。これにより、関連するブロックを修正できます。また、RMANの

```
RECOVER BLOCK
```

コマンドは、フィジカル・スタンバイ・データベース(使用できる場合)からブロックをリストアします。

フラッシュバック・データ・アーカイブ

フラッシュバック・データ・アーカイブは表領域に格納され、レコードの存続期間中に表のすべてのレコードに対して行われたトランザクションの変更が含まれます。

アーカイブされたデータは、UNDO表領域で提供される保存期間よりも長く保存でき、分析および修復のために非常に古いデータを取得するために使用されます。

Oracle Data Pumpとデータ転送

Oracle Data Pumpテクノロジを使用すると、データおよびメタデータをデータベース間で非常に高速に移動できます。Data Pumpは、次の計画メンテナンス・アクティビティを実行するために使用されます。

- 異なるプラットフォームへのデータベース移行
- プラガブル・データベースへのデータベース移行
- データベース・アップグレード

前述の計画メンテナンス・アクティビティを可能にするData Pump機能は次のとおりです。

- データベース全体を異なるデータベース・インスタンスに移動するフル・トランスポータブル・エクスポート/インポート
- データベース間で一連の表領域を移動するトランスポータブル表領域

関連項目:

データのトランスポート

データベース以外のファイルに対するOracleレプリケーション・テクノロジ

Oracle ASMクラスタ・ファイル・システム、Oracle Database File SystemおよびOracle Solaris ZFS Storage Applianceレプリケーションは、データベース以外のファイル用のOracleレプリケーション・テクノロジです。

表3-2 データベース以外のファイルに対するOracleレプリケーション・テクノロジ

テクノロジ	推奨される使用方法	コメント
Oracle ASM クラスタ・ファイル・システム	Oracle ASM、Oracle Clusterware および Oracle Enterprise Manager テクノロジと統合された単一ノードおよびクラスタ全体のファイル・システム・ソリューションの提供を推奨 Data Guard または Oracle GoldenGate との結合時に、疎結合されたフル・スタック・レプリケーション・ソリューションを提供します。	Oracle ACFS では、Oracle ASM インスタンスおよびディスク・グループのステータス更新やディスク・グループのリバランスなどの Oracle ASM の状態遷移に関与するために、Oracle ASM インスタンスとの通信を確立し、維持します。 実行可能ファイル、データベース・トレース・ファイル、データベース・アラート・ログ、アプリケーション・レポート、BFILE、構成ファイルなど、様々なデータベース・ファイルおよびアプリケーション・ファイルがサポートされま

テクノロジ	推奨される使用方法	コメント
		す。他にも、ビデオ、オーディオ、テキスト、イメージ、設計図、その他の汎用アプリケーションのファイル・データがサポートされます。 ポイント・イン・タイム・リカバリの発生時にデータベースの変更とファイル・システムの変更間で時間整合性をほぼ提供可能 NFS や CIFS などの標準 NAS ファイル・アクセス・プロトコルを使用してリモート・クライアントからアクセスおよびエクスポートできます。
Oracle Database File System,	データベースとデータベース以外のシステム間で強化された同期を提供する場合に推奨	データベースの変更とファイル・システムの変更間で完全に一貫性を保持するためにデータベースとの統合が可能 データベースに保存され、Oracle Active Data Guard と使用して障害時リカバリと読取り専用アクセスの両方を提供できるすべてのデータ Oracle データベースのすべての機能を利用可能
Oracle Solaris ZFS Storage Appliance レプリケーション	データベース以外のファイルに対する障害時リカバリ保護、特に Oracle Fusion Middleware のデータベース外に保存されたクリティカル・ファイルに推奨します。	Oracle Fusion Middleware バイナリ構成を含む、データベース以外のすべてのオブジェクトのレプリケート ポイント・イン・タイム・リカバリの発生時にデータベースの変更とファイル・システムの変更間で時間整合性をほぼ提供可能

Oracle ASMクラスタ・ファイル・システム

Oracle ASMクラスタ・ファイル・システム(ACFS)は、マルチプラットフォームのスケーラブル・ファイル・システムであり、Oracle Automatic Storage Management(Oracle ASM)機能を拡張して、Oracle Databaseの外部で保持されているカスタマ・ファイルをサポートするストレージ管理テクノロジです。

Oracle ACFSでは、実行可能ファイル、データベース・トレース・ファイル、データベース・アラート・ログ、アプリケーション・レポート、BFILE、構成ファイルなど、様々なデータベース・ファイルおよびアプリケーション・ファイルがサポートされます。他にも、ビデオ、オーディオ、テキスト、イメージ、設計図、その他の汎用アプリケーションのファイル・データがサポートされます。

Oracle ACFSでは次のOracle ASM機能を利用できます。

- Oracle ACFSの動的なファイル・システムのサイズ変更
- Oracle ASMディスク・グループ・ストレージへの直接アクセスによるパフォーマンスの最大化
- I/Oの並列性向上によるOracle ASMディスク・グループ・ストレージ全体でのOracle ACFSの分散の平均化
- Oracle ASMミラー化保護メカニズムによるデータの信頼性の確保

Oracle ACFSレプリケーションは、データベースに対するData Guardと同様、これによりネットワーク経由でのOracle ACFSファイル・システムのリモート・サイトへのレプリケーションが可能になり、ファイル・システムに対する障害時リカバリ機能が提供されます。Oracle ACFSレプリケーションはプライマリ・ファイル・システムのディスクに書き込まれた変更を取得し、その変更をレプリケーション・ログと呼ばれるファイルに記録します。このログは、関連するスタンバイ・ファイル・システムをホストするサイトに転送され、そこでバックグラウンド・プロセスがログを読み取り、ログに記録された変更をスタンバイ・ファイル・システムへ適用します。レプリケーション・ログに記録された変更がスタンバイ・ファイル・システムに正常に適用された後、レプリケーション・ログはプライマリおよびスタンバイ・ファイル・システムをホストしているサイトから削除されます。

Oracle ACFSの追加機能では、一般ファイルおよびアプリケーション・ファイルのスナップショット・ベースのレプリケーションが可能で、障害時リカバリおよびテスト/開発環境のためのHAソリューションが提供されます。ACFSに格納されたOracle Databaseでは、Oracle MultitenantおよびACFSスナップショット・テクノロジを利用して、プラガブル・データベースのスナップショット・クローンを迅速かつ効率的に作成できます。

Oracle Data GuardとOracle ACFSを結合し、スタンバイ・データベースでデータベースを保護するData Guardと、スタンバイ・ホストにファイル・システムの変更をレプリケートするOracle ACFSにより、フル・スタック高可用性ソリューションを提供できます。計画済停止の場合、ファイル・システムおよびデータベースはデータ損失なしで特定の時点との一貫性を維持します。

関連項目:

Oracle ACFS ASMクラスタ・ファイル・システム: 概要および使用方法

Oracle MAAホワイト・ペーパー『フル・スタック・ロール・トランジション-Oracle ACFSおよびOracle Data Guard』(http://www.oracle.com/goto/maa)

Oracle Database File System,

Oracle Database File System(DBFS)は、ファイルを格納するデータベースの利点と、リレーショナル・データを効率的に管理するデータベースの強度を活用して、データベースに格納されたファイルに対する標準のファイル・システム・インタフェースを実装します。

このインタフェースを使用すると、BLOBおよびCLOBプログラム・インタフェースを使用するために特別に記述されたプログラムに制限されずに、データベースにファイルを格納できるようになります。これで、ファイルに作用する任意のオペレーティング・システム(OS)プログラムを使用して、データベース内のファイルに透過的にアクセスできるようになります。たとえば、ETL(抽出、変換およびロード)ツールは、ステージング・ファイルを透過的にデータベースに格納します。

Oracle DBFSには次のような利点があります。

- フルスタックの統合リカバリおよびフェイルオーバー: ファイル・システムをデータベース構造に格納することにより、データベース・オブジェクトとファイル・システム・データの両方のポイント・イン・タイム・リカバリの容易な実行が可能です。
- 障害時リカバリ・システムの投資利益率(ROI): DBFSに含まれるファイルへのすべての変更も、OracleデータベースのREDOログ・ストリーム経由で記録され、Data Guardフィジカル・スタンバイ・データベースに渡すことができます。

Oracle Active Data Guardテクノロジを使用して、DBFSファイル・システムは、ソースとしてフィジカル・スタンバイ・データベースを使用して読取り専用でマウントできます。プライマリでの変更はスタンバイ・データベースに伝播され、スタンバイに適用されると表示できます。

- ファイル・システム・バックアップ: DBFSはデータベースにデータベース・オブジェクトとして格納されるので、標準RMANバックアップおよびリカバリ機能をファイル・システム・データに適用できます。データベースまたはデータベース内オブジェクト上で実行できるバックアップ、リストアまたはリカバリ操作は、DBFSファイル・システムに対して実行することもできます。

関連項目:

データベース・ファイル・システム(DBFS)

Oracle Solaris ZFS Storage Applianceレプリケーション

Oracle Solaris ZFS Storage Applianceシリーズでは、ソース・アプライアンスから任意の数のターゲット・アプライアンスに、手動で、スケジュールに従って、あるいは連続して、プロジェクトとシェアをスナップショットベースでレプリケートできます。

Oracle Solaris ZFS Storage Applianceシリーズでは、次のユースケースがサポートされます。

- 障害回復: レプリケーションを使用して障害時リカバリのためにアプライアンスをミラーリングできます。プライマリ・アプライアンス(またはデータ・センター全体)のサービスに影響を与える障害イベント時に、管理者は障害時リカバリ・サイトでサービスをアクティブ化し、最新のレプリケートされたデータを使用して引き継ぎます。プライマリ・サイトがリストアされると、障害時リカバリ・サイトがサービス中に変更されたデータは、プライマリ・サイトに移行して戻され、通常のサービスがリストアされます。このようなシナリオは、障害が発生する前に完全にテストすることができます。
- データ破損: レプリケーションを使用して、ターゲット・アプライアンスのクライアントが通常はソース・アプライアンスに直接到達できない、またはそのような設定では待機時間が非常に長くなるような状況で、データ(仮想マシン・イメージまたはメディア)を世界中のリモート・システムに分散できます。1つの例ではこのローカル・キャッシングのスキーマを使用して、読取り専用データ(ドキュメントなど)の待機時間を改善します。
- ディスク・ツー・ディスク・バックアップ: レプリケーションをテープ・バックアップが実行可能でない環境に対するバックアップ・ソリューションとして使用できます。たとえば、使用可能なバンド幅が不足しているため、またはリカバリの待機時間が長すぎるために、テープ・バックアップが実行可能でない場合があります。
- データ移行: レプリケーションを使用して、ハードウェアのアップグレード時または記憶域のリバランス時に、Oracle Solaris ZFS Storageアプライアンス間でデータおよび構成を移行できます。シャドウ移行はこの目的でも使用できます。

Oracle Solaris ZFS Storage Applianceのアーキテクチャによっても、これはOracle Fusion Middlewareの障害時リカバリに対しData Guardを補完するための理想的なプラットフォームになります。Oracle Fusion Middlewareにはデータベース外に保管された多数の重要なファイルがあります。これらのバイナリ、構成データ、メタデータ、ログなども、Oracle Fusion Middlewareの可用性を確保するためにデータ保護が必要です。これらに対し、ZFS Storage Applianceの組込みレプリケーション機能を使用してこのデータをリモートの障害時リカバリ・サイトに移動します。

Oracle Solaris ZFS Storage ApplianceをOracle Fusion Middlewareと使用する場合の利点には次のものがあります。

- Oracle Fusion Middlewareのリモート・レプリケーションの活用
- DRサイトのROIを向上させるデータベースのクローンおよびスナップショットを迅速に作成する機能

関連項目:

Oracle ZFS Storage Applianceソフトウェア

Oracle Multitenant

Oracle Multitenantはデータベースの統合に最適な方法です。マルチテナント・アーキテクチャでは、これまでの統合方法に付きものであったトレードオフはなくし、それぞれの一番よい特性が結合されています。

Oracle Multitenantは統合やプロビジョニング、アップグレードなどを簡素化することにより、ITコストの削減を支援します。この新しいアーキテクチャでは、コンテナ・データベース(CDB)に多数のプラガブル・データベース(PDB)を保持できます。アプリケーションでは、これらのPDBはスタンドアロン・データベースとして認識され、PDBにアクセスするためにアプリケーションを変更する必要はありません。複数のデータベースをPDBとして1つのCDBに統合することにより、複数のデータベースを1つのデータベースとして管理できるようになります。ビジネスで必要な場合には、分離してPDBを操作する柔軟性もあります。

Oracle Multitenantは、非コンテナ・データベース(非CDB)と同様に、Oracle Real Application Clusters、Oracle Data Guard、Oracle GoldenGateなどの高可用性機能と互換し、それらの機能を直接利用できるため、どのOracle MAAリファレンス・アーキテクチャでも使用できます。同じ高可用性要件を持つPDBを同じCDBにグループ化すると、それらすべてのPDBとそのアプリケーションは同じテクノロジと構成で管理および保護されます。

Oracle Multitenantを使用する利点

- 統合密度の高さ - 1つのCDBに、多数のPDBを格納できます。これらのPDBはバックグランド・プロセスとメモリー構造体を共有し、非CDBよりも多くのPDBを実行できます。これは、各非CDBのオーバーヘッドが削除または削減されるためです。CDBには最大4095個のPDBを格納できます。また、各PDBでは、同じCDB内の他のPDBと異なる文字セットを使用することもできます。この場合、CDBルートの文字セットはPDBのすべての文字セットのスーパーセットになります。ロジカル・スタンバイ・データベースは、この文字セットの混在もサポートして、一時ロジカル・スタンバイ・データベースを使用してローリング・アップグレード実行できます。
- クローン、リフレッシュ可能なクローン、PDBの再配置を含むオンラインのプロビジョン操作 - PDBを1つのCDBから切断して別のCDBに接続できます。PDBを同じCDBまたは別のCDBにクローニングすることも可能です。DBaaSまたはSaaS環境では、クローニングにより、ゴールド・イメージやシード・データベースを作成できます。このPDBをクローニングすれば、新規顧客用にデータベース環境を簡単に設定できます。
    - 停止時間がゼロに近いPDB再配置 – この機能では、クローン機能を使用してPDBを別のCDBに再配置する停止時間を大幅に短縮します。再配置が行われている間、ソースのPDBは開いたまま機能し続けます。アプリケーションの停止は非常に短時間に短縮され、ソースのPDBは一貫性のある状態になり、宛先のPDBは同期されてオンラインになります。この機能は別の新機能であるリスナー・リダイレクトも利用します。これにより、アプリケーションで同じ接続記述子を維持し、再配置後も宛先PDBに接続できます。
    - オンラインのプロビジョニングとクローニング – ソースのPDBを読取り専用モードにする必要なく、PDBのクローンを作成できます。ソースのPDBは読取り/書込みモードのままにすることができ、クローン操作中もアプリケーションにアクセスできます。
    - リフレッシュ可能なPDBのクローン – PDBのクローンは、設定された間隔で自動的にまたは手動でソースPDBへの変更を適用してリフレッシュするのと同じ方法で作成できます。リフレッシュ可能なクローンの場合は、読取り専用モードで維持する必要があります。クローンは読取り/書込みで開くことで通常のPDBに変換できます。リフレッシュ可能なクローンは、Exadataストレージ・スナップショットのテスト・マスターとして使用するのに

最適です。

- パッチ適用およびアップグレードの新オプション - CBDをアップグレードまたはパッチ適用すると、そのコンテナのPDBもアップグレードまたはパッチ適用されます。分離が必要な場合は、PDBを切断し、後のバージョンでCDBに接続できます。
- データベースのバックアップとリカバリ - 複数のデータベースをPDBとして統合することで、バックアップと障害リカバリなどの操作をコンテナ・レベルで実行できます。Oracle Multitenantでも、同じCDB内の、実行されているその他のPDBに影響を与えずに、個々のPDBを柔軟にバックアップおよびリストアできます。
- Oracle Data Guardを使用した操作 - CDBレベルでは、Data Guard構成が維持されます。Data Guardのロール・トランジション(フェイルオーバーかスイッチオーバーのいずれか)が実行されると、すべてのPDBが新しいプライマリ・データベースに移行されます。単一データベースの場合には必要になりますが、PDBごとに複数のData Guard構成を作成したり管理したりする必要はありません。Data Guard Standby-First PatchやData Guard一時ロジカル・ローリング・アップグレードなどの既存ツールを使用して停止時間を短くすることが可能で、コンテナ・レベルで実行されるため、すべてのPDBを一度の操作でメンテナンスできます。
    - Data Guard BrokerでのPDBの移行 – Data Guard Brokerが拡張されて、PDBをプライマリ・データベースまたはスタンバイ・データベースのCDBから別のCDBに自動的に移行できるようになりました。これは、PDBを同じバージョンで実行中の別のCDBまた上位バージョンで実行中CDBのいずれかに直接移行してアップグレード・プロセスを開始するために使用できます。この自動化を使用すると、スタンバイ・データベースでPDBファイルを使用して同じバージョンの別のCDBに接続し、単一のPDBフェイルオーバーに影響を与えることもできます。
    - サブセット・スタンバイ - Oracle Multitenantのユーザーは、スタンバイ・データベースへのレプリケーション用のCDBでPDBのサブセットを指定できます。これにより、どのスタンバイ・データベースにどのPDBを含めるか細かく指定できます。
- Oracle GoldenGateを使用した操作 - Oracle Multitenantには、Oracle GoldenGateで提供されているすべての機能があります。GoldenGateは、PDBレベルで柔軟に操作することができ、レプリケーションをCDB内のPDBのサブセットに対して実行できます。GoldenGateを使用して、CDBレベルまたは個々のPDBレベルで、停止時間が最小限またはゼロのアップグレードを実行できます。
- リソース管理 - Oracle Resource Managerで単一データベース間のリソース使用率を制御できるのと同じように、コンテナ内の個々のPDBのリソース使用率も制御できます。これにより、1つのPDBが、割り当てられている量を超えてシステム・リソースにアクセスすることはありません。PDBの制限で、SGA、バッファ・キャッシュ、共有プールおよびPGAメモリーの保証された最小値および最大値を指定できます。
- Oracle Flashback Databaseの操作 - 高速のポイント・イン・タイム・リカバリが必要な場合は、Oracle Multitenantの初期リリースを使用すると、CDBレベルでFlashback Databaseを使用できます。Oracle Multitenantでは、その他のPDBの可用性に影響を与えずに、個々のPDBでFlashback Databaseを使用できます。Flashback DatabaseはCDBレベルで実行でき、コンテナ内のすべてのPDBがフラッシュバックされます。プラガブル・データベースのフラッシュバック機能を使用して、個々のPDBをフラッシュバックできます。個々のPDBをフラッシュバックする場合、他のすべてのPDBは影響を受けません。
- Data Guard Broker PDBの移行またはフェイルオーバー - マルチテナントのブローカ構成では、本番PDBをコンテナ・データベースから同じシステムにある別のコンテナ・データベースに移動する必要がある場合があります。また、本番PDBで障害が発生したが、コンテナ・データベースおよび他のすべてのPDBが通常どおりに機能している場合、Data Guardスタンバイ・データベースから新しい本番コンテナ・データベースにPDBをフェイルオーバーする必要がある場合が

あります。新しいData Guard BrokerコマンドMIGRATE PLUGGABLE DATABASEを使用すると、単一PDBを別のコンテナ・データベースに移動したり、単一PDBをData Guardスタンバイから新しい本番コンテナ・データベースにフェイオーバーする操作を簡単に実行できます(Oracle Database 12cリリース2の新機能)。

関連項目:

- Oracle Multitenant管理者ガイド
- Oracle MAA技術概要『データベース統合のためのベスト・プラクティス』(https://www.oracle.com/database/technologies/high-availability/oracle-database-MAA-best-practices.html)

Oracle Sharding

Oracle Shardingはアプリケーションに対する拡張性および可用性の機能で、シャード・データベースで実行されるように明示的に設計されています。

Oracle Shardingを使用すると、ハードウェアやソフトウェアを共有しないOracleデータベースのプール間でデータを配布およびレプリケーションできます。データベースのプールは1つの論理データベースとしてアプリケーションに示されます。プールにデータベース(シャード)を追加するだけで、任意のレベル、任意のプラットフォームにアプリケーション(データ、トランザクションおよびユーザー)を柔軟にスケーリングできます。最大1000個のシャードまでのスケール・アップがサポートされています。

Oracle Shardingは、スケーラビリティに対して同様のアプローチを使用するカスタムのデプロイメントに比べ、より優れたランタイム・パフォーマンスとよりシンプルなライフサイクル管理を提供します。リレーショナル・スキーマ、SQL、およびその他のプログラム・インタフェース、複雑なデータ型のサポート、オンライン・スキーマの変更、マルチコア・スケーラビリティ、高度なセキュリティ、圧縮、高可用性、ACIDプロパティ、読取り一貫性、JSONを使用した開発者の俊敏性などを含むエンタープライズDBMSの利点も提供します。

関連項目:

Oracle Shardingの使用

Oracle Restart

Oracle Restartは、単一インスタンスの(クラスタ化されていない) Oracleデータベースおよびそのコンポーネントの可用性を向上します。

Oracle Restartは、単一インスタンス環境でのみ使用されます。Oracle Real Application Clusters(Oracle RAC)環境では、Oracle Clusterwareによってコンポーネントを自動再起動する機能が提供されます。

Oracle Restartをインストールすると、ハードウェアまたはソフトウェア障害の後、あるいはデータベースのホスト・コンピュータが再起動するときは必ず、データベース、リスナー、およびその他のOracleコンポーネントは自動的に再起動します。また、Oracleコンポーネントが、コンポーネントの依存性に応じて、確実に正しい順序で再起動します。

Oracle Restartでは、Oracle Restartと統合されているコンポーネント(SQL*Plus、リスナー制御ユーティリティ(LSNRCTL)、ASMCMDおよびOracle Data Guardなど)の状態を定期的に監視します。コンポーネントのヘルス・チェックに失敗すると、Oracle Restartはそのコンポーネントを停止し、再起動します。

Oracle Restartは、Oracle Databaseホームとは別にインストールするOracle Grid Infrastructureホームから実行され

ます。

統合されたクライアント・フェイルオーバー・アプリケーションは、Oracle Clusterwareにより管理されているロール・ベースのサービスや高速アプリケーション通知イベントに依存して、アプリケーションに障害の警告を行います。単一インスタンス・データベースで、統合されたクライアント・フェイルオーバーを実現するには、Oracle Restartが必要です。

関連項目:

Oracle Restart機能のインストールおよび構成の詳細は、『Oracle Database管理者ガイド』を参照してください。

オンライン再編成および再定義

可用性および管理性を強化する1つの方法として、データの再編成操作中、データベースへの通常のアクセス権をユーザーに与えます。

Oracle Databaseのオンライン再編成および再定義機能により、管理者は、データベースへの通常のアクセス権をユーザーに与えたまま、表の物理属性を変更したり、データおよび表構造の両方を変換したりするなど多くの柔軟性が得られます。この機能によって、データの可用性、問合せのパフォーマンス、レスポンス時間およびディスク領域の使用率が向上します。これらは、すべてミッションクリティカルな環境において重要で、アプリケーションのアップグレード・プロセスをより簡単、安全かつ高速なものにします。

Oracle Databaseのオンライン・メンテナンス機能を使用すると、アプリケーションのデータベース・オブジェクトに変更を加えるために必要なアプリケーションの停止時間が大幅に短縮(または排除)されます。

関連項目:

*Oracle Database*管理者ガイドのオンラインでの表の再定義を参照してください

Zero Data Loss Recovery Appliance

クラウド規模のZero Data Loss Recovery Appliance (通常、リカバリ・アプライアンスと呼ばれる)は、エンタープライズ内のすべてのOracleデータベースでデータ損失とバックアップ・オーバーヘッドを劇的に削減するエンジニアド・システムです。

Recovery Manager (RMAN)と統合したリカバリ・アプライアンスによって、クラウド規模の耐障害性の高いハードウェアと記憶域を使用し、多数のデータベースに対応する、集中管理された永久的増分バックアップ計画が実現します。リカバリ・アプライアンスでは、継続してバックアップのリカバリ可能性が検証されます。

リカバリ・アプライアンスは、次の理由からMAA推奨バックアップおよびリカバリ・アプライアンスです。

- リカバリ・アプライアンスから復元する場合のデータ損失の排除
- 最小限のバックアップ・オーバーヘッド
- エンドツーエンド・データ保護の可視性の向上
- クラウド規模の保護
- Oracle Sharding層を含むすべてのMAAリファレンス・アーキテクチャとの適切な統合

関連項目:

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、領域効率がよいソフトウェアのリポジトリ(より正確には「ゴールド・イメージ」)が維持されます。これは、任意の数のターゲット・マシンにプロビジョニングできる標準化されたソフトウェア・ホームです。

任意の数のホームを特定のゴールド・イメージからプロビジョニングでき、フリート・パッチ適用およびプロビジョニングによって系統情報が保持されるため、デプロイされたソフトウェアの出所が常に認識されます。ゴールド・イメージはシリーズに編成できるため、リリースの展開を追跡するグループを作成できます。様々な調整されたソリューション(特定のアプリケーションのOracle Databaseパッチ・バンドルなど)が異なるシリーズにグループ化されます。通知システムは、特定のシリーズで新しいイメージが使用可能になると関係者に通知します。フリート・パッチ適用およびプロビジョニングは、Oracle Grid Infrastructureの機能です。フリート・パッチ適用およびプロビジョニング・サーバーを構成するコンポーネントは、Oracle Grid Infrastructureによって自動的に管理されます。

フリート・パッチ適用およびプロビジョニングを使用して、データベース、クラスタウェア、ミドルウェアおよびカスタム・ソフトウェアをプロビジョニングできます。フリート・パッチ適用およびプロビジョニングには、Oracle Grid InfrastructureおよびOracle Databaseのデプロイメントを作成、構成、パッチ適用およびアップグレードする追加機能があります。これらの機能によって、メンテナンスが簡単になり、リスクと影響が軽減され、変更をバックアウトする必要がある場合にロールバックのオプションが提供されます。追加機能としては、ベース・マシンへのクラスタおよびデータベースのプロビジョニング、クラスタおよびOracle RACデータベースの拡張と縮小による簡単なオンデマンドの容量調整があります。これらの操作はすべて、1つのコマンドで実行されます。他の方法を使用した場合に必要となる多数の手動のステップが、このコマンドに置き換えられています。すべてのコマンドとその結果は、監査ログに記録されます。すべてのワークフローは、任意の環境に固有の要件をサポートするためにカスタマイズできます。

フリート・パッチ適用およびプロビジョニングの主な利点は、次のとおりです。

- 標準化を有効にして施行します
- プロビジョニング、パッチ適用およびアップグレードが簡略化されます
- メンテナンスの影響およびリスクが最小限に抑えられます
- 自動化が進み、タッチ・ポイントが減少します
- 大規模なデプロイメントがサポートされます

関連項目:

*Oracle Clusterware*管理およびデプロイメント・ガイドのフリート・パッチ適用およびプロビジョニングとメンテナンス

Oracle Fleet Patching and Provisioning(FPP)の紹介と技術的概要

エディションベースの再定義

計画的なアプリケーション変更には、データ、スキーマおよびプログラムの変更が含まれる場合があります。これらの変更は、主にパフォーマンス、管理性、および機能性を向上させる目的で行われます。例として、アプリケーションのアップグレードがあります。

エディションベースの再定義(EBR)を使用すると、アプリケーションの使用中にそのデータベース・コンポーネントをアップグレードでき、停止時間を最小化あるいは排除することができます。アプリケーションを使用中にアップグレードするには、アプリケーションのデータベース・コンポーネントを構成するデータベース・オブジェクトをコピーし、コピーしたオブジェクトを個別に再定義する必要があります。アプリケーションのユーザーは、この変更による影響を受けず、変更されていないアプリケーションを引き続き実行できます。変更が正しいことを確認できたら、アップグレードしたアプリケーションをすべてのユーザーが使用できるようにします。

関連項目:

*Oracle Database*開発ガイドのエディションベースの再定義の使用

4 計画外停止時間用Oracle Database高可用性ソリューション

Oracle Databaseは、可用性を高める高可用性ソリューションの統合スイートを提供しています。

これらのソリューションは、計画停止時間と計画外停止時間の両方を排除または最小化し、企業が24時間年中無休でビジネスの継続性を維持する場合に役立ちます。ただし、Oracleの高可用性ソリューションは停止時間の短縮のみでなく、全体的なパフォーマンス、スケーラビリティおよび管理性の改善にも役立ちます。

計画外停止時間の停止タイプとOracle高可用性ソリューション

計画外停止時間に対する各種のOracle MAA高可用性ソリューションについて、簡単にナビゲートできるマトリックスで説明します。

次の表は、参照先(ハイパーリンク)の項で説明されている機能を使用して、計画外停止時間の様々な原因に対処する方法を示しています。いくつかのOracleソリューションがリストされている場合、MAA推奨ソリューションはOracle MAAソリューション列に示しています。

表4-1 計画外停止時間の停止タイプとOracle高可用性ソリューション

停止範囲	Oracle MAAソリューション	メリット
サイトの障害	Oracle Data Guard とアプリケーションの継続的なサービスの有効化(MAA推奨)	● 統合されたクライアントとアプリケーションのフェイルオーバー ● 最も高速で簡単なデータベースのレプリケーション ● すべてのデータ型のサポート ● 伝播遅延の解消によるデータ損失ゼロ ● Oracle Active Data Guard 　● プライマリの負荷をオフロードするための読取り専用サービス、およびグローバル一時表や順序に対する DML のサポート 　● 少量の更新をプライマリにリダイレクトでき、ほとんどが読取りのレポートをスタンバイにオフロードすることが可能 ● Database In-Memory のサポート
	Oracle GoldenGate	● 柔軟な論理レプリケーション・ソリューション(ターゲットは読取り/書取りで開く) ● アクティブ-アクティブな高可用性(競合解決を含む)

停止範囲	Oracle MAAソリューション	メリット
		● 異機種プラットフォームおよび異機種データベースのサポート ● カスタムのアプリケーション・フェイルオーバーによる停止時間はゼロになる可能性

	Recovery Manager、Zero Data Loss Recovery Appliance および Oracle Secure Backup	● 完全に管理されたデータベース・リカバリおよび Oracle Secure Backup との統合
		● リカバリ・アプライアンス
		  ● エンドツーエンドのデータ保護をバックアップに提供
		  ● データベースのリストアでのデータ損失が減少
		  ● 非リアルタイム・リカバリ

インスタンスまたはコンピュータの障害	Oracle Real Application Clusters および Oracle Clusterware とアプリケーションの継続的なサービスの有効化(MAA 推奨)	● 統合されたクライアントとアプリケーションのフェイルオーバー ● 障害が発生したノードとインスタンスの自動リカバリ ● Oracle Real Application Clusters による最小限のアプリケーション低下
	Oracle RAC One Node とアプリケーションの継続的なサービスの有効化	● 統合されたクライアントとアプリケーションのフェイルオーバー ● オンライン・データベースの再配置による接続とインスタンスの別のノードへの移行 ● 従来のコールド・フェイルオーバー・ソリューションよりも高いデータベースの可用性
	Oracle Data Guard とアプリケーションの継続的なサービスの有効化	● 統合されたクライアントとアプリケーションのフェイルオーバー ● 最も高速で簡単なデータベースのレプリケーション ● すべてのデータ型のサポート ● 伝播遅延の解消によるデータ損失ゼロ ● Oracle Active Data Guard ● プライマリの負荷をオフロードするための読取り専用サービス、およびグローバル一時表や順序に対する DML のサポート

停止範囲	Oracle MAAソリューション	メリット
		● 少量の更新をプライマリにリダイレクトでき、ほとんどが読取りのレポートをスタンバイにオフロードすることが可能 ● Database In-Memory のサポート
	Oracle GoldenGate	● 柔軟な論理レプリケーション・ソリューション(ターゲットは読取り/書取りで開く) ● アクティブ-アクティブな高可用性(競合解決を含む) ● 異機種プラットフォームおよび異機種データベースのサポート ● カスタムのアプリケーション・フェイルオーバーによる停止時間はゼロになる可能性
ストレージ障害	Oracle Automatic Storage Management(MAA 推奨)	ミラー化およびオンライン自動リバランスによる、個別の障害グループへのデータの冗長コピーの配置。
	Oracle Data Guard(MAA 推奨)	● 統合されたクライアントとアプリケーションのフェイルオーバー ● 最も高速で簡単なデータベースのレプリケーション ● すべてのデータ型のサポート ● 伝播遅延の解消によるデータ損失ゼロ ● プライマリの負荷をオフロードするための Oracle Active Data Guard の読取り専用サービス、グローバル一時表および順序での DML のサポート ● Database In-Memory のサポート
	高速リカバリ領域を使用する Recovery Manager および Zero Data Loss Recovery Appliance (MAA 推奨)	完全に管理されたデータベース・リカバリとディスクおよびテープのバックアップ
	Oracle GoldenGate	● 柔軟な論理レプリケーション・ソリューション(ターゲットは読取り/書

停止範囲	Oracle MAAソリューション	メリット
		取りで開く) ● アクティブ-アクティブな高可用性(競合解決を含む) ● 異機種プラットフォームおよび異機種データベースのサポート ● カスタムのアプリケーション・フェイルオーバーによる停止時間はゼロになる可能性
データ破損	破損の予防、検出および修復(MAA 推奨) DB_BLOCK_CHECKING、DB_BLOCK_CHECKSUM および DB_LOST_WRITE_PROTECT などのデータベース初期化設定	様々なレベルのデータおよび REDO ブロック破損予防とデータベース・レベルでの検出
データ破損	Oracle Data Guard(MAA 推奨) Oracle Active Data Guard 自動ブロック修復 DB_LOST_WRITE_PROTECT 初期化パラメータ	● Oracle Active Data Guard スタンバイがある Data Guard 構成の場合 ● プライマリ・データベースで検出された物理ブロック破損は、スタンバイから取得されたブロックの良好なコピーを使用して自動的に修復されます(その逆にも修復されます) ● 修復はユーザーやアプリケーションに対して透過的であり、データ破損は明確に分離できます ● MAA の推奨初期化設定では、Oracle Active Data Guard および Oracle Exadata Database Machine で、最も包括的でフル・スタックの破損保護が実現されます。 ● DB_LOST_WRITE_PROTECT を有効にした場合 ● プライマリ・データベースで発生した書込み欠落がフィジカル・スタンバイ・データベースによって、またはプライマリ・データベースのメディア・リカバリ時に検出され、リカバリはデータベースの一貫性を保つために停止します ● Data Guard を使用してスタンバイ・データベースにフェイルオーバーすると、データの一部が失われます ● Data Guard Broker の PrimaryLostWrite プロパティでは、プライマリ・データベースで書込み欠落が検

停止範囲	Oracle MAAソリューション	メリット
		出された場合の SHUTDOWN、CONTINUE、FAILOVER および FORCEFAILOVER オプションがサポートされます。『Oracle Data Guard Broker』を参照してください ● DB_LOST_WRITE_PROTECT 初期化パラメータでは、書込み欠落が検出されます ● シャドウ消失書込み保護は、データが大きく破損する前に書込みが失われたことを検出します。Oracle Data Guard スタンバイ・データベースを必要とせずに、データベース、表領域またはデータファイルのシャドウ消失書込み保護を有効化できます。ワークロードへの影響は状況によって異なる場合があります。
	Dbverify、Analyze、データ・リカバリ・アドバイザと Recovery Manager、Zero Data Loss Recovery Appliance、および高速リカバリ領域を使用する ASM スクラブ (MAA 推奨)	これらのツールを使用すると、管理者が手動チェックを実行できるため、様々なデータ破損を検出し、可能な場合には修復できます。 ● Dbverify と Analyze では、物理的ブロック・チェックおよび論理的ブロック内チェックが実行されます。Analyze では、オブジェクト間の一貫性チェックを実行できます。 ● データ・リカバリ・アドバイザでは、データ破損が自動的に検出され、最適なリカバリ計画が推奨されます。 ● RMAN の操作では、次のことが可能です ● 物理およびブロック間の両方の論理チェックを実行します ● フラッシュバック・ログ、バックアップまたはスタンバイ・データベースを使用してオンライン・ブロックメディア・リカバリを実行して、物理ブロック破損からリカバリできます ● リカバリ・アプライアンス ● バックアップが有効であることを確認できる定期的なバックアップ検証を実行します ● リカバリ期間要件を入力し、それらの SLA がリカバリ・アプライアンスで管理される既存のバックアップで満たされない場合にアラートを受信できます ● ASM スクラブでは、標準冗長性および高冗長性のディスク・グループ内の ASM ペアを使用して、物理的および論理的データ破損が検出され、修復が試行されます。
データ破損	Oracle Exadata Database Machine およ	● Oracle ASM では、破損を検出し、良好なミラーがある場合、良好なブロックを返し、後続の書込み I/O 中にその破損を修

停止範囲	Oracle MAAソリューション	メリット
	び Oracle Automatic Storage Management(MAA 推奨) DIX + T10 DIF 拡張(該当する場合は MAA 推奨)	復。 ● Exadata が、不正または誤った記憶域 I/O を原因とするデータ破損を防止する、暗黙の HARD 対応チェックを提供。 ● Exadata では、ハード・ディスク・スクラブおよび修復が自動的に行われます。不良セクターの検出および修正。 ● DIX +T10 DIF 拡張では、ベンダーのホスト・アダプタからストレージ・デバイスに対するチェックサム検証を通じて、読取りおよび書込みのエンドツーエンドのデータ整合性が保証されます
	Oracle GoldenGate	● 柔軟な論理レプリケーション・ソリューション(ターゲットは読取り/書取りで開く)。パートナ・レプリカが破損している場合、フェイルオーバー・ターゲットとして論理レプリカを使用可能 ● アクティブ-アクティブな高可用性(競合解決を含む) ● 異機種プラットフォームおよび異機種データベースのサポート
人的エラー	Oracle セキュリティ機能(MAA 推奨)	人的エラーを防ぐためのアクセスの制限
	Oracle Flashback Technology(MAA 推奨)	● 不適切な結果のファイングレイン・エラー調査 ● ファイングレインおよびデータベース全体またはプラグイン・データベースの巻戻しとリカバリ機能
遅延または減速	Oracle Database および Oracle Enterprise Manager Oracle Data Guard (MAA 推奨)とアプリケーションの継続的なサービスの有効化	● Oracle Database では、インスタンスおよびデータベースの遅延またはクラスタの減速を自動的に監視し、ブロックしているプロセスまたはインスタンスを取り除いて、遅延の長期化や不要なノード削除を防ごうとします。 ● アプリケーションまたはレスポンス時間の減速を検出し、これらの SLA 違反に対応するように、Oracle Enterprise Manager またはカスタマイズしたアプリケーションのハートビートを構成できます。たとえば、Enterprise Manager のビーコンを構成して、アプリケーションのレスポンス時間を監視および検出できます。その後、特定のしきい値を超過したら、Enterprise Manager で Data Guard の `DBMS_DG.INITIATE_FS_FAILOVER` PL/SQL プロシージャをコールして、フェイルオーバーを開始できま

停止範囲	Oracle MAAソリューション	メリット
		す。『Oracle Data Guard Broker』の「ファスト・スタート・フェイルオーバーの管理」に関する項を参照してください。 ● Database In-Memory のサポート
ファイル・システム・データ	データベース以外のファイルに対する Oracle レプリケーション・テクノロジ	データベース以外のファイルを含むフル・スタック・フェイルオーバーの有効化

MAAリファレンス・アーキテクチャおよびマルチテナント・アーキテクチャの計画外停止の管理

MAAリファレンス・アーキテクチャの各MAAサービス・レベル層とマルチテナント・アーキテクチャでの高可用性ソリューションについて、簡単にナビゲートできるマトリックスで説明します。

DBaaSで多くのデータベースを管理している場合は、「Oracle MAAリファレンス・アーキテクチャ」の説明に従ってMAA層とOracle Multitenantを使用することをお薦めします。

次の表に、マルチテナント・アーキテクチャ内のデータベースに影響する可能性のある様々な計画外停止を示します。また、各MAAリファレンス・アーキテクチャで使用可能なOracle高可用性ソリューションで、その停止に対応するものも示します。

表4-2 MAAリファレンス・アーキテクチャおよびマルチテナント・アーキテクチャの計画外停止マトリックス

イベント	MAAアーキテクチャごとのソリューション	リカバリ期間(RTO)	データ消失(RPO)
インスタンス障害	BRONZE: Oracle Restart	インスタンスを再起動できる場合は数分	ゼロ
	SILVER: Oracle RAC (Oracle Real Application Clusters および Oracle Clusterware を参照)または Oracle RAC One Node と、アプリケーションの継続的なサービスの有効化	Oracle RAC の場合は数秒、Oracle RAC One Node の場合は数分	ゼロ
	GOLD: Oracle RAC (Oracle Real Application Clusters および Oracle	秒	ゼロ

イベント	MAAアーキテクチャごとのソリューション	リカバリ期間(RTO)	データ消失(RPO)
	Clusterware を参照)とアプリケーションの継続的なサービスの有効化		
	PLATINUM: Oracle RAC (Oracle Real Application Clusters および Oracle Clusterware を参照)とアプリケーションの継続的なサービスの有効化	アプリケーションの停止なし	ゼロ
永続的なノード障害(ただし、ストレージは使用可能)	BRONZE: リストアおよびリカバリ	数時間から 1 日	ゼロ
	SILVER: Oracle RAC (Oracle Real Application Clusters および Oracle Clusterware を参照)とアプリケーションの継続的なサービスの有効化	秒	ゼロ
	SILVER: Oracle RAC One Node とアプリケーションの継続的なサービスの有効化	分	ゼロ
	GOLD: Oracle RAC (Oracle Real Application Clusters および Oracle Clusterware を参照)とアプリケーションの継続的なサービスの有効化	秒	ゼロ
	PLATINUM: Oracle RAC (Oracle Real Application Clusters	秒	ゼロ

イベント	MAAアーキテクチャごとのソリューション	リカバリ期間(RTO)	データ消失(RPO)
	および Oracle Clusterware を参照)とアプリケーションの継続的なサービスの有効化		
ストレージ障害	すべて: Oracle Automatic Storage Management	停止時間ゼロ	ゼロ
データ破損	BRONZE/SILVER: 基本的な保護 一部の破損では、プラガブル・データベース(PDB)、マルチテナント・コンテナ・データベース(CDB)全体またはコンテナ以外のデータベース(非 CDB)のリストアおよびリカバリの回復が必要	1 時間から数日	● リカバリ不能な場合は、最後のバックアップ以降 ● リカバリ・アプライアンスではゼロまたはゼロ付近
	GOLD: Oracle Active Data Guard による包括的な破損保護および自動ブロック修復	● 自動ブロック修復によりゼロ ● 破損の原因が書込み欠落で、Data Guard ファスト・スタート・フェイルオーバーを使用している場合は、数秒から数分。	破損の原因が書込み欠落でない場合はゼロ
	PLATINUM: Oracle Active Data Guard による包括的な破損保護および自動ブロック修復 カスタムのアプリケーション・フェイルオーバーを使用する Oracle GoldenGate	● 自動ブロック修復によりゼロ ● Oracle GoldenGate のレプリカによりゼロ	Active Data Guard のファスト・スタート・フェイルオーバーおよび Oracle GoldenGate を使用する場合はゼロ

イベント	MAAアーキテクチャごとのソリューション	リカバリ期間(RTO)	データ消失(RPO)
	のレプリカ		
人的エラー	すべて: 論理障害はフラッシュバック・ドロップ、フラッシュバック表、フラッシュバック・トランザクション、フラッシュバック問合せ、プラガブル・データベースのフラッシュバックおよび UNDO により解決	検出時間による異なるが、それらのオブジェクトを使用して PDB やアプリケーションに分離。	論理障害により異なる
	すべて: RMAN ポイント・イン・タイム・リカバリ(PDB)またはプラガブル・データベースのフラッシュバックを必要とするデータベースおよび PDB 全体に影響を与える包括的な論理障害	検出時間により異なる	論理障害により異なる
データベース使用不可、システム障害、サイト障害、ストレージ障害、大規模な破損または災害	BRONZE/SILVER: リストアおよびリカバリ	数時間から数日	● 最後のデータベースおよびアーカイブのバックアップ以降 ● リカバリ・アプライアンスではゼロまたはゼロ付近
	GOLD: Active Data Guard ファスト・スタート・フェイルオーバーとアプリケーションの継続的なサービスの有効化	秒	ゼロからほぼゼロ
	PLATINUM: カスタムのアプリケーション・フェイルオーバーを使用する Oracle GoldenGate のレプリカ	ゼロ	Active Data Guard のファスト・スタート・フェイルオーバーおよび Oracle GoldenGate を使用する場合はゼロ

イベント	MAAアーキテクチャごとのソリューション	リカバリ期間(RTO)	データ消失(RPO)
パフォーマンスの低下	すべて: モニタリングと検出用の Oracle Enterprise Manager、リソース制限用のデータベース・リソース管理および進行中のパフォーマンス・チューニング	停止時間はないがサービスは低下	ゼロ

5 計画停止時間用Oracle Database高可用性ソリューション

計画停止時間は、計画外停止時間と同様、業務に悪影響を及ぼします。これは、特に、複数のタイムゾーンのユーザーをサポートする必要がある、または顧客に24時間年中無休でインターネット・アクセスを提供する必要があるグローバル企業に当てはまります。

計画停止時間中にデータベースの可用性を高める方法について学習するには、次のトピックを参照してください。

計画メンテナンス用Oracle高可用性ソリューション

Oracleでは、すべての計画メンテナンスに対して高可用性ソリューションを提供しています。

次の表に、各種のOracle高可用性ソリューション、および様々なメンテナンス・アクティビティで予想される停止時間を示します。

表5-1 計画メンテナンス用Oracle高可用性ソリューション

メンテナンス・イベント	高可用性ソリューションとターゲットの停止時間
動的およびオンライン・リソース・プロビジョニングまたは オンライン再編成および再定義	アプリケーションとデータベースの停止時間なし ● 初期化パラメータの動的な変更 ● オンラインのデータファイルの名前変更と再配置 ● 自動メモリー管理のチューニング ● オンライン再編成および再定義(表の管理と索引の管理) 『Oracle Database 管理者ガイド』、『Oracle Database リファレンス』(動的なパラメータを評価するため)、およびオンライン・データ再編成と再定義を参照してください
オペレーティング・システムのソフトウェアまたはハードウェアの更新とパッチ	データベースの停止時間なし(Oracle RAC および Oracle RAC One Node ローリングを使用) データベースの停止時間は数秒から数分(Standby-First Patch の適用の後に Data Guard スイッチオーバーを使用)
Oracle Database または Grid Infrastructure の個別パッチまたは診断パッチ	停止時間なし(データベース・オンライン・パッチ適用または停止時間なしの Oracle Grid Infrastructure パッチ適用を使用) データベースの停止時間なし(Oracle RAC および Oracle RAC One Node ローリングを使用) アプリケーションの停止時間なし(MAA ソリューションの継続的サービスのためのアプリケーション・チェックリストを使用)

メンテナンス・イベント	高可用性ソリューションとターゲットの停止時間
	データベースの停止時間は数秒から数分(Standby-First Patch の適用の後に Data Guard スイッチオーバーを使用)
クリティカル・パッチ・アップデート(CPU)プログラムの Oracle Database または Grid Infrastructure の四半期ごとの更新、または Oracle Grid Infrastructure リリースのアップグレード	データベースの停止時間なし(Oracle RAC および Oracle RAC One Node ローリングを使用) アプリケーションの停止時間なし(MAA ソリューションの継続的サービスのためのアプリケーション・チェックリストを使用) 停止時間は数秒から数分(Standby-First Patch の適用の後に Data Guard スイッチオーバーを使用) データベース OJVM を使用するアプリケーションのデータベースを四半期ごとにローリング更新する際には、特別な考慮事項が必要になります。詳細は、My Oracle Support の"Oracle JavaVM コンポーネント・データベース PSU/RU" (OJVM PSU/RU)パッチの RAC ローリング・インストール・プロセス(ドキュメント ID 2217053.1)を参照してください。
Oracle Database リリースのアップグレード (Oracle Database 11g から 12.2、12.2 から 19c など)	停止時間は数秒から数分(Data Guard 一時ロジカルまたは DBMS_ROLLING ソリューションを使用) 停止時間なし(Oracle GoldenGate を使用) 12.2 以上のデータベース・リリースの場合は Oracle Active Data Guard および DBMS_ROLLING を使用した自動データベース・アップグレード、または古いリリースの場合は Data Guard を使用したデータベースのローリング・アップグレードを参照してください。
Exadata データベース・サーバーのソフトウェア更新	データベースの停止時間なし(Oracle RAC ローリングを使用) アプリケーションの停止時間なし(MAA ソリューションの継続的サービスのためのアプリケーション・チェックリストを使用) 停止時間は数秒から数分(Standby-First Patch の適用の後に Data Guard スイッチオーバーを使用) Exadata ソフトウェアの更新を参照してください
Exadata ストレージ・サーバーまたは Exadata スイッチのソフトウェア更新	停止時間なし(Exadata `patchmgr` を使用) Exadata ソフトウェアの更新を参照してください

メンテナンス・イベント	高可用性ソリューションとターゲットの停止時間
データベース・サーバーまたは Oracle RAC クラスタの変更(ノードの追加、ノードの削除、データベース・サーバーの CPU またはメモリー・サイズの調整)	CPU の調整などの一部のハードウェアの変更は、データベース・サーバーを再起動することなくオンラインで行うことができます。ハードウェア固有のドキュメントを参照してください。 変更がオンラインでない場合 データベースの停止時間なし(Oracle RAC および Oracle RAC One Node ローリングを使用) アプリケーションの停止時間なし(MAA ソリューションの継続的サービスのためのアプリケーション・チェックリストを使用) 停止時間は数秒から数分(Standby-First Patch の適用の後に Data Guard スイッチオーバーを使用)
アプリケーション・アップグレード	停止時間なし(エディション・ベースの再定義を使用) 停止時間なし(Oracle GoldenGate を使用) エディション・ベースの再定義および Oracle GoldenGate のドキュメントを参照してください
フリート全体のソフトウェア・メンテナンス・イベント ● Oracle Database または Grid Infrastructure の個別パッチまたは診断パッチ ● クリティカル・パッチ・アップデート(CPU)プログラムの Oracle Database または Grid Infrastructure の四半期ごとの更新、または Oracle Grid Infrastructure リリースのアップグレード ● Exadata データベース・サーバーのソフトウェア更新 ● Exadata ストレージ・サーバーまたは Exadata スイッチのソフトウェア更新	次の高可用性ソリューションを活用して、フリート全体のソフトウェア・メンテナンス・イベントのターゲットの停止時間を実現する、フリート・パッチ適用およびプロビジョニングを使用します。 データベースの停止時間なし(Oracle RAC および Oracle RAC One Node ローリングを使用) アプリケーションの停止時間なし(次の場合に MAA ソリューションの継続的サービスのためのアプリケーション・チェックリストを使用) ● Oracle Database または Grid Infrastructure の個別パッチまたは診断パッチ ● クリティカル・パッチ・アップデート(CPU)プログラムの Oracle Database または Grid Infrastructure の四半期ごとの更新、または Oracle Grid Infrastructure リリースのアップグレード ● Exadata データベース・サーバーのソフトウェア更新 停止時間なし(Exadata ストレージ・サーバーまたは Exadata スイッチのソフトウェア更新の場合に Exadata `patchmgr` を使用)

移行用高可用性ソリューション

Oracle MAAでは、データベース移行による停止時間を短縮するために複数のソリューションを推奨しています。

次の表は、移行のための高可用性ソリューションの概要を示しています。

表5-2 移行用高可用性ソリューション

メンテナンス・イベント	高可用性ソリューションとターゲットの停止時間
オンプレミスの Oracle Exadata Database Machine または任意の Oracle Database クラウド・サービス (Oracle Exadata Database Service on Cloud@Customer を含む)への移行 サポートされているサービスおよびプラットフォームの詳細なリストは、ゼロ・ダウンタイム移行を参照してください	次のものを提供するゼロ・ダウンタイム移行ツールを使用します。 ● RMAN バックアップおよびリストアによる物理移行と、Oracle Data Guard を使用したオプションの低ダウンタイム・オプション。これは最も単純なターンキー移行ソリューションで、ソースとターゲットのシステム・プラットフォームが同じ(Linux から Linux など)、およびデータベース・バージョンが同じ(Oracle Database 19c から 19c)場合に最適です。 ● Oracle Data Pump による論理移行と、Oracle GoldenGate を使用したオプションの低ダウンタイム・オプションこれは、ソースとターゲットのシステム・プラットフォームが異なる(AIX から Linux など)、またはデータベースのメジャー・バージョンが異なる(Oracle Database 12c から 19c)場合にデータベースを移行する唯一のオプションです。 Oracle Autonomous Database に移行するには、次のものを提供する Oracle Cloud Infrastructure Database Migration サービス(またはゼロ・ダウンタイム移行ツール)を使用します ● Data Pump によるオフライン移行 ● Data Pump と Oracle GoldenGate によるオンライン移行
異なるサーバーまたはプラットフォームへのデータベースの移行	停止時間は数秒から数分(特定のプラットフォームの組合せの Oracle Data Guard を使用) 停止時間なし(Oracle GoldenGate を使用) Data Guard では、同じプラットフォームのプライマリとスタンバイの組合せが常にサポートされます。異機種プラットフォームについては、同一 Data Guard 構成の異機種間プライマリおよびフィジカル・スタンバイに関する Data Guard のサポート(Doc ID 413484.1)を参照してください
互換性のない文字セットへのデータベースの移行	停止時間なし(Oracle GoldenGate を使用) 文字セット移行を参照してください
別のコンテナ・データベースへのプラガブル・	停止時間は数秒から数分(プラガブル・データベースの再配置(PDB 再配置)を

メンテナンス・イベント	高可用性ソリューションとターゲットの停止時間
データベースの移行	使用) PDB の再配置を参照してください
新しい記憶域への移行	停止時間なし(Oracle Automatic Storage Management を使用(ストレージに互換性がある場合)) 特定のプラットフォームの組合せの Oracle Data Guard を使用 停止時間なし(Oracle GoldenGate を使用)
単一インスタンス・システムから Oracle RAC クラスタへのデータベースの移行	停止時間なし(Oracle RAC を使用(可能な場合))。Oracle Clusterware がインストールされたノードへの Oracle RAC の追加を参照してください 停止時間は数秒から数分(特定のプラットフォームの組合せの Oracle Data Guard を使用) 停止時間なし(Oracle GoldenGate を使用)

6 アプリケーションの継続的なサービスの有効化

基盤となるネットワーク、システムおよびデータベースが常に使用可能であれば、アプリケーションの継続的なサービスは簡単に実現します。

計画外停止や計画メンテナンス・アクティビティの際にも継続的なサービスを実現することは困難な場合があります。MAAデータベース・アーキテクチャおよびその構成と運用のベスト・プラクティスは、冗長性と、障害を許容および防止し、場合によっては自動修復する機能に基づいて構築されます。

ただし、障害の影響がデータベース・インスタンス、データベース・ノード、またはクラスタやデータ・センター全体に及ぶ場合は常に、アプリケーションの停止時間が発生する可能性があります。同様に、計画メンテナンス・アクティビティの中には、データベース・インスタンス、データベース・ノードまたはデータベース・サーバー全体の再起動が必要になるものがあります。

いずれの場合も、単純なチェックリストに従って、アプリケーションが接続されているデータベース・サービスを別のOracle RACインスタンスまたは別のデータベースに移行できれば必ず、アプリケーションの停止時間をなくすか、ほとんど発生させないようにすることが可能です。

アプリケーションの継続的なサービスを実現するための様々なレベルとオプションについては、「アプリケーションの継続的な可用性の構成」を参照してください。

計画メンテナンス・イベントのドレイン・タイムアウト

計画メンテナンス・イベントの場合、アプリケーションによっては実行中のトランザクションが完了するまでに時間がかかる場合があります。

ワークロードが実行中のトランザクションを正常に完了し、セッションを移動するまでの時間(DRAIN_TIMEOUT)は、ワークロードの特性によって異なります。短いOLTPトランザクションでは、1分のDRAIN_TIMEOUTで十分ですが、バッチ・ジョブには30分かかる場合があります。場合によっては、これらの長いトランザクションを、計画メンテナンス・ウィンドウ外の時間まで一時停止することをお薦めします。

DRAIN_TIMEOUTをより長くする構成には、計画メンテナンス・ウィンドウが拡張されるというトレードオフがあります。

次の表では、Oracle RACインスタンスのローリング再起動が発生する計画メンテナンス・イベントと、アプリケーションに影響を与える可能性がある、関連するサービスの排出タイムアウト変数について概説します。

表6-1 計画メンテナンス・イベントの排出タイムアウト変数

計画メンテナンス・イベント	アプリケーション・ドレイン・タイムアウト変数
Exadata データベース・ホスト(Dom0)のソフトウェアの変更	Exadata ホストは、10 分の最大タイムアウトでオペレーティング・システム(OS)のシャットダウンを処理します。 OS のシャットダウンによって、次の排出タイムアウト設定を持つ `rhphelper` がコールされます。 ● `DRAIN_TIMEOUT`: `drain_timeout` が定義されていないサービスに使用される値。デフォルトは 180 ● `MAX_DRAIN_TIMEOUT`: 特定のサービスに定義されている、より大きい `drain_timeout` 値をオーバーライドします。デフォルトは 300

計画メンテナンス・イベント	アプリケーション・ドレイン・タイムアウト変数
	それぞれのクラスタウェア管理サービスも、前述の値より小さくすることができる `drain_timeout` 属性によって制御されます。 関連項目: Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)
Exadata データベース・ゲスト(DomU)のソフトウェアの変更	Exadata の `patchmgr` および `dbnodeupdate` ソフトウェア・プログラムによって、次の排出タイムアウト設定を持つ `rhphelper` がコールされます。 `DRAIN_TIMEOUT`: `drain_timeout` が定義されていないサービスに使用される値。デフォルトは 180 `MAX_DRAIN_TIMEOUT`: 特定のサービスに定義されている、より大きい `drain_timeout` 値をオーバーライドします。デフォルトは 300 それぞれのクラスタウェア管理サービスも、前述の値より小さくすることができる `drain_timeout` 属性によって制御されます。 関連項目: Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)
Oracle Grid Infrastructure (GI)ソフトウェアの変更またはアップグレード	推奨されるステップは、Graceful Application Switchover in RAC with No Application Interruption (Doc ID 1593712.1)に記載されています。 例: `srvctl stop instance -o immediate -drain_timeout 600 -failover -force` それぞれのクラスタウェア管理サービスも、前述の値より小さくすることができる `drain_timeout` 属性によって制御されます。
Oracle Database ソフトウェアの変更	推奨されるステップは、Graceful Application Switchover in RAC with No Application Interruption (Doc ID 1593712.1)に記載されています。 例: `srvctl stop instance -o immediate -drain_timeout 600 -failover -force` それぞれのクラスタウェア管理サービスも、前述の値より小さくすることができる

計画メンテナンス・イベント	アプリケーション・ドレイン・タイムアウト変数
	`drain_timeout` 属性によって制御されます。

関連項目:

アプリケーションの継続的な可用性の構成

7 可用性を最大化するための運用前提条件

MAAの実装を成功させるには、次の運用ベスト・プラクティスを使用します。

可用性およびパフォーマンスのSLAの理解

高可用性およびパフォーマンスに関するサービス・レベル合意(SLA)を理解して文書化します。

- 「高可用性の概要」に示すように、高可用性の属性と、停止時間の様々な原因を理解します。
- 「高可用性要件」および「高可用性要件を文書化する方法」に示すように、部門、経営幹部および高可用性とパフォーマンスに関するサービス・レベル合意の技術チームから同意を得ます。

SLAを満たす高可用性アーキテクチャの実装および検証

高可用性およびパフォーマンスのサービス・レベル要件に関する合意事項がある場合は、次の作業を行います。

- 高可用性およびデータ保護 - 要件からのアーキテクチャの取得で説明されているOracle MAAの標準および検証済MAAリファレンス・アーキテクチャのいずれかに要件をマッピングします
- 「計画外停止時間用Oracle Database高可用性ソリューション」および「計画停止時間用Oracle Database高可用性ソリューション」の参照アーキテクチャに関連する停止および計画内メンテナンス・マトリックスを評価します
- 高可用性アーキテクチャにMAAアーキテクチャを実装するために必要なデータベース機能について学習します

テスト・プラクティスおよび環境の構築

ターゲットの高可用性SLAが満たされていることを確認するには、次のものを検証または自動化する必要があります。

- すべてのソフトウェア更新およびアップグレード・メンテナンス・イベント
- すべての修復操作(各種の計画外停止のための修復操作を含む)
- バックアップ、リストアおよびリカバリ操作

障害時リカバリとデータ保護のためにOracle Data Guardを使用する場合は、次のことをお薦めします。

- 定期的にスイッチオーバー操作を実行するか、アプリケーションおよびデータベースの完全なフェイルオーバー・テストを実行します
- アプリケーションおよびData Guardのスイッチオーバーを定期的に実行して、エンドツーエンドのロール・トランジション手順を検証します

よいテスト環境や適切なテスト・プラクティスは、本番環境で最高の安定性と可用性を達成するための重要な前提条件です。テスト環境であらゆる変更を完全に検証することで、本番システムで同じ変更を適用する前に、問題を事前に検知して防止および回避することができます。

実際の作業には次のようなものがあります。

テスト・システムおよびQA環境の構成

テスト・システムは本番MAA環境のレプリカ(MAAのGoldリファレンス・アーキテクチャを使用するなど)にする必要があります。テスト・システムが、実装する予定のMAAサービス・レベルで運用される標準のリファレンス・アーキテクチャと同一でない場合は、トレードオフがあります。本番を模倣したワークロードで、機能、パフォーマンスおよび可用性テストを実行することをお薦めします。可用性とパフォーマンスSLAが各変更後も維持されているかを評価し、本番環境に変更を適用している間になにか失敗した

場合に、明確なフォールバックまたは修復プロシージャの準備が整っていることを確認します。

適切に構成されたテスト・システムによって、多くの問題を回避できます。変更が検証されるのは同等の本番およびスタンバイ・データベース構成で、これには完全なデータ・セットが含まれ、本番を模倣するためのワークロード・フレームワーク(たとえば、Oracle Real Application Testing)が使用されるためです。

テスト・システムの廃止によってコストを削減しようとしないでください。この決断は後々、本番アプリケーションの安定性と可用性に影響を及ぼすことになります。テストおよびQAのためにシステム・リソースのサブセットのみを使用すると、次の表に示されているトレードオフがあります(MAAのGoldリファレンス・アーキテクチャの例です)。

表7-1 各種のテストおよびQA環境のトレードオフ

テスト環境	利点とトレードオフ
本番システムとスタンバイ・システムの完全レプリカ	検証: ● すべてのソフトウェアの更新およびアップグレード ● すべての機能テスト ● 本番スケールにおける完全パフォーマンス ● 完全な高可用性
本番システムの完全レプリカ	検証: ● すべてのソフトウェアの更新およびアップグレード ● すべての機能テスト ● 本番スケールにおける完全パフォーマンス ● 完全な高可用性からスタンバイ・システムをマイナスしたもの 検証できません: ● スタンバイ・データベースでの機能テスト、大規模なスケールでのパフォーマンス、高可用性および障害時リカバリ
スタンバイ・システム	検証: ● ほとんどのソフトウェア更新の変更 ● すべての機能テスト ● 完全なパフォーマンス - Data Guard スナップショット・スタンバイを使用する場合。ただし、フェイルオーバーが必要な場合はリカバリ時間が長くなる可能性があります ● ロール・トランジション ● リソース管理およびスケジューリング - スタンバイ・データベースとテスト・データベースが同じシステムに存在する場合は必要となります
共有システム・リソース	検証: ● ほとんどのソフトウェア更新の変更

テスト環境	利点とトレードオフ
	● すべての機能テスト 本番の模倣に十分なシステム・リソースを割当てできれば、この環境はパフォーマンス・テストに適している場合があります。ただし、通常は、環境に本番システム・リソースのサブセットが含まれ、パフォーマンスの検証には欠落が生じます。リソース管理およびスケジューリングが必要です。
システム・リソースの小規模バージョンまたはサブセット	検証: ● すべてのソフトウェア更新の変更 ● すべての機能テスト ● 限定的な、完全スケールの高可用性評価 検証できません: ● 本番スケールにおけるパフォーマンス・テスト
異なるハードウェアまたはプラットフォームのシステム・リソース(同一オペレーティング・システム)	検証: ● ソフトウェア更新の変更の一部 ● 限定的な、ファームウェア・パッチ適用テスト ● ハードウェアの新機能で制限されていないかぎり、すべての機能テスト ● 限定的な、本番スケールのパフォーマンス・テスト ● 限定的な、完全スケールの高可用性評価

関連項目:

『Oracle Databaseテスト・ガイド』

本番前の検証ステップの実行

ハードウェア、ソフトウェア、データベース、アプリケーションまたはすべての変更を本番前に検証およびテストすることは、安定性を維持するために重要です。上位レベルの本番前検証のステップは次のとおりです。

1. パッチまたはアップグレードのドキュメントや、対象の変更に関連するドキュメントを確認します。SLAで停止時間をゼロまたは最小限にすることが必要な場合、ローリング・アップグレードの実行の可能性を評価します。計画済停止時間を最小化または排除するローリング・アップグレードの機会を評価します。パッチまたは変更がStandby-First Patchに適格かどうか評価します。

   ノート:

   Standby-First Patchを使用すると、プライマリ・データベースは以前のソフトウェア・リリースのまま、パッチを最初にフィジカル・スタンバイ・データベースに適用できます(この適用対象は特定のソフトウェア・アップデートのタイプで、メジャー・リリース・アップグレードには適用されません。パッチ・セットおよびメジャー・リリースにはData Guardの一時ロジカル・スタ

ンバイおよびDBMS_ROLLINGの方法を使用してください)。変更作業が終了したら、スタンバイ・データベースへのスイッチオーバーを実行します。フォールバックは必要な場合にスイッチオーバーすることです。もう1つの方法として次のステップに進み、本番環境に変更を適用することもできます。詳細は、My Oracle Supportのノート1265700.1(https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1265700.1)『Oracle Patchの保証 - Data Guard Standby-First Patch適用』を参照してください。

2. テスト環境でアプリケーションを検証し、機能、パフォーマンスおよび可用性の要件を変更が満たしている、または上回っていることを確認します。手順を自動化して、フォールバック手順の文書化およびテストも確認します。これには、テストでパッチ適用の前後に取得されたメトリックを、本番システムで取得されたメトリックと比較することが必要です。Real Application Testingは、本番システムでワークロードを取得し、それをテスト・システムで再現するために使用できます。AWRとSQLパフォーマンス・アナライザは、パッチによって引き起こされたパフォーマンスの向上または低下を評価するために使用できます。

   本番環境を模倣するテスト・システムで新規ソフトウェアを検証し、機能、パフォーマンスおよび可用性の要件を変更が満たしている、または上回っていることを確認します。パッチまたはアップグレードの手順を自動化して、フォールバックを確認します。このステップを一通り実施することで、パッチまたはアップグレードの実施中および実施後における最も致命的な問題を回避できます。

3. 部門で決められているセキュリティ上の制約にも準拠しながら、アプリケーションを包括的に検証するには、Oracle Real Application Testingおよびテスト・データ管理機能を使用します。Oracle Real Application Testing(個別のデータベース・オプション)を使用すると、Oracle Databaseを現実的な環境でテストできます。Oracle Real Application Testingでは、本番環境にデプロイメントを行う前に、本番環境のワークロードを取得し、こらのワークロードに対するシステムの変更の影響を評価することで、システム変更が原因で不安定になるリスクを最小限にします。Oracle Real Application Testingの主要コンポーネントは、SQLパフォーマンス・アナライザおよびデータベース・リプレイです。テストするシステム変更の性質とその影響、およびテストを実行するシステムの種類に応じて、テストの実行にいずれかまたは両方のコンポーネントを使用できます。

   現実的な環境でテストを行う場合、テスト環境の本番ではないユーザーに機密データが公開されてしまうリスクがあります。Oracle Databaseのテスト・データの管理機能では、テスト・データに対してデータ・マスキングおよびデータ・サブセット化を実行し、このリスクを最小限にできます。

4. 適用可能な場合、すべての変更の最終的な本番前検証をData Guardスタンバイ・データベースで実行してから、それを本番に適用してください。適用可能な場合、Data Guard環境で変更を適用します。

5. 本番環境で変更を適用します。

関連項目:

Data Guard REDO適用およびStandby-First Patchの適用およびData Guard一時ロジカル・ローリング・アップグレード

『Oracle Data Guard概要および管理』のスナップショット・スタンバイ・データベースへのフィジカル・スタンバイ・データベースの変換および既存のフィジカル・スタンバイ・データベースを使用したローリング・アップグレードの実行

Oracle Databaseのローリング・アップグレード: Data Guardフィジカル・スタンバイ・データベースの使用
(http://www.oracle.com/goto/maa)

Oracle Patchの保証 - Data Guard Standby-First Patch適用(Doc ID 1265700.1)

セキュリティ・ベスト・プラクティスの設定および使用

適切なセキュリティ基準のないシステムまたはデータベースに配置された企業データは、深刻なリスクに直面している可能性があります。明確に定義されたセキュリティ・ポリシーは、システムに対する不正アクセスを遮断し、企業の機密情報を破壊行為から保護するのに役立ちます。適切なデータ保護により、セキュリティの侵害によるシステム停止の可能性を抑制できます。

関連項目:

『Oracle Databaseセキュリティ・ガイド』

変更管理手順の確立

システムの安定性を維持し、テスト・システムまたはMAAサービスレベル層の基本アーキテクチャのいずれかで厳密に評価されるまで変更がプライマリ・データベースに適用されないように、変更を管理および制御する手順を設けます。

変更をレビューし、変更管理チームからフィードバックと承認を取得します。

推奨パッチおよびソフトウェアの定期的な適用

最新の推奨パッチおよびソフトウェア・バージョンに対する定期的なテストと適用によって、最新のセキュリティおよびソフトウェアの修正がシステムで確実に行われるようにします。これは、安定性を維持し、多くの既知の問題を回避するために必要です。本番システムでアップグレードを実行する前に、テスト・システムですべての更新および変更を必ず検証してください。

さらに、Oracleヘルス・チェック・ツール

```
orachk
```

(非エンジニアド・システムおよびOracle Database Applianceをサポート)および

```
exachk
```

(Oracle Exadata Database Machine、Exalogic、Zero Data Loss Recovery ApplianceおよびBig Data Applianceなどのエンジニアド・システムをサポート)などでは、Oracleソフトウェアのアップグレードのアドバイス、重要なソフトウェア更新の推奨事項、パッチ適用とアップグレードの事前チェックが、システムおよびデータベースのヘルス・チェックやMAAの推奨事項とともに提供されます。

関連項目:

My Oracle Supportのノート756671.1 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=756671.1)『Oracle推奨パッチ -- Oracle Database』を参照してください。

My Oracle Supportのノート888828.1 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=888828.1)『Exadata Database MachineとExadata Storage Serverでサポートされるバージョン』を参照してください。

My Oracle Supportのノート1268927.2 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1268927.2)『ORAchk - Oracleスタックのヘルス・チェック』を参照してください。

My Oracle Supportのノート1070954.1 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1070954.1)『Oracle Exadata Database MachineのexachkまたはHealthCheck』を参照してください。

障害時リカバリ検証の実行

RTOやRPOなどの障害時リカバリ・サービス・レベル要件を満たしているか確認するには、障害時リカバリ検証が必要です。

スタンバイ・データベースとしてOracle GoldenGateレプリカを使用している場合でも、Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)、ZFS Storage、または別のサード・パーティのデータベース・バックアップを利用している場合でも、操作およびデータベース管理チームは、事前に設定したしきい値に従って、プライマリ・データベースの停止またはパフォーマンス低下時にいつでもデータベースとアプリケーションをフェイルオーバーまたは復元できるよう十分に準備しておくことが重要です。関係するチームは、それらを検出し、必要に応じてフェイルオーバーまたはリストアすることを決断できる必要があります。障害前に効率的に準備できると、全体的な停止時間が大幅に短縮されます。

Data GuardまたはOracle GoldenGateを高可用性、障害時リカバリ、データ保護のために使用する場合、3～6か月ごとに定期的なアプリケーションとデータベースのスイッチオーバーの操作を実行するか、完全なアプリケーションおよびデータベースのフェイルオーバー・テストを実行することをお薦めします。

バックアップを使用して障害時リカバリ・ソリューションを検証するには、定期的なRMANクロス・チェック、RMANバックアップ検証および完全なデータベースのリストアとリカバリが必要です。リカバリ・アプライアンスで固有のバックアップ・チェックと検証が自動的に実行されますが、定期的なリストアとリカバリのテストを行うこともお薦めします。

エスカレーション管理手順の確立

修復を妨げないためのエスカレーション管理手順を確立します。ほとんどの修復ソリューションは、適切に実施されていればMAAソリューションで自動的かつ透過的です。問題が発生するのは、プライマリ・データベースまたはシステムが可用性またはパフォーマンスのSLAに適合せず、フェイルオーバー手順がData Guardの一部のフェイルオーバー・シナリオのように自動で行われるのではない場合です。適切なエスカレーション・ポリシーが施行されず、決定が迅速に行われないと、停止時間が長引く可能性があります。

可用性が最優先である場合、修復およびフェイルオーバー操作を最初に実行し、アプリケーション・サービスが再構築されてから、根本原因分析(RCA)用のログおよび情報の収集を続行します。簡易なデータ収集には、トレース・ファイル・アナライザ・コレクタ(TFA)を使用します。

関連項目:

MAAのWebページ(http://www.oracle.com/goto/maa)

Oracle Supportのノート1513912.2『TFAコレクタ - 拡張診断収集のツール』(1513912.2)を参照してください。

高可用性のための監視およびサービス・リクエスト・インフラストラクチャの構成

高可用性環境を維持するには、停止時間が発生する前に、パフォーマンスおよび高可用性に関連したしきい値を検出し、それに対応する監視インフラストラクチャを構成する必要があります。

また、可能な場合は、ユーザーが操作しなくても、Oracleが障害を検出し、フィールド・エンジニアを派遣して、障害が発生したディスク、フラッシュ・カード、ファン、電源などのハードウェア・コンポーネントを置き換えることができます。

データベース・ヘルス・チェックの定期的な実行

Oracleデータベース・ヘルス・チェックは、ハードウェアおよびソフトウェアの構成とベスト・プラクティスに対するMAA準拠を評価するように設計されています。

Oracleヘルス・チェック・ツールはすべて、Oracle Grid Infrastructure、Oracle Databaseを評価し、自動MAAスコアカードを提供するか、または主要なアーキテクチャおよび構成の設定が失敗に対する耐性がない、または高速リカバリに対応していない場合にその強調部分を確認します。Exadata Database MachineなどのOracleの設計されたシステムには、数百の追加のソフトウェア、エラーおよび構成のチェックが存在します。

Oracleでは、定期的に(たとえば、Exadata Database Machineの場合毎月)最新のデータベース・ヘルス・チェックのダウンロードと実行、および主要なFAILURES、WARNINGS、およびINFOメッセージへの対処をお薦めしています。ツール

```
exachk
```

はOracle Exadata Database Machine、Exalogic、Zero Data Loss Recovery ApplianceおよびBig Data Applianceなどのエンジニアド・システムに対して使用し、

```
orachk
```

は非エンジニアド・システムおよびOracle Database Applianceに対して使用します。

さらに、計画メンテナンス・アクティビティの前と後に、ヘルス・チェックを実行することもお薦めします。

次のことを評価する必要があります。

- 計画内メンテナンス・ウィンドウより前の既存のまたは新しい重要なヘルス・チェック・アラート
- テスト後の計画内メンテナンス・ウィンドウへの新しい推奨事項の追加
- 既存のソフトウェアまたは重要なソフトウェアの推奨事項

関連項目:

My Oracle Supportのノート1268927.2 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1268927.2)『ORAchk - Oracleスタックのヘルス・チェック』を参照してください。

My Oracle Supportのノート1070954.1 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1070954.1)『Oracle Exadata Database MachineのexachkまたはHealthCheck』を参照してください。

高可用性のためのOracle Enterprise Manager監視インフラストラクチャの構成

可能性のある停止時間を回避するには、Enterprise Managerと、パフォーマンスおよび高可用性に関連したしきい値を検出し、それに対応する監視インフラストラクチャの両方を構成および使用する必要があります。

監視インフラストラクチャを使用すると、ユーザーは高可用性を監視しやすくなり、次の作業も実行できます。

- システム、ネットワーク、アプリケーション、データベースおよびストレージの統計の監視
- パフォーマンスおよびサービスの統計の監視
- システムまたはアプリケーションの問題の初期警告インジケータとなる、パフォーマンスおよび高可用性のしきい値の作成

- パフォーマンスおよび可用性アドバイスの提供
- 確立されたアラート、ツール、およびデータベース・パフォーマンス
- 設計されたシステムのハードウェア障害のアラートの受信

関連項目:

Enterprise ManagerのMAAベスト・プラクティス(http://www.oracle.com/goto/maa)

自動サービス・リクエスト・インフラストラクチャの構成

Oracle高可用性環境でのEnterprise Managerによる監視インフラストラクチャに加えて可能な場合は、ユーザーが操作しなくても、Oracleが障害を検出し、フィールド・エンジニアを派遣して、障害が発生したハードウェアを置き換えることができます。

たとえば、Oracle Automatic Service Request (ASR)は安全かつスケーラブルな、ユーザーがインストール可能なソフトウェア・ソリューションで、これは機能として使用できます。特定のハードウェア障害が発生したとき、OracleのSolarisサーバーおよびストレージ・システムの自動ケース生成を使用することで、このソフトウェアでは問題をより早く解決します。

関連項目:

My Oracle Supportのノート1185493.1 (https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1185493.1)の「Oracle自動サービス・リクエスト」を参照してください

最新のMAAベスト・プラクティスの確認

MAAソリューションには、Oracleテクノロジのすべてのスタックが含まれるため、MAAのページでOracle Fusion Middleware、Oracle Fusion Applications、Oracle Applications Unlimited、Oracle Exalytics、Oracle Exalogic、Oracle VM、Oracle Enterprise Manager Cloud ControlのMAAベスト・プラクティスを参照できます。

MAAソリューションおよびベスト・プラクティスは、http://www.oracle.com/goto/maaで継続的に開発および公開されています。

第II部 Oracle Database高可用性ベスト・プラクティス

8 Oracle Database高可用性ベスト・プラクティスの概要

Oracle DatabaseのOracle MAAベスト・プラクティスを採用することで、Oracle MAA Bronzeリファレンス・アーキテクチャのサービス・レベルを実現できます。

Bronzeアーキテクチャは、それがスタンドアロン・データベースであるか統合マルチテナント・データベースの一部であるかに関係なく、Oracle Database Enterprise Editionに含まれる高可用性機能を使用して、単一インスタンス・データベース構成に対して最高の可用性を実現します。

Bronzeアーキテクチャは、他のMAAリファレンス・アーキテクチャの基本構成です。Oracle Databaseのベスト・プラクティスは、Silver、GoldおよびPlatinum参照アーキテクチャにも実装する必要があります。ただし、そのアーキテクチャのベスト・プラクティスに特に明記されている場合は除きます。

Bronzeリファレンス・アーキテクチャのコンポーネント、サービス・レベルおよび利点、さらにBronzeベース上に構築されるMAAアーキテクチャの詳細は、高可用性リファレンス・アーキテクチャに関する対話型の図(https://www.oracle.com/webfolder/technetwork/tutorials/architecture-diagrams/high-availability-overview/high-availability-reference-architectures.html)を参照してください。

9 Oracle Database構成のベスト・プラクティス

すべてのOracle単一インスタンス・データベースを構成する場合にOracle MAAベスト・プラクティスを採用すると、停止が削減または回避され、破損のリスクが軽減され、リカバリ・パフォーマンスが向上します。

Oracle MAA Bronzeリファレンス・アーキテクチャは次のOracle Databaseのベスト・プラクティスを使用して構成し、これらはSilver (Oracle RAC)、Gold (Oracle Data Guard)およびPlatinum (Oracle GoldenGate)という他のMAAリファレンス・アーキテクチャの基本データベースの基本プラクティスでもあります。

サーバー・パラメータ・ファイル(SPFILE)の使用

サーバー・パラメータ・ファイル(SPFILE)は、データベースのすべてのインスタンスに関連付けられたすべてのデータベース初期化パラメータを保持する単一の集中管理パラメータ・ファイルです。このファイルにより、データベース・パラメータを管理するための簡単で確実な永続的環境が提供されます。SPFILEは、DATA ASMディスク・グループに配置することをお薦めします。

アーカイブ・ログ・モードと強制ロギングの有効化

データベースのARCHIVELOGモードでの実行とデータベースのFORCE LOGGINGモードの使用は、データベースのリカバリ操作のための前提条件です。

ARCHIVELOGモードでは、オンラインでのデータベース・バックアップが可能です。リストアされている時点より後の時点にデータベースをリカバリするには、このモードで運用する必要があります。Oracle Data Guardやフラッシュバック・データベースなどの機能を使用する場合、本番データベースはARCHIVELOGモードで実行する必要があります。

特定の表領域内でリカバリの必要がないデータを分離できる場合は、データベースのFORCE LOGGINGモードのかわりに、表領域レベルのFORCE LOGGING属性を使用できます。

代替ローカル・アーカイブ先の構成

ローカル・アーカイブ先(通常はLOG_ARCHIVE_DEST_1)では、代替ローカル宛先が別のASMディスク・グループに必要になります。この構成により、DB_RECOVERY_FILE_DESTがいっぱいになった場合や、なんらかの理由で使用できなくなった場合に、アーカイブ・ログ領域が不足してデータベースがハングすることを回避できます。

表9-1 代替ローカル・アーカイブ構成パラメータ

データベース・パラメータ	ローカル・アーカイブ先のLOG_ARCHIVE_DEST_nパラメータ設定
LOG_ARCHIVE_DEST_n	LOCATION=USE_DB_FILE_RECOVERY_DEST VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 データベースの DB_UNIQUE_NAME=db_unique_name ALTERNATE=ログ・アーカイブの他の宛先。log_archive_dest_[1-10]である必要があり

データベース・パラメータ	ローカル・アーカイブ先のLOG_ARCHIVE_DEST_nパラメータ設定
	ます
LOG_ARCHIVE_DEST_y	LOCATION=DB_RECOVERY_FILE_DEST に使用されるディスク・グループ以外のディスク・グループ。通常は DATA ディスク・グループです。 VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 ALTERNATE=プライマリ・ローカル・アーカイブ・ログの宛先: 通常は LOG_ARCHIVE_DEST_1 です
DB_RECOVERY_FILE_DEST	アーカイブ先(RECO ディスク・グループなど)
LOG_ARCHIVE_DEST_STATE_n	ENABLE
LOG_ARCHIVE_DEST_STATE_y	ALTERNATE

サンプル・パラメータ設定:

- LOG_ARCHIVE_DEST_1='LOCATION=USE_DB_FILE_RECOVERY_DEST VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 DB_UNIQUE_NAME=db_unique_name of the database ALTERNATE=LOG_ARCHIVE_DEST_10'
- LOG_ARCHIVE_DEST_10='LOCATION=+DATA VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1 REOPEN=5 DB_UNIQUE_NAME=db_unique_name of the database ALTERNATE=LOG_ARCHIVE_DEST_1'
- LOG_ARCHIVE_DEST_STATE_1 =enable
- LOG_ARCHIVE_DEST_STATE_10=alternate
- DB_RECOVERY_FILE_DEST=typically the RECO disk group

高速リカバリ領域の使用

高速リカバリ領域は、Oracle管理のディスク領域として、バックアップ・ファイルおよびリカバリ・ファイル用の集中管理されたディスクの場所を提供します。

高速リカバリ領域は、次のデータベース初期化パラメータを設定することで定義します。

- DB_RECOVERY_FILE_DESTには、高速リカバリ領域のデフォルトの位置を指定します。このパラメータをRECOディスク・グループに設定します。
- DB_RECOVERY_FILE_DEST_SIZE リカバリ領域の場所に作成されるデータベースのリカバリ・ファイルで使用される合計領域に対する厳密な制限(バイト)を指定します。

  このパラメータには、アーカイブ・ログ、フラッシュバック・ログおよびローカル・データベース・バックアップ・ファイルをローカルに格納するのに十分な大きさの値を設定します。ファイルをローカルに保持すると、バックアップのリストア後のリカバリ時間を短縮できます。RMANは、RMANバックアップおよびデータ保持のポリシーに従って、これらのファイルを自動的に管理します。通常、顧客は24時間分のデータを宛先に格納します

同じDB_RECOVERY_FILE_DEST_SIZEを共有する多数のデータベースをホストする場合には、領域全体を管理および監視する必要があります。たとえば、RECOディスク・グループの使用率が90%になったときに警告することをお薦めします。

フラッシュバック・データベースの有効化

フラッシュバック・データベースは、意図と異なるデータベースの変更を元に戻すための、ポイント・イン・タイム・リカバリに代わる方法です。

フラッシュバック・データベースを使用すると、データベース全体を復元して、一定期間内のデータベースの変更の影響を元に戻すことができます。その効果は、データベースのポイント・イン・タイム・リカバリを使用した場合と同様です。複雑な手順を使用することなく、単一のRMANコマンドまたはSQL*Plus文を実行してデータベースをフラッシュバックできます。

フラッシュバック・データベースを有効にするには、次に示すベスト・プラクティスを使用して、高速リカバリ領域を構成し、フラッシュバック保存目標を設定します。この保存目標では、フラッシュバック・データベースを使用してデータベースをどの程度巻き戻すかを指定します。

- フラッシュバック・データベースを有効化する前に、使用するアプリケーションのパフォーマンス・ベースラインを認識しておくと、オーバーヘッドの判断と、フラッシュバック・データベースの有効化におけるアプリケーション・ワークロードの影響の評価に役立ちます。
- フラッシュバック・データベースのフラッシュバック・ログを保持するために、高速リカバリ領域が十分あることを確認します。一般的な経験則では、フラッシュバック・ログ生成のサイズは、REDOログ生成のサイズと同程度になります。たとえば、DB_FLASHBACK_RETENTION_TARGETを24時間に設定し、1日に20GBのREDOがデータベースで生成される場合は、20から30GBのディスク領域をフラッシュバック・ログに使用できるようにします。
    - 高速リカバリ領域のサイズ指定を判断するもう1つの方法は、フラッシュバック・データベースを有効化して、データベースを短時間(2、3時間)稼働させることです。
      V$FLASHBACK_DATABASE_STAT.ESTIMATED_FLASHBACK_SIZEを問い合せて、高速リカバリ領域に必要な見積もり領域サイズを取得します。
    - DB_FLASHBACK_RETENTION_TARGETは目標であり、データベースを同じくらいフラッシュバックできるという保証はありませんので注意してください。フラッシュバック・ログが保存されている高速リカバリ領域で領域圧迫があるような場合は、最も古いフラッシュバック・ログが削除される可能性があります。フラッシュバックのポイント・イン・タイムを保証するには、保証付きリストア・ポイントを使用する必要があります。
- 高速リカバリ領域に十分なI/O帯域幅があることを確認します。FLASHBACK BUF FREE BY RVWR待機イベントが頻繁に発生すると、一般的には、フラッシュバック・データベースにおけるI/O帯域幅が不十分です。
- フラッシュバック・データベース操作の進捗状況は、V$SESSION_LONGOPSビューを問い合せることによって監視できます。進捗状況を監視するための問合せの例は次のとおりです

```
SELECT sofar, totalwork, units FROM v$session_longops WHERE opname = 'Flashback
Database';
```

- 同じポイントにフラッシュバックすることが必要な繰返しのテストについては、フラッシュバック・データベースを有効化するかわりにフラッシュバック・データベースの保証付きリストア・ポイントを使用します。これにより、領域の使用が最小限に抑えられます。
- フラッシュバックPDBは、CDB内の他のPDBに影響を与えずにプラガブル・データベースを巻き戻しできます。PDBリストア・ポイントを作成することもできます。

FAST_START_MTTR_TARGET初期化パラメータの設定

ファスト・スタート・フォルト・リカバリでは、初期化パラメータFAST_START_MTTR_TARGETによって、インスタンス障害またはシステム障害からのリカバリ時間を簡単に構成できます。

FAST_START_MTTR_TARGETパラメータは、リカバリ時間目標(RTO)の目標値、つまりインスタンスを起動してキャッシュ・リカバリの実行にかかる時間(秒単位)の目標値を指定します。このパラメータを設定すると、データベースはその目標を達成するために増分チェックポイントの書込みを管理します。このパラメータに現実的な値を選択した場合、選択したおおよその平均秒数内でデータベースがリカバリされます。

最初に、FAST_START_MTTR_TARGET初期化パラメータを300 (秒)またはリカバリ時間目標(RTO)の目標値に必要な値に設定します。この値を設定するか、さらに小さい値にすると、データベース・ライター(DBWR)がリカバリ目標を満たすようにアクティブになります。

さらに高い負荷でも処理できるように十分なIO帯域幅があることを確認します。FAST_START_MTTR_TARGETの監視およびチューニングの詳細は、『データベース・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。

ピーク負荷時のノードやインスタンスの障害などの場合の停止テストをお薦めします。

データ破損の防止

Oracle Databaseの破損の防止、検出および修復の機能は、保護対象のデータおよびトランザクションの内部知識と、包括的な高可用性ソリューションのインテリジェント統合を基に構築されています。

データ・ブロックは、認識されているOracleデータベース形式ではないか、または内容に内部的な一貫性がないと破損します。データ・ブロック破損は、内部のOracle制御情報やアプリケーションおよびユーザー・データにダメージを与えて、これがクリティカル・データおよびサービスの重大な損失につながる場合があります。

Oracle Databaseは、破損を検出すると、データをリカバリするためのブロック・メディア・リカバリおよびデータ・ファイル・メディア・リカバリを提供します。Oracle Flashbackテクノロジを使用すると、人的エラーまたはアプリケーション・エラーによって発生したデータベース全体の論理破損を元に戻すことができます。ツールは論理データ構造の予防的検証にも使用できます。たとえば、SQL*Plus ANALYZE TABLE文はブロック間の破損を検出します。

次に、データベースを破損から保護するためのベスト・プラクティスを示します。

- ディスク障害からの保護にディスクのミラー化を提供する場合は、Oracle Automatic Storage Management (Oracle ASM)を使用します。
- HIGH冗長性ディスク・タイプを使用して、Oracle ASMで破損を最適に修復します。

  ディスク・グループでOracle ASM冗長性を使用すると、I/Oエラーまたは破損の発生時にミラー・エクステントをデータベースで使用できます。継続的な保護を目的として、Oracle ASMの冗長性により、I/Oエラーの発生時にエクステントをディスク上の別の領域に移動できます。Oracle ASMによる冗長性メカニズムは、メディア・エラーを戻す不良セクターがある場合に役立ちます。
- (ほとんどの場合、人的エラーによって引き起こされた)論理破損からの高速ポイント・イン・タイム・リカバリと、フェイルオーバー後のプライマリ・データベースの高速復元には、Flashbackテクノロジを有効化します。
- Recovery Manager (RMAN)によるバックアップおよびリカバリ計画を実装し、RMANのBACKUP VALIDATE CHECK LOGICALスキャンを定期的に使用して破損を検出します。

  バックアップおよびリストア操作中の追加的なブロック・チェックにはRMANおよびOracle Secure Backupを使用しま

す。破損のチェックと修復、中央バックアップ検証、本番データベースへの影響の縮小、Enterprise Cloudバックアップおよびリカバリ・ソリューションを含め、バックアップおよびリカバリの検証にはZero Data Loss Recovery Applianceを使用します。

- データベース初期化パラメータDB_BLOCK_CHECKSUM=MEDIUMまたはFULLを設定します。
- DB_BLOCK_CHECKING=MEDIUMまたはFULLの設定値を評価しますが、これはアプリケーションでパフォーマンスを完全に評価した後にのみ行います。

LOG_BUFFER初期化パラメータを128 MB以上に設定

フラッシュバックが有効なデータベースでは、LOG_BUFFER初期化パラメータを128 MB以上に設定します。

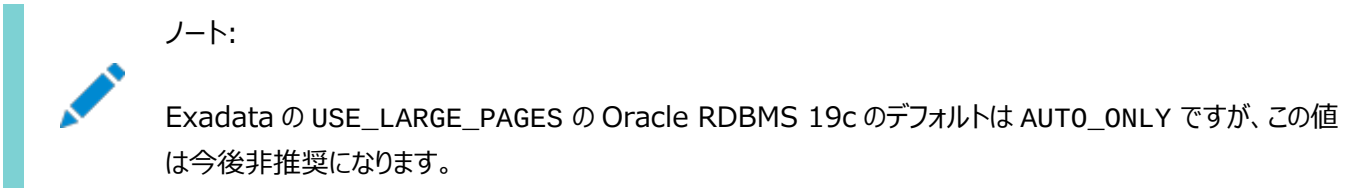
Set USE_LARGE_PAGES=ONLY

Linuxでは、データベースのSGAは、一貫したパフォーマンスと安定性を実現するために大きなページを利用する必要があります。

USE_LARGE_PAGESパラメータを使用して、これが行われるようにするには、次の2つの方法があります。

- USE_LARGE_PAGES=ONLY - インスタンスの起動前にHugepagesを事前に割り当てる必要があります。
- USE_LARGE_PAGES=AUTO_ONLY - Hugepagesはインスタンスの起動時に動的に取得されますが、メモリーが断片化されている場合、または別のインスタンスが同時にHugepagesを起動して動的に取得している場合は、この動的取得が失敗する可能性があります。

MAAのベスト・プラクティスはUSE_LARGE_PAGES=ONLYです。この推奨事項は、クラウド環境およびクラウド以外の環境に適用でき、すべてのクラウドおよびExadata自動化ツールでこの構成が適用されていることを確認します。



ノート:

Exadata の USE_LARGE_PAGES の Oracle RDBMS 19c のデフォルトは AUTO_ONLY ですが、この値は今後非推奨になります。

ビッグファイル表領域の使用

データベースが大きくなるにつれて、スモールファイル表領域にデータ・ファイルが追加されます。この表領域には、追加の管理、監視およびメンテナンスが必要ですが、Oracle Data Guard環境ではデータベースのオープン時間およびロール・トランジション時間に悪影響を及ぼします。

ビッグファイル表領域では、表領域ごとに1つの大きなデータ・ファイル(8Kブロック・サイズに最大32TB、32Kブロック・サイズに128TB)が可能です。1つのデータ・ファイルによってデータベース内のファイルの数が削減されるため、データベース・チェックポイント、データベース・オープンおよびロール・トランジション時間が改善し、管理コストも改善します。

推奨事項は次のとおりです:

- 新しいデータベースの設計およびデプロイメントでは、ビッグファイル表領域およびパーティション化を使用して、データ・ファイルの数を最小限に抑えます。大規模な表のパーティション化により、大きなビッグファイルが回避されます。妥当なビッグファイルは16TB以下です。
    - 保持ポリシーが異なる、またはアクセス要件が異なる大規模な表の場合、データベースおよびオブジェクト設計の一部としてOracleパーティション化を使用します。Oracleパーティション化は、潜在的なビッグファイル・サイズ制限に対処することもできます。

- 非常に大きい表領域の場合は、多数のスモールファイル・データ・ファイルのかわりにビッグファイル表領域を使用します。ビッグファイル表領域は、自動セグメント領域管理を備えたローカル管理表領域でのみサポートされます。
- ビッグファイル表領域の使用には、DB_BLOCK_SIZEの最大制限を理解すること以外に、負のトレードオフはありません。データベースのバックアップおよびリストア・パフォーマンスを向上させるには、ビッグファイル表領域がある場合は、RMAN SECTION SIZEパラメータを使用してバックアップおよびリストア操作をパラレル化する必要もあります。

- 多くのデータ・ファイルを含む既存のデータベースの場合は、データ・ファイルが最も多い表領域に焦点を当て、ALTER TABLE MOVEまたはオンライン再定義を使用して表またはパーティションをビッグファイル表領域に移行できるかどうかを評価します。

次の表に、データベース内のデータ・ファイルの数を9000データ・ファイルから最大100データ・ファイルまで減らし、フェイルオーバー時間を10倍に、スイッチオーバー時間を4回短縮したことを示す最新のData Guardパフォーマンス・テストを示します。

計画外停止/DR (フェイルオーバー)	**初期構成**	**チューニングされたMAA構成**
クローズしてマウント(C2M)	21 秒	1 秒
ターミナル・リカバリ(TR)	154 秒	2 秒
プライマリに変換(C2P)	114 秒	5 秒
新規プライマリのオープン(OnP)	98 秒	28 秒
PDB のオープンおよびサービスの開始(OPDB)	146 秒	16 秒
合計アプリケーション停止時間	533 秒(8 分 53 秒)	52 秒(90%低下)
計画DRスイッチ(スイッチオーバー)	**初期構成**	**チューニングされたMAA構成**
プライマリをスタンバイに変換	26 秒	21 秒
スタンバイをプライマリに変換(C2P)	47 秒	7 秒
新規プライマリのオープン(OnP)	152 秒	14 秒
PDB のオープンおよびサービスの開始(OPDB)	130 秒	39 秒
合計アプリケーション停止時間	355 秒(5 分 55 秒)	81 秒(78%低下)

多数のデータ・ファイルを含む既存のデータベースの場合、次の表では、表またはパーティションをビッグファイル表領域に移行するためのALTER TABLE MOVEまたはDBMS_REDEFINITIONの使用を比較します。

関連性またはユースケース	**DBMS_REDEFINITION**	**ALTER TABLE MOVE ONLINE**
アプリケーションへの影響	● 移動中に DDL の変更は許可されません ● アクティブ化中のアプリケーションのブラックアウト(秒)	● 移動中に DDL の変更は許可されません ● 最終スイッチ中のアプリケーション

関連性またはユースケース	DBMS_REDEFINITION	ALTER TABLE MOVE ONLINE
		のブラックアウトが不明です
アプリケーション機能がサポートされています	● DML がサポートされています ● PDML がサポートされています ● 移動中に DDL の変更は許可されません	● DML がサポートされています ● PDML はサポートされていません ● 移動中に DDL の変更は許可されません
索引の影響	● 移動中に使用可能です ● 索引は維持されます	● 移動中に使用可能です ● 移動後に索引は維持されます (UPDATE INDEXES 句を使用) ● 索引は個別に移動されます (REBUILD ONLINE)
領域要件	ダブル・スペースが必要です(表+索引)	ダブル・スペースが必要です(表+索引)
表パーティションの機能	1 回の実行でパーティション表全体を移動します	すべての索引を維持するためにパーティションを 1 つずつ移動します
統計管理	アクティブ化の前に新しい統計を作成できます	アクティブ化後に新しい統計が作成されます
進捗状況の監視,	V$ONLINE_REDEF ビューを問い合せて、表のオンライン再定義操作の進行状況を監視できます。	V$SESSION_LONGOPS?を問い合せます
失敗時の再開	再起動可能	不明
ロールバック	あり	N/A
制限	● 複数の LONG 列を保持している表は、オンラインで再定義できますが、これらの列は、CLOB に変換する必要があります。また、LONG RAW 列は、BLOB に変換する必要があります。LOB 列を持つ表は、オンライン再定義可能です。 ● 索引構成表を移動できます ● ドメイン索引を移動できます ● パラレル DML およびダイレクト・パス INSERT 操作は可能です DBMS_REDEFINITION で多数のレア・ケースの制	● LONG または RAW 列を含む表は移動できません ● パーティション化された索引構成表は移動できません。 ● 空間、XML、テキスト索引などの表でドメイン索引が定義されている場合、移動できません。 ● 表の移動中は、パラレル DML およびダイレクト・パス INSERT 操作は実行できません。 ● LOB、VARRAY、Oracle が提供する型またはユーザー定義オブ

関連性またはユースケース	DBMS_REDEFINITION	ALTER TABLE MOVE ONLINE
	限があります。『*Oracle Database* 管理者ガイド』の表のオンライン再定義の制限事項を参照してください	ジェクト型の列を含む索引構成表は移動できません。
ドキュメントとリファレンス	『*Oracle Database PL/SQL* パッケージおよびタイプ・リファレンス』の DBMS_REDEFINITION を参照してください	『*Oracle Database SQL* 言語リファレンス』の ALTER TABLE を参照してください

自動共有メモリー管理の使用およびメモリー・ページングの回避

SGA_TARGETパラメータを設定して自動共有メモリー管理を有効にし、USE_LARGE_PAGESデータベース初期化パラメータをAUTO_ONLYまたはONLYに設定し、USE_LARGE_PAGES ASM初期化パラメータをTRUEに設定します。

SGA_TARGETの設定に加えて、次のガイドラインを使用して、自動共有メモリー管理を有効にします。

- データベース・サーバー上のSGAとPGAのメモリー割当ての合計は、常にシステムの物理メモリーよりも小さくする必要がありますが、それでも同じデータベース・サーバー上で実行されているプロセス、PGAおよび他のアプリケーションに必要なメモリーを確保する必要があります。
- メモリー使用状況を正確に把握するには、V$PGASTATでオペレーティング・システム統計を問い合せて、PGAメモリーおよびホストベース・メモリーの使用状況を監視します。
- データベースおよびアプリケーションの数を調整するか、または割り当てられたメモリー設定を減らして、メモリー・ページングを回避します。

  PGA_AGGREGATE_LIMITを設定して、PGAメモリーの使用量に対して厳密な制限を指定します。PGA_AGGREGATE_LIMITの値を超過している場合は、Oracle Databaseにより、最もチューニングが困難なPGAメモリーを消費しているセッション・コールが最初に終了します。それでもPGAメモリーの使用量合計が上限を超えている場合は、次に、最も調整が困難なメモリーを使用しているセッションが終了します。

データベース初期化パラメータUSE_LARGE_PAGES=AUTO_ONLYまたはONLYを設定し、ASM初期化パラメータUSE_LARGE_PAGES=TRUEを設定します。

- init.oraパラメータUSE_LARGE_PAGES=ONLYを設定するか、ExadataシステムでAUTO_ONLYに設定して、データベース・インスタンスのSGA全体がHugePagesに格納されていることを確認します。

  データベース・インスタンスのUSE_LARGE_PAGES=ONLYを設定することをお薦めします。このパラメータを設定すると、SGAのすべてのメモリーをHugePagesから取得できる場合にのみ、インスタンスが起動するようになります。
- ASMインスタンスの場合、パラメータUSE_LARGE_PAGES=ONLYを(デフォルト値)のままにします。この設定によって、使用可能な場合に確実にHugePagesが使用されますが、さらに、HugePagesが構成されていない場合や構成が不十分な場合には、Grid Infrastructureの一部として確実にASMが起動します。
- HugePagesは、自動メモリー管理と互換性がないため、自動共有メモリー管理を使用します。

Oracle Clusterwareの使用

Oracle Clusterwareを使用すると、複数のサーバーが相互に通信することにより、1つの集合単位として機能するように見えます。Oracle Clusterwareには、単一インスタンスのOracleデータベースを含むすべてのOracleデータベース用に高可用性オプションがあります。Oracle Clusterwareは、アプリケーションの可用性を高くするための最小要件の1つです。

Oracle Clusterwareでは、Oracle Real Application Clusters (Oracle RAC)、Oracle RAC One NodeおよびOracle Restartの実行に必要なインフラストラクチャが提供されます。Oracle Grid Infrastructureは、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です。クラスタの場合、このソフトウェアにはOracle Clusterwareと Oracle ASMが含まれます。

スタンドアロン・サーバーの場合、Grid InfrastructureにはOracle RestartとOracle ASMが含まれます。Oracle Restartでは、単一インスタンスの(クラスタ化されていない) Oracleデータベース、Oracle ASMインスタンス、サービス、リスナーおよびサーバー上で実行されているその他のプロセスの起動および再起動が管理されます。ハードウェア障害またはソフトウェア障害後にサービスの割込みが発生すると、Oracle Restartは自動的にコンポーネントを再起動します。

Oracle Clusterwareは、リソースおよびリソース・グループを管理し、それらを構成する方法に基づいて可用性を高めます。リソースおよびリソース・グループは、Oracle Clusterwareで次の操作が実行されるように構成できます。

- クラスタまたはサーバーの起動時のリソースおよびリソース・グループの起動
- 障害が発生した場合のリソースおよびリソース・グループの再起動
- 各サーバーへのリソースおよびリソース・グループの再配置(他のサーバーを使用できる場合)

詳細は、*Oracle Clusterware*管理およびデプロイメント・ガイドのトピック、Oracle Database高可用性オプションおよびOracle Clusterwareを使用したアプリケーションの可用性の向上を参照してください。

10 Oracle Flashbackのベスト・プラクティス

Oracle Databaseフラッシュバック・テクノロジは、誤った操作による影響を選択的かつ効率的に取り消すことで、データベースが人的エラーを無効にできるようにする、独自の充実した一連のデータ・リカバリ・ソリューションです。

Oracle Databaseにフラッシュバックが導入される以前は、データベースの破損にかかる時間が数分でも、そのリカバリには数時間かかる場合がありました。フラッシュバックでは、エラーの修正にかかる時間は、エラーの発生にかかる時間とほぼ同じです。また、このエラーからのリカバリに必要な時間は、Oracle Database固有の機能であるデータベース・サイズに依存しません。

フラッシュバックでは、行、トランザクション、表およびデータベース全体を含むすべてのレベルでのデータベース・リカバリがサポートされています。フラッシュバックは、データを表示したり、任意の時点にデータを戻したり、進めるための増え続ける機能セットを提供し、高可用性と障害時リカバリのいくつかの重要なユースケースに対応します。機能およびユースケースのリストといくつかの主な例については、「Oracle Flashback Technology」を参照してください。

フラッシュバック機能を使用すると、データベースがオンラインである間に、履歴データを問い合せ、変更分析を実行し、セルフサービス修復を実行して、論理的な破損からリカバリできます。Oracle Flashback Technologyを使用すると、実際に、過去を元に戻すことができます。

Oracle Flashbackのパフォーマンス観測

構成および運用のベスト・プラクティスを採用し、推奨パッチを適用した後、プライマリ・データベースまたはスタンバイ・データベースでフラッシュバック・データベースが有効になっている状態で、Oracleでは次のパフォーマンス観測を確認しました。

- データベースまたはPDBを前の時間にフラッシュバックするには、ワークロードが非常に高い場合でも、通常、数秒および数分かかります。特定の量のREDOを適用するのにかかる時間の何分の1かで終了します。次に、いくつかの観測を示します。
    - 400 GBの変更で構成される大規模なバッチ・ワークロードのフラッシュバックが5分未満で完了しました。
    - 8GBの変更を含む大容量のOLTPのフラッシュバックが2分未満で完了しました。
    - 多くの変数があるため、フラッシュバックを完了するまでの時間を推定する経験則または計算はありません。これらの観測が確認されたテストは、記憶域I/O帯域幅などのシステム・ボトルネックを取り除くためにExadataで行われました。
- プライマリ・データベースへのOLTPワークロードの影響は、通常5パーセント未満です。
- 大容量の挿入(バッチ挿入など)または直接ロード操作による影響は、フラッシュバック・ブロックの新しい最適化が有効な場合、通常は5パーセント未満です。それ以外の場合は、影響は大幅に変動するため(2-40%の影響)、テストが必要です。

  「特定のアプリケーションのユースケースに対するOracle Flashbackのパフォーマンス・チューニング」で、ブロックの新しい最適化に関する説明と例外を取り上げたフラッシュバックのユースケースを参照してください。
- スタンバイ・システムが高速リカバリ領域の追加のI/Oスループット要件に対応できない場合、フラッシュバック・データベースを有効にすると、フィジカル・スタンバイ・データベースのピークREDO適用パフォーマンス率が低下する可能性があります。ただし、スタンバイでフラッシュバック・データベースが有効になっていても、フラッシュバックが有効な状態で実現可能なREDO適用率は非常に高く、アプリケーションのREDO生成率を超える可能性があります。

次のリストに、様々なOracle Databaseソフトウェア・リリースでの重要なフラッシュバック・マイルストンおよび主要なパフォーマンスの改善を示します。

Oracle Database 12cリリース2 (12.2)

- プラガブル・データベースのフラッシュバックを使用すると、他のPDBに影響を与えずに個々のPDBのフラッシュバックが可能になります。
- PDBリストア・ポイントを使用すると、簡単に別名をSCNに設定できます。その後、この別名は、フラッシュバックPDBまたはポイント・イン・タイム・リカバリに使用できます。

Oracle Database 19c

- プライマリ・データベースでリストア・ポイントを作成すると、自動的にスタンバイ・データベースに伝播され、対応するリストア・ポイントがスタンバイ・データベースに作成されます。
- Oracle Data Guard構成のプライマリ・データベースとスタンバイ・データベースの両方でフラッシュバック・データベースが有効になっている場合、プライマリ・データベースをフラッシュバックすると、スタンバイ・データベースも自動的にフラッシュバックされます。

Oracle Database 21c

- フラッシュバック・データ・アーカイブが有効な表の異なるデータベース・リリース間での移行
- データ・ファイルのサイズ変更操作のフラッシュバック・データベース・サポート
- PDBは、同じCDBインカネーションまたは祖先インカネーション内の孤立したPDBインカネーションにリカバリできます

Oracle Flashback構成のベスト・プラクティス

次に、Oracle Databaseでフラッシュバック・テクノロジを構成するためのOracle MAAのベスト・プラクティスを示します。

DB_FLASHBACK_RETENTION_TARGETの設定

DB_FLASHBACK_RETENTION_TARGET初期化パラメータを、次のいずれかの適用される条件で指定されている最大値に設定します。

- Oracle Data Guardフェイルオーバー後にフラッシュバック・データベースを利用して障害が発生したプライマリ・データベースを復元するには、DB_FLASHBACK_RETENTION_TARGETを60 (分)以上に設定して、障害が発生したプライマリの復元を有効化します。フラッシュバック・データベースを有効にする場合、復元が可能になる前に、十分なフラッシュバック・データをフラッシュバック・ログに生成するために数時間かかります。V$FLASHBACK_DATABASE_LOGを問い合せて、最も古いフラッシュバック時間を見つけることができます。
- 複数の停止が発生したケースを想定してください(ネットワークの停止後にプライマリ・データベースの停止が発生するなど)。この場合、フェイルオーバー時にプライマリ・データベースとスタンバイ・データベース間に転送ラグが生じる可能性があります。このような場合は、DB_FLASHBACK_RETENTION_TARGETを、対応可能な最大転送ラグを60 (分)にプラスした合計と等しい値に設定します。これにより、障害が発生したプライマリ・データベースを、スタンバイがプライマリになったときのSCNよりも前のSCNに確実にフラッシュバックできます。これはプライマリ復元の要件です。
- ユーザー・エラーまたは論理破損からの高速ポイント・イン・タイム・リカバリにフラッシュバック・データベースを使用中の場合、DB_FLASHBACK_RETENTION_TARGETを、リカバリ可能な過去の最も遠い時間と等しい値に設定します。
- ほとんどの場合、DB_FLASHBACK_RETENTION_TARGETは、プライマリとスタンバイで同じ値に設定する必要があります。

高速リカバリ領域のサイズ設定

フラッシュバック・データベースでは、独自のロギング・メカニズムによってフラッシュバック・ログが作成され、高速リカバリ領域(FRA)に格納されます。FRAに、ターゲット保持サイズおよびピーク・バッチ率のフラッシュバック・ログを格納するための十分な領域が割

り当てられていることを確認します。FRAのサイズ設定については、Oracleバックアップおよびリカバリのドキュメントで詳しく説明しますが、フラッシュバック・ログ生成のボリュームは、REDOログ生成の大きさとほぼ同じです。次の保守的な式とアプローチを使用します。

ターゲットFRA = 現在のFRA + DB_FLASHBACK_RETENTION_TARGET x 60 x ピークREDO率(MB/秒)

例:

- 現在のFRAまたはDB_RECOVERY_FILE_DEST_SIZE= 1000GB
- ターゲットDB_FLASHBACK_RETENTION_TARGET=360 (360分)
- AWRからの場合:
  - OLTPワークロードのピークREDO率は、データベースに対して3 MB/秒です
  - バッチ・ワークロードのピークREDO率は、データベースに対して30 MB/秒で、最長期間は4時間です
  - 6時間ウィンドウの最悪のREDO生成サイズは、(240分 x 30 MB/秒 x 60秒/分) + (120分 x 3 MB/秒 x 60秒/分) = 453,600 MB、または約443 GBです
- 提案されたFRAまたはDB_RECOVERY_FILE_DEST_SIZE= 443 GB +1000 GB = 1443 GB

FRAのサイズ設定を決定するための他の方法は、フラッシュバック・データベースを有効にして、データベース・アプリケーションを短期間(2-3時間)実行できるようにし、V$FLASHBACK_DATABASE_STAT.ESTIMATED_FLASHBACK_SIZEを問い合せることです。

DB_FLASHBACK_RETENTION_TARGETは目標であり、データベースを同じくらいフラッシュバックできるという保証はありませんので注意してください。フラッシュバック・ログが格納されているFRAで領域が不十分な場合は、最も古いフラッシュバック・ログを削除できます。FRA削除ルールの詳細は、『*Oracle Database*バックアップおよびリカバリ・ユーザーズ・ガイド』の高速リカバリ領域のメンテナンスに関する項を参照してください。保証付きリストア・ポイント(GRP)を使用して、フラッシュバックのポイント・イン・タイムを保証する必要があります。GRPが削除されるまで、必要なフラッシュバック・ログはGRPでリサイクルまたはパージされません。GRPがあるが領域が不十分な場合にデータベースが応答を停止する可能性があるため、GRPの対象期間に応じて、FRAに追加領域を割り当てる必要があります。

高速リカバリ領域の十分なI/O帯域幅の構成

OLTPワークロードの自動ワークロード・リポジトリ(AWR)レポートでFLASHBACK BUF FREE BY RVWR待機イベントが頻繁に発生し、大容量の挿入操作でFLASHBACK LOG FILE WRITE待機時間が30ミリ秒を超える場合、一般的には、フラッシュバック・データベースが有効化されていてI/O帯域幅が不十分です。

通常、フラッシュバックI/Oのサイズは1 MBです。全体的な書込みスループットは、データベースの強制ロギングが有効になっている場合にはREDO生成率と同様となり、または直接ロード操作の場合にはロード率と同一となる場合があります。簡潔化のために、1つの大きな共有ストレージGRIDを構成し、ディスクまたはLUNの外側の部分にDATAを構成して、ディスクまたはLUNの内側の部分にRECO (FRA)を構成します。これは、Exadataシステムの場合には自動的に実行されます。

LOG_BUFFERの設定

フラッシュバック・データベースのメモリー内にバッファ領域を追加するには、オペレーティング・システムの制限に応じて、初期化パラメータLOG_BUFFER=256MB以上を設定します。

Oracle Flashbackの運用ベスト・プラクティス

次に、フラッシュバック・データベースに対するOracle MAAの推奨される運用ベスト・プラクティスを示します。

- フラッシュバック・データベースを有効にする前後に、自動ワークロード・リポジトリ(AWR)またはOracle Enterprise Managerを使用してデータベース統計を収集し、フラッシュバック・データベースの有効化による影響を測定できます。

- Oracle Enterprise Managerを使用して、Enterprise Manager監視メトリック「リカバリ領域空き領域(%)」を、高速リカバリ領域(FRA)における領域の問題の予防的アラートに設定します。
- フラッシュバック・データベース操作の進捗状況は、V$SESSION_LONGOPSビューを問い合せることによって監視できます。たとえば、

  ```
  select * from v$session_longops where opname like 'Flashback%';
  ```

  フラッシュバック・データベース操作で詳細が必要な場合は、データベース・パラメータ_FLASHBACK_VERBOSE_INFO=TRUEを設定して、データベースのDIAGNOSTIC_DESTトレース・ディレクトリにフラッシュバック・データベース操作の詳細なトレースを生成します。
- フラッシュバック・データベースを使用してテスト・データベースで繰返しテストを実行する場合は、明示的にフラッシュバック・データベースを有効にせずに、保証付きリストア・ポイント(GRP)のみを使用することをお薦めします。領域の使用率およびフラッシュバック・パフォーマンスのオーバーヘッドを最小限に抑えるには、次の推奨アプローチに従います。

  ```
  Create Guaranteed Restore Point (GRP)
  Execute test
  loop
       Flashback database to GRP
       Open resetlogs
       Create new GRP
       Drop old GRP
       Execute
  testEnd loop
  ```

- 「REDO適用のトラブルシューティングおよびチューニング」で説明されているOracle Data Guard REDO適用のベスト・プラクティスに従います。

特定のアプリケーションのユースケースに対するOracle Flashbackのパフォーマンス・チューニング

OLTPワークロードのパフォーマンス・チューニング

flashback buf free by RVWR待機イベントは、フラッシュバック・データベースが有効になっている場合にのみ発生します。バッファの満杯によりディスクのフラッシュバック・ログにフラッシュバック・データを書き込むため、セッションはリカバリ・ライター(RVWR)を待機します。セッションは、RVWRによりバッファを解放できるまで待機することが必要になる場合があります。

このイベントがデータベースの上位待機イベントの1つになるとします。その場合、これは通常、高速リカバリ領域(FRA)のファイル・システムまたはストレージ・システムに、フラッシュバック書込みからの追加のI/Oに対応できる十分なI/O帯域幅がないためです。チューニングの考慮事項および対応するI/Oとストレージの統計の評価については、『*Oracle Database*バックアップおよびリカバリ・ユーザーズ・ガイド』のフラッシュバック・データベースに関する項を参照してください。

表10-1 上位5位のフォアグラウンド時間イベント

イベント	待機	回数	平均待機(ミリ秒)	%データベース時間	待機クラス
write complete waits	1,842	23,938	12995	33.68	構成
flashback buf free by RVWR	53,916	20,350	377	28.63	構成

イベント	待機	回数	平均待機(ミリ秒)	%データベース時間	待機クラス
cell single block physical read(セルの単一ブロックの物理読込み)	3,029,626	16,348	5	23.00	ユーザーI/O
buffer busy waits	6,248	5,513	882	7.76	同時実行性
DB CPU		1,757		2.47	

ダイレクト・パス操作のパフォーマンス・チューニング

AWRレポートまたはOracle Enterprise Managerで、v$sysstatにあるflashback log write bytesおよびphysical write bytesのシステム統計を確認します。

- flashback log write bytes = RVWRによってフラッシュバック・データベース・ログに書き込まれたフラッシュバック・データベース・データの合計サイズ(バイト)
- physical write bytes = データベース・アプリケーション・アクティビティ(他のインスタンス・アクティビティは含まれない)によるすべてのディスク書込みの合計サイズ(バイト)。

(flashback log write bytes) / (physical write bytes) < 5%の場合、フラッシュバックはパフォーマンスに影響しません。

それ以外の場合は、フラッシュバック・ブロックの新しい最適化機能を使用できる運用上の変更またはバグ修正を評価します(前述のパフォーマンス観測に関する項を参照)。さらに、上位の待機イベントの1つであっても、flashback log file sync待機イベントを無視します。

ブロックの新しい最適化が有効な例

この例の詳細は次のとおりです。

- flashback log write bytes = 1,223,442,432
- physical write bytes = 184,412,282,880

(flashback log write bytes) / (physical write bytes) = 0.0066 < 5%という結果は、直接ロード操作が行われるこの間隔内の物理書込みと比較して、フラッシュバック・データが何分の1かのわずかな量であることを意味します。この場合でも、flashback log file sync待機イベントは、次の表に示すように、データベース内の上位2番目の待機イベントでした。

表10-2 上位5位のフォアグラウンド時間イベント

イベント	待機	回数	平均待機(ミリ秒)	%データベース時間	待機クラス
direct path write	136,553	7,875	58	39.12	ユーザーI/O
flashback log file sync	91,566	5,887	64	29.25	ユーザーI/O
DB CPU		3,092		15.36	
log buffer space	20,545	1,737	85	8.63	構成
gc buffer busy release	1,277	487	382	2.42	クラスタ

ブロックの新しい最適化が有効でない例

この例の詳細は次のとおりです。

- flashback log write bytes= 184,438,194,176

- physical write bytes =184,405,925,888

(flashback log write bytes) / (physical write bytes) = 100% > 5%という結果は、この場合、すべての直接書込みでもフラッシュバック・ログ書込みが行われることを意味します。ここで、この場合の上位待機イベントを示します。

表10-3 上位5位のフォアグラウンド時間イベント

イベント	待機	回数	平均待機(ミリ秒)	%データベース時間	待機クラス
flashback log file sync	170,088	22,385	132	52.04	ユーザーI/O
direct path write	278,185	8,284	30	19.26	ユーザーI/O
flashback buf free by RVWR	38,396	5,048	131	11.74	構成
direct path read	220,618	4,242	19	9.86	ユーザーI/O
DB CPU		2,788		6.48	

従来型ロード操作のパフォーマンス・チューニング

次の例は、ブロックの新しい最適化を使用する場合と使用しない場合の、2つの従来型ロードを示しています。

ブロックの新しい最適化が有効でない例

次の例では、表をロードする直前に切り捨てるため、ブロックの新しい最適化を使用しません。ブロックの新しい最適化を使用しない従来型ロードの待機イベントでは、flashback log file syncにかかる合計待機時間が非常に長時間であることがわかります。これは、ブロックのイメージをバッファ・キャッシュに読み込み、ブロックをフラッシュバック・ログに書き込むために時間が必要であるためです。

表10-4 上位5位のフォアグラウンド時間イベント

イベント	待機	回数	平均待機(ミリ秒)	%データベース時間	待機クラス
flashback log file sync	235,187	13,728	58	30.82	ユーザーI/O
direct path write	558,037	10,818	19	24.29	ユーザーI/O
direct path read	459,076	8,419	18	18.90	ユーザーI/O
DB CPU		6,171		13.85	
flashback buf free by RVWR	79,463	4,268	54	9.58	構成

次のインスタンス統計では、ブロックの新しい最適化を追跡する統計において増加はほとんど見られません。

統計	合計	1秒当たり	1トランザクション当たり

統計	合計	1秒当たり	1トランザクション当たり
flashback cache read optimizations for block new	62	0.06	1.13
flashback direct read optimizations for block new	8	0.01	0.15
flashback log write bytes	177,533,280,256	177,245,433.67	3,227,877,822.84
flashback log writes	18,917	18.89	343.95

flashback cache read optimizations for block newがflashback log writesよりはるかに小さい場合、ブロックの新しい最適化は効果がありません。

前述のロード操作に最適なチューニングの推奨事項は、I/O帯域幅を増やすか、またはブロックの新しい最適化を活用できるようにロードの実行方法を変更することです。フラッシュバック保存目標の対象外となるまで待機するか、オブジェクトが削除された場合はごみ箱から削除することもできます。

ブロックの新しい最適化が影響しない例

ブロックの新しい最適化を使用する従来型ロードの待機イベントでは、次に示すように、他のデータベース待機と比較して、flashback log file syncにかかる合計時間が比較的少ないことがわかります。

表10-5 上位5位のフォアグラウンド時間イベント

イベント	待機	回数	平均待機(ミリ秒)	%データベース時間	待機クラス
direct path write	284,115	8,977	32	34.20	ユーザーI/O
DB CPU		6,284		23.94	
log buffer space	128,879	5,081	39	19.36	構成
flashback log file sync	139,546	3,178	23	12.11	ユーザーI/O
ラッチ: REDO 割当て	95,887	1,511	16	5.76	その他

インスタンス統計を見ると、ブロックの新しい操作を追跡する統計がロード中に大幅に増加したことがわかります。

統計	合計	1秒当たり	1トランザクション当たり
flashback cache read optimizations for block new	329	0.53	9.68
flashback direct read optimizations for block new	698,410	1,116.43	20,541.47
flashback log write bytes	1,197,514,752	1,914,271.66	35,221,022.12
flashback log writes	18,951	30.29	557.38

第III部 Oracle RACおよびClusterwareのベスト・プラクティス

- Oracle RACおよびClusterwareのベスト・プラクティスの概要

11 Oracle RACおよびClusterwareのベスト・プラクティスの概要

Oracle ClusterwareおよびOracle Real Application Clusters (RAC)は、クラスタ環境におけるOracleの戦略的な高可用性およびリソース管理データベース・フレームワークであり、Oracle MAA Silverリファレンス・アーキテクチャの不可欠な部分です。

Bronze MAAリファレンス・アーキテクチャにOracle RACを追加すると、Silver MAAリファレンス・アーキテクチャに昇格します。Silver MAAリファレンス・アーキテクチャは、リカバリ不能なデータベース・インスタンスまたはサーバー障害があった場合に、コールド再起動を待機できないデータベースまたはバックアップからのリストアを待機できないデータベースのために設計されています。

Silverリファレンス・アーキテクチャでは、ノードまたはインスタンスの障害およびほとんどのデータベースやシステムのソフトウェア更新について、停止時間ゼロとなる可能性があります。これは、Bronzeアーキテクチャでは実現できません。Silver MAAリファレンス・アーキテクチャについてさらに学習するには、「高可用性リファレンス・アーキテクチャ」を参照してください。

Oracle ClusterwareおよびOracle RACには、次の利点があります。

- 高可用性フレームワークおよびクラスタ管理ソリューション
  - 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を管理します
  - Oracleデータベース・リソースおよびサード・パーティ・エージェントなどの非Oracleデータベース・リソース用のHAフレームワークを提供します
- スケーラビリティと可用性を実現するアクティブ-アクティブ・クラスタリング
  - 高可用性 サーバーまたはデータベース・インスタンスが停止しても、機能しているインスタンスへの接続に影響はなく、停止したインスタンスへの接続は、Oracle RACクラスタ内の他のサーバーですでに実行され、オープンである機能しているインスタンスへすぐにフェイルオーバーされます
  - スケーラビリティおよびパフォーマンス Oracle RACは、容量の動的な追加や、複数のサーバーをまたぐ優先度の変更を必要とする大規模なアプリケーションや統合環境に最適です。個々のデータベースのインスタンスは、クラスタの1つ以上のノードで実行されます。同様に、データベース・サービスも、1つ以上のデータベース・インスタンスで使用可能です。追加のノード、データベース・インスタンスおよびデータベース・サービスを、オンラインでプロビジョニングできます。クラスタ間でワークロードを簡単に分散できるため、Oracle RACは、多数のデータベースを連結する際、Oracle Multitenantに最適なコンポーネントです。

次の表に、様々なOracle ClusterwareおよびReal Application Cluster構成のベスト・プラクティスを示します。

表11-1 Oracle RAC HAのユースケースおよびベスト・プラクティス

ユースケース	ベスト・プラクティス
認定済および検証済のクラスタウェア・ソフトウェア・スタック	Oracle Clusterware を使用して、サード・パーティ・クラスタウェアを回避します。 Oracle Database 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 クラスタウェアは、すべての Oracle Exadata システムに組み込まれています。
認定済および検証済のストレージ・アーキテクチャ	次の MAA の利点には、サード・パーティのボリューム・マネージャおよびクラスタ・ファイル・システムではなく、Oracle Automatic Storage Management (Oracle ASM)および Oracle

ユースケース	ベスト・プラクティス
	ASM クラスタ・ファイル・システム(Oracle ACFS)を使用します。 ● すべてのディスクにわたって作業を分散させることによる、ホット・スポットの排除 ● ディスクおよびストレージをオンラインで追加および削除することによる、ストレージ容量のスケーリングおよび調整 ● データベース・ストレージを管理する簡略化された均一な方法(ASMCMD、ASMCA、ExaCLI または oeadacli)を提供することによる、複雑さの軽減 ● ASM ディスク・グループ使用時の固有のデータ破損の検出および修復 ● 追加のドライバを必要としない統合された Oracle Grid Infrastructure (Clusterware +ASM)による、簡単な管理、パッチ適用およびメンテナンス 外部冗長性で ASM を使用する場合は、基礎となるストレージおよびネットワークが単一点障害なしで可用性が高いことを確認してください。 ASM ネイティブ冗長性を使用する場合、計画外停止やストレージ・ソフトウェアの更新時に最大限の保護を提供するために、高冗長性ディスク・グループの使用をお薦めします。デフォルトでは、Exadata デプロイメントでは、すべてのディスク・グループに対して高冗長性が使用されます(データとリカバリの宛先の両方)。 Oracle Automatic Storage Management Cluster File System (Oracle ACFS)は、マルチプラットフォームのスケーラブルなファイル・システムであり、Oracle Automatic Storage Management (Oracle ASM)の機能を拡張してすべてのカスタマ・ファイルをサポートし、非データベース・ファイルに活用できるストレージ管理テクノロジです。 これらのベスト・プラクティスは、すべての Oracle Exadata システムに組み込まれています。 Oracle Automatic Storage Management の概要を参照してください
認定済および検証済のネットワーク・アーキテクチャ	データベースおよびストレージ・ネットワーク・トポロジ全体に、単一点障害のない複数のネットワーク・パスがあることを確認します。 データベース・サービスに接続する場合は、組込みの仮想インターネット・プロトコル(VIP)アドレス、単一クライアント・アクセス名(SCAN)および結合されたクライアント・ネットワーク上に構成された複数のローカル SCAN リスナーを使用します。 バックアップまたは Data Guard トラフィックには、別個の高帯域幅の結合されたネットワークを使用します。 クラスタのインターコネクトとして使用されるプライベート・ネットワークの場合、Exadata 以外の顧客は、結合されたネットワークを使用するかわりに、ネットワーク冗長性のために Oracle HAIP を使用することをお薦めします。結合構成には、異なるネットワーク・カードおよびスイッチ設定で動作が異なる様々な属性があります。結合設定が正しく構成および検証されているため、この推奨事項は、Exadata 環境内のプライベート・クラスタのインターコネクトには適用されません。さら

ユースケース	ベスト・プラクティス
	に、Exadata では、可用性の高い結合されたネットワークを介して CLUSTER_INTERCONNECT パラメータを使用します。汎用システムでは、CLUSTER_INTERCONNECT と結合を使用せず、Oracle HAIP を使用してください。
クラスタ構成チェック	クラスタ検証ユーティリティ(CVU)を月次間隔で使用して、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 などの様々なクラスタおよび Oracle RAC コンポーネントを検証します。クラスタ検証ユーティリティのリファレンスを参照してください 包括的でプロアクティブなヘルス・チェックを実行し、Oracle RAC または Exadata のベスト・プラクティスに従っているかどうかを評価するには、ソフトウェア更新の前後に月次間隔で、Exadata RAC システムの場合には exachk を、または Exadata 以外の RAC システムには orachk を使用します。 ORAchk - Health Checks for the Oracle Stack (Doc ID 1268927.2)および Oracle Exadata Database Machine exachk or HealthCheck (Doc ID 1070954.1)を参照してください。 exachk と orachk の両方に CVU チェックが含まれていることに注意してください。exachk では、ストレージ、ネットワーク、クラスタウェア、ASM およびデータベースについて、ソフトウェアと構成のベスト・プラクティスおよびクリティカル・アラートが対象となります。 構成に関する推奨事項を CVU、exachk または orachk から組み込みます。
データベース・ノードまたはインスタンスの障害による停止時間を短縮する	通常、ほとんどのユースケースではデフォルト設定で十分です。ノード検出およびインスタンス・リカバリを迅速化する必要がある場合は、FAST_START_MTTR_TARGET のより小さい値を評価します FAST_START_MTTR_TARGET を減らすと、データベース・ライターのアクティビティが大幅に増加するため、I/O 帯域幅を追加する必要があります。 Exadata システムの場合、即時障害検出でリモート直接メモリー・アクセス(RDMA)を使用し、ほとんどの Oracle RAC クラスタで一般的に 30 秒かかる検出と比較して、サーバーの障害を 2 秒未満で迅速に確認できます。
ソフトウェア更新による停止時間を排除する	停止時間を回避するには、クラスタウェアまたはデータベース・ソフトウェアの更新(リリース更新など)に Oracle RAC ローリング更新を使用します。 可能な場合にはホーム外ソフトウェア更新を使用するため、ロールバックおよびフォールバックのユースケースが簡略化されます。 ソフトウェア・ゴールド・イメージを使用して、データベース opatch ユーティリティを実行する複雑

ユースケース	ベスト・プラクティス
	さを排除します。 単一の Oracle RAC クラスタまたは複数のクラスタ上のデータベースのフリートの場合は、Oracle Fleet Patching and Provisioning を使用します
アプリケーションおよびプロセスをクラスタで高可用性にする	クラスタ内でアプリケーション、プロセスまたはサーバーに障害が発生した場合、中断をできるだけ短くしてユーザーに透過的にする必要があります。たとえば、サーバーでアプリケーションに障害が発生した場合、そのアプリケーションをクラスタ内の別のサーバーで再起動して、そのアプリケーションの使用に対する影響を最小限にするか、またはなくしてしまうことができることを意味します。同様に、クラスタ内のサーバーに障害が発生した場合、引き続きユーザーにサービスを提供するために、そのサーバーで実行されているすべてのアプリケーションおよびプロセスを別のサーバーにフェイルオーバーする必要があります。組込みの `generic_application` リソース・タイプを使用すると、Oracle Clusterware はこれらのすべてのエンティティを管理して、高可用性、リソース・タイプ、カスタマイズ可能なスクリプト、アプリケーション・エージェント・プログラム、およびアプリケーションやプロセスに割り当てるリソース属性を確保できます。 Oracle Clusterware を使用して、クラスタに存在するサード・パーティのリソースおよびエージェントを管理します。 Oracle Clusterware を使用したアプリケーションの可用性の向上に関する項を参照してください
計画停止および計画外停止によるアプリケーションの停止時間を短縮する	クラスタウェア管理サービスおよびアプリケーションのベスト・プラクティスを活用して、アプリケーションの停止時間をなくします。 PDB のサービスを管理するには、SRVCTL を使用します。アプリケーションの接続にデフォルトのサービスを使用しないでください。高可用性のために、少なくとも 1 つの優先 Oracle RAC インスタンスおよび少なくとも 1 つの使用可能な追加の Oracle RAC インスタンスが必要です。 アプリケーションは HA 高速アプリケーション通知(FAN)をサブスクライブし、必要に応じて応答およびフェイルオーバーするように構成する必要があります。 「アプリケーションの継続的なサービスの有効化」および継続的な可用性 - MAA ソリューションの継続的なサービスのためのアプリケーション・チェックリストを参照してください
容量計画	アプリケーション・パフォーマンス要件を満たす十分なシステム・リソースがあることを確認するために、容量計画およびサイズ設定はデプロイメント前、およびその後は定期的に実行する必要があります。 容量計画では、データベースの増加または統合、追加のアプリケーション・ワークロード、追加のプロセス、または既存のシステム・リソースを損なうあらゆるものに対応する必要があります。

ユースケース	ベスト・プラクティス
	計画外停止または計画メンテナンス・イベント中にパフォーマンス要件がまだ満たされているかどうかを評価することも非常に重要です。

第IV部 Oracle Data Guardのベスト・プラクティス

12 Oracle Data GuardのMAAベスト・プラクティスの概要

Oracle Active Data Guardを使用してフィジカル・スタンバイ・データベースを追加すると、Silver MAAリファレンス・アーキテクチャがGold MAAリファレンス・アーキテクチャに昇格します。Oracle Data Guardのベスト・プラクティスを実装して、すべての計画外停止に対して停止時間を最小限に抑え、データ損失ゼロを実現します。

Oracle Active Data Guardは、Gold MAAリファレンス・アーキテクチャで想定される高可用性および包括的なデータ保護を実現する上で重要な役割を果たします。Goldリファレンス・アーキテクチャは、Oracle RACプライマリ・データベースとOracle Active Data Guard搭載のOracle RACスタンバイ・システム、さらにMAA構成およびライフ・サイクル操作で構成され、1つ以上のスタンバイ・データベースを作成、メンテナンス、管理および監視する包括的なサービス・セットを提供します。Oracle Active Data Guardは、ソフトウェア更新やメジャー・データベース・アップグレードなどあらゆるタイプの計画メンテナンス・アクティビティと、データベース障害、サイト停止、自然災害、データ破損などの計画外停止の際に、データを保護します。

Oracle Data Guardのベスト・プラクティスの目的は、テストおよび実証済のMAAベスト・プラクティスを実装して、Data Guardのデプロイメントを正常に完了し、安定させることです。次のステップでは、このタイプのアーキテクチャを計画、実装およびメンテナンスするためのOracle MAAベスト・プラクティスを取り上げます。

1. Oracle Data Guardアーキテクチャを計画し、アプリケーションやネットワークなど様々な事項を考慮します。
2. Oracle MAAベスト・プラクティスを使用して、Data Guardを構成およびデプロイします。
3. Data Guardデプロイメントをチューニングおよびトラブルシューティングします。

Gold MAAリファレンス・アーキテクチャについてさらに学習するには、高可用性リファレンス・アーキテクチャを参照してください。

13 Oracle Data Guardデプロイメントの計画

ITシステムおよびビジネス・プロセスの技術面と運用面の両方を含む特定の要件を分析し、Oracle Data Guardアーキテクチャ・オプションの可用性の影響を理解し、アプリケーションおよびネットワークの影響を考慮します。

Oracle Data Guardアーキテクチャ

Gold MAAリファレンス・アーキテクチャには4つのアーキテクチャ・パターンが用意されており、Oracle Active Data Guardを使用して単一点障害をなくすことができます。パターンは、ファスト・スタート・フェイルオーバーおよびHAオブザーバを使用する単一のリモート・アクティブ・スタンバイから、遠隔同期インスタンス、複数のスタンバイ、リーダー・ファームなどまで様々です。

Gold MAAリファレンス・アーキテクチャを計画する場合、高可用性リファレンス・アーキテクチャで各Goldアーキテクチャ・パターンの概要を参照し、要件に基づいて組み込む要素を選択します。

Oracle Data Guardデプロイメントのアプリケーションに関する考慮事項

Oracle Data Guardデプロイメントの計画の一環として、フェイルオーバーのシナリオで必要なリソースとアプリケーション可用性要件を考慮します。

サイト全体のフェイルオーバーかシームレス接続フェイルオーバーかの決定

最初のステップは、障害のためにプライマリ・データベースまたはプライマリ・サイトがアクセス不可か失われた場合に、ビジネス要件およびアプリケーション要件に最適なフェイルオーバー・オプションを評価することです。

次の表では、停止タイプごとの様々な条件を説明し、各シナリオでお薦めするフェイルオーバー・オプションを示します。

表13-1 様々な停止シナリオで推奨されるフェイルオーバー・オプション

停止タイプ	条件	推奨されるフェイルオーバー・オプション
プライマリ・サイト障害(すべてのアプリケーション・サーバーを含む)	障害が発生したプライマリ・データベースに接続されていた既存のすべてのアプリケーション・サーバー(または中間層サーバー)がプライマリ・サイトに含まれています。	サイト全体のフェイルオーバーが必要です
プライマリ・サイト障害(一部のアプリケーション・サーバーは引き続き稼働)	一部またはすべてのアプリケーション・サーバーに影響がなく、引き続き稼働しているアプリケーション・サーバーがセカンダリ障害時リカバリ・サイトの新しいプライマリ・データベースに再接続できます。 アプリケーション・サーバーとセカンダリ障害時リカバリ・サイトの新しいプライマリ・データベースとの間のネットワーク待機時間が異なっても、アプリケーションのパフォーマンスとスループットを引き続き許容できます。 通常、分析アプリケーションまたはレポート・アプリケーションでは、パフォーマンスへの顕著な影響なくクライ	停止時間を最小限に抑え、アプリケーションおよびデータベースの自動フェイルオーバーを可能にするために、シームレスな接続フェイルオーバーをお薦めします。

停止タイプ	条件	推奨されるフェイルオーバー・オプション
	アントとデータベースの間のより長いネットワーク待機時間を許容できますが、OLTP アプリケーションはアプリケーション・サーバーとデータベースの間のネットワーク待機時間が長くなると、パフォーマンスが大幅に低下する可能性があります。	
プライマリ・データベースまたはプライマリ・サーバー全体の障害	アプリケーション・サーバーは影響を受けず、ユーザーはセカンダリ障害時リカバリ・サイトの新しいプライマリ・データベースに再接続できます。 アプリケーション・サーバーとセカンダリ障害時リカバリ・サイトの新しいプライマリ・データベースとの間のネットワーク待機時間が異なっても、アプリケーションのパフォーマンスとスループットを引き続き許容できます。 通常、分析アプリケーションまたはレポート・アプリケーションでは、パフォーマンスへの顕著な影響なくクライアントとデータベースの間のより長いネットワーク待機時間を許容できますが、OLTP アプリケーションはアプリケーション・サーバーとデータベースの間のネットワーク待機時間が長くなると、パフォーマンスが大幅に低下する可能性があります。	パフォーマンスを許容できる場合は、停止時間を最小限に抑え、アプリケーションおよびデータベースの自動フェイルオーバーを可能にするために、シームレスな接続フェイルオーバーをお薦めします。 それ以外の場合は、サイト全体のフェイルオーバーが必要です。

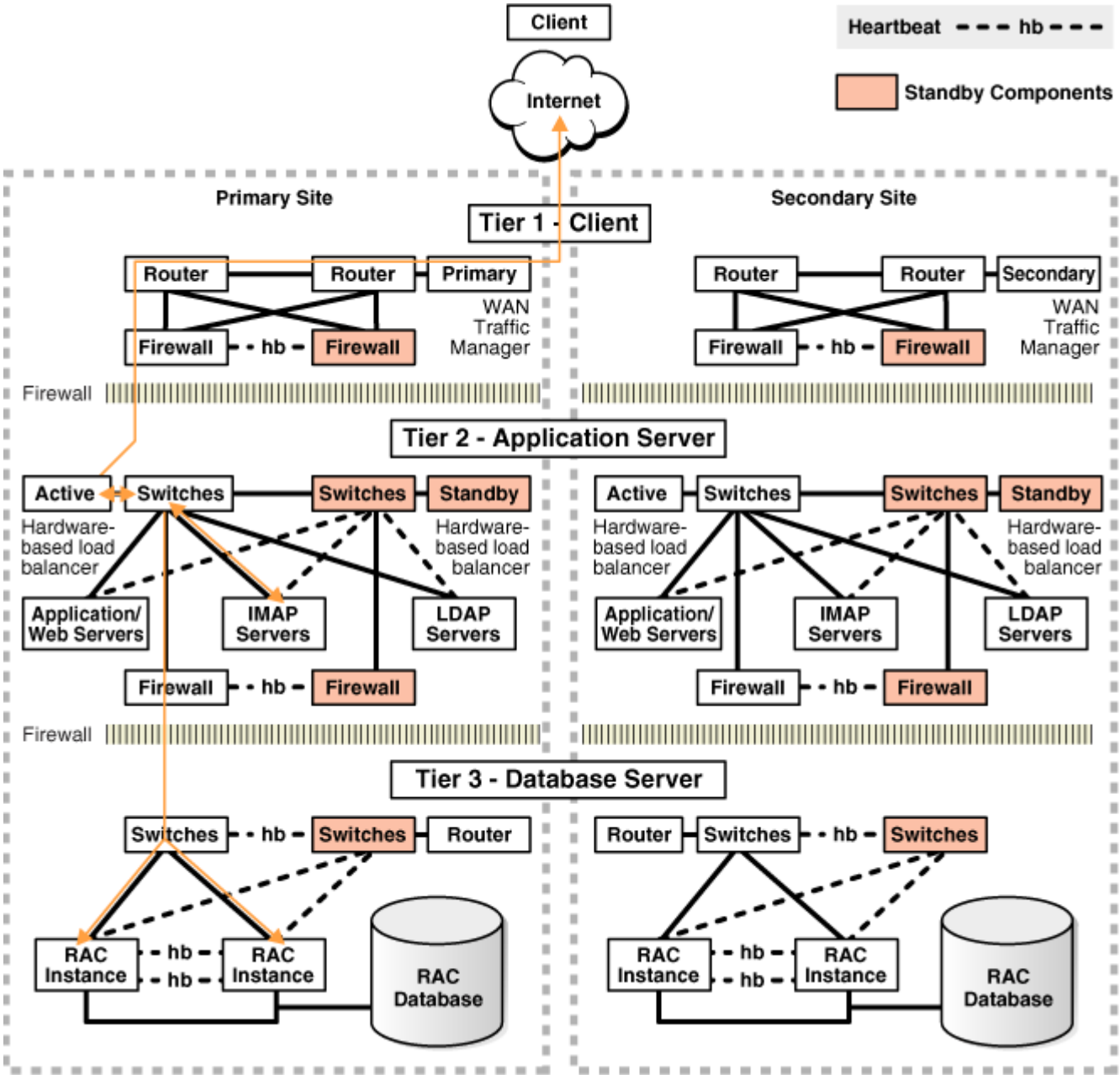
サイト全体のフェイルオーバーのベスト・プラクティス

サイト全体のフェイルオーバーとは、サイト全体が新しい一連のアプリケーション層と新しいプライマリ・データベースを使用する別のサイトにフェイルオーバーすることです。

サイト全体で障害が発生すると、アプリケーション層とデータベース層の両方が使用できなくなります。可用性を維持するために、本番データベースの冗長アプリケーション層と同期コピーをホスティングするセカンダリ・サイトに、アプリケーション・ユーザーをリダイレクトする必要があります。

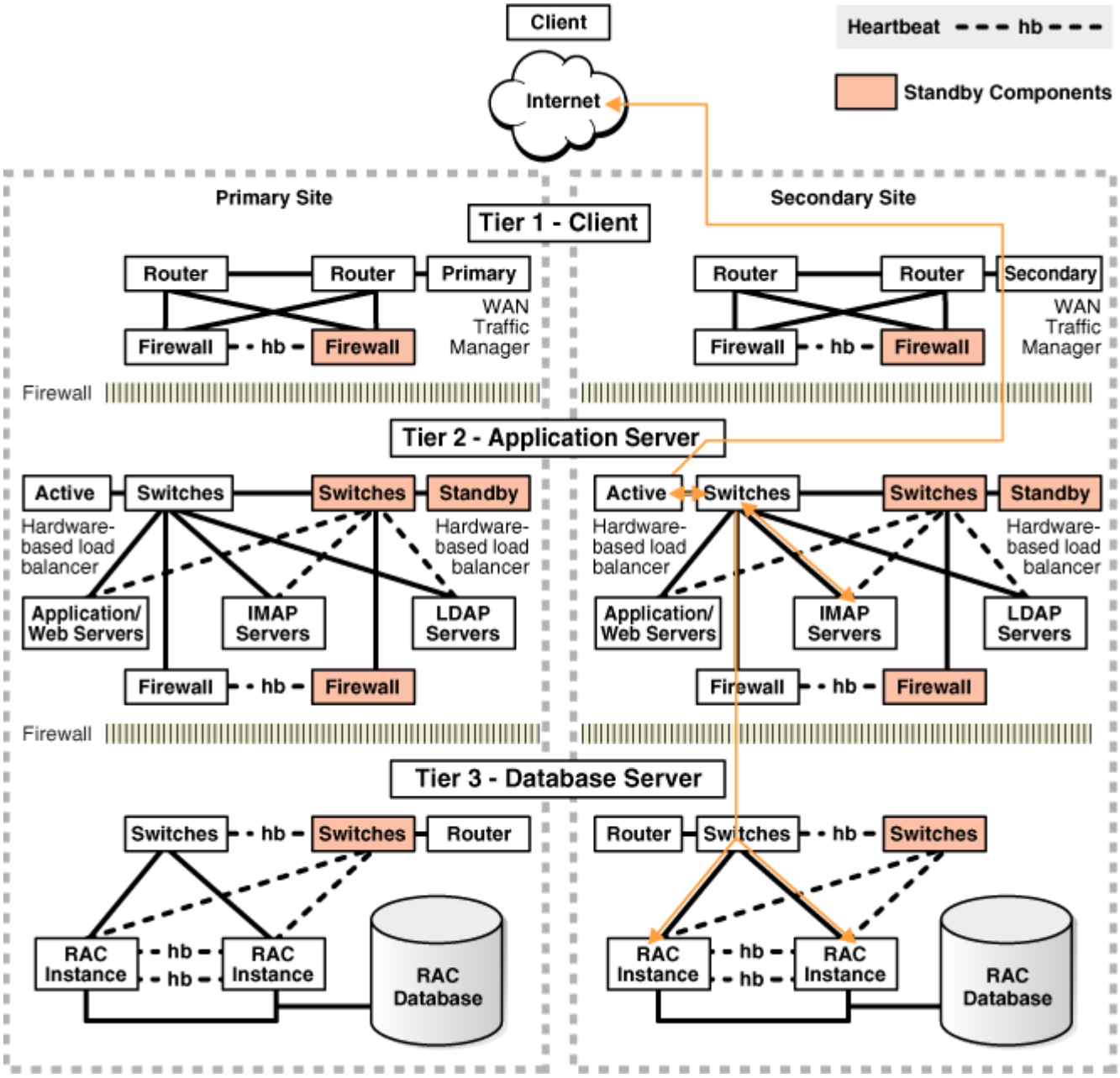
次の2つの図について考えてみます。最初の図は、フェイルオーバー前のネットワーク・ルートを示しています。クライアント・リクエストまたはアプリケーション・リクエストは、クライアント層のプライマリ・サイトに入り、プライマリ・サイトのアプリケーション・サーバー層およびデータベース・サーバー層にルーティングされます。

図13-1 サイト・フェイルオーバー前のネットワーク・ルート



次の2つ目の図は、サイト・フェイルオーバー完了後のネットワーク・ルートを示しています。クライアント・リクエストまたはアプリケーション・リクエストは、セカンダリ・サイトのクライアント層に入り、プライマリ・サイトと同じ経路をたどります。

図13-2 サイト・フェイルオーバー後のネットワーク・ルート



MAAベスト・プラクティスは、起動時間の発生を回避するためにスタンバイ・サイトで実行中のアプリケーション層をメンテナンスすること、およびOracle Data Guardを使用して本番データベースの同期コピーをメンテナンスすることです。サイト障害が発生すると、WANトラフィック・マネージャを使用してDNSフェイルオーバーが(手動または自動で)実行されて、すべてのユーザーがスタンバイ・サイトのアプリケーション層にリダイレクトされます。その間に、Data Guardフェイルオーバーによってスタンバイ・データベースがプライマリ本番ロールに遷移します。

Oracle Active Data Guardファスト・スタート・フェイルオーバーを使用して、データベース・フェイルオーバーを自動化します。アプリケーション・サーバーおよび非データベース・フェイルオーバーは、Oracle Site Guardを使用して自動化および調整できます。Oracle Site Guardは、セカンダリ・サイトでアプリケーション・サーバーを起動したり、Data Guardが自動的にフェイルオーバーする際に非データベース・メタデータを再同期するなどの操作を編成および自動化します。

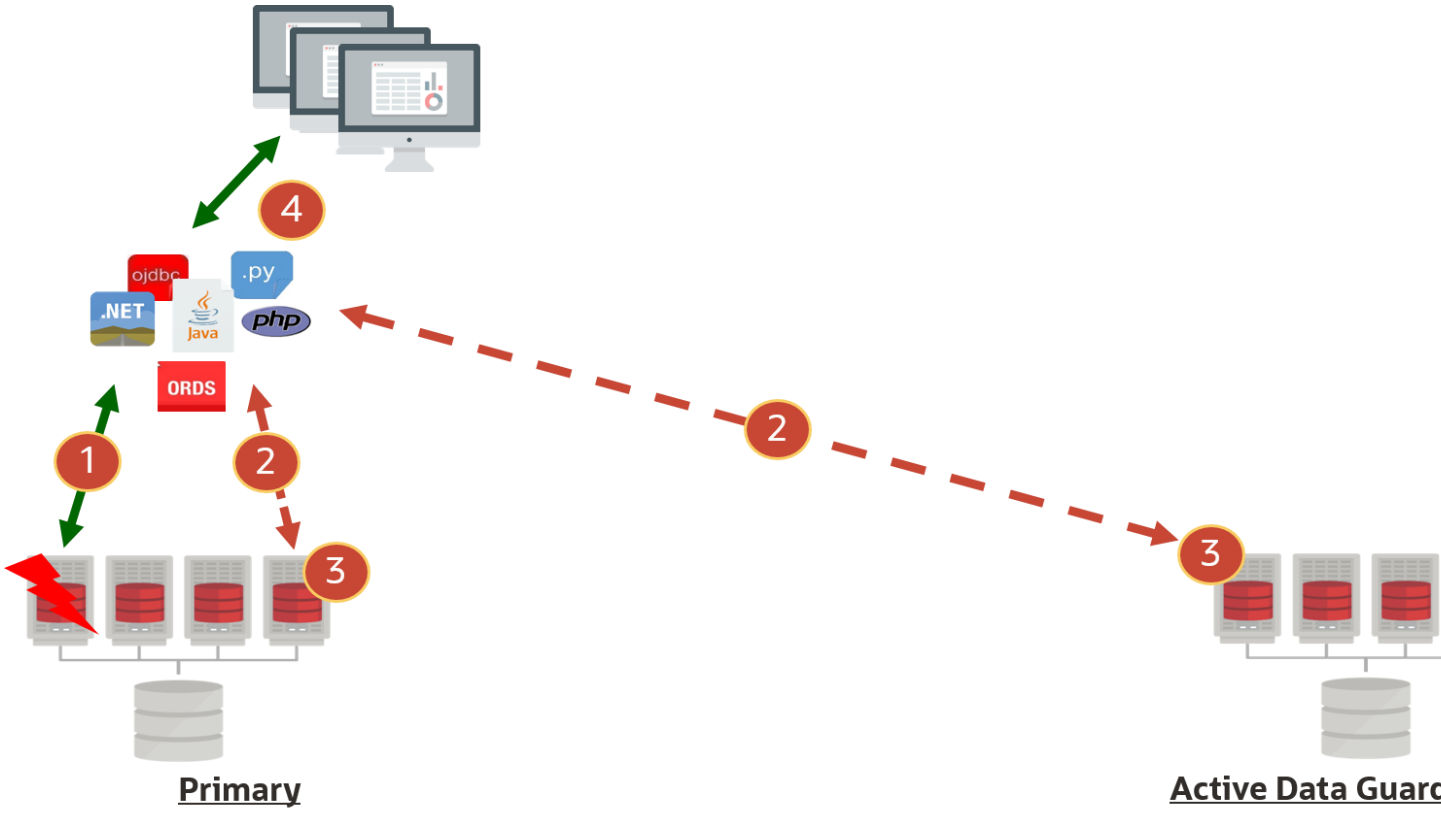
Oracle Site Guardの詳細は、Oracle Site Guard管理者ガイドを参照してください。

シームレスな接続フェイルオーバーの構成

Oracle Data Guard構成でシームレスなクライアント・フェイルオーバーを自動化する作業には、Data Guardフェイルオーバーの一部として新しいプライマリ・データベースにデータベース・サービスを再配置することや、障害の発生をクライアントに通知してTCPタイムアウトを解消すること、新しいプライマリ・データベースにクライアントをリダイレクトすることなどが含まれます。

次の図では、データベース・リクエストが停止またはタイムアウトによって中断し(1)、そのためセッションがOracle RACクラスタに再接続され(2)(またはスタンバイに再接続され(2))、データベース・リクエストが代替ノードで自動的にリプレイされ(3)、データベース・リクエストの結果がユーザーに返されます(4)。

図13-3 シームレスな接続フェイルオーバー



シームレスな接続フェイルオーバーを実現するには、Oracle RACおよびOracle Data Guardでインスタンスとデータベースの高速フェイルオーバーおよびスイッチオーバーの効果を最大化するように高速接続フェイルオーバー(FCF)をベスト・プラクティスとして構成します。FCFを使用すると、データベース・サービスが使用不可能になった場合に、クライアント(中間層アプリケーションまたはデータベースに直接接続する任意のプログラム)を迅速かつシームレスに使用可能なデータベース・サービスにフェイルオーバーできます。

- デフォルトのデータベース・サービスではないOracle Clusterware管理サービスを使用します(デフォルト・サービスの名前はデータベースまたはPDBと同じです)。作成するサービスでは、場所の透過性および高可用性の機能を提供します。
- 『アプリケーション・コンティニュイティ』ホワイト・ペーパーで説明されているように、タイムアウト、再試行および遅延が組み込まれた推奨の接続文字列を使用して、停止中に着信接続要求でエラーが表示されないようにします。
- 高速アプリケーション通知(FAN)は、サービス再開時およびロードの不均衡が発生した場合に、排出の開始、障害の分割、セッションのリバランスを行うための必須コンポーネントです。ノード障害やネットワーク障害などの停止の場合に、クライアントがFANによって中断されないと、アプリケーションの高速フェイルオーバーは行われません。FANは、すべてのフェイルオーバー・ソリューションに適用されます。
- メンテナンスが開始されたら、メンテナンス対象のインスタンスまたはノードからアプリケーション接続を排出します。ユニバーサル接続プール(UCP)かODP.NET接続プール、または接続テスト(あるいはその両方)でFANを有効にします。プールによってセッション移動のライフ・サイクル全体(メンテナンスの進行に応じたアプリケーション接続の排出およびリバランス)が提供されるため、FANを備えた接続プールが最適なソリューションです。
- セッションをフェイルオーバーする標準的なソリューションは、透過的アプリケーション・コンティニュイティ(TAC)を使用する

ことです。アプリケーションが読取り専用で、初期設定後にセッションのOracleセッション状態を変更しない場合は、透過的アプリケーション・フェイルオーバー(TAF)を使用します。あるいは、副次的作用またはコールバックを伴ってアプリケーション接続がフェイルオーバーする方法をカスタマイズする場合や、アプリケーションで一時表などの状態を使用していてクリーン・アップを行わない場合には、アプリケーション・コンティニュイティ(AC)を使用します。

フェイルオーバー・ソリューションを実装する場合は、前述のベスト・プラクティスとともに、ホワイト・ペーパー継続的な可用性 - MAAソリューションの継続的なサービスのためのアプリケーション・チェックリストの手順に従ってください。

ネットワーク・パフォーマンスの評価および最適化

Oracle Data Guardは、基礎となるネットワークを利用して、プライマリ・データベースからスタンバイ・データベースにREDOを送信します。ネットワークが正常で、ピークREDO生成率をサポートできることを保証することで、将来の転送ラグを回避できます。

転送ラグは、プライマリ・データベースがプライマリ・インスタンスのREDO生成率よりも速くスタンバイにREDOを送信できない場合に発生します。プライマリ・データベース障害が発生した場合、転送ラグによってデータ損失が発生する可能性があります。

ネットワーク評価は、次の評価で構成されます。

- ネットワークの信頼性
- ピークREDO生成率に対応するネットワーク帯域幅

ノート:

プライマリ・データベース・インスタンスの各インスタンスは、独自の REDO を生成し、単一のネットワーク・ストリームでスタンバイ・データベースに REDO を送信します。したがって、REDO 転送では、各ノードの単一プロセス・ネットワーク・スループットを最大にすることが重要です。

従来、ネットワークおよびREDO転送のスループットを削減できる領域があり、これによって転送ラグが生じる可能性があります。

1. ネットワーク・ファイアウォールまたはネットワーク暗号化

   ネットワーク・ファイアウォールおよびネットワーク(Oracle Netではなく)暗号化により、全体的なスループットが大幅に低下する可能性があります。oratcpツール(後述)を使用して、暗号化の有無にかかわらずスループットを検証し、適宜チューニングします。

   暗号化レベルを低くすると、スループットが大幅に向上することがあります。パフォーマンスとデータ損失の要件とともにセキュリティのニーズを満たすには、バランスが必要です。

2. REDO転送の圧縮

   データベース初期化パラメータにLOG_ARCHIVE_DEST_N属性COMPRESSION=ENABLEがある場合、Oracleバックグラウンド・プロセスは、ネットワーク・メッセージを送信する前にREDOを圧縮し、REDOを処理する前にREDOを圧縮解除する必要があります。これにより、REDOおよびネットワーク・スループット全体が削減されます。圧縮は、プライマリ宛先とスタンバイ宛先の間でネットワーク帯域幅が不十分な場合にのみ推奨されます。

3. Oracle Net暗号化

   REDOを含むOracle Netメッセージは送信前に暗号化し、その後、REDO処理の前に暗号化を解除する必要があるため、Oracle Netの暗号化レベルに応じて、REDOスループットの影響が異なります。

   データベース暗号化が透過的データ暗号化(TDE)ですでに有効化されている場合、REDOはすでに暗号化されてい

ますが、Oracle Net暗号化ではメッセージ・ヘッダーも暗号化できます。

4. REDO転送用のチューニングされていないネットワーク

    - オペレーティング・システムのソケット・バッファの最大サイズを増やすと、単一のプロセス・スループットが2から8倍増加する可能性があります。異なるソケット・バッファ・サイズでテストして、どの値が正の結果をもたらすかを確認し、スループットがピークREDOスループットより大きいことを確認します。
    - 様々なMTU設定とパフォーマンスを比較します。

      平均REDO書込みサイズが1500バイト未満の場合は、システム上でREDOを送信または受信するネットワーク・インタフェースのMTU=9000 (ジャンボ・フレームなど)などの様々なMTU設定を試みます。これにより、不要なネットワーク・ラウンドトリップが削減され、全体的なスループットが向上する可能性があります。

      また、SYNC転送の場合、Oracleの平均REDO書込みサイズ(Oracleメッセージの送信など)は、v$sysstatsまたはAWR統計のREDOサイズ/REDO書込みによって決定されるとおりに大幅に増加します。

      地理的なリージョン間でREDOを送信する場合、実験ではMTU=9000を使用すると、一部のネットワーク・トポロジでもメリットが得られることが示されています。oratcpでパフォーマンス・テストを実行し、結果をデフォルトのMTUおよびMTU=9000設定と比較します。

トポロジ情報の収集

Oracle Data Guard構成のトポロジとData Guardのパフォーマンスとの関連性を理解すると、Data Guardアーキテクチャに起因すると誤って判断されることがよくあるインフラストラクチャの脆弱性を排除するのに役立ちます。

次の上位レベルのアーキテクチャ情報の概要をまとめることをお薦めします。

- プライマリ・データベース・システムおよびスタンバイ・データベース・システムの説明(Oracle RACクラスタ内のノード数、データベース・ノードごとのCPUおよびメモリー、ストレージI/Oシステム)
- プライマリ・システムとスタンバイ・システムを接続するネットワーク・トポロジの説明
    - プライマリとスタンバイ間のネットワーク・スイッチおよびファイアウォール
    - ネットワーク帯域幅および待機時間

対称的なハードウェアおよび構成を備えたスタンバイ・データベースで、ピークREDO生成率をサポートできるようにネットワーク構成が適切にチューニングされている場合、転送ラグは1秒未満となります。

Data Guardのネットワーク使用状況の理解

ネットワークを最も頻繁に使用するData Guardライフ・サイクルのフェーズは次のとおりです。

- インスタンス化 - スタンバイ・データベース作成のこのフェーズでは、任意のホストからの並列処理を使用してファイルをコピーできます。ノード間のスループットを最大化する並列度を特定することは、スタンバイのインスタンス化プロセスを最適化するのに役立ちます。
- REDO転送(安定した状態): 通常のData Guard操作では、プライマリ・データベースは適用対象のスタンバイにREDOを送信します。プライマリ・データベースの各RACインスタンスは、それが実行されているホストから単一のストリームでREDOを送信します。各プライマリ・データベース・インスタンスの要件を理解し、単一のプロセスでスループット要件を達成できるようにすることは、スタンバイ・データベースがプライマリ・データベースに遅れないようにするために重要です。

インスタンス化のターゲットおよび目標の理解

大規模なデータベースのインスタンス化には数時間、極端な場合は数日かかることがあります。インスタンス化の計画を可能にし、プライマリ・システムとスタンバイ・システム間のスループットを最大化してできるだけタイミングよくインスタンス化を完了するには、まずインスタンス化時間の目標を決定します。次に、次に定義するプロセスに従って、プロセスごとのスループットを最大化し、プライマリ・ノードとスタンバイ・ノード間の最適な並列度を特定します。

REDO転送のスループット要件および平均REDO書込みサイズの理解

特定のData Guard構成に必要なネットワーク帯域幅は、プライマリ・データベースのREDO生成率によって決まります。

ノート:

プライマリ・データベースがすでに存在する場合は、必要なネットワーク帯域幅のベースラインを確立できます。既存のプライマリ・データベースがない場合は、このステップおよび今後のプロセスでのデータへの言及を読み飛ばしてください。

自動ワークロード・リポジトリ(AWR)ツールを使用してREDO生成率を特定できますが、スナップショットは多くの場合、30分または60分間隔であり、最大速度が低下する可能性があります。最大速度は比較的短い期間で発生することが多いため、既存のデータベースでの実行時に各ログのREDO生成率を計算する次の問合せを使用する方が正確です(必要に応じてタイムスタンプを変更します)。

```
SQL> SELECT THREAD#, SEQUENCE#, BLOCKS*BLOCK_SIZE/1024/1024 MB,
(NEXT_TIME-FIRST_TIME)*86400 SEC,
 (BLOCKS*BLOCK_SIZE/1024/1024)/((NEXT_TIME-FIRST_TIME)*86400) "MB/S"
FROM V$ARCHIVED_LOG
WHERE ((NEXT_TIME-FIRST_TIME)*86400<>0)
AND FIRST_TIME BETWEEN TO_DATE('2022/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
 AND TO_DATE('2022/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
AND DEST_ID=1 ORDER BY FIRST_TIME;
```

出力例:

```
THREAD# SEQUENCE# MB          SEC         MB/s
------- --------- ---------- ---------- ----------
      2      2291 29366.1963        831  35.338383
      1      2565 29365.6553        781 37.6000708
      2      2292 29359.3403        537  54.672887
      1      2566 29407.8296        813 36.1719921
      2      2293 29389.7012        678 43.3476418
      2      2294 29325.2217       1236 23.7259075
      1      2567 11407.3379       2658 4.29169973
      2      2295 24682.4648        477 51.7452093
      2      2296 29359.4458        954 30.7751004
      2      2297 29311.3638        586 50.0193921
      1      2568 3867.44092       5510 .701894903
```

ノート:

ピーク REDO 率を見つけるには、最高レベルの処理(OLTP のピーク期間、四半期末バッチ処理、年末バッチ処理など)中の時間を選択します。

この短い例では、最高速度は約52MB/sでした。ネットワークは、このアプリケーションの最大速度に30%を上乗せするか68MB/sを加算したものをサポートするのが理想的です。

平均REDO書込みサイズの検証

v$sysstatsを使用するか、様々なワークロードおよびピーク間隔についてAWRレポートを参照して、次に基づいて平均REDO書込みサイズを記録します

REDO書込み平均サイズ= "REDO SIZE" / "REDO WRITES"

oratcpのテストでこの平均REDO書込みサイズを使用します。平均REDO書込みサイズが1500バイトを超える場合は、様々なMTU設定を試してください。

現在のネットワーク・スループットの理解

Oracleユーティリティoratcptestは、どのOSユーザーでも実行できるiperf/qperfのような、ネットワーク帯域幅および待機時間を測定するための汎用ツールです。

oratcptestユーティリティには、次のようなネットワーク・ロードを制御するためのオプションが用意されています。

- ネットワーク・メッセージ・サイズ
- メッセージ間の遅延時間
- パラレル・ストリーム
- oratcptestサーバーがディスクにメッセージを書き込むかどうか。
- パケットの確認応答(ACK)を待機することによるData Guard SYNC転送のシミュレートまたはACKを待機しないことによるASYNC転送のシミュレート。

ノート:

このツールは、Oracle ネットワーク・ストリーミング転送と同様に、Data Guard 転送のようにソース・ホストからターゲット・ホストへの効率的なネットワーク・パケット転送をシミュレートできます。スループットは、ソース・サーバーとターゲット・サーバー間で使用可能なネットワーク帯域幅を飽和状態にする可能性があります。したがって、短期間のテストを実行し、同じネットワークを共有するその他の重要なアプリケーションについて検討することをお薦めします。

1つ以上のプロセスの既存のスループットの測定

既存のスループットを測定するには、次のタスクを実行します。

タスク1: oratcptestのインストール

1. MOSノート2064368.1からoratcptest.jarファイルをダウンロードします
2. JARファイルをクライアント(プライマリ)とサーバー(スタンバイ)の両方にコピーします

ノート:

oratcptest には JRE 6 以降が必要です

3. ホストにJRE 6以降があることを確認します
4. すべてのプライマリ・ホストおよびスタンバイ・ホストで、ヘルプを表示してJVMがJARファイルを実行できることを確認します

```
# java -jar oratcptest.jar -help
```

タスク2: 単一プロセスの既存のスループットの測定

Data Guardの非同期REDO転送(ASYNC)では、パケット確認応答を待機しないため、SYNC転送よりも高い速度を達成するストリーミング・プロトコルを使用します。

1. 受信(スタンバイ)側でテスト・サーバーを起動します。

```
java -jar oratcptest.jar -server [IP of standby host or VIP in RAC
 configurations] -port=<any available port number>
```

2. テスト・クライアントを実行します。(ステップ4で起動したサーバーのものと一致するように、サーバー・アドレスおよびポート番号を変更します。)

```
$ java -jar oratcptest.jar [IP of standby host or VIP in RAC configurations]
 -port=<port number> -mode=async -duration=120 -interval=20s
[Requesting a test]
        Message payload        = 1 Mbyte
        Payload content type   = RANDOM
        Delay between messages = NO
        Number of connections  = 1
        Socket send buffer     = (system default)
        Transport mode         = ASYNC
        Disk write             = NO
        Statistics interval    = 20 seconds
        Test duration          = 2 minutes
        Test frequency         = NO
        Network Timeout        = NO
        (1 Mbyte = 1024x1024 bytes)
(17:54:44) The server is ready.
                    Throughput
(17:55:04)     20.919 Mbytes/s
(17:55:24)     12.883 Mbytes/s
(17:55:44)     10.457 Mbytes/s
(17:56:04)     10.408 Mbytes/s
(17:56:24)     12.423 Mbytes/s
(17:56:44)     13.701 Mbytes/s
(17:56:44) Test finished.
               Socket send buffer = 2 Mbytes
                  Avg. throughput = 13.466 Mbytes/s
```

この例では、これらの2つのノード間の平均スループットは約13 MB/sで、問合せからの68 MB/sの要件を満たしていませんでした。

ノート:

このプロセスは、帯域幅が 1 日の様々な時間で変化するかどうかを判断するために、`-freq` オプションを使用して特定の頻度で実行されるようにスケジュールできます。たとえば、`-freq=1h/24h` を設定すると、1 時間ごとに 24 時間テストが繰り返されます。

タスク3: 複数プロセスの既存のスループットの測定

1. 2つの接続を(num_connパラメータを使用して)指定し、前述のテストを繰り返します。

```
$ java -jar oratcptest.jar <target IP address> -port=<port number>
 -duration=60s -interval=10s -mode=async [-output=<results file>] -num_conn=2
[Requesting a test]
        Message payload        = 1 Mbyte
        Payload content type   = RANDOM
        Delay between messages = NO
        Number of connections  = 2
        Socket send buffer     = (system default)
        Transport mode         = ASYNC
        Disk write             = NO
```

```
        Statistics interval    = 20 seconds
        Test duration          = 2 minutes
        Test frequency         = NO
        Network Timeout        = NO
        (1 Mbyte = 1024x1024 bytes)
(18:08:02) The server is ready.
                    Throughput
(18:08:22)     44.894 Mbytes/s
(18:08:42)     23.949 Mbytes/s
(18:09:02)     25.206 Mbytes/s
(18:09:22)     23.051 Mbytes/s
(18:09:42)     24.978 Mbytes/s
(18:10:02)     22.647 Mbytes/s
(18:10:02) Test finished.
          Avg. socket send buffer = 2097152
        Avg. aggregate throughput = 27.454 Mbytes/s
```

2. ステップ1を繰り返し実行し、3つの連続する値で集計スループットが増加しなくなるまで、num_connの値を毎回2ずつ増やします。たとえば、集計スループットが10個、12個および14個の接続でほぼ同じである場合、終了します。

> ノート:
>
> RMAN では、クラスタ内のすべてのノードをインスタンス化に使用できます。合計集計スループットを検出するには、My Oracle Support の『Creating a Physical Standby database using RMAN restore database from service』(ドキュメント ID 2283978.1)を参照してください。

3. すべてのクラスタ内の全ノードで同じテストを実行して、現在の合計集計スループットを検出します。プライマリのノード1からスタンバイのノード1、ノード2からノード2のように、すべてのノードで検出されたスループットを合計します。
4. ロールを逆にして、テストを繰り返します。
5. 最適な集計スループットを達成した接続の数をメモします。

データベースの合計サイズおよび合計集計スループットを使用して、データベースのコピーの完了にかかる時間を推定します。完全なインスタンス化では、コピー中に生成されたREDOを適用する必要もあります。データベースのアクティブ状態に基づいて、この推定時間に何パーセント(0%から50%)かさらに加算する必要があります。

見積り時間が目標を満たしている場合は、インスタンス化のための追加のチューニングは必要ありません。

1つ以上のプロセスによるREDO転送の最適化

前の単一プロセス・テストおよび複数プロセス・テストのスループットが目標を満たしている場合、追加のチューニングは必要ありません。より高いスループットが必要な場合、最大TCPソケット・バッファ・サイズをより大きい値に設定することが、スループットを増やす可能性がある主な方法です。

TCPソケット・バッファ・サイズの設定

TCPソケット・バッファは、受信データおよび送信データを一時的に格納するシステム・メモリー・バッファです。送信データは書込みバッファに格納されますが、受信データは読取りバッファに格納されます。読取りおよび書込みのソケット・バッファは別個に割り当てられます。バッファ(一般には読取りバッファ)が一杯になる(多くの場合、アプリケーションがバッファから十分な速さでデータを取得できない)と、データの送信が遅くなったり停止するために送信側にメッセージが送信されます。多くの場合、より大きなバッファを割り当てると、送信側を停止することなくワイヤからデータを取得する時間がアプリケーションに提供されるため、REDO転送が改善されます。

TCPソケット・バッファ・サイズのチューニングは、ASYNC転送を改善するための主な手法であり、場合によってはSYNC転送も

改善できます。

ノート:

ソケット・バッファ・サイズが比較的大きい場合は、TCP 選択的確認応答(SACK)をお薦めします。多くの場合、これはデフォルトで有効になっていますが、TCP 選択的確認応答の確認または有効化の詳細は、オペレーティング・システムのドキュメントを参照してください。

TCPソケット・バッファ・サイズを設定するには、次のタスクを実行します。

タスク1: 最適な最大ソケット・バッファ・サイズの決定

一連のテストを実行して、ターゲット・ネットワーク・リンク上の単一プロセスについて、最適な最大ソケット・バッファ・サイズを見つけます。

ノート:

帯域幅遅延積は、チャネルのネットワーク・リンク容量とラウンド・タイム(待機時間)の積です。ソケット・バッファ・サイズの最小推奨値は、特に待機時間の長い高帯域幅ネットワークの場合、3*BDP です。oratcptest を使用してソケット・バッファ・サイズをチューニングします。

タスク2: 最大ソケット・バッファ・サイズの一時的な設定

プライマリ・システムとスタンバイ・システムで、次の各ステップに従ってリクエストの最大ソケット・バッファ・サイズを設定します。これはメモリー内で行われ、なんらかの理由でサーバーを再起動された場合は維持されません。

rootとして次の各ステップを実行します。

1. まず、カーネル・パラメータnet.ipv4.tcp_rmemおよびnet.ipv4.tcp_wmemの現在のサイズを見つけます。返される値は、TCPが動的に割り当てるソケット・バッファの最小、デフォルトおよび最大のサイズです。ソケットの作成時に指定されたデフォルトよりも大きい値がプロセスで必要な場合、さらにバッファが最大値まで動的に割り当てられます。

```
# cat /proc/sys/net/ipv4/tcp_rmem
4096    87380   6291456
# cat /proc/sys/net/ipv4/tcp_wmem
4096    16384   4194304
```

2. 値を16MBまたは3*BDPの計算結果に変更します

```
# sysctl -w net.ipv4.tcp_rmem='4096 87380 16777216';
# sysctl -w net.ipv4.tcp_wmem='4096 16384 16777216';
```

ノート:

これらの値を増やすと、システムでのネットワーク・ソケットのシステム・メモリー使用量が増加する可能性があります。

ノート:

sysctl を使用して行われた変更は永続的ではありません。マシンの再起動後もこれらの変更を保持するために、/etc/sysctl.conf ファイルを更新します。適切な設定が決定されると、このプロセスの最後に構成ファイルを変更

するステップがあります。

タスク3: 単一プロセスのスループットのテスト

前のテストを再実行して、前のステップで設定した新しい最大値までソケット・バッファが動的に増加できるようにします。

(oracleとして)

サーバー(スタンバイ):

```
$ java -jar oratcptest.jar -server [IP of standby host or VIP in RAC configurations]
 -port=<port number>
```

クライアント(プライマリ):

```
$ java -jar oratcptest.jar <IP of standby host or VIP in RAC configurations>
 -port=<port number> -mode=async -duration=120s -interval=20s
```

ノート:

ソケット・バッファ・サイズの明示的なリクエストを制御するカーネル・パラメータは、このテストに設定されたものと異なるため、oratcptest sockbuf パラメータを使用しないでください。

テストが完了すると、クライアントおよびサーバーからの結果に、そのテスト中のソケット・バッファの値が表示されます。このドキュメントの執筆時点では、その値は実際のソケット・バッファ・サイズの半分であり、使用された実際のサイズを検出するには2倍にする必要があります。

クライアント

```
[Requesting a test]
        Message payload = 1 Mbyte
        Payload content type = RANDOM
        Delay between messages = NO
        Number of connections = 1
        Socket send buffer = 2 Mbytes
        Transport mode = ASYNC
        Disk write = NO
        Statistics interval = 20 seconds
        Test duration = 2 minutes
        Test frequency = NO
        Network Timeout = NO
        (1 Mbyte = 1024x1024 bytes)
(11:39:16) The server is ready.
                Throughput
(11:39:36) 71.322 Mbytes/s
(11:39:56) 71.376 Mbytes/s
(11:40:16) 72.104 Mbytes/s
(11:40:36) 79.332 Mbytes/s
(11:40:56) 76.426 Mbytes/s
(11:41:16) 68.713 Mbytes/s
(11:41:16) Test finished.
          Socket send buffer = 8388608
            Avg. throughput = 73.209 Mbytes/s
```

サーバー

```
The test terminated. The socket receive buffer was 8 Mbytes.
```

ノート:

oratcptest のレポートでは、ソケットに割り当てられたバッファの半分になっています。テスト中に使用された実際のソケット・バッファ・サイズについてレポートされた数値を 2 倍にします。

タスク4: 複数プロセスのスループットのテスト

num_conn=10に設定された10個のプロセスで、ピーク集計スループットに到達した場合など、最初のテストで決定されたnum_conn値を使用してテストを繰り返します。

クライアント

```
$ java -jar oratcptest.jar <IP of standby host or VIP in RAC configurations>
 -port=<port number> -mode=async -duration=120s -interval=20s -num_conn=10
[Requesting a test]
        Message payload        = 1 Mbyte
        Payload content type   = RANDOM
        Delay between messages = NO
        Number of connections  = 10
        Socket send buffer     = (system default)
        Transport mode         = ASYNC
        Disk write             = NO
        Statistics interval    = 20 seconds
        Test duration          = 2 minutes
        Test frequency         = NO
        Network Timeout        = NO
        (1 Mbyte = 1024x1024 bytes)
(19:01:38) The server is ready.
                    Throughput
(19:01:58)    266.077 Mbytes/s
(19:02:18)    242.035 Mbytes/s
(19:02:38)    179.574 Mbytes/s
(19:02:58)    189.578 Mbytes/s
(19:03:18)    218.856 Mbytes/s
(19:03:38)    209.130 Mbytes/s
(19:03:38) Test finished.
          Avg. socket send buffer = 8 Mbytes
        Avg. aggregate throughput = 217.537 Mbytes/s
```

ノート:

oratcptest のレポートでは、ソケットに割り当てられたバッファの半分になっています。テスト中に使用された実際のソケット・バッファ・サイズについてレポートされた数値を 2 倍にします。

サーバー(各接続で受信バッファが出力されます。各インスタンスのソケット・バッファ・サイズを2倍にします)

```
The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.

The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.
The test terminated. The socket receive buffer was 8 Mbytes.
```

データベースの合計サイズおよび合計集計スループットを使用して、データベースのコピーの完了にかかる時間を推定します。完全なインスタンス化では、コピー中に生成されたREDOを適用する必要もあります。データベースのアクティブ状態に基づいて、こ

の推定時間に何パーセント(0%から50%)かさらに加算する必要があります。

タスク5: テストの繰返し

スループットがさらに必要な場合は、tcp_rmemおよびtcp_wmemの値を大きくして前の2つのテストを繰り返します。これらの大きくした値は、他のソケットにも使用できますが、必要な場合にのみ動的に割り当てられることを理解してください。この表は、ソケット・バッファ・サイズごとに異なるスループットの結果を追跡するサンプル・データを示しています。

tcp_rmemの最大値	tcp_wmemの最大値	単一プロセスのスループット	単一ノードのマルチプロセスの最大集計スループット	単一ノードのマルチプロセスの並列度
6291456	4194304	13.5 MB/s	203 MB/s	16
8388608	8388608	48 MB/s	523 MB/s	14
16777216	16777216	73 MB/s	700 MB/s	14
33554432	33554432	132 MB/s	823 MB/s	14

タスク6: パラメータの永続的な設定

sysctlを使用した変更により、ホストの再起動後には保持されないメモリー内の値が変更されます。ソケット・バッファの最適なサイズが決定されたら、/etc/sysctl.confファイルを編集してサーバーの再起動後も保持されるようにカーネル・パラメータを設定します。

これは、プライマリ・システムおよびスタンバイ・システムのすべてのノードで実行する必要があります。

これらの変更を永続的にするには、既存の値を変更するか、存在しない場合はこれらの値をファイルに追加して/etc/sysctl.confを編集します。

```
net.ipv4.tcp_rmem='4096 87380 16777216'
net.ipv4.tcp_wmem='4096 16384 16777216'
```

タスク7: 大きいMTUの評価

Data Guardト転送で使用されるネットワーク・インタフェースを決定します。

平均REDO書込みサイズが現在のMTU設定(通常はデフォルトの1500など)より大きい場合、ジャンボ・フレーム(MTU=9000など)によってこれらの大規模なネットワーク・パケットのネットワークRTTが削減され、REDOスループット全体が向上するかどうかを評価します。

Linuxでテスト目的でData Guard転送ネットワーク・インタフェースのMTUを変更する例を次に示します。

```
ifconfig bondeth0 mtu 9000 up
```

前述したように、MTUサイズを大きくして同じoratcpパフォーマンス手法を繰り返し、スループットが向上したかどうかを確認します。

パフォーマンスが向上した場合は、システムおよびネットワーク・エンジニアと協力して、プライマリ・データベースとスタンバイ・データベースの両方のDG転送のMTUサイズを変更します。

このデータの使用方法

スループットの数値を使用してスループットを決定し、REDO転送およびインスタンス化の状況に役立てることができます。

REDO転送

単一プロセスのスループットがプライマリ・データベースの単一インスタンスのREDO生成率を超えない場合、スタンバイは、この間、プライマリに遅れをとるようになります。このような場合、ネットワーク・エンジニアリング・チームによるさらなる評価およびネットワーク・チューニングが必要になることがあります。

インスタンス化

すべてのノードの最大集計スループットが認識されると、インスタンス化の大まかな見積りを策定できます。たとえば、インスタンス化する2ノードRACに100 TBのデータベースがあり、各ノードが300 MB/秒を達成できる場合、データ・ファイルのコピーには約50時間かかります。インスタンス化するための追加作業で、その数値に何パーセントか(30%まで)加算されます。

```
300 MB/s * 60 seconds/minute * 60 minutes/hour * 2 nodes = ~2 TB/hr aggregate for
both nodes
100TB / 2TB/hr = ~50 hours
```

複数のノードを使用するなど、大規模なデータベースの最適化を使用してデータベースをインスタンス化するステップについては、『Creating a Physical Standby database using RMAN restore database from service』(ドキュメントID 2283978.1)を参照してください。

Oracle Data Guard保護モードの決定

Oracle Data Guardは3つの異なる保護モードで実行でき、モードごとに対応するパフォーマンス、可用性およびデータ損失の要件が異なります。このガイドを使用して、ビジネス要件に適合する保護モードおよび潜在的な環境制約を決定します。

最大保護モード: このモードでは、プライマリ・データベースの障害発生時に、それが複数の障害(たとえば、プライマリとスタンバイ間のネットワークに障害が発生し、しばらくしてからプライマリに障害が発生する場合)であっても、データの損失がないことが保証されます。このポリシーを実施するため、REDOがディスクに固定されたことを少なくとも1つのData Guardスタンバイが確認応答するまで、プライマリ・データベース・トランザクションのコミット成功は伝達されません。このような確認応答がない場合、プライマリ・データベースでは保護されていないトランザクションのコミットは不可能で、データベースは一時停止して最終的にはシャットダウンします。

プライマリ・データベースは操作可能でスタンバイ・データベースは操作不可能な場合に可用性を保つためのベスト・プラクティスは、最大保護構成で同期化されたスタンバイ・データベースを常に2つ以上保持することです。プライマリ・データベースの可用性は、少なくとも1つの同期化されたスタンバイ・データベースから確認応答を受け取っていれば、影響されません。

データ損失ゼロがデータベース可用性よりも重要な場合には、この保護モードを選択します。SYNC転送を有効にするときに、オーバーヘッドを許容できるかどうかを測定するには、ワークロードの影響分析をお薦めします。

最大可用性モード: このモードでは、プライマリ・データベースで構成に影響を与える最初の障害が発生した場合に、データ損失は起こらないことが保証されます。 最大保護モードとは異なり、最大可用性モードはいずれかのスタンバイ・データベースからの確認応答に対してNET_TIMEOUTの最大秒数を待機します。その後、コミット成功をアプリケーションに伝達して、次のトランザクションに移動します。プライマリ・データベースの可用性(つまりモードの名前と同じ)は、スタンバイとの通信ができない場合(たとえば、スタンバイまたはネットワークの停止が原因の場合)でも、影響を受けません。Data Guardは引き続きスタンバイにpingを送信し、可能になったら自動的に接続を再確立して、スタンバイ・データベースを再同期化します。ただし、プライマリとスタンバイが分岐した期間中にデータ損失が発生し、これはプライマリ・データベースに影響を与える二次障害になります。

このような理由から、保護レベルを監視(Enterprise Manager Grid Controlの使用が最も簡単)して、プライマリとスタンバイの間の通信の遮断は、二次障害が発生する前に迅速に解決することがベスト・プラクティスとなります。これは、最も一般的なデータ損失ゼロのデータベース保護モードです。

データ損失ゼロは非常に重要であるものの、すべてのスタンバイ・データベースにアクセスできなくなるという可能性の低い事態でも引き続きプライマリ・データベースを使用可能にする場合には、この保護モードを選択します。このソリューションを補完するには、

複数のスタンバイ・データベースを統合するか、遠隔同期インスタンスを使用してWAN全体でデータ損失ゼロのスタンバイ・ソリューションを実装します。SYNC転送を有効にするときに、オーバーヘッドを許容できるかどうかを測定するには、ワークロードの影響分析をお薦めします。

最大パフォーマンス・モードは、デフォルトのData Guardモードで、プライマリ・データベースのパフォーマンスまたは可用性に影響を与えない範囲で最高レベルのデータ保護を提供します。このモードでは、トランザクションのリカバリに必要なREDOデータがプライマリ・データベースにあるローカルのオンラインREDOログに書き込まれると、即座にそのトランザクションのコミットが許可されます(スタンバイ・データベースが存在しなかった場合と同じ動作)。Data Guardは、REDOを1)プライマリ・ログ・バッファからスタンバイ・データベースに直接送信するとともに、2)ローカル・オンラインREDOログ書込みに非同期で送信することで、プライマリ・サイトが失われた場合のデータ損失の可能性を非常に低く抑えます。スタンバイ確認応答の待機は発生しませんが、それでも、このデータ保護モードではデータ損失の可能性をゼロに近くすることができます。

最大可用性モードと同様に、Enterprise Manager Grid Controlで保護レベルを監視して、二次障害が発生する前にプライマリとスタンバイの間の通信の遮断を迅速に解決することがベスト・プラクティスです。

最小データ損失を許容でき、プライマリに対するパフォーマンスへの影響がゼロの場合には、このモードを選択します。

読取り専用スタンバイ・データベースへの問合せのオフロード

問合せおよびレポートのワークロードを読取り専用スタンバイ・データベースにオフロードすると、プライマリ・データベースのシステム・リソースが解放され、ユーザー、ワークロードさらにはデータベースまで追加できるようになります。

プライマリ・データベースとスタンバイ・データベースの両方のリソースを利用すると、業務およびアプリケーションでの合計システム使用率が高くなり、アプリケーションのスループットが向上する可能性もあります。

次のステップに従って、適切なワークロードをオフロードします。

1. 読取り専用または読取り多用のアプリケーション・モジュールを特定します。
    - 読取り専用のアプリケーション・サービスまたはモジュールがあるかどうかを評価します。
    - 小規模で短い読取り専用の問合せは、スタンバイ・データベースへのオフロードの候補として適しています。
    - 短いDML、特にレスポンス時間に依存するDMLは、スタンバイにオフロードしないでください。
    - 大規模なレポートまたは分析レポートは、オフロードの候補として適しています。
    - 主に読み取られるレポートや、低頻度のDMLが(通常はレポートの先頭または末尾)に含まれることがあるレポートは、オフロードの候補として適している場合があります。

      DMLリダイレクトを有効にするには、ADG_REDIRECT_DMLを参照してください。
2. オフロードの候補ごとに、予想されるアプリケーション・パフォーマンス、スループット、レスポンス時間または経過時間のサービス・レベルに関する情報を収集します。
    - オフロードに適した候補となる問合せおよびレポートを決定したら、それぞれに必要な予想および最大のレスポンス時間または経過時間を求めます。たとえば、大規模な分析レポートの中には、2時間以内に完了する必要があるものがあります。
    - 短い問合せの場合は、予想されるレスポンス時間とスループットの見込みを割り出します。
    - これらの要件は、次のステップで必要になるアプリケーション・パフォーマンスに関するサービス・レベル契約と呼ばれることもあります。
3. スタンバイ上の各候補のパフォーマンスをテストし、要件を満たしているかどうかを判断します。

- プライマリ・データベースとスタンバイ・データベースのデータは基本的に同じですが、独立したデータベース、独立したマシン、独立した構成であり、ワークロードが異なります。たとえば、Active Data Guardの読取り専用スタンバイ・データベースには、REDO適用のワークロードとオフロードされる問合せがありますが、プライマリ・データベースにはOLTP、バッチおよび問合せのワークロードがある場合があります。
- レポートされる経過時間、問合せレスポンス時間およびワークロード・パフォーマンスは、これらのシステム、構成およびワークロードの違いにより、プライマリとスタンバイで異なる場合があります。
- チューニングでは、システム・リソース、SQL計画および個々の問合せのCPUと待機プロファイルを理解する必要があります。チューニングに関する推奨事項は、プライマリ・データベースとスタンバイ・データベースの両方に適用できます。データベース・パフォーマンスの診断およびチューニングを参照してください。

4. パフォーマンス要件を満たす問合せのサブセットをオフロードし、処理能力を高めるためにプライマリ・データベースのリソースを解放します。
   - オフロードできる問合せおよびレポートを決定し、それらのアクティビティのパフォーマンスが許容できるようになったら、ワークロードの一部を徐々にオフロードして監視します。
   - チューニング後にREDO適用の速度を維持できなくなるほど、過剰なワークロードをスタンバイにオーバーサブスクライブおよびオフロードしないでください。スタンバイが遅れを取ると、実行可能なロール・トランジション・ターゲットとしてのそのスタンバイは失われ、ほとんどの場合、遅れているスタンバイは問合せのオフロードに使用できません。

特定の問合せが要件を満たしていない場合はどうなりますか。

1. パフォーマンス・エンジニアに相談し、Databaseパフォーマンス・チューニング・ガイドの推奨事項に従ってください。
2. 特定の問合せのレスポンス時間、スループットまたはレポート経過時間は、スタンバイ・システムとプライマリとで同じである保証はありません。システム・リソース、SQL計画、全体的なCPU作業時間および待機時間を分析します。

   たとえば、短い問合せのいずれかで、`standby query scn advance wait`がずっと長い経過時間の原因となっていることがわかります。この待機の増加は、Active Data GuardのREDO適用に起因します。問合せでデータ・ブロック内の特定の行が検出され、問合せのシステム・コミット番号(SCN)の時点でトランザクションがコミットされていないためロールバックする必要がある場合は、対応するUNDOを適用して、その問合せの読取り一貫性を取得する必要があります。対応するUNDO変更のREDOがREDO適用によってまだ適用されていない場合、問合せは待機する必要があります。このような待機の存在自体は問題ではなく、通常は数ミリ秒かかることがありますが、ワークロードによって異なり、Real Application Clusterデータベース・システムでの方が大きくなる可能性があります。

14 Oracle Data Guardの構成およびデプロイ

Oracle Data Guardを構成およびデプロイするには、次のOracle MAAベスト・プラクティスの推奨事項を使用します。

Oracle Data Guardの構成ベスト・プラクティス

次のトピックでは、Oracle Data Guard構成を構成するためのOracle MAAベスト・プラクティスについて説明します。

最初にOracle Database構成のベスト・プラクティスを適用

後続のOracle Data Guardのベスト・プラクティスを実装する前に、Oracle Database構成のベスト・プラクティスを適用します。

Oracle Data Guard構成ベスト・プラクティスは、一般的なOracle Database構成のベスト・プラクティスを補足するものであり、MAA Goldリファレンス・アーキテクチャで想定されるサービス・レベルを実現するのに役立ちます。つまり、Data Guard構成ではデータベース構成のすべてのベスト・プラクティスに従い、矛盾がある場合には一般的なデータベースの推奨事項よりもここで説明するData Guardの推奨事項を優先するということです。

詳細は、Oracle Database構成のベスト・プラクティスを参照してください。

Recovery Managerを使用したスタンバイ・データベースの作成

Oracle Data Guardスタンバイ・データベースはいくつかの方法で作成できますが、Oracle MAAでは、その簡易性から、`RMAN RESTORE ... FROM SERVICE`句を使用してフィジカル・スタンバイ・データベースを作成するアプローチが推奨されます。

このアプローチの詳細は、Creating a Physical Standby database using RMAN restore from service (Doc ID 2283978.1)を参照してください。

Oracle Data GuardによるOracle Data Guardブローカの使用

Oracle Data Guard Brokerを使用して、Oracle Data Guard構成を作成、管理および監視します。

このブローカのインタフェースである、Oracle Enterprise ManagerのData Guard管理ページ(ブローカのGUI)およびDGMGRLと呼ばれるData Guardコマンドライン・インタフェースを使用すると、すべてのData Guard管理操作をローカルまたはリモートで実行できます。

ブローカのインタフェースによって利便性が向上し、Data Guard構成の管理と監視が集中管理されます。ブローカは、Oracle Database Enterprise EditionおよびPersonal Editionの機能として使用可能できるだけでなく、Oracle Database、Oracle Enterprise ManagerおよびOracle Cloud Control Planeとも統合されます。

ブローカのインストールおよび構成の例

ブローカのインストールおよび構成の例を次に示します。ブローカ構成ベスト・プラクティスのすべての例で使用されています。

前提条件:

- プライマリ・データベース、スタンバイ・データベースおよびオブザーバは、障害分離を実現するために別々のサーバーおよびハードウェア上に存在すること。
- プライマリ・データベースとスタンバイ・データベースの両方で`SPFILE`を使用していること。

- DG_BROKER_START初期化パラメータをTRUEに設定していること。
- 構成にOracle RACデータベースが含まれる場合は、そのデータベースのDG_BROKER_CONFIG_FILEn初期化パラメータを、そのデータベースの全インスタンスについて同じ共有ファイルを指すように設定する必要があります。共有ファイルには、クラスタ・ファイル・システム上のファイル(該当する場合)、RAWデバイス上のファイルまたは自動ストレージ管理を使用して格納されたファイルを使用できます。

1. まだ存在しない場合は、プライマリ・データベースおよびスタンバイ・データベースに接続するOracle Net Services別名を作成します。これらの別名は、Data Guard構成の各ホストまたはメンバーのデータベース・ホームに存在する必要があります。Oracle RAC構成の場合、別名はSCAN名を使用して接続する必要があります。

```
chicago =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP)
     (HOST=prmy-scan)(PORT=1521)))
 (CONNECT_DATA =
   (SERVER = DEDICATED)
    (SERVICE_NAME = chicago)))
boston =
 (DESCRIPTION =
  (ADDRESS_LIST =
    (ADDRESS=(PROTOCOL= TCP)
     (HOST=stby-scan)(PORT=1521)))
 (CONNECT_DATA =
   (SERVER = DEDICATED)
    (SERVICE_NAME = boston)))
```

2. プライマリ・ホストで、DGMGRLに接続し、構成を作成します。

```
$ dgmgrl sys
Enter password: password
DGMGRL> create configuration 'dg_config' as  primary database is 'chicago'
connect identifier is chicago;

 Configuration "dg_config" created with primary database "chicago"

DGMGRL> add database 'boston' as connect identifier is boston;

 Database "boston" added

DGMGRL> enable configuration;
 Enabled.
```

3. デフォルトでは、ブローカは最大パフォーマンス・データベース保護モードのためにLOG_ARCHIVE_DEST_nを設定します。

   ブローカは、次に示すように、非同期トランスポートのデフォルト値を使用してリモート・アーカイブ先を構成します。

```
log_archive_dest_3=service="boston", ASYNC NOAFFIRM delay=0 optional
compression=disable max_failure=0 reopen=300 db_unique_name="boston"
net_timeout=30, valid_for=(online_logfile,all_roles)
```

REDO転送モードの構成

LogXptModeプロパティを次のいずれかのモードに設定して、各構成メンバーでREDO転送サービスを構成します。

- ASYNCは、このスタンバイ・データベースに対するREDO転送サービスを、LOG_ARCHIVE_DEST_n初期化パラメータのASYNC属性およびNOAFFIRM属性を使用して構成します。このモードをスタンバイREDOログ・ファイルとともに使用すると、パフォーマンスへの影響なしで数秒もかからない最小データ損失データ保護を実現します。

- FASTSYNCは、このスタンバイ・データベースに対するREDO転送サービスを、LOG_ARCHIVE_DEST_n初期化パラメータのSYNC属性およびNOAFFIRM属性を使用して構成します。最大可用性モード保護モードを使用している場合は、NOAFFIRM属性(デフォルト=AFFIRM)を指定した同期REDO転送モードを構成します。これは、スタンバイ・メモリー内でREDOが正常に受信および確認されたら、REDOがスタンバイREDOログに書き込まれる前にREDO受信を確認応答することで、同期REDO転送のパフォーマンスへの影響を最小限に抑えることができます。プライマリ・データベースのみで障害が発生した場合でも、データ損失ゼロの保護は保持されます。
- SYNCは、このスタンバイ・データベースに対するREDO転送サービスを、LOG_ARCHIVE_DEST_n初期化パラメータのSYNC属性およびAFFIRM属性を使用して構成します。このモードとスタンバイREDOログ・ファイルは、最大保護モードまたは最大可用性モードのいずれかで動作している構成に必要です。このREDO転送サービスを使用すると、プライマリ・データベースへのデータ損失ゼロのデータ保護が可能になりますが、プライマリとスタンバイの間のラウンド・トリップ待機時間が長い場合には(2ミリ秒を超える場合など)、パフォーマンスへの影響も大きくなる可能性があります。このオプションは、最大保護モードでは必須です。

次の例に示すように、EDIT DATABASE SET PROPERTYコマンドを使用して、ブローカ構成の転送モードを設定します。

```
DGMGRL> EDIT DATABASE 'boston' SET PROPERTY LogXptMode=ASYNC;
DGMGRL> EDIT DATABASE 'chicago' SET PROPERTY LogXptMode=FASTSYNC;
DGMGRL> EDIT DATABASE 'SanFran' SET PROPERTY LogXptMode=SYNC;
```

ブローカ構成の検証

構成全体の問題を識別するには、次のステップを使用して検証します。

1. SHOW CONFIGURATIONコマンドを使用して、ブローカ構成のステータスを表示します。

```
DGMGRL> show configuration;
 Configuration – dg
   Protection Mode: MaxPerformance
   Members:
   chicago - Primary database
   boston - Physical standby database
 Fast-Start Failover: DISABLED
 Configuration Status:
 SUCCESS   (status updated 18 seconds ago)
```

   構成ステータスがSUCCESSの場合は、ブローカ構成全体が正常に動作しています。ただし、構成ステータスがWARNINGまたはERRORである場合は、構成の一部に問題があることを意味します。WARNINGステータスまたはERRORステータスに付随する追加のエラー・メッセージを使用して、問題を識別できます。次のステップでは、構成内の各データベースを調べて、特定のエラーに関連する内容を絞り込みます。

2. プライマリ・データベースおよびスタンバイ・データベースに対する警告を識別するには、SHOW DATABASEコマンドを使用してステータスを表示します。

```
DGMGRL> show database chicago

 Database – chicago

   Role:               PRIMARY
   Intended State:     TRANSPORT-ON
   Instance(s):
     tin1
     tin2

 Database Status:
 SUCCESS
```

データベース・ステータスがSUCCESSの場合、データベースは正常に動作しています。ただし、データベース・ステータスがWARNINGまたはERRORである場合は、データベースの一部に問題があることを意味します。WARNINGステータスまたはERRORステータスに付随する追加のエラー・メッセージを使用して、現在の問題を識別できます。

スタンバイ・データベースでSHOW DATABASEコマンドを繰り返し、エラー・メッセージを評価します。

3. Oracle Database 12.1以降でデータベースを検証します。

   Oracle Database 12.1以降の場合、Data Guard Brokerには前述のコマンドに加えてVALIDATE DATABASEコマンドもあります。

```
DGMGRL> validate database chicago
  Database Role:    Primary database
   Ready for Switchover:  Yes
DGMGRL> validate database boston;

   Database Role:     Physical standby database
   Primary Database:  tin

   Ready for Switchover:  No
   Ready for Failover:    Yes (Primary Running)

   Capacity Information:
     Database  Instances        Threads
     tin       2                2
     can       1                2
     Warning: the target standby has fewer instances than the
     primary database, this may impact application performance

   Standby Apply-Related Information:
     Apply State:      Not Running
     Apply Lag:        Unknown
     Apply Delay:      0 minutes
```

   VALIDATE DATABASEコマンドは、SUCCESSステータスまたはWARNINGステータスを提供しないため、アクションを実行する必要があるかどうかを判断するための調査が必要になります。

ファスト・スタート・フェイルオーバーの構成

ファスト・スタート・フェイルオーバーでは、プライマリ・データベースが消失した場合、ブローカにより、事前に選択されたスタンバイ・データベースに自動的にフェイルオーバーできます。プライマリ・データベース、クラスタまたはサイトに障害が発生した場合に厳しいRTO要件を満たすためには、ファスト・スタート・フェイルオーバーを有効にする必要があります。

ファスト・スタート・フェイルオーバーでは、ターゲット・スタンバイ・データベースがプライマリ・データベース・ロールに迅速かつ確実にフェイルオーバーされます。このとき、手動ステップを実行してフェイルオーバーを起動する必要はありません。ファスト・スタート・フェイルオーバーは、ブローカ構成でのみ使用できます。

プライマリ・データベースに複数のスタンバイ・データベースがある場合、FastStartFailoverTargetプロパティを使用して複数のファスト・スタート・フェイルオーバー・ターゲットを指定できます。このようなターゲットはターゲット候補と呼びます。ブローカは、FastStartFailoverTargetプロパティでの指定順に基づいて、ターゲットを選択します。指定されたファスト・スタート・フェイルオーバー・ターゲットに問題が発生し、フェイルオーバーのターゲットにできない場合、ブローカはファスト・スタート・フェイルオーバー・ターゲットを別のターゲット候補のいずれかに自動的に変更します。

ファスト・スタート・フェイルオーバーでは、任意の保護モードを使用できます。最大保護モードおよび最大可用性モードは、データが消失しないことを保証する自動フェイルオーバー環境を提供します。最大パフォーマンス・モードによる自動フェイルオーバー環境では、消失データがFastStartFailoverLagLimit構成プロパティで指定したデータ量(秒単位)以下であることが保証されます。このプロパティは、自動フェイルオーバーの実行に許容される最大消失データ量を示します。このプロパティは、ファス

ト・スタート・フェイルオーバーが有効化され、構成が最大パフォーマンス・モードで動作している場合にのみ使用されます。

1. 次の例に示すように、FastStartFailoverThresholdプロパティを設定することで、プライマリ・データベースを使用できないことが検出されてからフェイルオーバーを開始するまでにオブザーバおよびターゲット・スタンバイ・データベースが何秒待機するかを指定します。

```
DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold = seconds;
```

ファスト・スタート・フェイルオーバーが発生するのは、オブザーバとスタンバイ・データベースの両方が本番データベースとの接続を失ってからFastStartFailoverThresholdに設定されている値を超える期間が経過したときと、構成の状態が同期されていること(最大可用性)またはラグが構成済のFastStartFailoverLagLimitを超えないこと(最大パフォーマンス)に両者が同意したときです。

FastStartFailoverThresholdの最適な値は、可能なかぎり迅速にフェイルオーバー(停止時間を最小化)することと、一時ネットワークの不規則性や可用性に影響しないその他の短期間のイベントが原因でフェイルオーバーが不必要にトリガーされることとのトレードオフを考慮して判断します。

FastStartFailoverThresholdのデフォルト値は30秒です。

次の表は、様々なユースケースにおけるFastStartFailoverThresholdの推奨設定を示しています。

表14-1 FastStartFailoverThresholdの最小値の推奨設定

構成	**最小値の推奨設定**
単一インスタンスのプライマリ、短い待機時間、信頼性の高いネットワーク	15 秒
単一インスタンスのプライマリ、WAN を介した待機時間の長いネットワーク	30 秒
Oracle RAC のプライマリ	Oracle RAC ミスカウント + 再構成時間 + 30 秒

2. トポロジ内でオブザーバを配置する場所を決定します。

独自の可用性ドメイン(AD)またはデータ・センター内にあるプライマリ、スタンバイおよびオブザーバとともにファスト・スタート・フェイルオーバーをデプロイするのが理想的な状態ですが、2つの可用性ドメインまたは単一の可用性ドメインのみを使用する構成をサポートする必要があります。次に、オブザーバを配置する場合の推奨事項を2つのユースケースで示します。

デプロイメント構成1: リージョンが2つで、それぞれのリージョンにADが2つあります。

- 初期プライマリ・リージョンでは、プライマリ・データベースがAD1にあり、高可用性オブザーバが2つあります(1つ目のオブザーバはAD2に、2つ目のHAオブザーバはAD1にあります)
- 初期スタンバイ・リージョンでは、スタンバイ・データベースがAD1にあり、ロール変更後に使用される高可用性オブザーバが2つあります(1つ目のオブザーバはAD2に、2つ目のHAオブザーバはAD1にあります)
- オブザーバの場合、MAAでは少なくとも2つのオブザーバ・ターゲットを同じプライマリ・リージョンの異なるADに存在させることをお薦めします

デプロイメント構成2: リージョンが2つで、それぞれのリージョンにADが1つのみあります

- 初期プライマリ・リージョンには、プライマリ・データベースと、オブザーバをホストする2つの軽量サーバーがあります
- 初期スタンバイ・リージョンには、スタンバイ・データベースと、オブザーバをホストする2つの軽量サーバーがあります(ロール変更がある場合)

3. オブザーバ高可用性を構成します。

   1つのData Guard Broker構成を監視するには、最大で3つのオブザーバを登録できます。各オブザーバは、START OBSERVERコマンドの発行時に指定する名前で識別されます。オブザーバをバックグラウンド・プロセスとして起動することもできます。

   ```
   DGMGRL> sys@boston
   Enter password: password
   DGMGRL> start observer number_one in background;
   ```

   高可用性のために追加のオブザーバを同じホストまたは別のホストで起動できます。

   ```
   DGMGRL> sys@boston
   Enter password: password
   DGMGRL> start observer number_two in background;
   ```

   Data Guard Brokerでファスト・スタート・フェイルオーバーを調整できるのはプライマリ・オブザーバのみです。他の登録済オブザーバはすべてバックアップ・オブザーバとみなされます。

   オブザーバをバックグラウンドに配置していない場合、オブザーバはSTART OBSERVERコマンドを発行したときに作成される継続的に実行されるプロセスです。そのため、別のDGMGRLセッションからSTOP OBSERVERコマンドを発行するまで、オブザーバ・コンピュータ上のコマンドライン・プロンプトは戻されません。コマンドを発行し、ブローカ構成と対話するには、別のDGMGRLクライアント・セッションにより接続する必要があります。

ファスト・スタート・フェイルオーバーを正しく構成したので、次の条件でフェイルオーバーをトリガーできます。

- データベースに障害が発生してすべてのデータベース・インスタンスが停止している
- I/Oエラーのためにデータ・ファイルがオフラインになっている
- オブザーバとスタンバイ・データベースのどちらも本番データベースへのネットワーク接続を失っており、スタンバイ・データベースが同期状態にあることが確認される
- ユーザー構成可能な条件

必要に応じて、ファスト・スタート・フェイルオーバーを起動できる次の条件を指定できます。これらのユーザー構成可能な条件はデフォルト値のままにし、自動フェイルオーバーを起動しないようにすることをお薦めします。

- データ・ファイルがオフライン(書込みエラー)
- ディクショナリが破損
- 制御ファイルが破損
- ログ・ファイルにアクセスできない
- スタック・アーカイバ
- ORA-240 (制御ファイル・エンキューのタイムアウト)

これらの条件のいずれかが検出されると、FastStartFailoverPmyShutdownの設定に関係なく、オブザーバがスタンバイにフェイルオーバーし、プライマリが停止します。ユーザー構成可能な条件の場合、ファスト・スタート・フェイルオーバーのしきい値は無視され、フェイルオーバーがすぐに続行されます。

複数のスタンバイ・データベースを使用するファスト・スタート・フェイルオーバー

FastStartFailoverTarget構成プロパティは、プロパティが設定されているデータベースがプライマリ・データベースの場合、ファスト・スタート・フェイルオーバー状態にあるときにターゲット・データベースとして動作できる1つ以上のスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。このような指定可能なターゲット・データベースは、ファスト・スタート・フェイルオーバー・ターゲット候補と呼びます。

FastStartFailoverTarget構成プロパティには、フィジカル・スタンバイの名前のみを設定できます。スナップショット・スタンバイ・データベース、遠隔同期インスタンスまたはZero Data Loss Recovery Applianceの名前は設定できません。

フィジカル・スタンバイ・データベースが1つのみ存在している場合、ブローカはそのデータベースを、ファスト・スタート・フェイルオーバーが有効化されたときに、プライマリ・データベースのこのプロパティのデフォルト値として選択します。複数のフィジカル・スタンバイ・データベースが存在する場合、ブローカはプロパティ定義内の指定順に基づいて1つを選択します。ファスト・スタート・フェイルオーバーが有効化されると、ターゲットが検証されます

ログ・バッファの最適な設定

非同期REDO転送でOracle Data Guardを使用する場合、LOG_BUFFERを256MB以上に設定します。

そうすることで、非同期REDO転送では、ログ・バッファからREDOを読み取り、オンラインREDOログへのディスクI/Oを避けることができます。REDO生成率が非常に高いワークロード(たとえば、データベース・インスタンス当たり50MB/秒を超える)の場合、LOG_BUFFERは、使用されているプラットフォームに許可されている最大値まで増やすことができます。

ノート:

Linux プラットフォームの最大 LOG_BUFFER 設定は 2GB で、Windows の最大設定は 1GB です。

送信および受信バッファ・サイズの設定

REDO転送プロセス(特に受信/スタンバイ側)では、通常、待機時間が長いネットワーク・パスでTCPソケット・バッファを増やすことでメリットが得られます。TCPソケット・バッファは、TCPスタックによって動的に管理できます。

tcp_rmemおよびtcp_wmemの最大値を設定すると、カーネルは必要に応じてソケットに割り当てられたバッファを動的に変更できます。

帯域幅遅延積(BDP)は、チャネルのネットワーク・リンク容量とラウンド・タイム(待機時間)の積です。ソケット・バッファ・サイズの最小推奨値は、特に待機時間の長い高帯域幅ネットワークの場合、3*BDPです。oratcptestを使用してソケット・バッファ・サイズをチューニングします。

rootとして、tcp_rmemおよびtcp_wmemの最大値を3*<帯域幅遅延積>に設定します。この例では、BDPは16MBです。これらのパラメータの他の2つの値は、システムに現在設定されているままにする必要があります。

```
# sysctl -w net.ipv4.tcp_rmem='4096 87380 16777216';
# sysctl -w net.ipv4.tcp_wmem='4096 16384 16777216';
```

sysctlを使用してこれらの値を変更すると、メモリー内でのみ動的に変更され、システムの再起動時に失われます。さらに、Linuxシステムではこれらの値を/etc/sysctl.confに設定します。ファイルに値がない場合は、これらのエントリを追加します。

```
net.ipv4.tcp_rmem='4096 87380 16777216'
net.ipv4.tcp_wmem='4096 16384 16777216'
```

同期トランスポートの場合にのみSDUサイズを65535に設定

Oracle Net Servicesでは、セッション・データ・ユニット(SDU)のサイズを調整してデータ転送を制御できます。オラクル社でのテストでは、SDUパラメータを最大値の65535に設定すると、同期トランスポートのパフォーマンスが向上しました。

SDUは、ローカル・ネーミング構成ファイル(tnsnames.ora)およびリスナー構成ファイル(listener.ora)のSDUパラメータを使用して接続単位ごとに設定するか、sqlnet.oraファイルのDEFAULT_SDU_SIZEプロファイル・パラメータですべてのOracle Net Services接続を対象に設定します。

オンラインREDOログの適切な構成

REDOログ・スイッチは、REDOの転送と適用のパフォーマンスに大きく影響します。プライマリ・データベースおよびスタンバイ・データベースのオンラインREDOログのサイズを設定するには、次のベスト・プラクティスに従います。

オンラインREDOログについては、次のガイドラインに従います。

- すべてのオンラインREDOログ・グループは、同じサイズのログ(バイト)を持つ必要があります。
- オンラインREDOログは、高パフォーマンスのディスク(DATAディスク・グループ)に存在する必要があります。
- Oracle RACインスタンスでREDOのスレッドごとに3つ以上のオンラインREDOログ・グループを作成します。
- Oracle RAC環境の共有ディスクにオンラインREDOログ・グループを作成します。
- 高冗長性ディスク・グループに配置されていないかぎり、オンラインREDOログを多重化します(ログ・グループごとに複数のメンバー)。
- ログ・スイッチが1時間に12回(5分ごと)以下になるようにオンラインREDOログのサイズを設定します。ほとんどの場合、ピーク時のワークロードであっても、15分から20分ごとのログ・スイッチが最適です。

REDOログ・サイズの設定

プライマリ・データベースのピークREDO生成率に基づいてREDOログ・サイズを設定します。

最大速度を確認するには、最大ワークロードを含む期間に次の問合せを実行します。最大速度は、月末、四半期末または年次で確認できます。REDO適用をこれらのワークロード中に一貫して実行するためには、最大速度を処理するようにREDOログ・サイズを設定します。

```
SQL> SELECT thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
first_time)*86400 sec,
 blocks*block_size/1024/1024)/((next_time-first_time)*86400) "MB/s"
 FROM v$archived_log WHERE ((next_time-first_time)*86400<>0) and first_time
 between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
 and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS') and dest_id=1 order by
first_time;
   THREAD#  SEQUENCE#         MB        SEC       MB/s
---------- ---------- ---------- ---------- ----------
         2       2291 29366.1963        831  35.338383
         1       2565 29365.6553        781 37.6000708
         2       2292 29359.3403        537  54.672887
         1       2566 29407.8296        813 36.1719921
         2       2293 29389.7012        678 43.3476418
         2       2294 29325.2217       1236 23.7259075
         1       2567 11407.3379       2658 4.29169973
         2       2295 29452.4648        477 61.7452093
         2       2296 29359.4458        954 30.7751004
         2       2297 29311.3638        586 50.0193921
         1       2568 3867.44092       5510 .701894903
```

次のチャートを使用して、ピーク生成率に基づいてREDOログ・サイズを選択します。

表14-2 推奨REDOログ・サイズ

ピークREDO率	推奨REDOログ・サイズ
<= 1 MB/s	1 GB
<= 5 MB/s	4 GB
<= 25 MB/s	16 GB
<= 50 MB/s	32 GB
> 50 MB/s	64 GB

スタンバイREDOログ・グループの使用

可用性およびパフォーマンスを向上させるために、すべてのプライマリ・データベースおよびスタンバイ・データベースでスタンバイREDOログ・グループを構成します。

REDOログ・スレッドごとに、スレッドがOracle RACデータベース・インスタンスに関連付けられます。スタンバイREDOログ・グループの数は、オンラインREDOログ・グループの数以上(>=)である必要があります。

スタンバイREDOログ・グループを作成する際には、次の追加のガイドラインを考慮してください。

- すべてのオンラインREDOログとスタンバイREDOログ・グループは、同じサイズのログ(バイト)を持つ必要があります。スタンバイREDOログは、オンラインREDOログと同じサイズでない場合には使用されません。
- すべてのスタンバイREDOログ・グループは、プライマリ・データベースとスタンバイ・データベースの両方で同じサイズのログ(バイト)を持つ必要があります。
- スタンバイREDOログは、高パフォーマンスのディスク(DATAディスク・グループ)に存在する必要があります。
- スタンバイREDOログは、高冗長性ディスク・グループに配置されていないかぎり、多重化する必要があります(ログ・グループごとに複数のメンバー)。Data Guardは欠落しているREDOをフェッチできるため、いずれの場合もスタンバイREDOログの多重化はオプションです。
- Oracle RAC環境では、スタンバイREDOログは共有ディスクに作成します。
- Oracle RAC環境では、各スタンバイREDOログ・グループにスレッドを割り当てます。

次の例では、REDOスレッドごとに3つのログ・グループを作成します。

```
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 1 GROUP 7 ('+DATA') SIZE 4194304000,
GROUP 8 ('+DATA') SIZE 4194304000, GROUP 9 ('+DATA') SIZE 4194304000;
SQL> ALTER DATABASE ADD STANDBY LOGFILE THREAD 2 GROUP 10 ('+DATA') SIZE 4194304000,
GROUP 11 ('+DATA') SIZE 4194304000, GROUP 12 ('+DATA') SIZE 419430400
```

オンラインREDOログのスレッド数およびグループ番号を確認するには、次のようにV$LOGビューを問い合せます。

```
SQL> SELECT * FROM V$LOG;
```

ALTER DATABASE ADD STANDBY LOGFILE THREAD文の結果を確認するには、次のようにV$STANDBY_LOGビューを問い合せます。

```
SQL> SELECT * FROM V$STANDBY_LOG;
```

データ破損の防止

Oracle Databaseの破損の防止、検出および修復の機能は、保護対象のデータおよびトランザクションの内部知識と、包括的な高可用性ソリューションのインテリジェント統合を基に構築されています。

データ破損が検出されると、Oracle Data Guard、ブロック・メディア・リカバリおよびデータ・ファイル・メディア・リカバリでデータをリカバリできます。人為的エラーまたはアプリケーション・エラーで引き起こされたデータベース全体の論理破損は、Oracle Flashbackテクノロジによって元に戻すことができます。

ツールは論理データ構造の予防的検証にも使用できます。たとえば、SQL*Plus ANALYZE TABLE文はブロック間の破損を検出します。

これらのベスト・プラクティスを使用して、最も包括的なデータ破損防止および検出を実現します。

- ブロック破損の拡大を防止するには、フィジカル・スタンバイ・データベースでOracle Data Guardを使用します。Oracle Data GuardはOracleデータをデータ損失および破損、書込みの欠落から保護する最適なソリューションです。
- 次の表に示すように、Data Guardプライマリ・データベースおよびスタンバイ・データベースでOracle Databaseブロック破損初期化パラメータを設定します。

  表14-3 ブロック破損初期化パラメータの設定

プライマリ・データベース上の設定...	スタンバイ・データベース上の設定...
DB_BLOCK_CHECKSUM=MEDIUM または FULL	DB_BLOCK_CHECKSUM=MEDIUM または FULL
DB_LOST_WRITE_PROTECT=TYPICAL	DB_LOST_WRITE_PROTECT=TYPICAL
DB_BLOCK_CHECKING=FALSE*	DB_BLOCK_CHECKING=MEDIUM または FULL

  * PRIMARYでDB_BLOCK_CHECKINGをMEDIUMまたはFULLに設定することをお薦めしますが、アプリケーションでパフォーマンスを完全に評価した後にのみ設定してください。

- パフォーマンス・オーバーヘッドはブロック変更ごとに発生するため、パフォーマンス・テストはDB_BLOCK_CHECKINGパラメータの設定時に特に重要です。プライマリ・データベースまたはスタンバイ・データベースのいずれかで、DB_BLOCK_CHECKING=MEDIUMを最小の設定(索引ブロックではなくデータ・ブロック上のブロック・チェック)にすることを強くお薦めします。DB_BLOCK_CHECKINGをMEDIUMまたはFULLにすることを有効化するパフォーマンス・オーバーヘッドがプライマリ・データベースで許容されない場合、スタンバイ・データベースでDB_BLOCK_CHECKINGをMEDIUMまたはFULLに設定します。

次の推奨事項も、データ破損からの保護に役立ちます。

- ディスク障害からの保護にディスクのミラー化を提供する場合は、Oracle Automatic Storage Management (Oracle ASM)を使用します。
- 最適な破損の修復にはOracle ASM HIGH REDUNDANCYを使用します。ディスク・グループでOracle ASM冗長性を使用すると、I/Oエラーまたは破損の発生時にミラー・エクステントをデータベースで使用できます。継続的な保護を目的として、Oracle ASMの冗長性により、I/Oエラーの発生時にエクステントをディスク上の別の領域に移動する機能が提供されます。Oracle ASMによる冗長性メカニズムは、メディア・エラーを戻す不良セクターがある場合に役立ちます。

- (ほとんどの場合、人為的エラーによって引き起こされた)論理破損からの高速ポイント・イン・タイム・リカバリと、フェイルオーバー後のプライマリ・データベースの高速復元には、Oracle Flashbackテクノロジを有効化します。
- バックアップおよびリストア操作中の追加的なブロック・チェックには、RMANを使用します。Recovery Manager (RMAN)によるバックアップおよびリカバリ計画を実装し、RMANのBACKUP VALIDATE CHECK LOGICALスキャンを定期的に使用して破損を検出します。
- 破損のチェックと修復、中央バックアップ検証、本番データベースへの影響の縮小、Enterprise Cloudバックアップおよびリカバリ・ソリューションを含め、バックアップおよびリカバリの検証にはZero Data Loss Recovery Applianceを使用します。

フェイルオーバー後の復元のためのフラッシュバック・データベースの使用

プライマリ・データベースとスタンバイ・データベースの両方でフラッシュバック・データベースを有効化すると、元のプライマリ・データベースが損害を受けていなかった場合に、フェイルオーバー後に元のプライマリ・データベースを新しいスタンバイ・データベースとして復元できます。

フラッシュバック・データベースが有効であれば、スイッチオーバー・プロセスで障害が発生した場合でも、そのプロセスを簡単に元に戻すことができます。

スタンバイ・データベースでは、DB_FLASHBACK_RETENTION_TARGETをプライマリと同じ値に設定します。DB_FLASHBACK_RETENTION_TARGET初期化パラメータを、次のいずれかの適用される条件で指定されている最大値に設定します。

- Data Guardフェイルオーバー後にフラッシュバック・データベースを利用して障害が発生したプライマリ・データベースを復元するには、ほとんどの場合、DB_FLASHBACK_RETENTION_TARGETを120 (分)以上に設定して、障害が発生したプライマリの復元を有効化します。
- ユーザー・エラーまたは論理破損からの高速ポイント・イン・タイム・リカバリにフラッシュバック・データベースを使用中の場合、DB_FLASHBACK_RETENTION_TARGETを、データベースのリカバリを行う先の過去の最も遠い時間と等しい値に設定します。論理破損を24時間以内に検出して修復できる場合は、DB_FLASHBACK_RETENTION_TARGETを1440 (分)以上に設定します。

強制ロギング・モードの使用

プライマリ・データベースをFORCE LOGGINGモードで運用すると、データベースのすべてのデータ変更がログに記録されます。FORCE LOGGINGモードにより、スタンバイ・データベースとプライマリ・データベースとの一貫性が維持されます。

NOLOGGING操作でロード・パフォーマンスが必要なためにこのモードを使用できない場合は、適切なロギング・モードの有効化でその他のオプションを参照してください。

ALTER DATABASE FORCE LOGGING文を発行すると、強制ロギングを即座に有効化できます。FORCE LOGGINGを指定した場合、Oracleはログに記録されていない進行中のすべての操作が終了するまで待機します。

バインドされたRTOおよびRPOへのファスト・スタート・フェイルオーバーの構成(MAA Gold要件)

プライマリ・データベース、クラスタまたはサイトに障害が発生した場合に厳しいRTO要件を満たすためには、ファスト・スタート・フェイルオーバーを有効にする必要があります。Data Guardのファスト・スタート・フェイルオーバーでは、Data Guardオブザーバが2つのクォーラムを提供し、データベースの一貫性を維持し、データベースのスプリット・ブレインを防止します。

ファスト・スタート・フェイルオーバーでは、プライマリ・データベースが消失した場合にData Guardブローカにより、事前に選択されたスタンバイ・データベースに自動的にフェイルオーバーできます。ファスト・スタート・フェイルオーバーでは、ターゲット・スタンバイ・データベースが迅速かつ確実に、プライマリ・データベース・ロールに切り替えられます。このとき、手動ステップを実行してフェイルオーバーを起動する必要はありません。ファスト・スタート・フェイルオーバーは、Data Guardブローカ構成でのみ使用できます。

プライマリ・データベースに複数のスタンバイ・データベースがある場合、FastStartFailoverTargetプロパティを使用して複数のファスト・スタート・フェイルオーバー・ターゲットを指定できます。このようなターゲットはターゲット候補と呼びます。ブローカは、FastStartFailoverTargetプロパティでの指定順に基づいて、ターゲットを選択します。指定されたファスト・スタート・フェイルオーバー・ターゲットに問題が発生し、フェイルオーバーのターゲットにできない場合、ブローカはファスト・スタート・フェイルオーバー・ターゲットを別のターゲット候補のいずれかに自動的に変更します。

ファスト・スタート・フェイルオーバーでは、任意のData Guard保護モードを使用できます。最大保護モードおよび最大可用性モードは、データが消失しないことを保証する自動フェイルオーバー環境を提供します。最大パフォーマンス・モードによる自動フェイルオーバー環境では、消失データがFastStartFailoverLagLimit構成プロパティで指定したデータ量(秒単位)以下であることが保証されます。このプロパティは、自動フェイルオーバーの実行に許容される最大消失データ量を示します。このプロパティは、ファスト・スタート・フェイルオーバーが有効化され、構成が最大パフォーマンス・モードで動作している場合にのみ使用されます。

1. 次の例に示すように、FastStartFailoverThresholdプロパティを設定することで、プライマリ・データベースを使用できないことが検出されてからフェイルオーバーを開始するまでにオブザーバおよびターゲット・スタンバイ・データベースが何秒待機するかを指定します。

   ```
   DGMGRL> EDIT CONFIGURATION SET PROPERTY FastStartFailoverThreshold = seconds;
   ```

   ファスト・スタート・フェイルオーバーが発生するのは、オブザーバとスタンバイ・データベースの両方が本番データベースとの接続を失ってからFastStartFailoverThresholdに設定されている値を超える期間が経過したときと、構成の状態が同期されていること(最大可用性モード)またはラグが構成済のFastStartFailoverLagLimitを超えないこと(最大パフォーマンス・モード)に両者が同意したときです。

   FastStartFailoverThresholdの最適な値は、可能なかぎり迅速にフェイルオーバー(停止時間を最小化)することと、一時ネットワークの不規則性や可用性に影響しないその他の短期間のイベントが原因でフェイルオーバーが不必要にトリガーされることとのトレードオフを考慮して判断します。

   FastStartFailoverThresholdのデフォルト値は30秒です。

   次の表は、様々なユースケースにおけるFastStartFailoverThresholdの推奨設定を示しています。

構成	最小値の推奨設定
単一インスタンスのプライマリ、短い待機時間、信頼性の高いネットワーク	15 秒
単一インスタンスのプライマリ、WAN を介した待機時間の長いネットワーク	30 秒
Oracle RAC のプライマリ	Oracle RAC ミスカウント + 再構成時間 + 30 秒 Exadata システムの場合、最小設定は

構成	最小値の推奨設定
	30 秒

2. トポロジ内でオブザーバを配置する場所を決定します。

   Data Guard Brokerオブザーバは2つのクォーラムを提供し、データベースの一貫性を維持し、スプリット・ブレインを防止します。Data Guardファスト・スタート・フェイルオーバーでは、常に1つのプライマリ・データベースのみが存在することが保証され、トランザクションをプライマリ・データベースにルーティングすることで外部の一貫性が保証されます。独自の可用性ドメイン(AD)またはデータ・センター内にあるプライマリ、スタンバイおよびオブザーバとともにファスト・スタート・フェイルオーバーをデプロイするのが理想的な状態ですが、2つの可用性ドメインまたは単一の可用性ドメインのみを使用する構成をサポートする必要があります。次に、オブザーバを配置する場合の推奨事項を2つのユースケースで示します。

   - デプロイメント構成1: リージョンが2つで、それぞれのリージョンにADが2つあります。
     - 初期プライマリ・リージョンでは、プライマリ・データベースがAD1にあり、高可用性オブザーバが2つあります(1つ目のオブザーバはAD2に、2つ目のHAオブザーバはAD1にあります)
     - 初期スタンバイ・リージョンでは、スタンバイ・データベースがAD1にあり、ロール変更後に使用される高可用性オブザーバが2つあります(1つ目のオブザーバはAD2に、2つ目のHAオブザーバはAD1にあります)
     - オブザーバの場合、MAAでは少なくとも2つのオブザーバ・ターゲットを同じプライマリ・リージョンの異なるADに存在させることをお薦めします
   - デプロイメント構成2: リージョンが2つで、それぞれのリージョンにADが1つのみあります
     - 初期プライマリ・リージョンには、プライマリ・データベースと、オブザーバをホストする2つの軽量サーバーがあります
     - 初期スタンバイ・リージョンには、スタンバイ・データベースと、オブザーバをホストする2つの軽量サーバーがあります(ロール変更がある場合)

3. オブザーバ高可用性を構成します。

   1つのData Guard Broker構成を監視するには、最大で3つのオブザーバを登録できます。各オブザーバは、START OBSERVERコマンドの発行時に指定する名前で識別されます。オブザーバをバックグラウンド・プロセスとして起動することもできます。

```
DGMGRL> sys@boston
Enter password:
DGMGRL> start observer number_one in background;
```

   高可用性のために追加のオブザーバを同じホストまたは別のホストで起動できます。

```
DGMGRL> sys@boston
Enter password:
DGMGRL> start observer number_two in background;
```

   Data Guard Brokerでファスト・スタート・フェイルオーバーを調整できるのはプライマリ・オブザーバのみです。他の登録済オブザーバはすべてバックアップ・オブザーバとみなされます。

   オブザーバをバックグラウンドに配置していない場合、オブザーバはSTART OBSERVERコマンドを発行したときに作成されて継続的に実行されるプロセスです。そのため、別のDGMGRLセッションからSTOP OBSERVERコマンドを発行するまで、オブザーバ・コンピュータ上のコマンドライン・プロンプトは戻されません。コマンドを発行し、ブローカ構成と対話する

には、別のDGMGRLクライアント・セッションにより接続する必要があります。

ファスト・スタート・フェイルオーバーのトリガー

ファスト・スタート・フェイルオーバーを正しく構成したので、次の条件でフェイルオーバーをトリガーできます。

- データベースに障害が発生してすべてのデータベース・インスタンスが停止している
- I/Oエラーのためデータ・ファイルがオフラインになっている
- オブザーバとスタンバイ・データベースのどちらも本番データベースへのネットワーク接続を失っており、スタンバイ・データベースが同期状態にあることが確認される
- ユーザー構成可能な条件

必要に応じて、ファスト・スタート・フェイルオーバーを起動できる次の条件を指定できます。これらのユーザー構成可能な条件はデフォルト値のままにし、自動フェイルオーバーを起動しないようにすることをお薦めします。

- データファイルがオフライン(書込みエラー)
- ディクショナリが破損
- 破損した制御ファイル
- アクセス不可能なログ・ファイル
- スタック・アーカイバ
- ORA-240 (制御ファイル・エンキューのタイムアウト)

これらの条件のいずれかが検出されると、FastStartFailoverPmyShutdownの設定に関係なく、オブザーバがスタンバイにフェイルオーバーし、プライマリが停止します。ユーザー構成可能な条件の場合、ファスト・スタート・フェイルオーバーのしきい値は無視され、フェイルオーバーがすぐに続行されます。

複数のスタンバイ・データベースを使用するファスト・スタート・フェイルオーバー

FastStartFailoverTarget構成プロパティは、プロパティが設定されているデータベースがプライマリ・データベースの場合、ファスト・スタート・フェイルオーバー状態にあるときにターゲット・データベースとして動作できる1つ以上のスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。このような指定可能なターゲット・データベースは、ファスト・スタート・フェイルオーバー・ターゲット候補と呼びます。

FastStartFailoverTarget構成プロパティには、フィジカル・スタンバイの名前のみを設定できます。スナップショット・スタンバイ・データベース、遠隔同期インスタンスまたはZero Data Loss Recovery Applianceの名前は設定できません。

フィジカル・スタンバイ・データベースが1つのみ存在している場合、ブローカはそのデータベースを、ファスト・スタート・フェイルオーバーが有効化されたときに、プライマリ・データベースのこのプロパティのデフォルト値として選択します。複数のフィジカル・スタンバイ・データベースが存在する場合、ブローカはプロパティ定義内の指定順に基づいて1つを選択します。ファスト・スタート・フェイルオーバーが有効化されると、ターゲットが検証されます

スタンバイAWRの構成

Oracle Database 12c (12.2)以降、自動ワークロード・リポジトリ(AWR)のスナップショットはスタンバイ・データベースから取得できます。

スタンバイAWRは、Active Data Guardスタンバイ・データベースのリカバリおよびレポートのワークロードに関するパフォーマンスの問題を特定するための最適なツールです。My Oracle Supportノート『How to Generate AWRs in Active Data Guard Standby Databases』(ドキュメントID 2409808.1)に従って、スタンバイAWRを構成し、レポートを生成します。

複数のスタンバイ・データベースの構成

複数のスタンバイ・データベースが含まれるOracle Data Guard構成には、ローカル・スタンバイ・データベースとリモート・スタンバイ・データベースの両方の利点があります。

ローカル・スタンバイ・データベースでは、データ損失ゼロのフェイルオーバーを実現し、アプリケーションの停止時間を数秒に短縮できます。リージョン障害が発生し、プライマリ・スタンバイ・システムとローカル・スタンバイ・システムにアクセスできなくなった場合、アプリケーションとデータベースはリモート・スタンバイにフェイルオーバーできます。複数スタンバイ構成の特徴および利点の詳細は、Gold: 複数のスタンバイ・データベースを参照してください。

複数のスタンバイ・データベースが含まれるOracle Data Guard構成の管理

Oracle Data Guard Brokerにより、Oracle Data Guard構成内の複数のデータベースにまたがる管理タスクおよび運用タスクが自動化されます。また、ブローカにより、単一のOracle Data Guard構成内のすべてのシステムが監視されます。

マルチメンバーData Guard構成では、次のREDO転送先がサポートされています。

- Oracle Data Guardスタンバイ・データベース
- 遠隔同期インスタンス(詳細は、遠隔同期インスタンスの使用を参照)
- Oracle Streamsダウンストリーム取得データベース
- Zero Data Loss Recovery Appliance (リカバリ・アプライアンス)

複数のスタンバイ・データベースとREDOルート

Oracle Data Guard BrokerのRedoRoutesプロパティを使用すると、デフォルトの動作(プライマリ・データベースが生成したREDOを構成内にある他のすべてのREDO転送先に送信する)をオーバーライドできます。

デフォルト以外のREDO転送トポロジの例として、フィジカル・スタンバイまたは遠隔同期インスタンスが、プライマリ・データベースから受け取ったREDOを1つ以上の宛先に転送するものや、指定された宛先に対して使用されるREDO転送モードが、どのデータベースがプライマリ・ロールであるかによって異なるものがあります。

プライマリ・データベース(North_Sales)と2つのフィジカル・スタンバイ・データベース(Local_SalesとRemote_Sales)がある構成を考えます。Local_Salesデータベースは、高可用性とより単純なアプリケーションおよびデータベースのフェイルオーバーのために、プライマリと同じデータ・センターに置かれています。Remote_Salesデータベースは、障害回復目的で、リモート・データセンターに置かれています。

North_Salesから両方のデータベースにREDOを送信するのではなく、RedoRoutesブローカ・プロパティを使用して、ローカルのフィジカル・スタンバイ・データベースが、North_Salesから受け取ったREDOをRemote_Salesに転送するように、リアルタイム・カスケーディングを構成することができます。そのためには、North_SalesおよびLocal_SalesでRedoRoutesプロパティを次のように設定します。

- North_Salesデータベースでは、North_Salesがプライマリ・ロールである場合、同期転送モードを使用してLocal_SalesデータベースにREDOを送信するように、RedoRoutesプロパティを指定します。このルールにより、プライマリがRemote_Salesデータベースに直接REDOデータを送信することがなくなります。
- Local_Salesデータベースでは、North_Salesがプライマリ・ロールの場合、Local_Salesが、North_Salesから受け取ったREDOをRemote_Salesに転送するように、RedoRoutesプロパティを指定する必要があります。

実行時のRedoRoutes構成を表示するには、SHOW CONFIGURATIONコマンドを使用します。たとえば:

```
DGMGRL> SHOW CONFIGURATION;
Configuration - Sales_Configuration
  Protection Mode: MaxAvailability
```

```
  Members:
 North_Sales  - Primary database
    Local_Sales  - Physical standby database
      Remote_Sales  - Physical standby database (receiving current redo)
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

Remote_Sales宛先のREDOルート・ルールで、非同期REDO転送属性が明示的に指定されることで、その宛先へのREDOのリアルタイム・カスケーディングが有効にされたことに注意してください。(リアルタイム・カスケードにはOracle Active Data Guardオプションのライセンスが必要です。)

REDOのリアルタイム・カスケーディングを無効にするには、非同期REDO転送属性を指定しないでください。たとえば:

```
DGMGRL> EDIT DATABASE 'Local_Sales' SET PROPERTY 'RedoRoutes' = '(North_Sales :
Remote_Sales)';
```

詳細は、RedoRoutesを参照してください。

リモート代替宛先に対するRedoRoutesプロパティの使用

RedoRoutesプロパティを使用すると、REDOデータの受信元のメンバーに障害が発生しても端末メンバーがREDOデータを受信できるように、リモート代替宛先を設定することができます。

前述の例を使用すると、プライマリ・データベースとしてNorth_Salesを使用し、スタンバイ・データベースLocal_Salesに障害が発生した場合にREDOデータを直接Remote_Salesに送信できます。PRIORITY属性を使用して、Local_Salesの障害が解決された時点でRemote_SalesへのREDOの送信を再開できるように指定することもできます。

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY
 'RedoRoutes' = '(LOCAL : ( Local_Sales ASYNC PRIORITY=1, Remote_Sales ASYNC
PRIORITY=2 ))';
Property "RedoRoutes" updated
DGMGRL> EDIT DATABASE 'Local_Sales'
 SET PROPERTY 'RedoRoutes' = '(North_Sales : Remote_Sales ASYNC)';
Property "RedoRoutes" updated
DGMGRL> SHOW CONFIGURATION;
Configuration - Sales_Configuration
  Protection Mode: MaxPerformance
  Members:
  North_Sales    - Primary database
    Local_Sales  - Physical standby database
      Remote_Sales - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

RedoRoutes構成全部を表示するには、SHOW CONFIGURATION VERBOSEコマンドを使用します。たとえば:

```
DGMGRL> SHOW CONFIGURATION VERBOSE;
Configuration - Sales_Configuration
  Protection Mode: MaxPerformance
  Members:
    North_Sales - Primary database
      Local_Sales - Physical standby database
        Remote_Sales - Physical standby database
      Remote_Sales - Physical standby database (alternate of Local_Sales)
  Properties:
    FastStartFailoverThreshold      = '180'
    OperationTimeout                = '30'
    TraceLevel                      = 'USER'
    FastStartFailoverLagLimit       = '300'
    CommunicationTimeout            = '180'
```

```
    ObserverReconnect               = '0'
    FastStartFailoverAutoReinstate  = 'TRUE'
    FastStartFailoverPmyShutdown    = 'TRUE'
    BystandersFollowRoleChange      = 'ALL'
    ObserverOverride                = 'FALSE'
    ExternalDestination1            = ''
    ExternalDestination2            = ''
    PrimaryLostWriteAction          = 'CONTINUE'
    ConfigurationWideServiceName    = 'c0_CFG'
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

複数のスタンバイ・データベースを使用するファスト・スタート・フェイルオーバー

Oracle Data GuardのFastStartFailoverTargetブローカ構成プロパティは、プロパティが設定されているデータベースがプライマリ・データベースの場合、ファスト・スタート・フェイルオーバーのシナリオにおいてターゲット・データベースとして動作できる1つ以上のスタンバイ・データベースのDB_UNIQUE_NAMEを指定します。

このような指定可能なターゲット・データベースは、ファスト・スタート・フェイルオーバー・ターゲット候補と呼びます。FastStartFailoverTargetプロパティには、フィジカル・スタンバイの名前のみを設定できます。スナップショット・スタンバイ・データベース、遠隔同期インスタンスまたはZero Data Loss Recovery Applianceの名前は設定できません。

フィジカル・スタンバイ・データベースが1つのみ存在している場合、ファスト・スタート・フェイルオーバーが有効になっていると、ブローカはそのデータベースをプライマリ・データベースのFastStartFailoverTargetのデフォルト値として選択します。複数のフィジカル・スタンバイ・データベースが存在する場合、ブローカはプロパティ定義内の指定順に基づいてスタンバイを1つ選択します。ファスト・スタート・フェイルオーバーが有効になっていると、ターゲットは検証されます。

FastStartFailoverTargetも参照してください。

FastStartFailoverTargetの設定

2つ以上のスタンバイ・データベースがある場合、プライマリ・データベースのFastStartFailoverTarget構成プロパティを設定して目的のファスト・スタート・フェイルオーバーのターゲット・スタンバイ・データベースを指定します。

ファスト・スタート・フェイルオーバーが実際に有効になると、Oracle Data Guard Brokerにより、今度は反対にターゲット・スタンバイ・データベースのこのプロパティが将来のターゲット・スタンバイ・データベースとしてプライマリ・データベースを指すように設定されます。このプロパティは自動的に設定されるため、ターゲット・スタンバイ・データベースでこのプロパティを設定する必要はありません。たとえば:

```
DGMGRL> edit database moe set property ='curly,larry';
Property "faststartfailovertarget" updated
```

FastStartFailoverTargetの構成後、ファスト・スタート・フェイルオーバーの有効化に進みます。ファスト・スタート・フェイルオーバーが有効になっていると、プライマリ・データベースまたはターゲット・スタンバイ・データベースのFastStartFailoverTarget構成プロパティは変更できません。

FastStartFailoverTargetプロパティを変更して別のスタンバイ・データベースを指すようにするには、ファスト・スタート・フェイルオーバーを無効化し、FastStartFailoverTargetプロパティを設定し、さらにファスト・スタート・フェイルオーバーを再有効化してください。このアクションは、プライマリ・データベースまたはスタンバイ・データベースの可用性や稼働時間には影響しません。

FastStartFailoverTargetを設定したスイッチオーバー

FastStartFailoverTargetを設定してファスト・スタート・フェイルオーバーが有効になっている場合でも、プライマリ・デー

タベースのFastStartFailoverTargetデータベース・プロパティに指定されたのと同じスタンバイ・データベースがロール変更の対象となるかぎり、スイッチオーバーまたは手動フェイルオーバーを実行できます。

ファスト・スタート・フェイルオーバー・ターゲットではないスタンバイにスイッチオーバーしようとすると、ORA-16655が発生します。

```
DGMGRL> switchover to curly
Performing switchover NOW, please wait...
Error: ORA-16655: specified standby database not the current fast-start failover
target standby
```

プライマリ・ファスト・スタート・ターゲットではないスタンバイにスイッチオーバーするには、次のようにします。

1. ファスト・スタート・フェイルオーバーを無効にします。

   ```
   DGMGRL> DISABLE FAST_START FAILOVER;
   ```

2. FastStartFailoverTargetプロパティを編集して、最初にスイッチオーバー先となるスタンバイをリストします。

   ```
   DGMGRL> edit database moe set property FastStartFailoverTarget='curly,larry';
   Property "faststartfailovertarget" updated
   ```

3. ファスト・スタート・フェイルオーバーを有効にします。

   ```
   DGMGRL> ENABLE FAST_START FAILOVER;
   ```

4. スイッチオーバー操作を実行します。

   ```
   DGMGRL> switchover to curly
   Performing switchover NOW, please wait...
   ```

ファスト・スタート・フェイルオーバーの停止処理

スタンバイ・データベースまたはインスタンスが停止しているか、REDOの転送に問題があるなどの理由で、プライマリ・データベースのファスト・スタート・フェイルオーバーのターゲット・スタンバイ・データベースが使用できなくなった場合、プライマリのファスト・スタート・フェイルオーバー・ターゲットはFastStartFailoverTargetプロパティで構成されている次のターゲットに自動的に切り替えられます。

ターゲット・スイッチを有効にするために、複数のpingサイクルが必要になる場合があります: 1つは現在のターゲットが有効でないことを認識するためのpingで、もう1つはターゲット・スイッチを提示してファイナライズするためのpingです。

元のファスト・スタート・フェイルオーバー・ターゲットがオンラインに戻っても、元のターゲットへの切替えは自動的には実行されません。停止後に元のターゲットに戻すには、ファスト・スタート・フェイルオーバーを無効にしてから有効にする必要があります。

Oracle Active Data Guard遠隔同期ソリューション

データ損失ゼロをサポートするために、プライマリ・データベースとスタンバイ・データベースの間にOracle Data Guard遠隔同期インスタンスをデプロイできます。これは、プライマリ・データベースからREDOを受け取り、そのREDOをOracle Data Guard構成の他のメンバーに送信するリモートのOracle Data Guardの宛先です。

遠隔同期について

遠隔同期は、データ損失ゼロの保護を実装する必要がある場合に、障害時リカバリ・サイトの場所の柔軟性を高めることができるOracle Active Data Guard機能です。

Oracle Data Guard同期トランスポートをすでにデプロイしているユーザーであっても、現在のスタンバイよりもプライマリに近い遠隔同期インスタンスを構成することで、本番データベースへのパフォーマンスの影響を軽減できます。

WANのディスタンスを超える、またはパフォーマンスが低いネットワークでの同期REDO転送では、プライマリ・データベースのパフォーマンスに対する影響が大きすぎてデータ損失ゼロの保護をサポートできないことがよくあります。Oracle Active Data Guard遠隔同期は、2つ目のスタンバイ・データベースまたは複雑な操作を必要とせずに、リモート・スタンバイ・データベースへのデータ損失ゼロのフェイルオーバーを実行する機能です。

遠隔同期を使用すると、同期REDO転送用のプライマリの許容範囲内である距離に遠隔同期インスタンス(軽量Oracleインスタンス)をデプロイすることで、これが可能になります。遠隔同期インスタンスでは、同期転送を使用してプライマリからREDOを受信し、非同期転送を使用して最大29個のリモート・スタンバイ・データベースにREDOを転送します。

遠隔同期インスタンスはOracle Active Data Guard遠隔同期機能の一部で、Oracle Active Data Guardのライセンスが必要です。

遠隔同期インスタンスへのオフロード

遠隔同期インスタンスは、リモート・スタンバイ・データベースで受信したREDOのギャップを解消するオーバーヘッドを(たとえば、ネットワークまたはスタンバイ・データベースの停止後に)プライマリからオフロードし、プライマリ・データベースのパフォーマンスに影響を与えずにREDO転送圧縮を実行してWAN帯域幅を節約できます。

REDO圧縮では、Advanced Compression Optionのライセンスが必要です。

さらに、REDO転送暗号化も遠隔同期インスタンスにオフロードできます。Advanced Security Option (ASO)を含め、MAAテスト時の暗号化は、プライマリのパフォーマンスにもスタンバイ・データベースの有効性にも影響しないことがわかっています。

Oracle NetおよびOracle Data Guardとテストおよび統合されているため、暗号化にはASOを使用することをお薦めします。

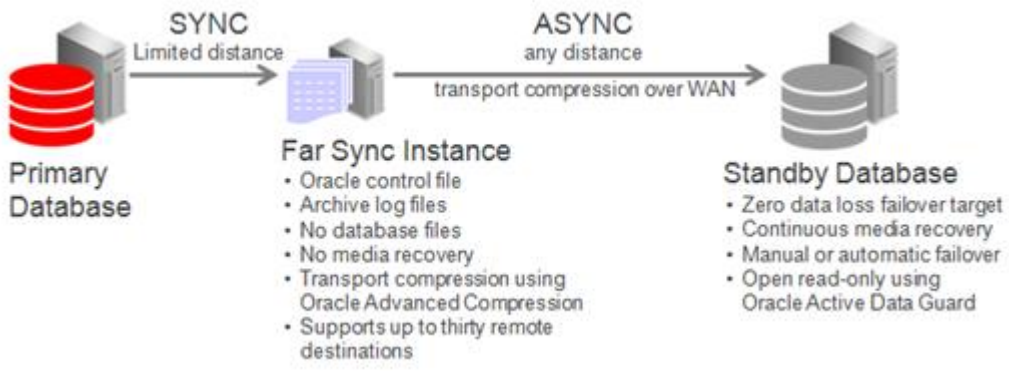
Oracle Advanced Security Optionはライセンス供与されたオプションです。

遠隔同期デプロイメント・トポロジ

Oracle Active Data Guard遠隔同期は、2つ目のスタンバイ・データベースまたは複雑な操作を必要とせずに、リモート・スタンバイ・データベースへのデータ損失ゼロのフェイルオーバーを実行する機能です。

Data Guardでこれを可能にするには、遠隔同期インスタンス(制御ファイル、SPFILE、パスワード・ファイルおよびスタンバイ・ログ・ファイルのみを含む軽量Oracleインスタンス。データベース・ファイルまたはオンラインREDOログはありません)を同期転送のプライマリの許容範囲内にある距離にデプロイします。遠隔同期インスタンスでは、同期転送を使用してプライマリからREDOを受信し、非同期転送を使用して最大29個のリモート・スタンバイ・データベースにREDOをすぐに転送します。遠隔同期インスタンスでは、REDOを新しいOracle Databaseバックアップ、ロギングおよびリカバリ・アプライアンスに転送することもできます。

図14-1 遠隔同期アーキテクチャの概要



次のユースケースは、遠隔同期インスタンスで実装できる様々なアーキテクチャの選択肢の利点を示しています。

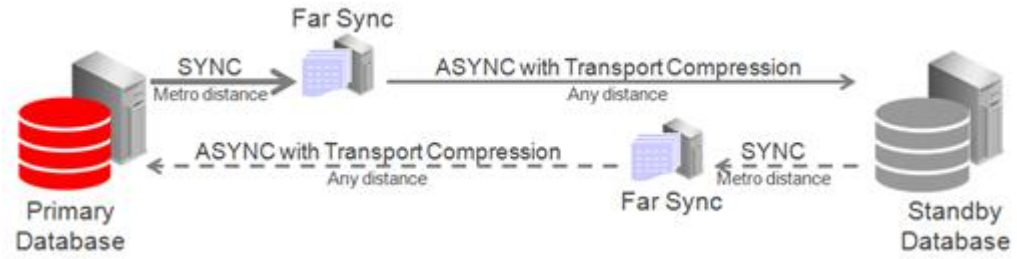
ケース1: ロール・トランジション後のデータ損失ゼロの保護

これは、プライマリ・データベースが高可用性遠隔同期インスタンスを使用して、データ損失ゼロのフェイルオーバーをリモート・スタンバイ・データベースに拡張する最も基本的な例です。

高可用性遠隔同期インスタンスは、サイト障害から分離するためにプライマリ・データベースとは別の場所にデプロイすることが理想的ですが、大都市地域の距離内(ネットワークRTT 5 ms以下、パフォーマンス・テストの対象)にします。別の場所を確保できない場合でも、遠隔同期インスタンスを同じデータ・センター内にデプロイすると、サイト全体の障害を除くすべてのリカバリ不能な停止に対して高速でデータ損失ゼロのフェイルオーバーを実現できるという利点があります。

スタンバイ・データベースがスタンバイ・ロールである間、リモート高可用性遠隔同期インスタンスはアイドル状態になります。これは、スタンバイ・データベースがプライマリ・データベース・ロールに遷移するとアクティブになって、新しいスタンバイ(古いプライマリ)へのデータ損失ゼロのフェイルオーバーが可能になります。元のプライマリ・データベースに対してローカルな高可用性遠隔同期インスタンスは、スタンバイ・ロールのときには非アクティブになります。

図14-2 遠隔同期によって容易になるロール・トランジション



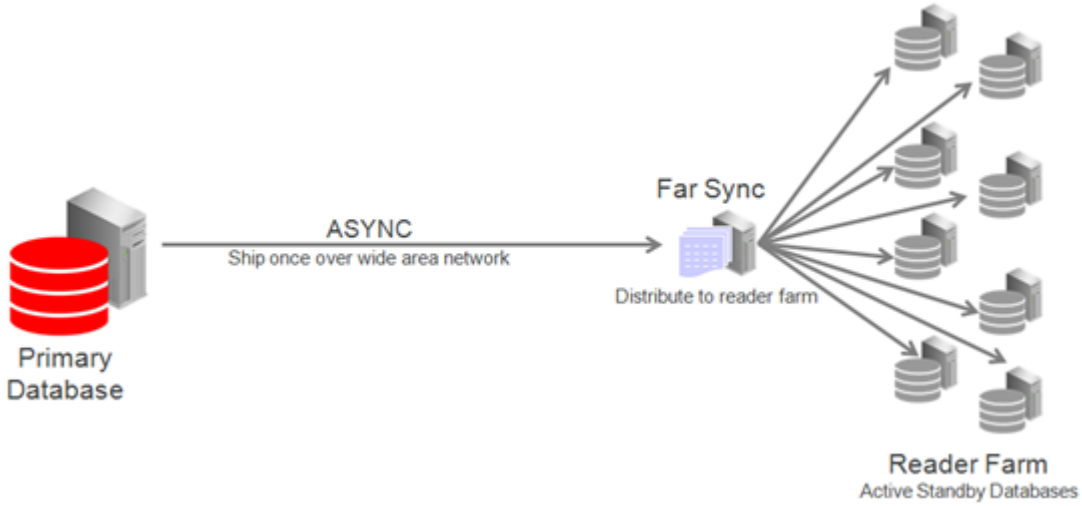
高可用性遠隔同期オプションの詳細は、「遠隔同期インスタンスの高可用性トポロジ」を参照してください。

ケース2: リーダー・ファーム・サポート

遠隔同期は、最大30のリモート宛先をサポートできるため、リーダー・ファームをサポートするための非常に便利なツールとなります。Active Data Guard構成では、複数のアクティブ・スタンバイ・データベースを設定して、読取りパフォーマンスを容易に拡張できます。

この例では、リーダー・ファームはプライマリ・データベースのリモートの場所に構成されます。プライマリはいったんWAN経由でリモート宛先にある遠隔同期インスタンスに送信され、遠隔同期はリーダー・ファーム内のすべてのアクティブ・スタンバイ・データベースにREDOをローカルに分散します。

図14-3 遠隔同期によるリーダー・ファームへのREDOの送信



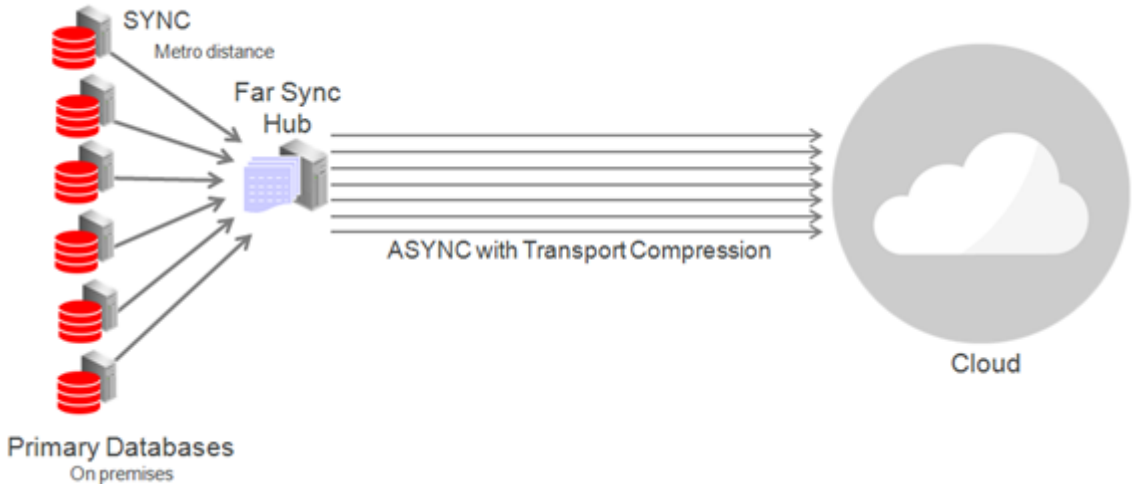
ケース3: 遠隔同期ハブを使用したクラウド・デプロイメント

遠隔同期は、非常に軽量なプロセスです。単一の物理サーバーで複数の遠隔同期インスタンスをサポートでき、各インスタンスがリモート宛先へのデータ損失ゼロのフェイルオーバーを実現します。

次の図では、プライマリ・データベースが単一の物理マシンに送信され、そのマシンは複数の遠隔同期インスタンスで構成された遠隔同期ハブとして動作しています。プライマリおよび遠隔同期ハブはオンプレミスですが、スタンバイ・データベースはクラウドにリモートでデプロイされています。

この構成のすべてのシステム(プライマリ・データベース・ホスト、スタンバイ・データベース・ホストおよび遠隔同期インスタンス・ホスト)は、Data Guard Support for Heterogeneous Primary and Physical Standbys in Same Data Guard Configuration (Doc ID 413484.1)で説明されているように、Data Guard構成で互換性の確保に必要な通常の要件を満たしている必要があります。

図14-4 遠隔同期ハブのアーキテクチャ



遠隔同期高可用性トポロジ

遠隔同期インスタンスの可用性を高く保つには、次のデプロイメント・トポロジを考慮してください。

Oracle Real Application Clustersでの遠隔同期インスタンスのデプロイ

遠隔同期インスタンスは、Oracle RACクラスタに配置できます。この構成では、遠隔同期インスタンスは一度に1つのサーバーでのみアクティブになり、他のサーバーは高可用性のための自動フェイルオーバーを提供します。この方法の特徴は次のとおりです。

- アクティブな遠隔同期インスタンスまたはノードに障害が発生した場合、データ損失の可能性が最も低く、ブラウンアウトします。
- 遠隔同期インスタンス障害後、データ損失ゼロの保護を短時間で再開できます。
- このソリューション単独で、クラスタ障害には対処しません。

最重要アプリケーションには、互いの代替として構成されて異なる場所にデプロイされたOracle RAC遠隔同期インスタンスのペアが適しています。これにより、(インスタンス、ノード、クラスタおよびサイトの停止中に)最も堅牢なHAおよびデータ保護を実現できます。

代替宛先および複数の遠隔同期インスタンスでの遠隔同期インスタンスのデプロイ

2つの別個の遠隔同期インスタンスを異なる物理マシンで構成し、それぞれが他方の代替宛先として機能して、非Oracle RAC環境での遠隔同期インスタンスの高可用性を実現します。プライマリ・データベースで定義された各宛先には

ALTERNATEキーワードが含まれ、他方の遠隔同期インスタンスを代替として割り当てています。アクティブな遠隔同期インスタンスがエラー状態に入ると、代替遠隔同期インスタンスを指している代替宛先は自動で有効になります。遠隔同期インスタンスを代替宛先として定義することで、一時的に再同期化状態に落ちながら新しい宛先が用意されると、最大可用性保護が維持されます。

この方法の特徴は次のとおりです。

- 遠隔同期インスタンス転送障害(インスタンスまたはネットワークの停止)後、データ損失ゼロの適用範囲が維持されます。
- 障害テストから、次のことがわかっています。
    - 遠隔同期インスタンス障害時、SYNC REDO転送が開始する間、約3.5秒のパフォーマンス・ブラウンアウトが発生しました(ネットワーク同期サービス - NSS)。
    - ネットワーク障害時、宛先のnet_timeoutパラメータの設定と等しい短時間のブラウンアウトが認められました。
- マシンの停止に対するHAでは、各遠隔同期インスタンスが別個のハードウェア上にあることを前提としています。
- サイトの停止に対するHAでは、遠隔同期インスタンスが別個のサイトにデプロイされることを前提としています。
- 遠隔同期インスタンスの停止中のアプリケーション・ブラウンアウトおよび再同期化時間がOracle RACによる遠隔同期と比較して長くなります

ターミナル・スタンバイでの遠隔同期インスタンスの代替宛先としてのデプロイ

遠隔同期インスタンスの停止中にデータ保護を維持する最も簡単な方法は、ターミナル・スタンバイを直接指す代替LOG_ARCHIVE_DEST_n (端末フェイルオーバー・ターゲット)を作成することです。WANネットワーク待機時間が原因でプライマリのパフォーマンスに影響が及ばないようにするには、リモート宛先への非同期トランスポートが最も有力な選択肢です。

非同期トランスポートではゼロに近いデータ損失保護を実現できますが(公開までに1秒もかかりません)、スタンバイ確認応答を待機しないため、データ損失ゼロは保証できません。この構成では、遷移を実行するためにレベルはターゲットに対して強制力がある必要があるため、スイッチオーバー(計画済のイベント)の前に保護レベルを最大パフォーマンスに落とす必要があります。保護レベルおよび転送方法の変更は、停止時間を必要としない動的操作です。

遠隔同期インスタンスの停止中に、REDO転送は代替宛先を使用して自動的にフェイルオーバーします。遠隔同期インスタンスが修復されて操作が再開されると、トランスポートは自動的に遠隔同期インスタンスに切り替わり、データ損失ゼロの保護がリストアされます。

この方法の特徴は次のとおりです。

- 管理対象となる追加ハードウェアまたは遠隔同期インスタンスはありません。
- 遠隔同期インスタンスの停止中、データ損失ゼロの適用範囲が失われます。遠隔同期インスタンスが動作を再開し、スタンバイが完全に同期されるまで、データ保護レベルがASYNC転送によるUNSYNCHRONIZEDに落ちます。

遠隔同期デプロイメント・トポロジの選択

REDOの送受信に関して、遠隔同期インスタンスの高可用性のための構成はすべて等しく機能します。構成は、最大データ損失(RPO)に対するアプリケーションの許容度および様々な障害シナリオでのアプリケーション・ブラウンアウト期間に基づいて選択する必要があります。

- Oracle RACにデプロイされた遠隔同期インスタンスでは、ブラウンアウトが最小で保護が最適になりますが、クラスタまたはサイトの停止は対象に含まれません。最重要アプリケーションには、互いの代替として構成されて異なる場所にデ

プロイされたOracle RAC遠隔同期インスタンスのペアが適しています。最も堅牢な遠隔同期高可用性(インスタンス、ノード、クラスタおよびサイトの障害)の保護を実現できます。

- RAC以外の環境で代替遠隔同期インスタンスを使用すると、各インスタンスを個別の物理データベース・サーバーに配置できます。この構成では、遠隔同期インスタンスを異なるサイトにデプロイすることで保護を実現します。データ保護が不可欠でも、コストが重要な考慮事項であるアプリケーションには、単一ノードの遠隔同期インスタンスのペアをそれぞれが他方の代替としてデプロイすることが最適です。しかし、一方の遠隔同期インスタンスから他方への転送の遷移中、アプリケーション・ブラウンアウトが若干増え、再同期時間が長くなります。転送が遠隔同期インスタンス間で遷移する際に、1秒の停止でもプライマリ・データベースに影響する場合には、データが損失する可能性があります。
- ターミナル・スタンバイ代替構成では、遠隔同期インスタンスが使用できない間、データ損失ゼロの保護がないことをアプリケーションで許容する必要がありますが、追加ハードウェアの実装が不要です。遠隔同期インスタンスの停止中にデータ損失の可能性が高くなることを許容でき、低コストが主な考慮事項であるアプリケーションには、非同期REDO転送を使用してターミナル・スタンバイを代替場所として構成することが最適です。ターミナル・スタンバイを代替宛先として使用するには、遠隔同期インスタンスの停止を解決するために必要な期間中に構成が非同期モードで実行されることを受け入れる必要があります。このアプローチの利点は、追加のハードウェアまたはソフトウェアをデプロイまたは管理する必要がないことです。遠隔同期インスタンスの停止中にデータ損失の可能性が高くなることを許容でき、低コストが主な考慮事項であるアプリケーションには、ASYNC REDO転送を使用してターミナル・スタンバイを代替場所として構成することが最適です。
- 遠隔同期ハブは、単一の物理ホスト上の複数のData Guard構成の遠隔同期インスタンスを効率的に統合する方法です。データ損失ゼロのサービス・レベル・カテゴリを含むクラウド・デプロイメントでは、遠隔同期ハブをデプロイして、単一の物理マシンまたはクラスタ上の複数のデータ損失ゼロ構成の遠隔同期インスタンスを効率的に統合できます
- データ保護が不可欠でも、コストが重要な考慮事項であるアプリケーションには、単一ノードの遠隔同期インスタンスのペアをそれぞれが他方の代替としてデプロイすることが最適です。

遠隔同期構成のベスト・プラクティス

次に、同期REDO転送先に適用されるベスト・プラクティスに加えて必要な遠隔同期構成のベスト・プラクティスを示します。

- プライマリ・データベースと遠隔同期インスタンスの間のネットワークでは次のことが必要です。
    - レスポンス時間およびプライマリ・データベースのスループットへの影響がビジネス要件を超えないように、ラウンド・トリップ待機時間を十分に低くします。影響の程度はアプリケーション固有であり、テストで検証する必要があります。通常、経験的には、ラウンド・トリップ待機時間が5ミリ秒未満である場合に成功する可能性が高くなることがわかっています。ただし、待機時間がこれよりも長くても成功するデプロイメントはあります。
    - ネットワークを共有する他のトラフィックに加えて、プライマリ・データベースと遠隔同期インスタンスとの間にピークREDOボリュームに対応するために十分な帯域幅を提供します。REDO転送圧縮を使用すると、ネットワーク帯域幅の要件を減らすことができます。
    - ネットワーク・コンポーネントの障害を許容する冗長なネットワーク・リンクを確保するのが理想的です。
- 標準Oracle Data Guardネットワークのベストプラクティス(適切なTCP送受信バッファ・サイズを帯域幅遅延積の3倍に設定するなど)。オンラインREDOログの適切な構成を参照してください。
- 遠隔同期インスタンスのスタンバイREDOログは十分なIOPS (書込み数/秒)能力を持つストレージに配置して、他のアクティビティからのIOPSに加えて、ピーク・アクティビティ時にプライマリ・データベースのLGWRプロセスのI/Oを超えても対応できるようにする必要があります。これは、重要な考慮事項です。たとえば:

    - 遠隔同期インスタンスにプライマリ・データベースよりもパフォーマンスが低いディスクがある場合、受信と同じ速度でREDOをリモート宛先に転送できず、アーカイブ・ログのギャップが生じることがあります。
    - REDOギャップ解消のシナリオの場合、たとえば、スタンバイでの計画メンテナンスまたはネットワークの停止のために、ピークREDOの頂点でギャップ解消用の追加I/Oリクエストが受信されます。
    - 遠隔同期インスタンスにあるパフォーマンスが低いディスクはプライマリ・データベースへの確認応答を遅らせるため、プライマリ・データベースとスタンバイ・データベースの間のラウンドトリップ時間が長くなり、アプリケーションのレスポンス時間に影響を及ぼします。この影響を解消するには、プライマリ・データベースと遠隔同期インスタンスとの間で遠隔同期を使用します。
- 遠隔同期インスタンスは、スタンバイ・データベースと同じスタンバイREDOログのベスト・プラクティスに従う必要があります。オンラインREDOログの適切な構成を参照してください。
- 代替遠隔同期がアクティブ化されたら同期転送に最速で戻るようにするには、代替遠隔同期インスタンスのスタンバイREDOログを手動でクリアしてから使用する必要があります。たとえば:

```
ALTER DATABASE CLEAR LOGFILE GROUP 4, GROUP 5, GROUP 6;
```

- Oracle MAAパフォーマンス・テストによって、小規模の遠隔同期インスタンスSGAは遠隔同期インスタンスやプライマリ・データベースのパフォーマンスに影響を及ぼさないことが示されています。システム・リソースを節約するために、遠隔同期の動作に必要な最小限のSGAを構成できます。
    - CPU_COUNT=4を設定します。圧縮も暗号化も使用しない場合には、値を1または2に設定できます。
    - テスト中にCPU_COUNTを削減しても、遠隔同期インスタンスのパフォーマンスには影響はありません。
- ロール・トランジション後もデータ損失ゼロの保護を維持するには、プライマリ・データベースとスタンバイ・データベースの両方に対して遠隔同期インスタンスを構成します。スタンバイ・データベースに近接して構成された2つ目の遠隔同期インスタンスは、スタンバイがプライマリ・データベースになるまでアイドル状態であるため、逆方向の同期REDO転送が可能です。

  Data Guard Broker構成では、ターゲット・スタンバイ・サイトから保護モードを強制できないかぎり、最大可用性モードである間はスイッチオーバー(計画済のロール移行)を実行できません。スタンバイ・データベースに独自の遠隔同期インスタンスがない場合は、ロールを元に戻した後に元のプライマリ・データベースに非同期REDOを送信するように構成する必要があります。これにより、プライマリ・データベースの保護モードを最大可用性から最大パフォーマンスに初めて落とした場合を除いて、スイッチオーバーの実行が防止されます。
- 遠隔同期では、ネットワーク待機時間および遠隔同期インスタンス・ハードウェアのI/O速度に応じて、同期転送と比較してプライマリ・データベースのパフォーマンスが4%から12%向上します。
- CPU、I/Oおよびネットワークの所定の要件を満たします。
    - 遠隔同期インスタンスを仮想マシンに配置しても、物理ハードウェア構成のパフォーマンスは低下しません。
    - 複数のData Guard構成を処理している複数の遠隔同期インスタンスは、同じ物理サーバー、クラスタまたは仮想マシンを共有できます。
- 遠隔同期サーバーでアーカイブを管理する必要がある場合があります。

Active Data Guardの遠隔同期アーキテクチャの構成

次の各トピックでは、Active Data Guardの遠隔同期アーキテクチャの構成例を示します。

遠隔同期インスタンスの構成

次の例は、遠隔同期インスタンスをOracle Data Guard Broker構成に追加する方法を示しています。

最初のステップでは、プライマリ・データベース・サーバーから独立している、または障害が分離されていて、アプリケーション・パフォーマンスで許容できるようにプライマリ・サーバーと遠隔同期サーバー間のネットワーク待機時間が一貫して低い(たとえば、5ミリ秒未満)遠隔同期スタンバイ・インスタンスを追加します。

次の例では、プライマリ・データベースNorth_Salesに対して遠隔同期インスタンスFS1が作成されます。

```
DGMGRL> ADD FAR_SYNC FS1 AS CONNECT IDENTIFIER IS FS1.example.com;
Far Sync FS1 added
DGMGRL> ENABLE FAR_SYNC FS1;
Enabled.
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxPerformance
  Members:
  North_Sales  - Primary database
    FS1        - Far Sync
    South_Sales - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

遠隔同期インスタンスが構成に追加された後、最大可用性モードをサポートするようにREDO転送を設定してから、次の例に示すように保護モードをアップグレードします。

```
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : FS1 SYNC)';
DGMGRL> EDIT FAR_SYNC 'FS1' SET PROPERTY 'RedoRoutes' = '(North_Sales : South_Sales
ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxAvailability
  Members:
  North_Sales  - Primary database
    FS1          - Far Sync
      South_Sales - Physical standby database

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

スイッチオーバーまたはフェイルオーバー後に、リモート・スタンバイ・データベースSouth_Salesがプライマリ・データベースになった場合に最大可用性保護モードを保持できるようにするには、2番目の遠隔同期インスタンスを構成に追加して、South_SalesがREDOを同期モードで送信できるようにし、ロール・トランジション後に新しいターミナル・データベースNorth_SalesにREDOが送信されるようにします。

次の例は、ブローカ構成に2番目の遠隔同期インスタンス(FS2)を追加する方法を示しています。

```
DGMGRL> ADD FAR_SYNC FS2 AS CONNECT IDENTIFIER IS FS2.example.com;
Far Sync FS2 added
DGMGRL> EDIT FAR_SYNC 'FS2' SET PROPERTY 'RedoRoutes' = '(South_Sales : North_Sales
ASYNC)';
DGMGRL> ENABLE FAR_SYNC FS2;
Enabled.
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'RedoRoutes' = '(LOCAL : FS2 SYNC)';
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxAvailability
  Members:
  North_Sales  - Primary database
```

```
    FS1          - Far Sync
      South_Sales - Physical standby database
      FS2         - Far Sync (inactive)

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

HA遠隔同期インスタンスの設定

代替HA遠隔同期インスタンスは、プライマリ・データベースおよびリモート・スタンバイ・データベース用に作成した遠隔同期インスタンスに高可用性を提供するために設定します。

次の例は、Oracle Data Guard Broker構成のプライマリ・データベースの遠隔同期インスタンス(FS1)に2つ目の遠隔同期インスタンス(FS1a)を追加して、プライマリ遠隔同期インスタンスが使用できなくなった場合にREDO転送で代替遠隔同期インスタンスが使用されるようにする方法を示しています。

```
DGMGRL> ADD FAR_SYNC FS1a AS CONNECT IDENTIFIER IS FS1a.example.com;
Far Sync FS1a added
DGMGRL> EDIT DATABASE 'North_Sales' SET PROPERTY 'RedoRoutes' = ' (LOCAL:(FS1 SYNC
PRIORITY=1, FS1a SYNC PRIORITY=2))';
DGMGRL> EDIT FAR_SYNC 'FS1' SET PROPERTY 'RedoRoutes' = '(North_Sales : South_Sales
ASYNC)';
DGMGRL> EDIT FAR_SYNC 'FS1a' SET PROPERTY 'RedoRoutes' = '(North_Sales : South_Sales
ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxAvailability
  Members:
  North_Sales  - Primary database
    FS1          - Far Sync
    FS1a          - Far Sync
      South_Sales - Physical standby database

Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS
```

プライマリで代替遠隔同期インスタンスを追加した後、次の例を使用して、スタンバイで代替遠隔同期インスタンス(FS2a)を追加します。

```
DGMGRL> ADD FAR_SYNC FS2a AS CONNECT IDENTIFIER IS FS2a.example.com;
Far Sync FS2a added
DGMGRL> EDIT DATABASE 'South_Sales' SET PROPERTY 'RedoRoutes' = ' (LOCAL:(FS2 SYNC
PRIORITY=1, FS2a SYNC PRIORITY=2))';
DGMGRL> EDIT FAR_SYNC 'FS2' SET PROPERTY 'RedoRoutes' = '(South_Sales : North_Sales
ASYNC)';
DGMGRL> EDIT FAR_SYNC 'FS2a' SET PROPERTY 'RedoRoutes' = '(South_Sales : North_Sales
ASYNC)';
DGMGRL> EDIT CONFIGURATION SET PROTECTION MODE AS MaxAvailability;
DGMGRL> SHOW CONFIGURATION;
Configuration - DRSolution
  Protection Mode: MaxAvailability
  Members:
    North_Sales - Primary database
      FS1 - Far Sync
      FS1a - Far Sync
        South_Sales - Physical standby database
          FS2 - Far Sync (inactive)
          FS2a - Far Sync (inactive)
Fast-Start Failover: DISABLED
Configuration Status:
```

```
SUCCESS
```

Oracle RACまたはOracle Clusterwareによる遠隔同期インスタンスの構成

遠隔同期インスタンスが(たとえばOracle Restart、Oracle Real Application Clusters (Oracle RAC)またはOracle RAC One Nodeのインストールの)Oracle Clusterwareを使用してサーバーまたはクラスタにデプロイされる場合は、SRVCTLユーティリティを使用して、デフォルトのオープン・モードのマウントを指定します。

次のようなコマンドを使用できます。

```
srvctl modify database -d db_unique_name -startoption MOUNT
```

Data Guardおよび高速オフライン暗号化を使用したデータベースの暗号化

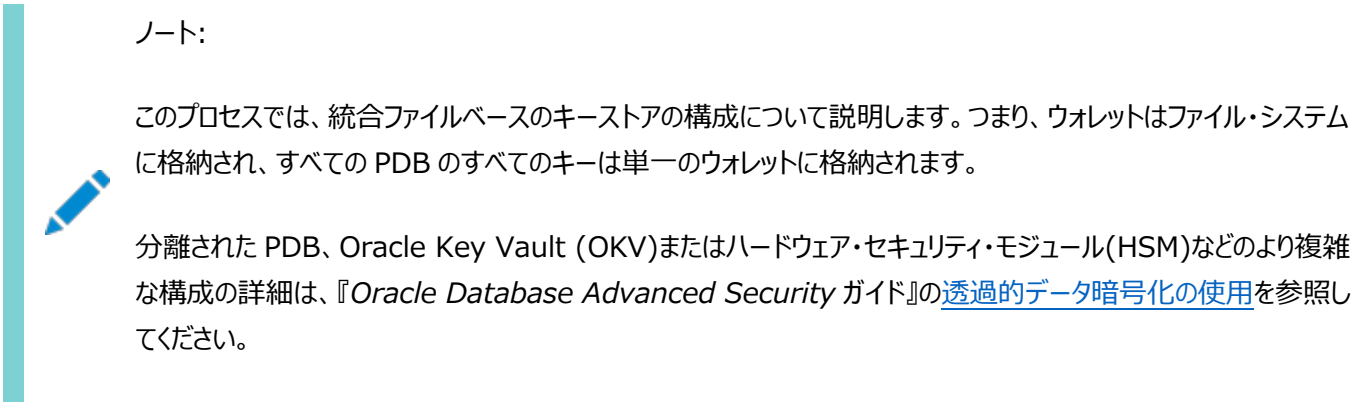
透過的データ暗号化(TDE)を使用したデータベースの暗号化は、スタンバイ・データベースとオフライン暗号化を使用して、停止時間を最小限に抑え、追加の領域要件なしで、より迅速に実行できます。

この2フェーズ・プロセスでは、スタンバイ・データベースはオフラインで暗号化され、その後スイッチオーバーが行われ、新しいスタンバイ・データベース(以前のプライマリ)でオフライン暗号化が繰り返されます。

最新のOracleリリースでは、オンライン暗号化も使用できます。オンライン暗号化は一部のユーザーには適していますが、表領域の変換中に追加のストレージが必要で、各ブロックが新しい暗号化データ・ファイルに読み書きされるため、オンライン暗号化は時間のかかるプロセスになる場合があります。高速オフライン暗号化では、マウントされたスタンバイ・データベースで各データ・ファイルが直接インプレースで暗号化されます。

ステップ1: 透過的データ暗号化(TDE)の構成

様々なTDE構成オプションがあります。Oracleリリースごとに要件が異なります。データベース・リリースの『*Oracle Database Advanced Security*ガイド』の透過的データ暗号化の概要を確認して、TDEの構成オプションおよび影響を理解することをお薦めします。



ノート:

このプロセスでは、統合ファイルベースのキーストアの構成について説明します。つまり、ウォレットはファイル・システムに格納され、すべての PDB のすべてのキーは単一のウォレットに格納されます。

分離された PDB、Oracle Key Vault (OKV)またはハードウェア・セキュリティ・モジュール(HSM)などのより複雑な構成の詳細は、『*Oracle Database Advanced Security* ガイド』の透過的データ暗号化の使用を参照してください。

次に、統合ファイルベースのキーストアを構成するために必要な基本パラメータを示します。パラメータはプライマリ・データベースとスタンバイ・データベースで構成されますが、値が異なる場合があります。

パラメータ	構成のベスト・プラクティス
WALLET_ROOT	Oracle Database 18c 以降では、すべてのデータベース・ウォレットのルート・ディレクトリを指定するためのベスト・プラクティスは、WALLET_ROOT データベース・パラメータの構成です。クラスタ化されたデータベースの場合、WALLET_ROOT で指定された場所は、ASM ディスクなどの共有の

パラメータ	構成のベスト・プラクティス
	場所である必要があります。 `ALTER SYSTEM SET WALLET_ROOT='+DATA/db_unique_name' SCOPE=SPFILE SID='*';` ノート: WALLET_ROOT は静的パラメータです。変更を有効にするには、データベースを再起動する必要があります。TDE_CONFIGURATION は、WALLET_ROOT を設定してデータベースを再起動するまで設定できません。 詳細は、『*Oracle Database* リファレンス』の WALLET_ROOT を参照してください。
TDE_CONFIGURATION	TDE_CONFIGURATION データベース・パラメータは、キーストアのタイプを設定します。 TDE_CONFIGURATION は動的ですが、WALLET_ROOT を構成してデータベースを再起動した後にのみ設定できます。 `ALTER SYSTEM SET TDE_CONFIGURATION='KEYSTORE_CONFIGURATION=FILE' SCOPE=SPFILE SID='*';` 詳細は、『*Oracle Database* リファレンス』の TDE_CONFIGURATION を参照してください。
TABLESPACE_ENCRYPTION	データベース・パラメータ TABLESPACE_ENCRYPTION は、Oracle 19c (19.16)で使用できます。TABLESPACE_ENCRYPTION は、ENCRYPT_NEW_TABLESPACES のかわりとなるもので、作成時に新しい表領域を暗号化するかどうかを指定します。 TABLESPACE_ENCRYPTION パラメータと ENCRYPT_NEW_TABLESPACES パラメータの両方が設定されている場合、TABLESPACE_ENCRYPTION が優先されます。 TABLESPACE_ENCRYPTION の値は次のとおりです。 ● AUTO_ENABLE - 新しく作成されたすべての表領域が暗号化されます。これは、Oracle 19c (19.16)以降ではオーバーライドできない Oracle Cloud のデフォルトです。 ● MANUAL_ENABLE - 表領域を CREATE 文の ENCRYPTION 句で暗号化するかどうかを手動で制御します。 ● DECRYPT_ONLY - 表領域は暗号化されません。この設定はハイブリッド Data Guard 構成で使用され、ここではクラウド・データベースの暗号化中にオンプレミス・データベースが暗号化されないままになります。 `ALTER SYSTEM SET TABLESPACE_ENCRYPTION=MANUAL_ENABLE SCOPE=BOTH SID='*';` 詳細は、『*Oracle Database* リファレンス』の TABLESPACE_ENCRYPTION を参照して

パラメータ	構成のベスト・プラクティス
	ください。
ENCRYPT_NEW_TABLESPACES	Oracle Database 19c (19.16)より前の ENCRYPT_NEW_TABLESPACES パラメータは、新しい表領域を暗号化するかどうかを指定します。 ENCRYPT_NEW_TABLESPACES の値は次のとおりです。 ● CLOUD_ONLY - 表領域が Oracle Cloud で作成されると、暗号化句が含まれているかどうかに関係なく、デフォルトの暗号化アルゴリズムで透過的に暗号化されます。デフォルトの暗号化アルゴリズムはステップ 2 に示すように変更できますが、AES128 がデフォルトのアルゴリズムです。 ● ALWAYS - 新しい表領域は、データベースが Oracle Cloud にあるかどうかに関係なく透過的に暗号化されます。 ● DDL - 表領域を ENCRYPTION 句で暗号化するかどうかを手動で制御します。 詳細は、『*Oracle Database* リファレンス』の ENCRYPT_NEW_TABLESPACES を参照してください。

次の表に、Oracle Databaseリリースに基づいて構成するTDEパラメータを示します。

Oracleリリース	WALLET_ROOTおよびTDE_CONFIGURATION	TABLESPACE_ENCRYPTION	ENCRYPT_NEW_TABLESPACES
Oracle 19c (19.16)以降	あり	あり	なし
Oracle 18c から 19c (19.15)	あり	なし	あり

ステップ2: デフォルトの暗号化アルゴリズムの設定

TDEのデフォルトの暗号化アルゴリズムはAES128です。Oracle 21c以降のリリースでは、TABLESPACE_ENCRYPTION_DEFAULT_ALGORITHMパラメータを使用してアルゴリズムを設定できますが、ウォレットを作成する前にこの設定を構成する必要があります。同様に、_tablespace_encryption_default_algorithmは、パッチ30398099を含むOracle 19c以前で使用できます。

この設定によって、TABLESPACE_ENCRYPTION=AUTO_ENABLE、ENCRYPT_NEW_TABLESPACES=ALWAYS、およびこのプロセスで使用されるオフライン暗号化の新しい表領域で使用される暗号化アルゴリズムが決まります。

プライマリ・データベースおよびスタンバイ・データベースで、次が発行されます。

```
-- for Oracle 21c and later
ALTER SYSTEM SET "tablespace_encryption_default_algorithm"='AES256' scope=both;
-- for Oracle 19c and earlier
ALTER SYSTEM SET "_tablespace_encryption_default_algorithm"='AES256' scope=both;
```

ステップ3: 暗号化ウォレットの作成およびマスター・キーの設定

TDEドキュメントでは、ウォレットまたはキーストアの作成、およびプライマリ・データベースでのマスター暗号化キーの設定について

詳細に説明します。

詳細は、『*Oracle Database Advanced Security*ガイド』の統一モードのソフトウェア・キーストアおよびTDEマスター暗号化キーの構成を参照してください。

スタンバイの暗号化後にプライマリ・データベースを暗号化しないままにする場合でも、ハイブリッドData Guardユースケースでは、マスター・キーをプライマリ・データベースに設定する必要があります。このキーは、REDO適用時およびロール・トランジション後のスタンバイのデータの暗号化に使用されます。このキーは、ロール・トランジション後に暗号化されたプライマリ・クラウド・データベースからデータを復号化するために使用されます。

ステップ4: スタンバイ・データベース環境へのウォレット・ファイルのコピー

スタンバイ・データベースで暗号化操作を実行するには、スタンバイ・データベースに暗号化ウォレットと自動ログイン・キーストアのコピーが必要です。ファイルはプライマリ・データベースからスタンバイ・データベースに適宜コピーします。

WALLET_ROOTで定義された場所から、実行します。ターゲット・ディレクトリがスタンバイに存在しない場合は、手動で作成する必要があります。

各ノードにファイルをコピーします。

```
ASMCMD> cp +DATA/PRIMARY_ORACLE_UNQNAME/TDE/cwallet.sso /tmp
ASMCMD> cp +DATA/PRIMARY_ORACLE_UNQNAME/TDE/ewallet.p12 /tmp
<primary host>$ scp /tmp/cwallet.sso ewallet.p12 oracle@standby_host:/tmp
<standby host> ASMCMD> cp /tmp/cwallet.sso +DATA/STANDBY_db_unique_name/TDE/
<standby host> ASMCMD> cp /tmp/ewallet.p12 +DATA/STANDBY_db_unique_name/TDE/
```

または、ASMからASMにファイルを直接コピーすることもできます。

```
ASMCMD>cp cwallet.sso sys/password@stbyhost1.+ASM1:+DATA/STANDBY_ORACLE_UNQNAME/TDE/
ASMCMD>cp ewallet.p12 sys/password@stbyhost1.+ASM1:+DATA/STANDBY_ORACLE_UNQNAME/TDE/
```

ステップ5: Data Guardのヘルスの確認

オフライン暗号化プロセスを開始する前に、スタンバイ・データベースがプライマリに遅れていないことを確認します。管理対象リカバリは暗号化プロセス中に停止する必要があるため、スタンバイ・データベースがプライマリに遅れていないことを確認すると、暗号化プロセス後に適用する必要があるREDOギャップが削減されます。

プライマリまたはスタンバイ・データベースで、REDO適用ラグを検索し、次の例に示すようにスタンバイ・データベースを検証します。Data Guard BrokerコマンドVALIDATE DATABASEは、潜在的な構成のギャップをリストします。ギャップに対処し、「Ready for Switchover」および「Ready for Failover」のステータスが両方ともYESであることを確認します。

```
DGMGRL> SHOW CONFIGURATION LAG
Configuration - dgconfig
 Protection Mode: MaxPerformance
 Members:
primary_db  - Primary database
 standby_db - Physical standby database
       Transport Lag:    0 seconds (computed 1 second ago)
       Apply Lag:      0 seconds (computed 1 second ago)
Fast-Start Failover: Disabled
Configuration Status:
SUCCESS  (status updated 11 seconds ago)
DGMGRL> VALIDATE DATABASE <standby>
 Database Role:   Physical standby database
  Primary Database: primary
 Ready for Switchover: Yes
 Ready for Failover:  Yes (Primary Running)
```

ステップ6: スタンバイ・データベースのマウントおよびリカバリの停止

オフライン暗号化プロセスをデータ・ファイルに対して直接実行する前に、スタンバイ・データベースをマウントし、リカバリを停止する必要があります。暗号化プロセス中にスタンバイのすべてのインスタンスを使用して、複数のファイルを同時に暗号化できます。

```
$ srvctl stop database -d standby -o immediate
$ srvctl start database -d standby -o mount
DGMGRL> EDIT DATABASE standby SET STATE=APPLY-OFF;
```

REDO転送サービスは、アーカイブ・ログがスタンバイ・データベースに存在するようにREDOを引き続き送信します。このプロセスでは、暗号化プロセス中に障害が発生した場合にリカバリ・ポイント目標(RPO)が維持されます。

非常にアクティブなデータベースの場合、必要な数のアーカイブ・ログが重要になる可能性があるため、リカバリ領域に十分な領域があることを確認してください。

ステップ7: スタンバイ・データベースでのデータ・ファイルのインプレースおよびパラレルな暗号化

TEMP表領域の暗号化プロパティは、作成後に変更できません。TEMP表領域を暗号化するには、暗号化して作成する必要があります。

暗号化されたTEMP表領域を使用するには、ENCRYPTION句を使用して新しいTEMP表領域を作成し、それをデフォルトの一時表領域にします。次に、元のTEMP表領域を削除します。

```
SQL> CREATE TEMPORARY TABLESPACE TEMP_ENC ENCRYPTION ENCRYPT;
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE TEMP_ENC;
```

暗号化された表領域の機密データから生成されるUNDOおよびTEMPメタデータは、すでに暗号化されているため、UNDOおよびTEMP表領域の暗号化はオプションです。

1. スタンバイ・データベースがマウントされ、キーストアがオープンしていることを確認します。

```
SQL> select inst_id,database_role,open_mode from gv$database;
  INST_ID DATABASE_ROLE OPEN_MODE
---------- ---------------- --------------------
  1 PHYSICAL STANDBY MOUNTED
 2 PHYSICAL STANDBY MOUNTED
SQL> col WRL_PARAMETER format a40
SQL> set linesize 120 pagesize 9999
SQL> select * from gv$encryption_wallet;
INST_ID WRL_TYPE WRL_PARAMETER  STATUS
------- -------- ---------------------------------------- ------
  1 file +DATA/ORACLE_UNQNAME/TDE OPEN
```

2. データ・ファイルを暗号化します。

   オフライン暗号化コマンドは、各データ・ファイルを単一のプロセスで暗号化しますが、複数のデータ・ファイルを別々のセッションでパラレルで暗号化できます。各セッションは、CPUコアを完全に使用できます。各インスタンスでホストのコア数以下のセッション数を発行することをお薦めします。

   次の問合せを使用して、データ・ファイルを変換するスクリプトを生成できます。スクリプトを複数のスクリプトに分割し、個々のセッションで小さいスクリプトをそれぞれ実行します。最も効率的なプロセスは、複数の小さなファイルを個々のスクリプトに配置しながら、大きなファイルを個別に暗号化することです。

   ノート:

   シード・データベース・ファイルを暗号化する必要はありません。

```
set lines 120
set pages 9999
spool encrypt.sql
```

```
select 'alter session set container='||pdb.name||';'||chr(10)||'alter database
datafile '||chr(39)||df.name||chr(39)||' encrypt;' COMMAND
from v$tablespace ts, v$datafile df, v$pdbs pdb where ts.ts#=df.ts# and
ts.con_id=df.con_id and df.con_id=pdb.con_id and pdb.name <> 'PDB$SEED';
spool off
COMMAND
--------------------------------------------------------------------------------
----------------------------------------
alter session set container=ORADBP11;
alter database datafile
'+DATA/DB_UNIQUE_NAME/E73F249E7030C3B8E0537B544664A065/DATAFILE/system.336.1113
852973' encrypt;
alter session set container=ORADBP11;
alter database datafile
'+DATA/DB_UNIQUE_NAME/E73F249E7030C3B8E0537B544664A065/DATAFILE/sysaux.335.1113
852973' encrypt;
alter session set container=ORADBP11;
alter database datafile
'+DATA/DB_UNIQUE_NAME/E73F249E7030C3B8E0537B544664A065/DATAFILE/undotbs1.337.11
13852973' encrypt;
<...>
```

3. TEMPファイルは、CREATE文のENCRYPTION句を使用して削除および再作成することで暗号化できます。V$TEMPFILEビューを使用して、既存のTEMPファイルを識別します。
4. V$DATAFILE_HEADER.ENCRYPTEDを問い合せて、すべてのデータ・ファイルが暗号化されていることを確認します。ファイルの暗号化の完了後、ENCRYPTED列はファイルが暗号化されているか(YES)いないか(NO)を示します。シードPDBに属するデータ・ファイルを除くすべてのデータ・ファイルを暗号化する必要があります。

ステップ8: REDO適用の再開とスタンバイ・データベースへのキャッチ・アップ

すべてのデータ・ファイルが暗号化されていることを確認した後、スタンバイ・データベースは、暗号化プロセス中に生成されたプライマリからのすべてのREDOを適用する必要があります。適用する必要があるREDOの量に応じて、スタンバイ・データベースでREDOをキャッチ・アップするには、次の方法をお薦めします。

- ギャップが小さい場合は、管理対象リカバリを再起動し、適用ラグが0になるまでREDOギャップを適用します。

  プライマリ・データベースまたはスタンバイ・データベースで、次を実行します

  ```
  DGMGRL> edit database standby set state=apply-on;
  ```

- 暗号化プロセスに時間がかかり、プライマリ・データベースがアクティブだった場合、ギャップが大きくなる可能性があります。多くの場合、増分ロール・フォワード・アプローチを使用して、適用の停止後に変更されたブロックのみをコピーする方が高速です。

  このプロセスについては、My Oracle Supportノート[サービスからのデータベースのリカバリによるスタンバイ・データベースのロール・フォワードの方法(ドキュメントID 2850185.1)]に記載されています。ロール・フォワードが完了してもリカバリは必要ですが、このプロセスによって、大きなギャップを埋めるまでの時間が大幅に短縮される可能性があります。

ステップ9: Data Guardスイッチオーバーの実行によるプライマリ・データベースでの暗号化の開始

プライマリ・データベースを暗号化する準備が整うまで、暗号化されていないプライマリ・データベースが暗号化されていないREDOをスタンバイに送信して、スタンバイによって無期限に暗号化するようにできます。

プライマリ・データベースを暗号化する準備ができたら、データベース・ロールを切り替え、Data Guardスイッチオーバーを実行して、暗号化されたスタンバイ・データベースを新しいプライマリ・データベースとし、暗号化されていないプライマリ・データベースを新しいスタンバイにすると、便利です。

スタンバイになった元のプライマリ・データベースで、ステップ5から8を繰り返してデータ・ファイルを暗号化し、REDOにキャッチ・アッ

プします。

ステップ11: Data Guardスイッチオーバーの実行(オプション)

スタンバイ・データベースとプライマリ・データベースの両方が暗号化された後、元のプライマリ/スタンバイ・データベース・ロールに戻す場合は、Data Guardスイッチオーバーを実行して元のロールを再設定します。

15 Oracle Data Guardのチューニングおよびトラブルシューティング

REDO転送、REDO適用またはロール・トランジションが想定した要件を満たしていない場合は、次のガイドラインを使用してデプロイメントをチューニングおよびトラブルシューティングします。

Oracle Data Guardのチューニングおよびトラブルシューティングの概要

Oracle Data Guard構成から最適なパフォーマンスを得るには、監視、アセスメントおよびパフォーマンス・チューニングに次のOracle MAAベスト・プラクティスを使用します。

- Oracle DatabaseおよびOracle Data Guard構成ベスト・プラクティスが適用されていることを確認します。

  評価およびチューニング時の前提は、Oracle DatabaseおよびData Guardのすべての構成ベスト・プラクティスがすでに環境に統合されていることです。チューニングを実行する前に、これらのベスト・プラクティスへの準拠を評価します。

- REDO転送サービスの評価およびチューニング

  Oracle Data Guardでは、パフォーマンスを最適化するために、REDO転送を自動的にチューニングします。ただし、パフォーマンスの問題が発生した場合は、REDO転送サービスを監視およびチューニングできます。

  最大パフォーマンス・データ保護モードでの非同期REDO転送が、デフォルトのOracle Data Guard構成です。非同期REDO転送のチューニングでは、主に、プライマリ、スタンバイおよびネットワーク・リソースがワークロードを処理するのに十分であることを確認し、それらのリソースのボトルネックを確実に監視するようにします。

  同期REDO転送では、データ損失ゼロのパフォーマンスが幾分低下しますが、MAA推奨の適切な方法を使用すると、影響を監視して評価し、リソースを適切に分散させることができます。

- REDO適用の評価およびチューニング

  ほとんどの場合、スタンバイが常に最新の状態であれば、デフォルトのOracle設定で良好なメディア・リカバリのパフォーマンスを実現できます。ただし、アプリケーションおよびデータベースのサイズとスループットが増えた場合は、メディア・リカバリ操作をさらにチューニングすると、スタンバイ・データベースでリカバリ時間またはREDO適用スループットをいっそう最適化できます

- ロール・トランジションの評価およびチューニング

  適切な計画と実装に基づいたOracle Data GuardおよびActive Data Guardのロール・トランジションにより、停止時間を効果的に最小化し、ビジネスへの影響を最小限に抑えながらデータベース環境をリストアできます。フィジカル・スタンバイ・データベースおよびOracle Maximum Availability Architecture (MAA)ベスト・プラクティスを使用してパフォーマンスをテストしたところ、スイッチオーバーおよびフェイルオーバーを数秒に短縮できました。

REDO転送のトラブルシューティングおよびチューニング

Oracle Data Guard REDO転送のパフォーマンスは、プライマリ・システムとスタンバイ・システム、それらを接続するネットワークおよびI/Oサブシステムのパフォーマンスに直接依存します。

ほとんどのOracle Data Guard構成では、REDO転送のトラブルシューティングおよびチューニングを行うことで、データ損失をゼロまたは最小限に抑えることができます。

ここに示すガイダンスでは、MAA構成のベスト・プラクティスに従っていることを前提としています。前提条件として、Oracle Data Guardの構成ベスト・プラクティスが実装されていることを確認します。

転送を全体的に向上させるには、次のトピックで説明するデータ収集およびトラブルシューティングの方法を活用します。この方法では、実際にREDO転送の問題があるかどうかを評価するために必要なデータの収集、およびREDO転送スループットを最適化するためにチューニングできることをガイドします。

- トポロジ情報の収集
- 転送ラグの検証およびREDO転送構成の理解
- 転送ラグのトラブルシューティングのための情報の収集
- プライマリでのREDO生成率履歴の比較
- 転送ネットワークの評価およびチューニング
- システム・リソースの収集および監視
- 高度なトラブルシューティング: 非同期REDO転送を使用したネットワーク時間の決定
- 同期REDO転送のチューニングおよびトラブルシューティング

トポロジ情報の収集

Oracle Data Guard構成のトポロジとData Guardのパフォーマンスとの関連性を理解すると、Data Guardアーキテクチャに起因すると誤って判断されることがよくあるインフラストラクチャの脆弱性を排除するのに役立ちます。

次の上位レベルのアーキテクチャ情報の概要をまとめることをお薦めします。

- プライマリ・データベース・システムおよびスタンバイ・データベース・システムの説明(Oracle RACクラスタ内のノード数、データベース・ノードごとのCPUおよびメモリー、ストレージI/Oシステム)
- プライマリ・システムとスタンバイ・システムを接続するネットワーク・トポロジの説明
    - プライマリとスタンバイの間のネットワーク・コンポーネント/デバイス
    - ネットワーク帯域幅および待機時間

対称のハードウェアおよび構成を持つスタンバイ・データベースで、ネットワーク構成が適切にチューニングされている場合、転送ラグは10秒未満であり、ほとんどの場合1秒未満となります。

転送ラグの検証およびREDO転送構成の理解

スタンバイ・データベースにラグがあるかどうか、およびこれが転送ラグまたは適用ラグかどうかを判断するには、V$DATAGUARD_STATSビューを問い合せます。

```
SQL> select name,value,time_computed,datum_time from v$dataguard_stats where
name='%lag';
```

DATUM_TIME列は、メトリックの計算に使用されたデータを受信した、スタンバイ・データベースでのローカル時刻です。ラグ・メトリックは、プライマリ・データベースから定期的に受信されるデータに基づいて計算されます。複数の問合せにおいてこの列の値が変化しない場合は、スタンバイ・データベースがプライマリ・データベースからデータを受信していないことを示している。このシナリオで考えられるデータ損失は、V$DATAGUARD_STATSの最終データ時刻からスタンバイの現在時刻までになります。

スタンバイ・インスタンスが最後に起動されてからの転送ラグまたは適用ラグの値の履歴を示すヒストグラムを取得するには、V$STANDBY_EVENT_HISTOGRAMビューを問い合せます。

```
SQL> select * from v$standby_event_histogram where name like '%lag' and count
>0;
```

ある期間にわたって転送ラグまたは適用ラグを評価するには、その期間の開始時のV$STANDBY_EVENT_HISTOGRAMのス

ナップショットを取得し、その期間の終了時に取得したスナップショットと比較します。

```
SQL> col NAME format a10
SQL> select NAME,TIME,UNIT,COUNT,LAST_TIME_UPDATED from V$STANDBY_EVENT_HISTOGRAM
where
 name like '%lag' and count >0 order by LAST_TIME_UPDATED;
NAME TIME UNIT COUNT LAST_TIME_UPDATED
---------- ---------- --------------- ---------- --------------------
transport lag 41 seconds 3 01/05/2022 16:30:59
transport lag245 seconds    1   01/05/2022 16:31:02
transport lag 365 seconds    2 01/05/2022 16:31:03
transport lag 451 seconds    2 01/05/2022 16:31:04
```

REDO転送ラグが高いことを確認した場合は、「転送ラグのトラブルシューティングのための情報の収集」を使用して、このREDO転送の調査を続行します。転送ラグはないものの、REDO適用ラグが高い場合は、「REDO適用のトラブルシューティングおよびチューニング」の方法を使用して適用ラグに対処します。

転送ラグのトラブルシューティングのための情報の収集

許容できないREDO転送ラグが発生した場合は、次の情報を収集し、質問について調査します。

- 転送ラグはいつ発生しましたか。V$DATAGUARD_STATSおよびV$STANDBY_EVENT_HISTOGRAMのデータを記録して、ラグがいつ開始したか、およびラグが時間の経過とともにどのように変化しているかを示します。
- 転送ラグは、日次バッチ操作では毎日午前0時、大規模バッチ操作では毎月、四半期末では毎四半期など、特定の期間中に発生しますか。
- 有効なOracle Data Guard転送ではLOG_ARCHIVE_DEST設定を確認し、REDO COMPRESSIONまたはENCRYPTIONが有効かどうかを確認します。送信前にREDOを圧縮または暗号化し、スタンバイでの受信時に圧縮解除または暗号化解除する必要があるため、REDO転送スループット全体に悪影響を及ぼす可能性があります。その変更が最新かどうか、およびこれらの設定属性の無効化をテストできるかどうかを確認します。
- Oracle Net設定をチェックして、Oracle Net暗号化が有効になっているかどうかを評価します。Oracle Net暗号化が有効な場合、いつ、どのレベルで有効化されましたか。REDOは送信前に暗号化され、スタンバイでのREDOの受信時に暗号化解除されるため、Oracle Netの暗号化によってREDOスループットが大幅に低下する可能性があります。オプションで、暗号化レベルを無効化または低減して、REDO転送ラグが減少するかどうかを確認します。

プライマリでのREDO生成率履歴の比較

大規模なバッチ・ジョブ、データ・ロード、データ・ポンプ操作、CREATE TABLE AS SELECT、PDML操作、または月末、四半期末、年度末のバッチ更新など、短期間についてプライマリ・データベースのREDO生成率が例外的に高くなる場合があります。

プライマリ・データベースからREDO生成履歴を取得し、REDO転送時またはREDO適用ラグの開始時と比較します。新しいプラガブル・データベースや新しいアプリケーション・サービスの追加など、追加ワークロードが原因で、REDO生成率が例外的に高いのかどうかを確認します。これにより、この追加のロードに対応するためにさらにチューニングが必要になる場合があります。

トラブルシューティングの一環として、次の情報を収集するか、次の質問に対処します。

- この問合せを使用して、プライマリ・データベースのREDO生成率の日次履歴を収集します。

```
SQL> select trunc(completion_time) as "DATE", count(*) as "LOG SWITCHES",
round(sum(blocks*block_size)/1024/1024) as "REDO PER DAY (MB)"
from v$archived_log
where dest_id=1
group by trunc(completion_time) order by 1;
```

- REDOラグまたは転送ラグが開始する6時間前以降の、ログごとのREDO生成率を収集します。

```
SQL> alter session set nls_date_format='YYYY/MM/DD HH24:MI:SS';
SQL> select thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
first_time)*86400 sec, blocks*block_size/1024/1024)/((next_time-
first_time)*86400) "MB/s" from v$archived_log
where ((next_time-first_time)*86400<>0)
and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
and dest_id=1 order by first_time;
```

- REDOラグまたは転送ラグが開始する6時間前の自動ワークロード・リポジトリ(AWR)レポートから、REDO生成率の毎時のスナップショットを収集します。

  デフォルトでは、Oracle Databaseによりスナップショットが1時間ごとに自動生成されますが、スナップショットを手動で作成して、自動生成されるスナップショットとは異なるタイミングで統計を取得することが必要になる場合があります。既存のスナップショットの情報を表示するには、DBA_HIST_SNAPSHOTビューを使用します。

  AWRおよびスナップショットおよびAWRレポートの生成の詳細は、『Oracle Databaseパフォーマンス・チューニング・ガイド』のスナップショットの作成に関する項を参照してください。

- このプライマリREDO生成率は、以前の履歴と比較して例外的に高いですか。
- 可能な場合は、高いREDO生成率に対応するワークロードを特定し、それが一時的かどうか、またはチューニング可能かどうかを評価します。

  たとえば、大規模なパージ操作では、REDO生成ボリュームを減らすためにパーティション操作を切り捨てるか、または削除することを検討してください。

転送ネットワークの評価およびチューニング

REDO転送は、スタンバイ・データベースのバックグラウンド・プロセスにREDOを送信するプライマリ・データベース・インスタンスのバックグラウンド・プロセスで構成されます。ネットワークがOracle Data Guard REDO転送用に最適化されているかどうかを評価できます。

非同期REDO転送が構成されている場合、REDOデータは、大規模なパケットで非同期的に、スタンバイにストリーミングされます。ネットワークを介した非同期REDO転送をチューニングするには、単一プロセスのネットワーク転送を最適化する必要があります。

同期REDO転送が構成されている場合、次のREDO書込みに進む前に、各REDO書込みをプライマリ・データベースとスタンバイ・データベースで確認する必要があります。LOG_ARCHIVE_DEST設定の一部としてFASTSYNC属性を使用することで、スタンバイ同期転送を最適化できますが、ネットワーク待機時間が長いと(5ミリ秒を超えるなど)、REDO転送スループット全体に影響を及ぼします。

続行する前に、まず「ネットワーク・パフォーマンスの評価および最適化」を参照して、次のことを実行します。

- プライマリのREDO生成率をサポートするのに十分なネットワーク帯域幅があるかどうかを評価する
- REDO転送をチューニングするための最適なTCPソケット・バッファ・サイズを決定する
- REDO転送をチューニングするために、ソケット・バッファ・サイズに対するオペレーティング・システムの制限をチューニングする
- REDO書込みサイズの最適なMTU設定を決定する
- REDO転送のネットワーク・スループットを向上させるためのMTUをチューニングする

ネットワーク構成がチューニングされている場合は、転送ラグ(「転送ラグの検証およびREDO転送構成の理解」を参照)が許容レベルまで小さくなっているかどうかを評価します。その場合は、目標を達成したので終了できます。それ以外の場合は、残りのチューニングおよびトラブルシューティングに関する項に進みます。

システム・リソースの収集および監視

Oracle Linux OSwatcherまたはOracle Exadata ExaWatcherのデータを収集して、システム・リソースを分析します。

OSWatcher (oswbb)は、パフォーマンス問題の診断のサポートを支援するために、オペレーティング・システムおよびネットワークのメトリックを収集およびアーカイブすることを目的としたUNIXシェル・スクリプトのコレクションです。ベスト・プラクティスとして、実行中のOracleインスタンスが存在するすべてのノードでOSWatcherをインストールして実行する必要があります。パフォーマンスに関する問題の場合、Oracleサポートはこのデータを使用して、データベースの外部で発生する可能性のあるパフォーマンスに関する問題を診断できます。

OSWatcherは、OSWatcher (Doc ID 301137.1)からダウンロードできます。

ExaWatcherは、Exadataシステムのストレージ・サーバーおよびデータベース・サーバーのパフォーマンス・データを収集するユーティリティです。収集されるデータには、iostat、セル統計(cellsrvstat)、ネットワーク統計などのオペレーティング・システム統計情報が含まれます。

詳細は、『Oracle Exadata Database Machineメンテナンス・ガイド』のExaWatcherチャートを使用したExadata Database Machineのパフォーマンスの監視に関する項を参照してください。

Data Guardのリソース要件を満たすためのチューニング

REDO転送は、次の場合に影響を受ける可能性があります。

- プライマリ・データベースまたはスタンバイ・データベースが完全にCPUバウンドである
- プライマリ・データベースまたはスタンバイ・データベースのI/Oシステムが飽和状態である
- ネットワーク・トポロジでREDO生成率をサポートできない

プライマリ・データベース・システムに次のものがあるかどうかを評価します。

- ログ・ライター・プロセス(LGWR)がフォアグラウンドを効率的にポストするのに十分なCPU使用率
- ローカル・ログ書込みでピーク時のI/O待機時間を低く保つために十分なI/O帯域幅
- 同じインタフェースの他のネットワーク・アクティビティと組み合せてピークREDO率ボリュームを処理できるネットワーク・インタフェース
- チューニングおよびトラブルシューティングのためにプライマリ・データベースから収集された自動ワークロード・リポジトリ(AWR)、アクティブ・セッション履歴(ASH)およびOSwatcherまたはExawatcherデータ

スタンバイ・データベース・システムに次のものがあるかどうかを評価します。

- スタンバイ・データベースでREDOを受信するOracle Data Guardプロセスであるリモート・ファイル・サーバー(RFS)がスタンバイREDOログに効率的に書き込むのに十分なCPU使用率
- ローカル・ログ書込みでピーク時のI/O待機時間を低く保つために十分なI/O帯域幅
- 同じインタフェースの他のネットワーク・アクティビティと組み合せてピークREDO率ボリュームを受信できるネットワーク・インタフェース
- チューニングおよびトラブルシューティングのためにスタンバイ・データベースから収集されたAWR、ASHおよびOSwatcherまたはExawatcherデータ

ノート:

スタンバイ・データベースで発生する最大の問題は、I/O 帯域幅が十分でないためにスタンバイ・ログ書込みの待機時間が長くなることです。この問題は、Data Guard 遠隔同期を使用することで軽減できます。

システム構成がチューニングされ、前述のリソース制約が解除されている場合は、転送ラグ(「転送ラグの検証およびREDO転送構成の理解」を参照)が許容レベルまで小さくなっているかどうかを評価します。その場合は、目標を達成しました。

高度なトラブルシューティング: 非同期REDO転送を使用したネットワーク時間の決定

続行する前に、まず「ネットワーク・パフォーマンスの評価および最適化」を参照してください。

十分なリソース(特にネットワーク帯域幅)がある場合、非同期REDO転送では、非常に高いワークロードを維持できます。リソースが制約される場合、非同期REDO転送が遅れ始めると、スタンバイ・データベースで転送ラグが増大します。

非同期REDO転送(ASYNC)は、REDOデータをトランザクション・コミットに対して非同期で転送します。トランザクションは、そのトランザクションによって生成されたREDOがリモート・スタンバイ・データベースに正常に送信されたことを確認するのを待機せずにコミットできます。ASYNCを使用すると、制限された帯域幅によって前のトランザクションのREDOがスタンバイ・データベースに即時に送信できなくても、プライマリ・データベースのログ・ライター・プロセス(LGWR)は、コミット成功を引き続き確認します(水が排出されるよりも速く、シンクが水でいっぱいになる様子を想像してください)。

ASYNCでは、TT00プロセスを使用して、プライマリ・データベースのログ・バッファから直接REDOを送信します。TT00プロセスが速度を維持できず、REDOをスタンバイ・データベースに転送する前にログ・バッファがリサイクルされる場合、TT00プロセスは、ディスク上のオンラインREDOログ・ファイル(ORL)からの読取りおよび送信に自動的に遷移します。TT00送信が現在のREDO生成に追い付くと、ログ・バッファからの直接読取りおよび送信に自動的に戻ります。

TT00が元のORLの送信を完了する前に2つ以上のログ・スイッチが存在する場合でも、TT00は、現在のオンライン・ログ・ファイルの内容の読取りに遷移して戻ります。元のORLと現在のORLの間にアーカイブされたすべてのORLは、Oracle Data GuardのREDOギャップ解消プロセスを使用して自動的に送信されます。

プライマリ・データベースとスタンバイ・データベースの両方における、ネットワーク帯域幅、CPU、メモリー、ログ・ファイルI/Oなどの十分なリソースは、非同期Data Guard構成のパフォーマンスにとって重要です。

非同期転送を制約しているリソースを特定するには、プライマリ・データベースとスタンバイ・データベースの両方でイベント16421を設定することで有効にできるkrsb統計を使用します。

```
alter session set events '16421 trace name context forever, level 3';
```

このイベントは非常に軽量であり、プライマリ・データベースまたはスタンバイ・データベースのパフォーマンスには影響しません。

この動的イベントは、すべてのプライマリ・インスタンスおよびスタンバイ・インスタンスに設定する必要があり、指定された順序の送信が完了すると、TT00またはリモート・ファイル・サーバー(RFS)のトレース・ファイルに統計が書き込まれます。トレース・ファイルを調べると、krsb_end統計はファイルの先頭と末尾に示されます。ファイルの末尾にある統計では、どこで非同期送信に時間がかかったかについての見通しが得られます。たとえば:

```
krsb_end: Begin stats dump for T-1.S-593
 max number of buffers in use       10
 Operation elapsed time (micro seconds)  474051333
 File transfer time (micro seconds)    474051326
 Network Statistics
 LOG_ARCHIVE_DEST_2 : OCI REQUEST
  Total count : OCI REQUEST       2748
  Total time  : OCI REQUEST       81374
  Average time : OCI REQUEST       29
 LOG_ARCHIVE_DEST_2 : NETWORK SEND
```

```
  Total count : NETWORK SEND      2748
  Total time  : NETWORK SEND      286554724
  Average time : NETWORK SEND      104277
  Total data buffers queued       9644
  Total data buffers completed      9644
  Total bytes written          9885272064
  Total bytes completed synchronously  9885272064
  Average network send size (blocks)   7025
  Average network send buffers      3.51
  Average buffer turnaround time     240889
  Throughput (MB/s)           19.89
 Total network layer time        286636098
 Percentage of time in network      60.47
 Disk Statistics
  Total count : DISK READ        11531
  Total time  : DISK READ        12335132
  Average time : DISK READ        1069
  Read-ahead blocks           14151680
  Log buffer blocks           266915
  Disk stall blocks           4888576
  Total count : BUFFER RELEASE     9643
  Total time  : BUFFER RELEASE     7229
  Average time : BUFFER RELEASE     0
 Total disk layer time          12342361
 Percentage of time in disk layer    2.60
 Data Guard Processing Statistics
  Total count : SLEEP          198
  Total time  : SLEEP          172351312
  Average time : SLEEP          870461
 Total DG layer time           175072867
 Percentage of time in DG layer     36.93
 Remote Server-Side Network Statistics
 LOG_ARCHIVE_DEST_2 : NETWORK GET
  Total count : NETWORK GET       8242
  Total bytes : NETWORK GET       9885272064
  Total time  : NETWORK GET       453233790
  Average time : NETWORK GET       54990
 Total server-side network layer time  453233790
 Percentage of time in network      95.61
 Remote Server-Side Disk Statistics
 LOG_ARCHIVE_DEST_2 : DISK WRITE
  Total count : DISK WRITE       9644
  Total time  : DISK WRITE       8731303
  Average time : DISK WRITE       905
 LOG_ARCHIVE_DEST_2 : DISK NOSTALL REAP
  Total count : DISK NOSTALL REAP    9644
  Total time  : DISK NOSTALL REAP    579066
  Average time : DISK NOSTALL REAP    60
 LOG_ARCHIVE_DEST_2 : BUFFER GET
  Total count : BUFFER GET       9644
  Total time  : BUFFER GET       3607
  Average time : BUFFER GET       0
 Total server-side disk layer time    9313976
 Percentage of time in disk layer    1.96
 Remote Server-Side Data Guard Processing Statistics
 LOG_ARCHIVE_DEST_2 : PUBLISH RTA BOUNDARY
  Total count : PUBLISH RTA BOUNDARY  8948
  Total time  : PUBLISH RTA BOUNDARY  3665841
  Average time : PUBLISH RTA BOUNDARY  409
 LOG_ARCHIVE_DEST_2 : VALIDATE BUFFER
  Total count : VALIDATE BUFFER     9644
  Total time  : VALIDATE BUFFER     1403088
  Average time : VALIDATE BUFFER     145
 Total Server-Side DG layer time     11503560
 Percentage of time in DG layer     2.43
krsb_end: End stats dump
```

前述の出力は、転送ラグが発生し始めたばかりのテスト実行から取得したものです。ネットワークの輻輳が増加し、ネットワーク・レイヤーで待機する時間が50%を超えて増加したことによる遅延を確認できます。転送ラグが圧縮または暗号化の結果である場合、Data Guardレイヤーに費やされる時間の割合がその大部分となります。

krsb統計を無効にするには、イベント16421をレベル1に設定します。

```
alter session set events '16421 trace name context forever, level 1';
```

同期REDO転送のチューニングおよびトラブルシューティング

続行する前に、まず「ネットワーク・パフォーマンスの評価および最適化」を参照してください。

次のトピックでは、同期REDO転送を評価する方法について説明します。

- 同期トランスポートでデータの整合性を確保する方法の理解
- 同期REDO転送環境でのパフォーマンスの評価
- log file sync待機イベントが誤解を招く理由
- 外れ値の原因の理解
- 同期REDO転送リモート書込みの影響
- 同期REDO転送パフォーマンスのトラブルシューティングの例

同期トランスポートでデータの整合性を確保する方法の理解

次のアルゴリズムにより、Oracle Data Guard同期REDO転送構成でデータの整合性を確保できます。

- プライマリ・データベースのオンラインREDOログに対するログ・ライター・プロセス(LGWR)のREDO書込みと、スタンバイREDOログに対するData Guardネットワーク・サービス・サーバー(NSS)のREDO書込みは同一です。
- スタンバイ・データベースのData Guard Managed Recovery Process (MRP)は、REDOがプライマリ・データベースのオンラインREDOログに書き込まれていないかぎり、REDOを適用できません。ただし、唯一の例外がData Guardフェイルオーバーの操作中です(プライマリが停止している場合)。

  REDOの同期送信に加えて、NSSおよびLGWRは、スタンバイ・リカバリがスタンバイREDOログ(SRL)から適用できる安全なREDOブロック境界に関する情報を交換します。これにより、スタンバイが受信しても、まだプライマリがオンラインREDOログにコミットされたことを確認していないREDOがスタンバイによって適用されることが回避されます。

次のような障害シナリオが考えられます。

- プライマリ・データベースのLGWRがオンラインREDOログに書き込むことができない場合、LGWRおよびインスタンスはクラッシュします。インスタンスまたはクラッシュ・リカバリは、オンラインREDOログ内の最後にコミットされたトランザクションにリカバリし、コミットされていないすべてのトランザクションをロールバックします。現在のログが完了し、アーカイブされます。
- スタンバイでは、一部のスタンバイREDOログが、対応するオンラインREDOログとサイズが一致する正しい値で完了します。スタンバイREDOログから欠落しているREDOブロックは、(REDOログ全体を再送信しないで)転送されます。
- プライマリ・データベースがクラッシュして自動または手動によるデータ損失ゼロのフェイルオーバーが発生した場合は、Data Guardフェイルオーバー操作の一部がターミナル・リカバリを実行し、現在のスタンバイREDOログを読み取ってリカバリします。

  リカバリでスタンバイREDOログ内のすべてのREDOの適用が終了すると、新しいプライマリ・データベースが起動し、新しく完了したログ・グループがアーカイブされます。新規および既存のすべてのスタンバイ・データベースが、オンラインREDOログ内のREDOを破棄し、一貫性のあるシステム変更番号(SCN)にフラッシュバックして、新しいプライマリ・データベースからのアーカイブのみを適用します。再度、Data Guard環境が(新しい)プライマリ・データベースと同期し

ます。

同期REDO転送環境でのパフォーマンスの評価

Oracle Data Guard同期REDO転送環境(SYNC)でパフォーマンスを評価する場合、様々な待機イベントが相互にどのように関連しているかを把握することが重要です。同期REDO転送を有効にした場合の影響は、アプリケーションによって異なります。

その理由を理解するために、コミットが発行されたときにログ・ライター・プロセス(LGWR)が実行する作業の次の説明を考慮してください。

1. フォアグラウンド・プロセスは、コミットのためにLGWRをポストします(ログ・ファイル同期が開始されます)。同時コミット・リクエストがキューに入っている場合、LGWRは未処理のすべてのコミット・リクエストをまとめてバッチ処理して、連続的なREDOストランドを生成します。
2. LGWRがCPUを待機します。
3. LGWRがREDO書込みを開始します(REDO書込み時間が開始されます)。
4. Oracle RACデータベースの場合、LGWRが現在の書込みを他のインスタンスにブロードキャストします。
5. 事前処理後、SYNCスタンバイがある場合、LGWRはリモート書込みを開始します(SYNCリモート書込みが開始されます)。
6. LGWRがローカル書込み(ログ・ファイル・パラレル書込み)を発行します。
7. SYNCスタンバイがある場合、LGWRはリモート書込みの完了を待機します。
8. I/Oステータスの確認後、LGWRはREDO書込み時間/SYNCリモート書込みを終了します。
9. Oracle RACデータベースの場合、LGWRはブロードキャストackを待機します。
10. LGWRがディスク上のSCNを更新します。
11. LGWRがフォアグラウンドをポストします。
12. フォアグラウンドがCPUを待機します。
13. フォアグラウンドがログ・ファイル同期を終了します。

次の方法を使用してパフォーマンスを評価します。

- バッチ・ロードの場合、これらのプロセスのほとんどが固定された期間内に完了する必要があるため、経過時間を監視することが最も重要な要因となります。これらの操作のデータベース・ワークロードは、通常のOLTPワークロードとは大きく異なります。たとえば、書込みのサイズが大幅に大きくなる可能性があるため、ログ・ファイル同期平均を使用しても、正確に把握または比較できません。
- OLTPワークロードの場合、AWRレポートから1秒当たりのトランザクション・ボリューム(自動ワークロード・リポジトリ(AWR)から)およびREDO率(1秒当たりのREDOサイズ)を監視します。この情報により、同期REDO転送を有効にすることで、アプリケーションのスループットとその影響を明確に把握できます。

log file sync待機イベントが誤解を招く理由

通常、プライマリ・データベースのlog file sync待機イベントは、管理者が同期REDO転送(SYNC)を有効にした場合の影響を評価するときに最初に参照する場所です。

SYNCを有効にする前の平均log file sync待機が3ミリ秒で、後の平均が6ミリ秒であった場合、SYNCがパフォーマンスに100パーセント影響を与えたと想定します。SYNCの影響を測定するためにlog file sync待機時間を使用することはお薦めしません。これは、平均は非常に誤解を招きやすいためであり、また、レスポンス時間およびスループットに対するSYNCの実際の影響はイベントが示す値よりもはるかに低くなる可能性があるためです。

ユーザー・セッションがコミットすると、ログ・ライター・プロセス(LGWR)はCPUを取得し、I/Oを送信し、I/Oの完了を待機してか

ら、CPUに戻って、コミットが完了したフォアグラウンド・プロセスをポストします。この期間全体がlog file sync待機イベントでカバーされます。LGWRが自身の作業を実行している間、ほとんどの場合、コミット中の他のセッションがあり、それらのセッションはLGWRの終了を待ってからそれぞれのコミットを処理する必要があります。待機しているセッションのサイズと数は、アプリケーションが持つセッションの数と、それらのセッションがコミットされる頻度によって決まります。コミットをこのようにバッチ処理することを一般にアプリケーション同時実行性と呼びます。

たとえば、通常、ログ書込み(log file parallel write)の実行に0.5msかかり、サービス・コミット(log file sync)の実行に1 msかかり、平均ではコミットごとに100個のセッションを処理しているとします。ストレージ層の異常によって1つのコミットのログ書込みI/Oの完了に20ミリ秒かかった場合、log file parallel writeに起因する長い待機は1つのみであるのに、最大2,000個のセッションがlog file syncで待機する可能性があります。1つの長い外れ値で多数のセッションが待機していると、log file syncの平均が大幅に偏ることがあります。

次の表は、特定の期間のlog file sync待機イベントに対するV$EVENT_HISTOGRAMからの出力を示しています。

表15-1 ログ・ファイル同期待機イベントのV$EVENT_HISTOGRAM出力

ミリ秒	待機数	合計待機に対する割合
1	17610	21.83%
2	43670	54.14%
4	8394	10.41%
8	4072	5.05%
16	4344	5.39%
32	2109	2.61%
64	460	0.57%
128	6	0.01%

この出力は、log file sync待機時間の92%が8ミリ秒未満で、大部分が4ミリ秒(86%)未満であることを示しています。8ミリ秒を超える待機は外れ値であり、待機時間全体に占める割合がわずか8%ですが、それらの外れ値で待機しているセッションの数のため(コミットのバッチ処理のため)、平均が偏ります。SYNCの影響を評価するメトリックとしてlog file sync平均待機時間が使用されている場合、偏った平均が誤解を招く可能性があります。

外れ値の原因の理解

プライマリ・データベースまたはスタンバイ・データベースでのI/Oの中断やネットワーク待機時間のスパイクがあると、同期REDO転送でlog file syncの外れ値が高くなる可能性があります。この影響は、スタンバイ・システムのI/Oサブシステムがプライマリ・システムのI/Oサブシステムよりも劣る場合に確認できます。

多くの場合、管理者がスタンバイ・システムに開発やテストなど複数のデータベースをホストするため、I/Oレスポンスが妨げられることがあります。iostatを使用してI/Oを監視し、ディスクが最大IOPSに達しているかどうかを確認することが重要です。これは、このことがSYNC書込みのパフォーマンスに影響するためです。

ログ・スイッチが頻繁に発生することは、外れ値の重大な原因です。次のように、プライマリでログ・スイッチが発生したときにスタンバイで何が起こるかを考慮してください。

1. スタンバイのリモート・ファイル・サーバー(RFS)プロセスで、スタンバイREDOログ・ヘッダーの更新を終了する必要があります。

2. 次に、RFSは追加のヘッダー更新を使用して新しいスタンバイREDOログに切り替えます。
3. ログを切り替えると、スタンバイで強制的に完全なチェックポイントが実行されます。

   これにより、バッファ・キャッシュ内のすべての使用済バッファがディスクに書き込まれ、書込みI/Oでスパイクが発生します。スタンバイ・ストレージ・サブシステムのパフォーマンスがプライマリ・データベースのパフォーマンスと同一でない非対称の構成では、I/O待機時間が長くなります。
4. 前のスタンバイREDOログをアーカイブして、読取りI/Oと書込みI/Oの両方を増やす必要があります。

同期REDO転送リモート書込みの影響

同期REDO転送(SYNC)を有効にすると、コミット処理の通常のローカル書込みに加えて、リモート書込み(スタンバイREDOログへのリモート・ファイル・サーバー(RFS)書込み)が導入されます。

このリモート書込みでは、ネットワーク待機時間およびリモートI/O帯域幅に応じて、コミット処理時間を増やすことができます。コミット処理には時間がかかるため、ログ・ライター・プロセス(LGWR)がその作業を終了してコミット・リクエストの作業を開始するまで待機しているセッションが増えます。つまり、アプリケーション同時実行性が増大しています。データベース統計および待機イベントを分析することで、アプリケーション同時実行性の増大を確認できます。

次の表の例を考えてみます。

表15-2 同期転送によるアプリケーション同時実行性増大の影響

SYNC	REDO率	ネットワーク待機時間	AWRからのTPS	ログ・ファイル同期平均(ミリ秒)	ログ・ファイル・パラレル書込み平均(ミリ秒)	RFS random I/O	SYNCリモート書込み平均(ミリ秒)	REDO書込みサイズ(KB)	REDO書込み
遅延	25MB	0	5,514.94	0.74	0.47	NA	NA	10.58	2,246,356
あり	25MB	0	5,280.20	2.6	.51	.65	.95	20.50	989,791
影響	0	-	-4%	+251%	+8.5%	NA	NA	+93.8%	-55.9%

前述の例では、SYNCを有効にすると、REDO書込みの数は減りますが、各REDO書込みのサイズは大きくなります。REDO書込みのサイズが増加するため、I/O (ローカルとリモートの両方)の実行に費やされる時間が長くなる可能性があります。1回の待機当たりの作業量が増えるため、log file sync待機時間が長くなります。

ただし、アプリケーション・レベルでは、コミットごとに処理されるセッションが増加しても、トランザクション率またはトランザクション・レスポンス時間への影響がほとんど変化しない場合があります。このため、データベース待機イベントに完全に依存するのではなく、アプリケーション・レベルでSYNCの影響を測定することが重要となります。また、これはSYNCがアプリケーションに与える実際の影響を把握する上でlog file sync待機イベントがなぜ誤解を招く指標であるかを示す好例でもあります。

同期REDO転送パフォーマンスのトラブルシューティングの例

同期REDO転送のパフォーマンスを調べるには、次に示すように、ローカルREDO書込み待機時間、書込み当たりの平均REDO書込みサイズおよびREDO書込み全体の待機時間を計算します。

次の待機イベントを使用して計算を行います。

- ローカルREDO書込み待機時間= 'log file parallel write'
- リモート書込み待機時間= 'SYNC remote write'
- 書込み当たりの平均REDO書込みサイズ= 'redo size' / 'redo writes'
- フォアグラウンド別に表示される平均コミット待機時間=「ログ・ファイル同期」

次の表に、Oracleデータベースの自動作業リポジトリ(AWR)レポートからの統計を示します。SYNCが無効になっているベースラインとパフォーマンスの影響を比較するために、ネットワーク待機時間が1ミリ秒のローカル・スタンバイに対して同期REDO転送(SYNC)が有効化されています。

表15-3 Oracle Databaseでの同期REDO転送のパフォーマンスの評価

メトリック	**ベースライン(SYNCなし)**	**SYNC**	**影響**
REDO 率(MB/秒)	25	25	変化なし
ログ・ファイル同期	0.68	4.60	+576%
ログ・ファイル・パラレル書込み平均(ミリ秒)	0.57	0.62	+8.8%
TPS	7,814.92	6224.03	-20.3%
RFS random I/O	NA	2.89	NA
SYNC リモート書込み平均(ミリ秒)	NA	3.45	NA
REDO 書込み	2,312,366	897,751	-61,2%
REDO 書込みサイズ(KB)	10.58	20.50	+93.8%

前述の例では、SYNCを有効にした後、log file sync待機平均が大幅に増加していることがわかります。ローカル書込みはほぼ一定していますが、log file sync増加の最大要因はSYNCリモート書込みが追加されたことでした。SYNCリモート書込みのうち、ネットワーク待機時間はゼロであるため、スタンバイREDOログへのリモート書込みに焦点を当てると、平均時間は2.89ミリ秒になります。プライマリとスタンバイが同じハードウェアを使用し、SYNCリモート書込み平均時間がプライマリのlog file parallel write平均時間と類似していることを考えると、これは緊急危険信号です。

前述の例では、スタンバイREDOログは複数のメンバーを持ち、低パフォーマンスのディスク・グループに配置されています。スタンバイREDOログのメンバーを1つに減らして高速のディスク・グループに配置すると、次の表に示すような結果が得られます。

表15-4 スタンバイREDOログのメンバーを1つに減らして高速のディスク・グループに配置した後のSYNCパフォーマンス

メトリック	**ベースライン(SYNCなし)**	**SYNC**	**影響**
REDO 率(MB/秒)	25	25	変化なし
ログ・ファイル同期	0.67	1.60	+139%
ログ・ファイル・パラレル書込み	0.51	0.63	+23.5%
TPS	7714.36	7458.08	-3.3%
RFS random I/O	NA	.89	NA

メトリック	ベースライン(SYNCなし)	SYNC	影響
SYNC リモート書込み平均(ミリ秒)	NA	1.45	NA
REDO 書込み	2,364,388	996,532	-57.9%
REDO 書込みサイズ(KB)	10.61	20.32	+91.5%

REDO適用のトラブルシューティングおよびチューニング

ほとんどのOracle Data Guard構成では、REDO適用のトラブルシューティングとチューニングを行うことで、適用ラグを最小化できます。REDO適用のパフォーマンスは、スタンバイ・システムのパフォーマンスに直接依存します。

ここに示すガイダンスでは、MAA構成のベスト・プラクティスに従っていることを前提としています。前提条件として、Oracle Data Guardの構成ベスト・プラクティスが実装されていることを確認します。

適用パフォーマンスを全体的に向上させるには、次のトピックで説明するデータ収集およびトラブルシューティングの方法を使用します。

REDO適用およびREDO適用のパフォーマンス期待値の理解

スタンバイ・データベース・リカバリは、DMLおよびDDLのすべての操作をリプレイするプロセスです。プロセスの概要は次のとおりです。

1. REDOはプライマリ・データベースから受信され、スタンバイREDOログ(SRL)に書き込まれます。データベースがOracle RACデータベースの場合、各スレッド(インスタンス)は割り当てられたSRLに格納されます。
2. ログ・マージ・プロセスは、リカバリ・コーディネータとも呼ばれ、REDOのスレッドをマージし、結果の変更ベクトルをメモリー・バッファに配置します。
3. リカバリ・ワーカー・プロセスは、必要なデータ・ブロックを特定し、バッファ・キャッシュに読み込みます(まだ存在しない場合)。次に、ワーカー・プロセスは、バッファ・キャッシュ内のブロックに変更ベクトルを適用します。
4. チェックポイント時間に、データベース・ライター・プロセスは検証済バッファ変更をデータ・ファイルに書き込み、システム・コミット番号(SCN)と呼ばれるデータベースのチェックポイント・タイムスタンプを拡張します。チェックポイントは、リカバリ・プロセスで最も広範囲に及ぶI/O負荷となる可能性があります。

REDO適用パフォーマンスの期待値

パフォーマンスおよび結果の適用率は、主に、リカバリ中のワークロードのタイプと、リカバリのために割り当てられ、使用可能なシステム・リソースによって異なります。

プライマリ・データベース・システムとスタンバイ・データベース・システムを対称にすること(同等のI/Oサブシステム、メモリー、CPUリソースなど)をお薦めします。このようにお薦めする主な理由は、どちらのデータベースがプライマリ・データベースであるかに関係なく、アプリケーションのパフォーマンスが同じレベルになるようにするためですが、REDO適用のパフォーマンスにも、プライマリ・データベースとスタンバイ・データベースを対称にすることで大きな利点がもたらされます。データ保護(DB_BLOCK_CHECKING、DB_BLOCK_CHECKSUM、DB_LOST_WRITE_PROTECT)などの機能には、Oracle Active Data Guardを使用したスタンバイ・データベースのレポートと同様に、CPUおよびI/Oリソースが必要です。

ほとんどの場合、REDO適用パフォーマンスはREDO生成率に遅れずについていく必要があり、その結果、対称的なシステム・リソースで適用ラグがほぼゼロになります。ワークロードのピーク時には、ワークロードが通常のレベルに戻ると自然にほぼゼロまで減少するREDO適用ギャップが、若干存在することがあります。

OLTPワークロード

オンライン・トランザクション処理(OLTP)ワークロードは数多くの異なるブロックに対してわずかな変更を行うため、OLTPワークロードのリカバリは、非常にI/O集中型となる場合があります。これにより、リカバリ中に、小さいランダム・ブロックのバッファ・キャッシュへの読込みが大量に発生します。その後、データベース・ライターは、書込みI/Oの大規模なバッチを実行してバッファ・キャッシュを維持し、データベースのチェックポイントを定期的に実行します。したがって、OLTPワークロードのリカバリでは、最適な率を実現するために、ストレージ・サブシステムで1秒当たりのI/O (IOPS)を多数処理する必要があります。これが、プライマリ・データベース・システムとスタンバイ・データベース・システムを対称にすることをお薦めするもう1つの理由です。

リソース・ボトルネックのないOracle Exadata Database Machineクオータ・ラック・システムのswingbenchによって生成されたOLTPワークロードのリカバリ・テストでは、約150 MB/秒の適用率を達成しました。より大規模なExadataシステムでは、単一インスタンスのREDO適用で200 MB/秒を超える速度が顧客によって確認されています。Exadata以外のシステムでは、I/OおよびネットワークのスループットがIOPSが低くなるため、このような速度を達成することはより困難です。

バッチ・ワークロード

OLTPワークロード・リカバリとは対照的に、バッチ・ワークロードのリカバリは、バッチ・ワークロードが大量の順次読取りおよび書込みで構成されるため、より効率的です。より多くのREDO変更が発生するものの、読取りおよび変更するデータ・ブロックが大幅に減少するため、OLTPワークロードと比較してREDO適用率が大幅に向上します。また、バッチ直接ロード操作リカバリの最適化により、効率が高まり、リカバリ速度もさらに向上します。

妨げとなるシステム・リソースのボトルネックを伴わないバッチ・ロードまたはパラレルDML (PDML)ワークロードを使用すると、小規模なExadata Database Machineクオータ・ラック・システムでの内部REDO適用テストで、適用率は約200-300 MB/秒となります。より大規模なExadataシステムのバッチ・ワークロードでは、単一インスタンスのREDO適用で600 MB/秒を超える適用率が顧客によって確認されています。これらの率はExadata以外のシステムで実現できますが、このような要求の高いワークロードを処理するには、システム・リソースの容量とスケーラブルなネットワークおよびI/Oサブシステムが必要です。

混合ワークロード

様々なOLTPワークロードとバッチ・ワークロードが混在するアプリケーションにおいて、プライマリ・データベースのREDO生成率が類似していてもスタンバイ・データベースのリカバリ速度が異なるのは、OLTPとバッチのリカバリ・パフォーマンス・プロファイルが異なり、システムの形状が異なるためです。様々なExadataシステムの様々な混合ワークロードで、100-1100 MB/秒のREDO適用率が、顧客により達成されています。これらの率はExadata以外のシステムで実現できますが、このような要求の高いワークロードを処理するには、システム・リソースの容量とスケーラブルなデータベース・コンピュート、ネットワークおよびI/Oサブシステムが必要です。このような極端なREDO適用率は、Exadata以外のシステムではほとんど実現されません。

REDO適用パフォーマンスの期待値のキャッチ・アップ

リアルタイムのREDO適用と比較すると、キャッチ・アップ期間中のREDO適用では、さらにシステム・リソースが必要になる場合があります。大きなREDOギャップがある場合、推奨事項については、「非常に大きいREDO適用ギャップの対処」を参照してください。

適用ラグの検証

リカバリ・パフォーマンスは、プライマリ・データベースのワークロード・タイプおよびREDO生成率によって異なります。適用率が低いからといって、必ずしもリカバリ・パフォーマンスに関する問題を示すわけではありません。ただし、永続的または増加する適用ラグ(不随する転送ラグなし)は、リカバリ・パフォーマンスのボトルネックを示す最適な目安となります。

適用ラグおよび転送ラグを識別して定量化するには、スタンバイ・データベースのV$DATAGUARD_STATSビューを問い合せます。

```
SQL> select name, value, time_computed, datum_time from v$dataguard_stats where
name='%lag';
```

DATUM_TIME列は、メトリックの計算に使用されたデータを受信した、スタンバイ・データベースでのローカル時刻です。ラグ・メトリックは、プライマリ・データベースから定期的に受信されるデータに基づいて計算されます。複数の問合せにおいてこの列の値が変化しない場合は、スタンバイ・データベースがプライマリ・データベースからデータを受信していないことを示している。このシナリオで考えられるデータ損失は、V$DATAGUARD_STATSの最終データ時刻からスタンバイの現在時刻までになります。

スタンバイ・インスタンスが最後に起動されてからの転送ラグまたは適用ラグの値の履歴を示すヒストグラムを取得するには、V$STANDBY_EVENT_HISTOGRAMビューを問い合せます。

```
 SQL> select * from v$standby_event_histogram where name like '%lag' and count >0;
```

ある期間にわたって転送ラグまたは適用ラグを評価するには、その期間の開始時のスタンバイ・データベースにおけるV$STANDBY_EVENT_HISTOGRAMのスナップショットを取得し、その期間の終了時に取得したスナップショットと比較します。

```
SQL> col NAME format a10
SQL> select NAME,TIME,UNIT,COUNT,LAST_TIME_UPDATED from V$STANDBY_EVENT_HISTOGRAM
 where name like '%lag' and count >0 order by LAST_TIME_UPDATED;
```

出力例:

```
NAMETIME   UNITCOUNT LAST_TIME_UPDATED
---------- ------ ---------- ----- -------------------
apply lag    23 seconds3     02/05/2022 16:30:59
apply lag   135 seconds1     02/05/2022 16:31:02
apply lag   173 seconds2     02/05/2022 16:32:03
apply lag   295 seconds2     02/05/2022 16:34:04
```

転送ラグによって適用ラグが発生する場合があります。転送ラグがほぼゼロで高い適用ラグが確認される場合は、「情報の収集」でこのREDO適用の調査を続行します。

高い転送ラグが確認された場合は、まず「REDO転送のトラブルシューティングおよびチューニング」の方法を使用して、転送ラグに対処します。

情報の収集

許容できない適用ラグが発生した場合は、次の情報を収集します。

- 適用ラグはいつ発生しましたか。

  15分から30分ごとにV$DATAGUARD_STATSおよびV$STANDBY_EVENT_HISTOGRAMのデータを記録して、ラグがいつ開始され、過去24時間に時間の経過とともにラグがどのように変化したかを確認します。

  ```
  SQL>select name, value, time_computed, datum_time from v$dataguard_stats where
  name='%lag';

  SQL>select * from v$standby_event_histogram where name like '%lag' and count
  >0;
  ```

- 適用ラグは、日次バッチ操作では毎日午前0時、大規模バッチ操作では毎月、四半期末では毎四半期など、特定の期間に発生しますか。
- スタンバイ自動作業リポジトリ(AWR)レポートV$RECOVERY_PROGRESSからデータを収集し、適用ラグの前および期間中に30分間隔で複数のスタンバイAWRスナップショットを取得します。

  How to Generate AWRs in Active Data Guard Standby Databases (Doc ID 2409808.1)を参照

してください。

たとえば:

```
SQL> set lines 120 pages 99
SQL> alter session set nls_date_format='YYYY/MM/DD HH24:MI:SS';
SQL> select START_TIME, ITEM, SOFAR, UNITS from gv$recovery_progress;
```

サンプル出力:

```
START_TIME     ITEM                 SOFAR UNITS
------------------- ------------------------------ ---------- ---------
2022/02/28 23:02:36 Log Files                   8 Files
2022/02/28 23:02:36 Active Apply Rate             54385 KB/sec
2022/02/28 23:02:36 Average Apply Rate            12753 KB/sec
2022/02/28 23:02:36 Maximum Apply Rate            65977 KB/sec
2022/02/28 23:02:36 Redo Applied                2092 Megabytes
2022/02/28 23:02:36 Last Applied Redo              0 SCN+Time
2022/02/28 23:02:36 Active Time                 41 Seconds
2022/02/28 23:02:36 Apply Time per Log             1 Seconds
2022/02/28 23:02:36 Checkpoint Time per Log           0 Seconds
2022/02/28 23:02:36 Elapsed Time                168 Seconds
2022/02/28 23:02:36 Standby Apply Lag              2 Seconds
```

REDOボリュームに関してアプリケーションのスループットを判断する最も簡単な方法は、通常およびピークのワークロード中にプライマリ・データベースで自動ワークロード・リポジトリ(AWR)レポートを収集し、本番データベースによって生成される1秒当たりのREDOデータのバイト数を確認することです。その後、REDOの生成速度をV$RECOVERY_PROGRESSビューのActive Apply Rate列と比較して、スタンバイ・データベースがその速度を維持できるかどうかを確認できます。

適用ラグが想定を上回る場合は、V$RECOVERY_PROGRESSビューを問い合せてREDO適用のパフォーマンスを評価します。このビューには、次の表で説明する列が表示されます。

Average Apply RateにはREDOの到着の待機に費やされたアイドル時間が含まれていて適用パフォーマンスが的確に示されないため、最も有用な統計はActive Apply Rateです。

表15-5 V$RECOVERY_PROGRESSビューの列

列	説明
Average Apply Rate	適用済 REDO/経過時間には、REDO のアクティブな適用に費やされた時間および REDO の到着の待機に費やされた時間が含まれます
Active Apply Rate	REDO 適用時間/アクティブ時間は過去 3 分間の移動平均で、率には、REDO の到着の待機に費やされた時間は含まれません
Maximum Apply Rate	REDO 適用時間/アクティブ時間は、過去 3 分間の移動平均で達成されたピーク測定スループットまたは最大率です。率には、REDO の到着の待機に費やされた時間は含まれません
Redo Applied	適用されたデータの合計量(バイト)
Last Applied Redo	適用された最後の REDO のシステム変更番号(SCN)およびタイムスタンプ。これは REDO ストリームに格納される時間であるため、プライマリを基準にしてスタンバイ・データベースの場所を比較するために使用できます。
Apply Time per Log	ログ・ファイルで REDO をアクティブに適用するために費やされた平均時間。
Checkpoint	ログ境界チェックポイントに費やされた平均時間。

列	説明
Time per Log	
Active Time	REDO を適用する合計期間(REDO の待機は含まれません)
Elapsed Time	REDO を適用する合計期間(REDO の待機が含まれます)
Standby Apply Lag	REDO 適用が適用されていない秒数。スタンバイがプライマリに遅れている可能性があります。
Log Files	現在までに適用されたログ・ファイルの数。

アクティブ・セッション履歴

スタンバイAWRが使用できない場合、またはスタンバイ・データベースがオープン読取り専用モードでない場合は、V$ACTIVE_SESSION_HISTORYビューを使用して上位の待機を収集できます。スタンバイAWRは、提供される追加情報および詳細のためにお薦めしますが、場合によっては次の問合せが役立ちます。

過去30分間の上位10件の待機を選択する場合(30を現在の時間からさかのぼる他の分数に置き換えます):

```
select * from (
select a.event_id, e.name, sum(a.time_waited) total_time_waited
from v$active_session_history a, v$event_name e
where a.event_id = e.event_id and a.SAMPLE_TIME>=(sysdate-30/(24*60))
group by a.event_id, e.name order by 3 desc)
where rownum < 11;
```

2つのタイムスタンプ間の待機を選択する場合(例では、2021/01/01 00:00:00から2021/01/01 03:00:00までの3時間の期間を示しています):

```
select * from (
select a.event_id, e.name, sum(a.time_waited) total_time_waited
from v$active_session_history a, v$event_name e
where a.event_id = e.event_id
and a.SAMPLE_TIME
between to_date('2021/01/01 00:00:00','YYYY/MM/DD HH24:MI:SS') and
to_date('2021/01/01 03:00:00','YYYY/MM/DD HH24:MI:SS')
group by a.event_id, e.name
order by 3 desc)
where rownum < 11
/
```

プライマリでのREDO生成率履歴の比較

大規模なバッチ・ジョブ、データ・ロード、データ・ポンプ操作、CREATE TABLE AS SELECT、PDML操作、または月末、四半期末、年度末のバッチ更新など、短期間についてプライマリ・データベースのREDO生成率が例外的に高くなる場合があります。

プライマリ・データベースからREDO生成履歴を取得し、REDO転送時またはREDO適用ラグの開始時と比較します。新しいプラガブル・データベース(PDB)や新しいアプリケーション・サービスの追加など、追加ワークロードが原因で、REDO生成率が例外的に高いのかどうかを確認します。この追加のロードに対応するためにさらにチューニングが必要になる場合があります。

トラブルシューティングの一環として、次の情報を収集するか、次の質問に対処します。

- この問合せを使用して、プライマリ・データベースのREDO生成率の日次履歴を収集します。

  ```
  SQL> select trunc(completion_time) as "DATE", count(*) as "LOG SWITCHES",
  round(sum(blocks*block_size)/1024/1024) as "REDO PER DAY (MB)"
  ```

```
from v$archived_log
where dest_id=1
group by trunc(completion_time) order by 1;
```

- REDOラグまたは転送ラグが開始する6時間前以降の、ログごとのREDO生成率を収集します。

```
SQL> alter session set nls_date_format='YYYY/MM/DD HH24:MI:SS';
SQL> select thread#,sequence#,blocks*block_size/1024/1024 MB,(next_time-
first_time)*86400 sec, blocks*block_size/1024/1024)/((next_time-
first_time)*86400) "MB/s" from v$archived_log
where ((next_time-first_time)*86400<>0)
and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
and dest_id=1 order by first_time;
```

- このプライマリREDO生成率は、以前の履歴と比較して例外的に高いですか。
- 可能な場合は、高いREDO生成率に対応するワークロードを特定し、それが一時的かどうか、またはチューニング可能かどうかを評価します。

  たとえば、大規模なパージ操作では、REDO生成ボリュームを減らすためにパーティション操作を切り捨てるか、または削除することを検討してください。

単一インスタンスREDO適用のチューニング

単一インスタンスREDO適用(SIRA)のチューニングは、反復プロセスであり、マルチインスタンスREDO適用(MIRA)の評価よりさらに前の必須の前提条件です。反復プロセスは次で構成されます

1. システム・リソース・ボトルネックの評価と対処
2. 上位スタンバイ・データベースの待機イベントに基づくチューニング

システム・リソース・ボトルネックの評価

まず、CPU使用率やI/Oサブシステムなどのシステム・リソースを評価します。`top`や`iostat`などのユーティリティまたはOSwatcherやExaWatcherの統計を使用して、これらのリソースに競合があるかどうかを判断します。リソースのボトルネックに対処してREDO適用に必要なリソースを解放すると、適用パフォーマンスを改善できます。

REDO適用は、次の場合に影響を受ける可能性があります。

- 管理対象リカバリ・ノードが完全にCPUバウンドである
- スタンバイ・データベースのI/Oシステムが飽和状態である
- スタンバイ・データベースSGA (特にバッファ・キャッシュ)が、プライマリ・データベースと同じサイズ(またはそれ以上)でない

最適なリカバリ・パフォーマンスのためには、スタンバイ・データベース・システムに次のものが必要です。

- リカバリ・コーディネータ(PR00)およびリカバリ・ワーカー(PRnn)に十分なCPU使用率
- 最大速度時に短いI/O待機時間を維持するために十分なI/O帯域幅
- 同じインタフェースの他のネットワーク・アクティビティに加えて、ピークREDO率ボリュームを受信できるネットワーク・インタフェース
- 対称SGAおよびバッファ・キャッシュに対応するために十分なメモリー。ログ・バッファおよびバッファ・キャッシュのサイズは、通常、REDO適用パフォーマンスに最も影響を及ぼします

収集内容と収集方法は、次のとおりです。

- スタンバイ自動作業リポジトリ(AWR)レポートを30分以下の間隔で収集します。

  『Oracle Databaseパフォーマンス・チューニング・ガイド』のActive Data Guardスタンバイ・データベースでの自動ワークロード・リポジトリの管理に関する項を参照してください

- よりリアルタイムで詳細な待機については、アクティブ・セッション履歴(ASH)データを収集します。

  『Oracle Databaseパフォーマンス・チューニング・ガイド』のアクティブ・セッション履歴レポートの生成に関する項を参照してください

- Oracle Linux OSwatcherまたはOracle Exadata ExaWatcherのデータを収集して、システム・リソースを分析します。

  Exadataシステムについては、『Oracle Exadata Database Machineメンテナンス・ガイド』のExaWatcherチャートを使用したExadata Database Machineのパフォーマンスの監視に関する項を参照してください

- topプロセス情報を収集し、topまたはpsコマンドを使用して、リカバリ・コーディネータ(PR00)がCPUバウンドかどうかをチェックします。

リソース・ボトルネックの一般的なインジケータおよび原因は次のとおりです。

- CPUアイドル時間が少ない場合は、システムがCPUバウンドであることを示している可能性があります
- ディスクまたはフラッシュのサービス時間が長いか、またはIOPSが高い場合は、I/Oの競合または飽和を示している可能性があります
- 多数のアクティブ・データベースを持つサイズ不足のシステムおよび共有システムでは、これらのリソースの競合が発生する可能性があります
- Active Data Guardスタンバイでワークロードをレポートする場合も、競合が発生する可能性があります

データベース待機イベントの評価によるREDO適用のチューニング

システム・リソース・ボトルネックがないことを確認したら、次に、スタンバイ自動作業リポジトリ(AWR)レポートを参照して、スタンバイ・データベースの待機イベントを評価します。

データベース待機イベントを評価する前に、リカバリに関連するプロセス・フロー中に待機が発生する場所を理解することが重要です。

1. REDOは、リモート・ファイル・サーバー(RFS)プロセスによってスタンバイで受信されます。

   RFSプロセスは、スレッドごとに新しく受信したREDOをそのスレッドの現在のスタンバイREDOログに書き込みます。RFS書込み操作は、rfs random I/O待機イベントによって追跡されます。

2. REDOが書き込まれると、リカバリ・コーディネータ・プロセス(pr00)がスレッドごとにスタンバイREDOログ(またはアーカイブ・ログ)からREDOを読み取ります。

   この読取りI/Oは、log file sequential read待機イベントによって追跡されます。

3. リカバリ・コーディネータは、すべてのスレッドのREDOをマージして、リカバリ・ワーカーのメモリー・バッファに配置します。

   リカバリ・メモリー・バッファへの書込みおよび読取りの待機イベントは、parallel recovery read buffer free待機イベントおよびparallel recovery change buffer free待機イベントによって追跡されます。

4. リカバリ・プロセスは、メモリー・バッファからREDOまたは変更ベクトルを取得し、変更をデータ・ブロックに適用するプロセスを開始します。

まず、リカバリ・ワーカーがリカバリを必要とするデータ・ブロックを判別し、そのブロックがバッファ・キャッシュにまだ存在しない場合には読み込みます。

リカバリ・ワーカーによるこの読取りI/Oは、recovery read待機イベントによって追跡されます。

5. いずれかのスレッドのプライマリでログが切り替えられると、スタンバイは同時にそのスレッドのスタンバイREDOログのスイッチを調整します。

以前のバージョンでは、スタンバイでログ・スイッチを実行すると、チェックポイント全体が強制的に実行されて、バッファ・キャッシュからスタンバイのデータ・ファイルにすべての使用済バッファがフラッシュされます。Oracle Database 18c以降では、チェックポイントも定期的に発生するため、すべてのフェーズにわたってチェックポイントI/Oが償却されます。

チェックポイントでは、複数のデータベース・ライター・プロセス(DBWR)は、db file parallel write待機イベントによって追跡される書込み時間とともに、データ・ファイル・ブロックをデータ・ファイルに書き込みます。チェックポイントが完了するまでの合計時間は、checkpoint complete待機イベントでカバーされます。

適用フェーズでは、通常、単一のCPUでリカバリ・コーディネータ・プロセス(pr00)の使用率が高くなりますが、チェックポイント・フェーズでは、DBライター・プロセス(dbwn)のCPU使用率が増加し、データファイルへの書込みI/Oの増加が示されます。

次の表は、リカバリ・プロセスに関連する待機イベントの説明とチューニングのアドバイスを示しています。

表15-6 リカバリ・プロセスの待機イベント

列	説明	チューニングに関する推奨事項
ログ・ファイル順次読取り	パラレル・リカバリ・コーディネータは、オンライン REDO ログ、スタンバイ REDO ログまたはアーカイブ REDO ログからの I/O を待機しています。	アーカイブ・ログ、スタンバイ REDO ログまたはオンライン REDO ログが存在する ASM ディスク・グループまたはストレージ・サブシステムの I/O 帯域幅をチューニングするか、拡大します。
Parallel recovery read buffer free	このイベントは、すべての読取りバッファがワーカーによって使用されていることを示し、通常はリカバリ・ワーカーがコーディネータに遅れていることを示します。	`_log_read_buffers` を最大の 256 に増やします
Parallel recovery change buffer free	パラレル・リカバリ・コーディネータは、リカバリ・ワーカーがバッファを解放するのを待機しています。このイベントも、リカバリ・ワーカーがコーディネータに遅れていることを示します。	データ・ファイルが存在する ASM ディスク・グループまたはストレージ・サブシステムの I/O 帯域幅をチューニングするか、拡大します。
Data file init write	パラレル・リカバリ・コーディネータは、ファイルの自動拡張などによるファイルのサイズ変更が終了するのを待機しています。	これはチューニング不可能なイベントですが、プライマリ・データベースのワークロードを評価して、データ・ファイルのサイズ変更操作が非常に多い理由を把握します。オプションで、AUTOEXTEND が有効な場合に、より大きな NEXT サイズを使用します
Parallel recovery	パラレル・リカバリ・コーディネータは、すべてのリカバリ・ワーカーが同期制御メッセージに応答するまで待機し	N/A。これはアイドル・イベントです。

列	説明	チューニングに関する推奨事項
control message reply	ています。	
Parallel recovery slave next change	パラレル・リカバリ・ワーカーは、コーディネータから変更が転送されるのを待機しています。このイベントは、実質的にはリカバリ・ワーカーのアイドル・イベントです。リカバリ・ワーカーが使用する CPU 時間を特定するには、このイベントの経過時間を起動済ワーカーの数で割り、その値を合計経過時間から引きます。	N/A。これはアイドル・イベントです。
DB File Sequential Read	パラレル・リカバリ・ワーカー(またはシリアル・リカバリ・プロセス)は、同期データ・ブロックのバッチ読取りが完了するのを待機しています。	データ・ファイルが存在する ASM ディスク・グループまたはストレージ・サブシステムの I/O 帯域幅をチューニングするか、拡大します。
Checkpoint completed	リカバリは、チェックポイントの完了を待機しており、REDO 適用は現時点で変更を適用していません。	データ・ファイルが存在する ASM ディスク・グループまたはストレージ・サブシステムの I/O 帯域幅をチューニングするか、拡大します。 また、`checkpoint completed` 待機イベントが db file parallel write 待機イベントより低くなるまで、db_writer_processes の数を増やします。 プライマリおよびスタンバイでのオンライン・ログ・ファイル・サイズを増やして、ログ・スイッチの境界における完全なチェックポイントの数を減らすことも検討します。
Recovery read	パラレル・リカバリ・ワーカーは、バッチ・データ・ブロック I/O を待機しています。	データ・ファイルが存在する ASM ディスク・グループの I/O 帯域幅をチューニングするか、拡大します。
Recovery apply pending and/or recovery receive buffer free (MIRA)	Recovery apply pending = 適用ワーカーがすべての保留中の変更を特定の SCN まで適用するのを、logmerger プロセスが待機した時間(センチ秒単位)。 Recovery receive buffer free = インスタンス上の受信者のプロセスが、受信したバッファを次の変更に備えて解放できるように、適用ワーカーが受信済バッファから変更を適用するのを待機するために費やした	`_mira_num_local_buffers` および `_mira_num_receive_buffers` を増やします これらのパラメータは、それ自体の値(MB)に適用インスタンスの数を乗算した合計に等しい共有プールの領域を使用します。

列	説明	チューニングに関する推奨事項
	時間(センチ秒単位)。	

スタンバイ・データベースでのAWRの生成の詳細は、How to Generate AWRs in Active Data Guard Standby Databases (Doc ID 2409808.1)を参照してください。

必要に応じたマルチインスタンスREDO適用の有効化

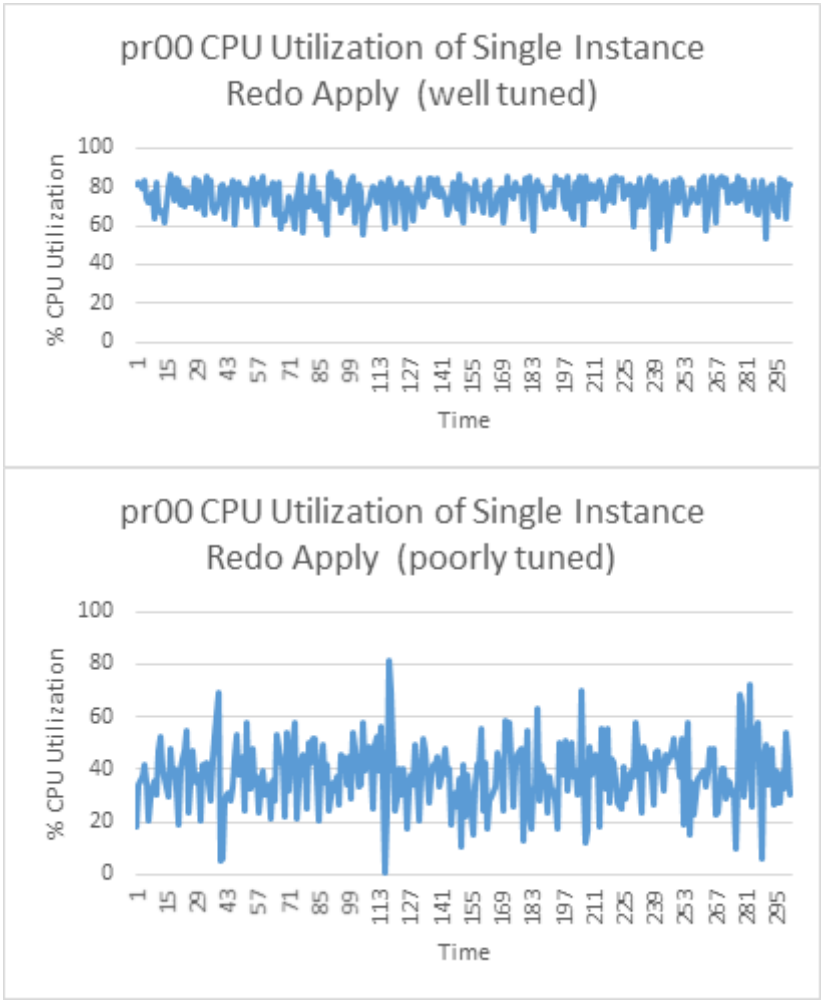
マルチインスタンスREDO適用(MIRA)には、スタンバイ・データベースのOracle RACデータベース・インスタンス全体で複数のリカバリ・コーディネータおよびREDO適用(ワーカー)プロセスを実行することで、REDO適用を改善できる可能性があります。MIRAはOracle Databaseの以降のリリース向けに最適化されており、REDO適用の利点はワークロードによって異なります。

MIRAを考慮する際の前提条件

- 単一インスタンスREDO適用(SIRA)は完全にチューニングされており、I/Oバウンドではありません。
- リカバリ・コーディネータ(PR00)はCPUバウンドです。

  1時間にわたり、リカバリ・コーディネータ/ログ・マージ・プロセスora_pr00_<SID>のCPU使用率を確認します。その時間の大部分で、コーディネータ・プロセスのCPU使用率%が70%を超える場合、ボトルネックが発生している可能性があり、MIRAによりリカバリ・パフォーマンスを改善できます。

  次に、pr00のCPU使用率を示す、topコマンドからの出力例を2つ示します。



図sira_poorly_tuned.pngの説明

リカバリ・コーディネータのCPU使用率が数回の短時間のスパイクのみで70%を大幅に下回る場合、CPUバウンドではありません。リソースに問題があるか、追加のチューニングによってパフォーマンスが改善する可能性があります。リカバリ・コーディネータがCPUバウンドでない場合は、SIRAのチューニングに戻ります。

- ほとんどのMIRA最適化はOracle Database 19cに実装されており、以前のデータベース・リリースでは使用できません。実際に、Oracle Database 19.13以降のデータベース・リリースを使用することをお薦めします。これは、このリリースには一部の重要な修正が含まれるためです(29924147、31290017、31047740、31326320、30559129、31538891、29785544、29715220、29845691、30421009、30412188、30361070、32486528、33821145、28389153など)。
- InfiniBandネットワーク・ファブリックまたはRDMA over Converged Ethernet (RoCE)ネットワーク・ファブリックに基づくすべてのOracle Exadata Database Machineシステムには、次の表に示すように、プライマリ・データベースで追加のステップが必要です。

  表15-7 MIRAを有効にするためのOracle Exadata Database Machineの前提条件

Exadataシステム	データベース・リリース	ステップ
永続メモリー(PMEM)のあるExadata ストレージ・セル	19.13 以降	追加ステップはありません
PMEM なし	19.13 以降	すべてのインスタンスに対して動的パラメータ `_cache_fusion_pipelined_updates_enable` =FALSE を設定します
任意の Exadata システム	19.12 以前	1. パッチ 31962730 を適用します 2. すべてのインスタンスに対して動的パラメータ `_cache_fusion_pipelined_updates_enable` =FALSE を設定します

ノート:

MIRA では、動的パラメータ`_cache_fusion_pipelined_updates_enable` または静的パラメータ `_cache_fusion_pipelined_updates` を FALSE に設定して生成された REDO のみをリカバリできます。

マルチインスタンスREDO適用およびチューニングの有効化

1. マルチインスタンスREDO適用(MIRA)を有効にするには、適用インスタンスの数を指定します。

   以前の単一インスタンスREDO適用(SIRA)のチューニング変更はすべてそのままにします。MIRAに対するMAAの推奨事項は、すべてのスタンバイ・データベース・インスタンスを適用に使用することです。

2. 次のいずれかの方法を使用して、MIRAを有効にします。
   - Oracle Data Guard Brokerプロパティを設定します

     `'ApplyInstances'=<#|ALL>`
   - または、次を実行します

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM
SESSION INSTANCES ALL;
```

3. MIRAのチューニング後にシステム・リソースの競合を確認します。

   「システム・リソース・ボトルネックの評価」で説明されている同じプラクティスに従います。

4. ここで説明されている待機イベントに基づいてMIRAをチューニングします。

   「データベース待機イベントの評価によるREDO適用のチューニング」の方法に従います。

   recovery apply pendingまたはrecovery receive buffer freeが待機イベントの上位にある場合は、次のようにします。

   - この待機イベントを減らすには、`_mira_num_receive_buffers`および`_mira_num_local_buffers`を100ずつ増分的に増やします。

     これらのパラメータは、インスタンス間でブロックを渡すための追加のバッファ領域を提供します。追加のバッファ領域に対応するためにSGAに十分なメモリーがあるかどうかを評価します。

     関連する各MIRA Oracle RACインスタンスの追加メモリー要件 = (_mira_num_receive_buffers + _mira_num_local_buffers) * (RACインスタンスの数 * 2) MB

     例: _mira_num_receive_buffers=500および_mira_num_local_buffers=500の場合、(500+500) *(4-node RAC *2) = SGAからの8000MB

   - `_mira_rcv_max_buffers=10000`を設定します

非常に大きいREDO適用ギャップの対処

適用ラグが24時間を超える場合は、すべてのREDOを適用するのではなく、スタンバイ・ロールフォワードの方法を使用してギャップをスキップすることを検討してください。How to Roll Forward a Standby Database Using Recover Database From Service (12.2 and higher) (Doc ID 2850185.1)を参照してください

このアプローチでは、変更されたOracleデータ・ブロックをプライマリ・データベースから直接プルし、すべてのREDOを適用するために必要な時間の半分に大きなREDOギャップを緩和できる可能性があります。

このアプローチの短所は次のとおりです。

- REDO適用およびスタンバイ・データベースに固有の、論理破損および書込み欠落の検出チェックおよびバランシングがスキップされます
- これらのコマンドを発行し、完了後にREDO適用を再開するには、手動操作が必要です。

データ・ブロックは、引き続き物理破損について検証されます。

データ保護を犠牲にしたREDO適用率の改善

プライマリに遅れないよう、さらに高いREDO適用率を達成するために、REDO適用をチューニングできない状況は非常にまれです。このような場合、REDO適用のパフォーマンスを向上させるために、推奨されるデータ保護設定をオフにすることが必要になる場合があります。

次の表は、いくつかの考えられる個別変更とその潜在的な利益およびトレードオフを示しています。

変更	潜在的な利益	潜在的なトレードオフ

変更	潜在的な利益	潜在的なトレードオフ
REDO 適用を停止し、サービスからのリカバリを使用する How to Roll Forward a Standby Database Using Recover Database From Service (12.2 and higher) (Doc ID 2850185.1)を参照してください	ギャップが 24 時間を超える場合など、大規模な REDO 転送または REDO 適用ギャップからリカバリするための最適化されたアプローチ	論理ブロックまたは書込み欠落のデータ保護チェックが行われません REDO ブロックのチェックサム検証が行われません
Active Data Guard ではなくスタンバイをマウントする	REDO 適用のパフォーマンスが 5-10% 向上する可能性はありますが、大部分はバッチ・ワークロードが対象です	物理破損の自動ブロック修復がリアルタイムで行われません スタンバイに対して問合せがリアルタイムで行われません スタンバイがアプリケーションのしきい値を超えて遅れている場合、前述のいずれのトレードオフも関連しない可能性があります
スタンバイで DB_BLOCK_CHECKING を無効化または削減する	REDO 適用中の CPU 使用率が削減されます CPU リソースが制限されている場合、この変更により REDO 適用が 10-40% 向上する可能性があります	まれに発生する可能性のある論理ブロック破損が検出されず、スタンバイに伝播される場合があります
フラッシュバック・データベースを無効化する	RECO のフラッシュバック IOPS 要件が排除されます ストレージ IOPS が制約リソースである場合、この変更が REDO 適用パフォーマンスに役立つ可能性があります	スタンバイを即時に巻き戻すことができなくなります
プライマリおよびスタンバイで DB_LOST_WRITE_PROTECT を無効にする	書込み欠落を検出するためにプライマリで生成されたブロック読取り REDO により、スタンバイでの追加の読取り IOPS が排除されます IOPS 容量が飽和状態の場合、この変更はオプションです	プライマリ・データベースまたはスタンバイ・データベースで、書込み欠落が早期に検出されません

ロール・トランジション、アセスメントおよびチューニング

綿密な計画、構成およびチューニングに基づいたOracle Data Guardのロール・トランジションにより、停止時間を効果的に最小化し、ビジネスへの影響を最小限に抑えながらデータベース環境をリストアできます。

フィジカル・スタンバイ・データベースを使用する場合、Oracle MAAのテスト結果では、Oracle Data Guardによるスイッチオーバーとフェイルオーバーの時間は数秒間に短縮されました。この項では、スイッチオーバーとフェイルオーバーのベスト・プラクティスについて説明します。ベスト・プラクティスに従う一方で、Oracle RACのスイッチオーバー時間は約30秒、単一インスタンス・データベースでは10秒未満であることが確認されています。検出時間は別々です。

ロール・トランジション前のData Guardヘルス・チェックの前提条件

次の前提条件を完了してから、スイッチオーバー操作を実行してください。

スイッチオーバーの1か月前

スイッチオーバー操作を実行する1か月以上前に、MOSノート「Oracle Database 19cの重要な推奨個別パッチ(ドキュメントID 555.1)」を参照して、リリースに影響する可能性がある重要な問題を特定してください。その後、既存の「Oracle Data Guardの構成ベスト・プラクティス」を参照して、推奨されるData Guard構成プラクティスがすべて適用されていることを確認します。

アプリケーション・フェイルオーバーが検証済であることを確認します。構成のガイダンスについては「アプリケーションの継続的な可用性の構成」を、PDBサービスの推奨事項については「Oracle Multitenantのベスト・プラクティスの概要」を参照してください。

Data Guardロール・トランジションの実行中にアプリケーション・サービスの可用性に影響を与える、一般的な構成の問題を次に示します。

- PDBの保存された状態またはトリガーが使用され、Data Guardロール・トランジション中に失敗します
- アプリケーション・サービスのPDBごとにOracleクラスタウェア管理の個別のサービスを使用するかわりに、PDBのデフォルト・サービスが利用されます
- ウォレット/セキュリティ設定がスタンバイで異なっています

アプリケーション・サービスとアプリケーション・フェイルオーバーの準備を確実にするには、次の点に留意します。

1. PDBのデフォルト・サービス、SAVED STATE (再配置操作時を除く)またはデータベース・トリガーを使用してロールベースのサービスを管理しないでください。
2. アプリケーション・サービスには、PDBごとにクラスタウェア管理の個別サービスを使用し、そのアプリケーション・サービスを利用してデータベースに接続します。
3. クラスタウェア管理のアプリケーション・サービスを定義する場合は、どのPDBとサービスを起動するか、どのOracle RACインスタンスとデータベース・ロールで起動するかを定義します。
4. Data Guardの場合、ロールを各クラスタウェア管理サービスに割り当てることで、常にロールベースのサービスを使用します。
5. Data Guardロール・トランジション後にエンドツーエンドのアプリケーション・フェイルオーバーをテストします

スイッチオーバーの数日前

1. Data Guardブローカのトレース・レベルを設定します。

   Data GuardブローカのTraceLevel構成プロパティは、構成のすべてのメンバーに対してブローカが実行するトレースの量を制御するために使用されます。プロパティをUSERに設定すると、トレース対象は、完了した操作と、操作また

はヘルス・チェックから発生する警告またはエラー・メッセージに制限されます。プロパティをSUPPORTに設定すると、問題のトラブルシューティングに必要な低レベルの情報が含まれ、トレースの量が増大します。

```
DGMGRL> SET TRACE_LEVEL SUPPORT;
```

2. ロール・トランジションのメトリックを有効にします。

時間管理インタフェース(TMI)イベントは、Oracleで特定のコールが実行されるたびにアラート・ログに行を追加するオーバーヘッドの低いイベントです。

アラート・ログ内のこれらのエントリ(またはタグ)は、コールの開始と終了を表します。次のトピックの表は、主要なスイッチオーバー操作とフェイルオーバー操作の説明を示しています。時間がどこに費やされているかを判断するには、これが最も正確な方法です。

すべてのデータベースで、データベース・レベル・イベントを16453、トレース名をcontext forever、レベルを15に設定します。このトレースを有効にする方法が2つあります。EVENTデータベース・パラメータを使用する方法と、システム・レベルでEVENTSを設定する方法です。違いは、EVENTパラメータが動的ではなく、再起動後も保持されることです。SET EVENTSは動的ですが、データベースの再起動後は保持されません。次の例を参照してください。

```
ALTER SYSTEM SET EVENT='16453 trace name contextforever, level 15' scope=spfile
sid='*';

ALTER SYSTEM SET EVENTS '16453 trace name context forever, level 15';
```

3. データベースのスイッチオーバーおよびフェイルオーバーの準備状況の検証

VALIDATEコマンドを使用して、ロール変更を実行する前に包括的なデータベース・チェックを実行できます。このコマンドは、次の項目を確認します。

- スタンバイ・データベースに、失われたREDOデータがあるかどうか
- フラッシュバックが有効かどうか
- 構成されている一時表領域ファイルの数
- オンライン・データ・ファイルの移動が進行中かどうか
- フィジカル・スタンバイ・データベースの場合、オンラインREDOログがクリアされているかどうか
- プライマリ・データベースの場合、スタンバイREDOログがクリアされているかどうか
- オンライン・ログ・ファイル構成
- スタンバイ・ログ・ファイル構成
- 適用関連のプロパティ設定
- 転送関連のプロパティ設定
- 自動診断リポジトリにエラーがあるかどうか(制御ファイルの破損、システム・データ・ファイルの問題、ユーザー・データ・ファイルの問題など)

スイッチオーバーの前に発行する必要がある、3つの主要なVALIDATEコマンドを次に示します。

- `VALIDATE DATABASE VERBOSE standby;`

  VALIDATE DATABASEコマンドは、データベースのサマリーを表示し、エラーまたは警告が検出されたらレポートします。VALIDATE DATABASE VERBOSEは、簡潔なサマリーの全内容に加え、検証されたすべての項目を表示します。

- `VALIDATE DATABASE standby SPFILE;`

  VALIDATE DATABASE SPFILEコマンドは、相違点がない場合はパラメータの違いが見つからないことを報告し、プライマリ・データベースと指定されたスタンバイ・データベースで値が異なるパラメータのリストを報告し

ます。

- VALIDATE NETWORK CONFIGURATION FOR ALL;

  VALIDATE NETWORK CONFIGURATIONコマンドは、構成のメンバー間のネットワーク接続性チェックを実行します。各接続性チェックの接続識別子は、関連付けられたデータベースのDGConnectIdentifierプロパティに基づいて生成されます。

例:

Oracle Data Guard BrokerのVALIDATE DATABASEコマンドは、スイッチオーバーおよびフェイルオーバーの準備状況に関連する情報を収集します。

スタンバイ・データベースとプライマリ・データベースがアクセス可能であることと、適用ラグがターゲット・データベースのApplyLagThreshold未満であることが検証されています。これらのデータ・ポイントが有益である場合は、次に示すように、コマンド出力に"Ready for Failover: Yes"と表示されます。また、REDO転送が実行中の場合は、コマンド出力に"Ready for Switchover: Yes"と表示されます。

```
DGMGRL> validate database [verbose] database_name
Database Role: Physical standby database
 Primary Database: standby_db_unique_name
Ready for Switchover: Yes
 Ready for Failover: Yes (Primary Running)
```

VALIDATE DATABASEは、オンラインREDOログがクリアされたかどうか、一時表領域の数、プライマリとスタンバイの間のパラメータの不一致、フラッシュバック・データベースのステータスなど、スイッチオーバー時間およびデータベース・パフォーマンスに影響する可能性のある追加情報をチェックします。

ほとんどのフェイルオーバーでは、プライマリ・データベースがクラッシュしているか、使用できなくなっています。Ready for Failoverという出力は、VALIDATE DATABASEが発行されたときにプライマリ・データベースが実行されているかどうかを示します。この状態からのフェイルオーバーは可能ですが、構成にプライマリ・データベースが2つあるスプリット・ブレインというシナリオを回避するため、フェイルオーバーを発行する前にプライマリ・データベースを停止することをお薦めします。ファスト・スタート・フェイルオーバーが使用されている場合、ブローカではフェイルオーバー時のスプリット・ブレインの回避のみが保証されます。

また、構成監視ツールとして、VALIDATE DATABASE VERBOSE standby、VALIDATE DATABASE standby SPFILE;およびVALIDATE NETWORK CONFIGURATION FOR ALL;を定期的に実行する必要があります。

Data Guard Brokerを使用したロール・トランジションの開始

Oracle Data Guard BrokerのSWITCHOVERコマンドを使用してスイッチオーバーを開始し、FAILOVERコマンドを使用してフェイルオーバーを開始します。

スイッチオーバー操作またはフェイルオーバー操作の一環として、ブローカは次のことを実行します。

- 新しいプライマリ・データベースからREDO転送を構成します
- 新しいスタンバイ・データベースでREDO適用を開始します
- ブローカ構成内の他のスタンバイ・データベースが実行可能で、新しいプライマリからREDOを受信していることを確認します
- Oracle ClusterwareとGlobal Data Servicesを統合して、ロール変更後に正しいサービスが開始されるようにします

Data Guardスイッチオーバーを発行する前に、長時間実行されるレポート作成やジョブ(永続的な接続を作成する監視、監

査、データベース・バックアップなど)を一時停止または停止します。

スイッチオーバーを開始するようにブローカを構成するには、SYSまたはSYSDBAとしてログインして、次を発行します。

```
DGMGRL> SWITCHOVER TO database_name;
```

フェイルオーバーを開始するようにブローカを構成するには、次を実行します:

```
DGMGRL> FAILOVER TO database_name [IMMEDIATE];
```

デフォルトでは、FAILOVERはフェイルオーバー前に受信したすべてのREDOを適用します。IMMEDIATE句を指定すると、保留中のREDOがスキップされ、すぐにフェイルオーバーされます。

SWITCHOVERコマンドおよびFAILOVERコマンドは多重呼出し不変であり、万一遷移が失敗した場合には再発行できます。

Data Guardロール・トランジションの監視

Data Guardロール・トランジションが行われている間は、Data Guard Brokerのメッセージを参照してください。詳細なロール・トランジションのステータスを抽出するには、プライマリおよびスタンバイのアラート・ログと、Data Guardスイッチオーバーおよびフェイルオーバーのメッセージとタグのブローカ・ログを参照してください。

主要なスイッチオーバー操作とアラート・ログ・タグ

スイッチオーバーは、次の4つの主なステップに分けられます。

1. `Convert to Standby` - 既存の本番セッションを終了し、制御ファイルをスタンバイ制御ファイルに変換し、スタンバイにメッセージを送信してスイッチオーバーを続行します。

   `Convert to Standby` - これらのステップは、元のプライマリのアラート・ログにあります。残りのステップはすべて、元のスタンバイ・アラート・ログにあります。

2. `Cancel Recovery` - 残りのREDOを適用し、リカバリを停止します。
3. `Convert to Primary` - インスタンス(1つのインスタンス、次に他のすべてのインスタンス)の(マウント状態への)2段階クローズ、オンラインREDOログのクリア、制御ファイルのプライマリ制御ファイルへの変換およびData Guard Brokerのブックキーピング。
4. `Open New Primary` - すべてのインスタンスのパラレル・オープン。

表15-8 時間管理インタフェース・イベントが有効なステップを定義するアラート・ログ・タグ

ステップ	ステージ	有効な時間管理インタフェース・イベント
Convert To Standby(primary alert log)	BEGIN	TMI: dbsdrv switchover to target BEGIN <DATE> <TIMESTAMP>
Convert To Standby(primary alert log)	END	TMI: kcv_switchover_to_target send 'switchover to primary' msg BEGIN <DATE> <TIMESTAMP>
Cancel Recovery(standby alert log)	BEGIN	TMI: kcv_commit_to_so_to_primary wait for MRP to die BEGIN <DATE> <TIMESTAMP>

ステップ	ステージ	有効な時間管理インタフェース・イベント
Cancel Recovery(standby alert log)	END	TMI: kcv_commit_to_so_to_primary wait for MRP to die END <DATE> <TIMESTAMP>
Convert to Primary (standby alert log)	BEGIN	TMI: kcv_commit_to_so_to_primary BEGIN CTSO to primary <DATE> <TIMESTAMP>
Convert to Primary (standby alert log)	END	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary(standby alert log)	BEGIN	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary(standby alert log)	END	TMI: adbdrv END 10 <DATE> <TIMESTAMP>

主要なフェイルオーバー操作とアラート・ログ・タグ

すべてのフェイルオーバー・ステップは、フェイルオーバーが実行されたターゲット・スタンバイのアラート・ログに記載されています。

1. Cancel Recovery - リカバリを停止し、(マウントされている)すべてのインスタンスをパラレルにクローズします。
2. Terminal Recovery - スタンバイREDOログをアーカイブし、未適用のREDOをリカバリします。
3. Convert to Primary - オンラインREDOログをクリアし、制御ファイルをスタンバイ制御ファイルに変換します。
4. Open Primary - すべてのインスタンスをパラレルにオープンします。

表15-9 時間管理インタフェース・イベントが有効なステップを定義するフェイルオーバー・アラート・ログ・タグ

ステップ	ステージ	有効な時間管理インタフェース・イベント
Cancel Recovery	BEGIN	TMI: adbdrv termRecovery BEGIN <DATE> <TIMESTAMP>
Cancel Recovery	END	TMI: adbdrv termRecovery END <DATE> <TIMESTAMP>
Terminal Recovery	BEGIN	TMI: krdsmr full BEGIN Starting media recovery <DATE> <TIMESTAMP>
Terminal Recovery	END	TMI: krdemr full END end media recovery <DATE> <TIMESTAMP>
Convert to Primary	BEGIN	TMI: kcv_commit_to_so_to_primary BEGIN CTSO to primary <DATE> <TIMESTAMP>
Convert to Primary	END	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary	BEGIN	TMI: adbdrv BEGIN 10 <DATE> <TIMESTAMP>
Open Primary	END	TMI: adbdrv END 10 <DATE> <TIMESTAMP>

ロール・トランジション後の検証

SHOW CONFIGURATION VERBOSEコマンドを使用して、スイッチオーバーまたはフェイルオーバーと、スタンバイの回復が成功したことを確認します。

```
DGMGRL> SHOW CONFIGURATION VERBOSE;
Configuration - DRSolution
Protection Mode: MaxAvailability
Members:
    South_Sales - Primary database
    North_Sales - Physical standby database
    Fast-Start Failover: DISABLED
    Configuration Status:
     SUCCESS
```

スイッチオーバー操作時の問題のトラブルシューティング

Data Guardのスイッチオーバーまたはフェイルオーバー操作の失敗後に最も重要となる目標は、データベースとアプリケーションの可用性を可能なかぎり早く再開することです。

診断情報のソース

Oracle Data Guard Brokerのアクティビティに関する情報は、いくつかの形式で提供されます。

- データベース・ステータスの情報 - SHOW DATABASE VERBOSE db_unique_nameコマンドを使用して、データベースの簡単な説明(名前、ロールなど)、データベースの状態、健全性チェックの問題に関する情報を入手できます。

  ```
  DGMGRL> SHOW DATABASE VERBOSE db_unique_name
  ```

- Oracleアラート・ログ・ファイル - ブローカでは、ブローカ構成に含まれる各データベースのインスタンスごとのアラート・ログ・ファイルに重要情報が記録されます。Oracle Data Guardをトラブルシューティングする場合は、このような情報のアラート・ログ・ファイルを確認できます。
- Oracle Data Guard "ブローカ・ログ・ファイル" - ブローカ構成に含まれる各データベースのインスタンスごとに、ブローカのDMONプロセスにより重要な動作とステータス情報がブローカ・ログ・ファイルに記録されます。このファイルはOracle Data Guardの障害を診断する際に役立ちます。TraceLevel構成プロパティは、ブローカ・ログ・ファイルで報告される診断情報のレベルを指定するために使用されます。ブローカ・ログ・ファイルは、アラート・ログと同じディレクトリに作成され、drc<$ORACLE_SID>.logという名前が付けられます。

初期問題の修正後のスイッチオーバーの再試行

報告された問題がすぐに修正できる場合は、スイッチオーバー操作を再試行できます。

報告された問題が修正できない場合や修正後もスイッチオーバー操作が失敗する場合は、スイッチオーバー用に別のデータベースを選択するか、構成をスイッチオーバー前の状態にリストアしてからスイッチオーバーを再試行します。または、「スイッチオーバー失敗後のロールバックによる稼働時間の最大化」を参照してください。

```
DGMGRL> SWITCHOVER TO database_name;
```

スイッチオーバー失敗後のロールバックによる稼働時間の最大化

フィジカル・スタンバイ・データベースでエラーが発生し、適切なタイミングでスイッチオーバーを続行できない場合は、新しいフィジカル・スタンバイ・データベースをプライマリ・ロールに戻して、データベースの停止時間を最小限に抑えます。

次のステップを実行します。

1. 新しいスタンバイ・データベース(古いプライマリ)を停止し、マウントします。
2. 新しいスタンバイ・データベースでREDO Applyを開始します。
3. 新しいスタンバイ・データベースがいつでもプライマリ・ロールに戻せることを確認します。

   スタンバイ・データベースでV$DATABASEビューのSWITCHOVER_STATUS列を問い合せます。値TO PRIMARYまたはSESSIONS ACTIVEは、スタンバイ・データベースでプライマリ・ロールに切り替える準備が完了していることを示します。値TO PRIMARYまたはSESSIONS ACTIVEが戻されるまで、この列の問合せを続行します。

4. 次の文を発行して、新しいスタンバイ・データベースを変換してプライマリ・ロールに戻します。

```
SQL> ALTER DATABASE SWITCHOVER TO target_db_name;
```

   ステップ4が失敗した場合は、失敗したスイッチオーバーのロールバックとやり直しを参照してください。

Data Guardのパフォーマンス観測

Data Guardロール・トランジション期間

Oracle Data GuardおよびOracle MAA Goldリファレンス・アーキテクチャは、プライマリ・データベース、クラスタまたはサイトに障害が発生した場合やアクセスできない場合に、障害時リカバリおよび高可用性ソリューションを提供します。

Data Guard環境はそれぞれ異なり、ロール・トランジションを実行する時間は大きく異なる場合があります。SGAサイズ、Oracle RACインスタンスの数、PDBの数、データ・ファイルおよびロール・トランジション時のデータベースへの接続を含む(これらに限定されない)変数は、特定のロール・トランジションの長さに影響します。

通常、Data Guardのスイッチオーバー(計画メンテナンス)は、Data Guardフェイルオーバー(計画外停止)より少し長くなります。

次の情報は、ロール・トランジションを最適化する方法を学習することを目的としています。

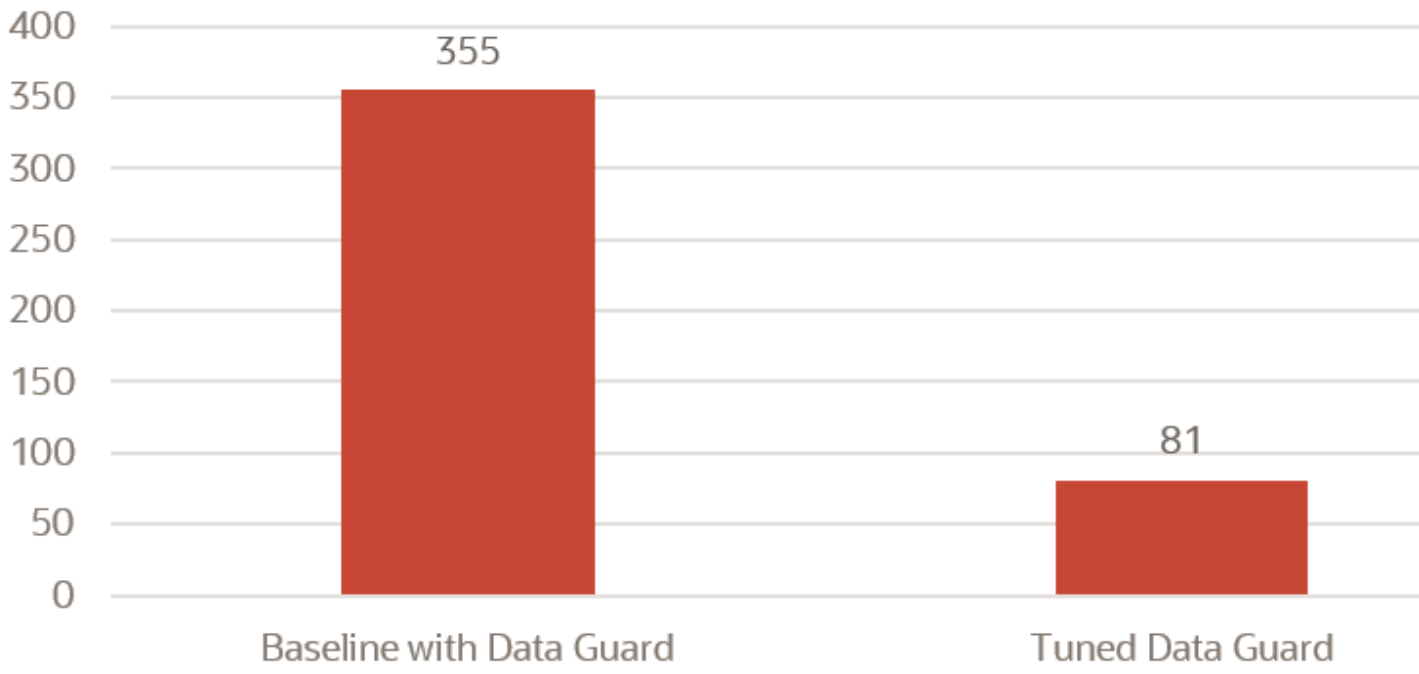
Data Guardスイッチオーバー期間

計画メンテナンスのアプリケーション停止時間を最小化しようとする場合:

- 計画メンテナンス期間の前に、バッチ・ジョブまたは実行時間が長いレポートを回避または遅延します。ピーク処理期間も回避する必要があります。
- Data Guardスイッチオーバーは正常であり、ソース・プライマリ・データベースの停止が必要であるため、すべてのアプリケーション排出タイムアウトを考慮します。Oracle Clusterwareサービスの排出属性および設定については、「アプリケーションの継続的なサービスの有効化」を参照してください。
- 単一インスタンス(非RAC)でのData Guardスイッチオーバー操作は、30秒未満にできます。
- Real Application ClusterでのData Guardスイッチオーバー操作は異なりますが、30秒から7分にすることができます。より多くのPDB (25を超えるPDBなど)がある場合、より多くのアプリケーション・サービス(200サービスなど)がある場合、およびデータベースに多数のデータ・ファイル(1000個のデータ・ファイルなど)がある場合、期間は長くなることがあります。

次のグラフと表は、MAAチューニング推奨事項を実装したときにスイッチオーバー操作時間がどれだけ短縮する可能性があるかの一例を示しています。結果は異なります。

図15-1 計画メンテナンス: DRスイッチの期間(秒)



計画DRスイッチ(スイッチオーバー)	初期構成	チューニングされたMAA構成
プライマリをスタンバイに変換	26 秒	21 秒
スタンバイをプライマリに変換(C2P)	47 秒	7 秒
新規プライマリのオープン(OnP)	152 秒	14 秒
PDB のオープンおよびサービスの開始(OPDB)	130 秒	39 秒
合計アプリケーション停止時間	355 秒(5 分 55 秒)	81 秒(78%低下)

「チューニングした」時間は、次のMAA推奨プラクティスを実装することで達成されています。

- ビッグファイル表領域の使用
- Oracle Data Guardの構成ベスト・プラクティス
- ロール・トランジション、アセスメントおよびチューニング

Data Guardフェイルオーバー期間

DRシナリオでアプリケーションの停止時間を最小化しようとする場合:

- リカバリ時間目標(RTOまたはデータベースの停止時間)およびリカバリ・ポイント目標(RPOまたはデータ損失)を制限するには、自動検出およびフェイルオーバーが必要です。『Oracle Data Guard Broker』のファスト・スタート・フェイルオーバーを参照してください。
- データベース管理者は、`FastStartFailoverThreshold`を設定して、自動フェイルオーバーを開始する前に適切な検出時間を決定できます。『Oracle Data Guard Broker』のファスト・スタート・フェイルオーバーの有効化タスク4: FastStartFailoverThreshold構成プロパティの設定を参照してください。

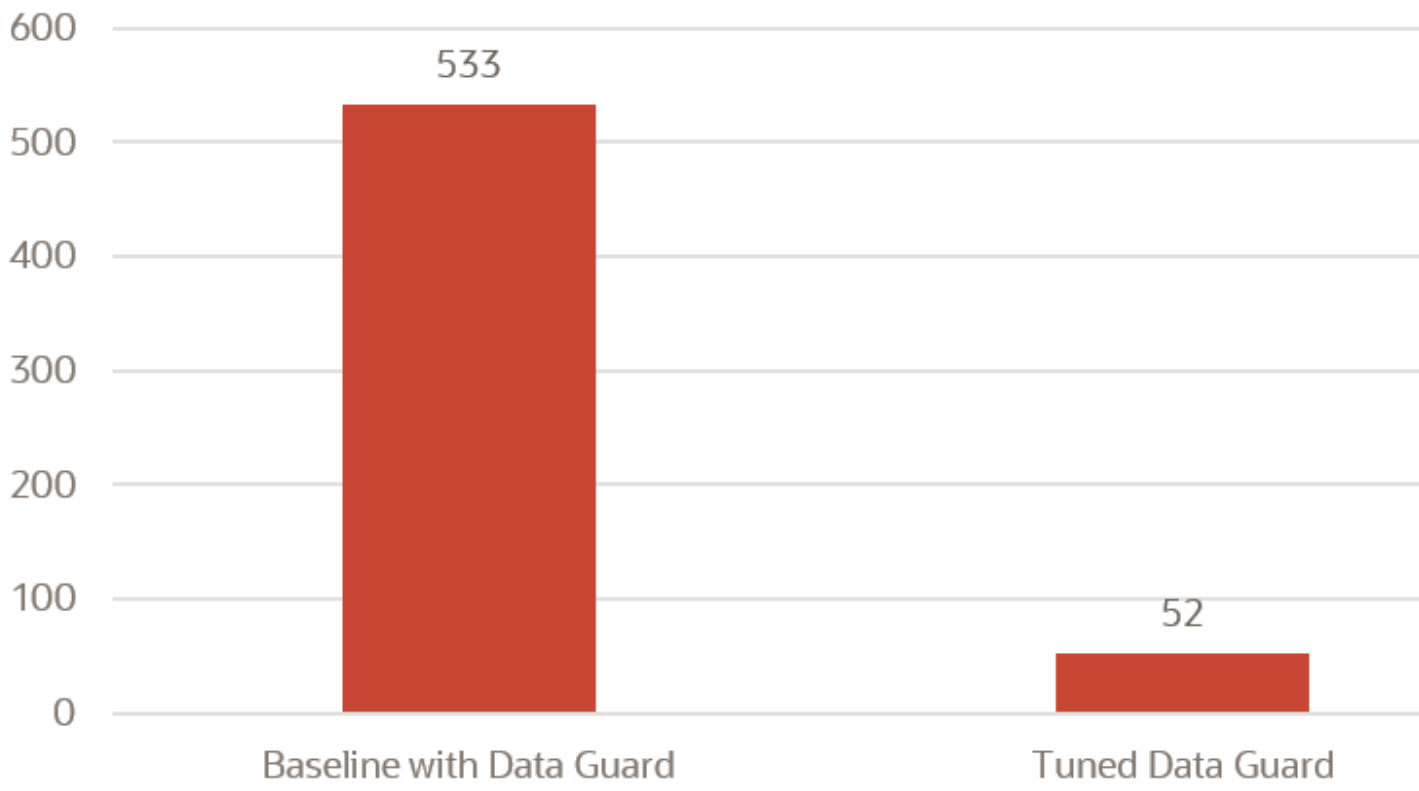
  信頼性の高いネットワークの場合、MAAの推奨設定は5秒から60秒です。Oracle RACを再起動すると、プライマリの一時エラーからもリカバリできます。このしきい値を高く設定すると、再起動が完了してフェイルオーバーを回避できます

が、一部の環境では負担が大きくなる可能性があります。トレードオフは、実際のフェイルオーバーが必要な場合にアプリケーションの停止時間が増加するということです。

- 単一インスタンス(非RAC)でのData Guardフェイルオーバー操作は、20秒未満にできます。
- Real Application ClusterでのData Guardのフェイルオーバー操作は異なりますが、20秒から7分にすることができます。より多くのPDB (25を超えるPDBなど)がある場合、より多くのアプリケーション・サービス(200サービスなど)がある場合、およびデータベースに多数のデータ・ファイル(1000個のデータ・ファイルなど)がある場合、期間は長くなる場合があります。

次のグラフと表は、MAAチューニング推奨事項を実装したときにフェイルオーバー操作時間がどれだけ短縮する可能性があるかの一例を示しています。結果は異なります。

図15-2 計画外DRフェイルオーバーの期間(秒)



計画外停止/DR (フェイルオーバー)	初期構成	チューニングされたMAA構成
クローズしてマウント(C2M)	21 秒	1 秒
ターミナル・リカバリ(TR)	154 秒	2 秒
プライマリに変換(C2P)	114 秒	5 秒
新規プライマリのオープン(OnP)	98 秒	28 秒
PDB のオープンおよびサービスの開始(OPDB)	146 秒	16 秒
合計アプリケーション停止時間	533 秒(8 分 53 秒)	52 秒(90%低下)

「チューニングした」時間は、次のMAA推奨プラクティスを実装することで達成されています。

- Data Guardのファスト・スタート・フェイルオーバーを評価し、異なるFastStartFailoverThreshold設定でテストします
- ビッグファイル表領域の使用
- Oracle Data Guardの構成ベスト・プラクティス
- ロール・トランジション、アセスメントおよびチューニング

カスタマの事例

次の表に、Oracleカスタマからの実際のData Guardロール・トランジション期間の観測を示します。

プライマリおよびData Guardの構成	観測されたRTO
Database Cloud (DBCS)で大量の OLTP ワークロードを使用する単一インスタンス・データベースのフェイルオーバー。Data Guard のしきい値は 5 秒です。	20 秒
大量の OLTP ワークロードを使用する 4 ノード RAC による大規模な商業銀行 POC の結果	51 秒(計画外 DR) 82 秒(計画 DR)
大量の OLTP ワークロードを使用する ExaDB-D の 2 ノード RAC MAA テスト	78 秒
大量の OLTP ワークロードを使用する ADB-D の 2 ノード RAC MAA テスト(25 PDB、250 サービス)	104 秒
大量の OLTP ワークロードを使用する ADB-D の 2 ノード RAC MAA テスト(50PDB、600 サービス)	180 秒
大量の OLTP ワークロードを使用する ADB-D の 2 ノード RAC MAA テスト(4 CDB、合計 100 PDB、500 サービス)	164 秒
12 ノード RAC、400GB の SGA、4000 以上のデータ・ファイルの Oracle SaaS フリート(数千) (ノート: データ・ファイルの数を数百に削減すると、停止時間を数分短縮できます)	6 分未満
四半期ごとのサイト・スイッチを使用する 7-12 ノード RAC のサード・パーティの SaaS フリート(数千)	5 分未満

Data Guardによるアプリケーション・スループットおよびレスポンス時間の影響

Data Guardの最大パフォーマンス保護モードまたはASYNC転送を有効にすると、アプリケーションのスループットおよびレスポンス時間の影響はほぼゼロになります。スループットおよびアプリケーション・レスポンス時間は、通常、このような場合にはまったく影響しません。

Data Guardの最大可用性モード、最大保護モードまたはSYNC転送では、アプリケーションのパフォーマンスへの影響が異なります。そのため、SYNC転送を有効にする前に、アプリケーションのパフォーマンス・テストが常に推奨されます。チューニングされたネットワークおよび低ラウンドトリップ待機時間(RTT)では、データ損失ゼロ・ソリューションを保持するために、パラレルに使用可能なすべてのSYNCスタンバイ・データベースに対してすべてのログ・コミットを確認する必要がある場合でも、影響は無視できることもあります。

次に、アプリケーションのスループットへの影響の例を示しますが、アプリケーションの影響はワークロードによって異なります。

図15-3 MTU=9000によるアプリケーションの影響

ネットワークRTT待機時間(x軸)が低いと、アプリケーション(TPSまたはy軸)のスループットが低下することに注意してください。

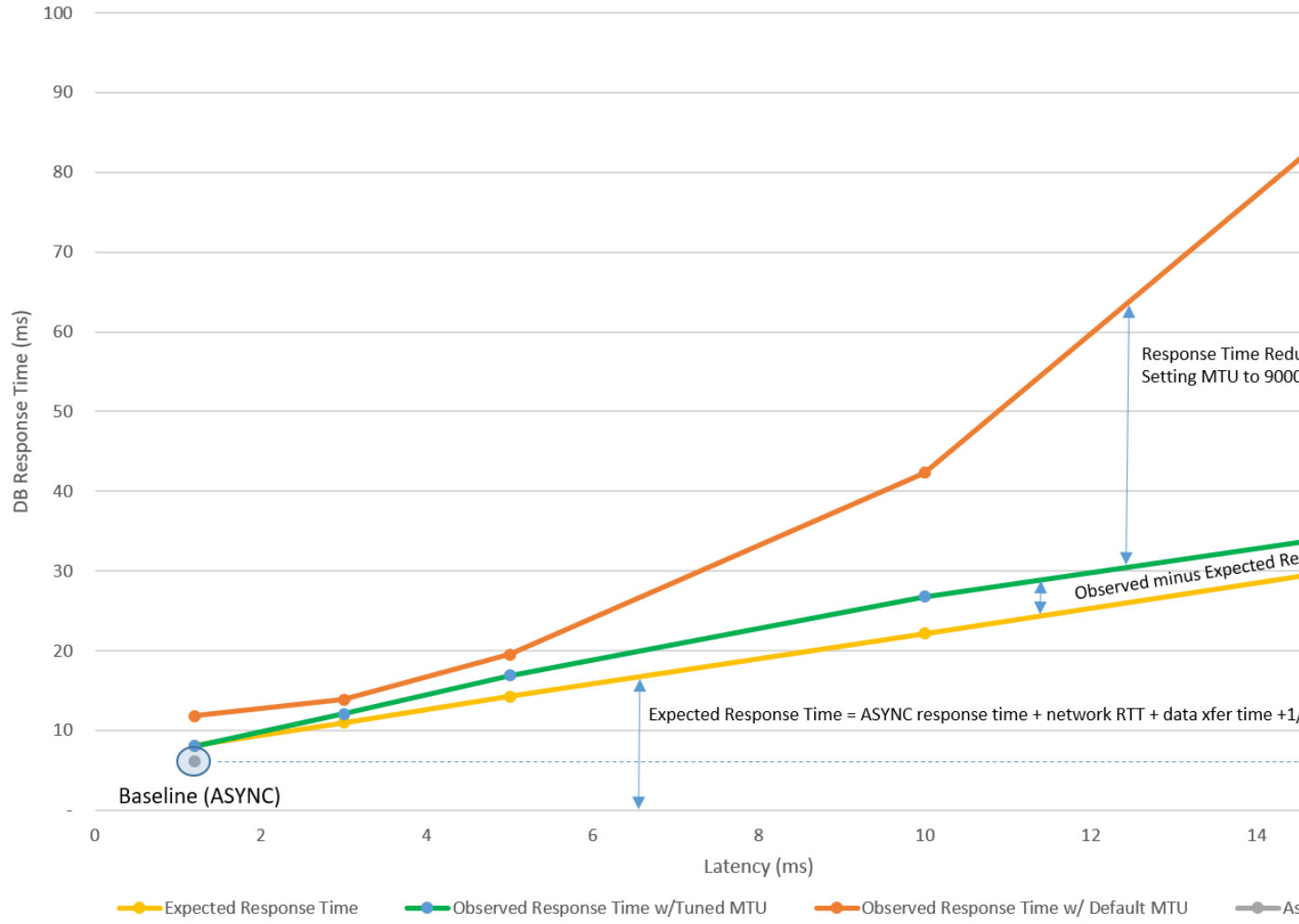
このネットワーク環境では、ログ・メッセージ・サイズがSYNCで大幅に増加したため、MTUを1500 (デフォルト)から9000 (ジャンボ・フレームなど)に増やすと、大幅に向上しました。MTUサイズが大きいほど、REDO送信リクエスト当たりのネットワーク・パケットの数が少なくなります。

ソケット・バッファ・サイズやMTUを含むネットワークのチューニングの詳細は、「ネットワーク・パフォーマンスの評価および最適化」を参照してください。

RTT待機時間が長くなってスループットが大幅に低下しても、アプリケーションで同時実行性が増大する可能性がある場合は、TPSを増やすことができます。前述のチャートでは、最後の2つの列で、ユーザーを追加することでワークロードの同時実行性が向上しました。

SYNC転送を使用したアプリケーション・レスポンス時間は長くすることもできますが、各アプリケーションのワークロードおよびネットワーク・チューニングによって異なります。SYNC転送では、すべてのログ書込みがスタンバイSYNC確認応答を待機する必要があります。この追加の待機によって、コミットの確認応答を待機するフォアグラウンドが増えます。コミットはスタンバイ・データベースで確認する必要があり、より多くのフォアグラウンドがコミットを待機しているため、次のチャートに示すように、平均ログ書込みサイズが大きくなると、REDO/データ転送時間に影響を与えます。

図15-4 チューニング済およびデフォルトMTUのデータベース・レスポンス時間(ミリ秒)と待機時間(ミリ秒)



この例では、AWRレポートから、平均REDO書込みサイズが大幅に増え、チューニングMTUによってレスポンス時間の影響が削減されたことがわかりました。ソケット・バッファ・サイズやMTUを含むネットワークのチューニングについては、「ネットワーク・パフォーマンスの評価および最適化」を参照してください。

ネットワークのチューニング後、レスポンス時間の影響が予測可能になり、少なくなりました。レスポンスの影響は、アプリケーションのワークロードによって異なります。

Data Guardで最高のアプリケーション・パフォーマンスを得るには、次のプラクティスを使用します。

- 最初にData Guardを使用せずにアプリケーションをチューニングします。ASYNC転送についても同様のパフォーマンスが確認されます
- 「Oracle Data Guardの構成ベスト・プラクティス」を実装します
- 「REDO転送のトラブルシューティングおよびチューニング」の方法の使用
- SYNCを使用してネットワークをチューニングし、アプリケーションのパフォーマンスを向上させます。「ネットワーク・パフォーマンスの評価および最適化」を参照してください

- SYNC転送のスループットの向上に役立つアプリケーション・ワークロード固有の変更は次のとおりです。
  - 同時実行性またはユーザーの追加を評価して、スループットを向上させます。
  - データ損失ゼロを必要としない特定のセッション内の重要でないワークロードの場合は、拡張COMMIT_WRITE属性をNOWAITに評価します。

    この場合、確認応答を受信する前にコミットできます。REDOは永続REDOログに送信されますが、非同期で実行されます。リカバリは、適用されるREDO内のすべての永続コミット済トランザクションに対して保証されます。『Oracle Databaseリファレンス』のCOMMIT_WRITEを参照してください。

16 Oracle Data Guard構成の監視

Oracle Data Guard構成を監視するには、次のOracle MAAベスト・プラクティスの推奨事項を使用します。

ブローカを使用したOracle Data Guard構成ヘルスの監視

Oracle Data Guard Brokerは、ヘルス・チェックを1分に1回発行し、構成ステータスを更新します。ヘルス・チェックをすぐに強制的に実行するには、コマンドshow configuration verboseを実行します。

プライマリ・データベースのヘルス・チェックでは、次の条件が満たされているかどうかを判別します。

- データベースが、ユーザーが指定してブローカ構成ファイルに記録された状態であるかどうか。
- データベースが正しいデータ保護モードであるかどうか。
- データベースでサーバー・パラメータ・ファイル(SPFILE)が使用されているかどうか
- データベースがARCHIVELOGモードであるかどうか。
- REDO転送サービスにエラーがあるかどうか。
- データベース設定が、ブローカの構成可能なプロパティで指定された設定と一致しているかどうか。
- REDO転送設定が、スタンバイ・データベースのREDO転送関連プロパティで指定された設定と一致しているかどうか。
- 現行のデータ保護レベルが構成済のデータ保護モードと一致しているかどうか。
- プライマリ・データベースで、すべてのスタンバイ・データベースのすべてのギャップを解決できるかどうか。

スタンバイ・データベースのヘルス・チェックでは、次の条件が満たされているかどうかを判別します。

- データベースが、ユーザーが指定してブローカ構成ファイルに記録された状態であるかどうか。
- データベースでサーバー・パラメータ・ファイル(SPFILE)が使用されているかどうか
- データベース設定が、ブローカの構成可能なプロパティで指定された設定と一致しているかどうか。
- ファスト・スタート・フェイルオーバーが有効化されている場合は、プライマリ・データベースおよびターゲット・スタンバイ・データベースが同期化されているかどうか、またはラグ制限内であるかどうか。

構成全体の警告を確認するには、SHOW CONFIGURATIONコマンドを使用してステータスを表示します。

```
DGMGRL> show configuration;
Configuration – dg
  Protection Mode: MaxPerformance
  Members:
  tin - Primary database
    can - Physical standby database
Fast-Start Failover: DISABLED
Configuration Status:
SUCCESS   (status updated 18 seconds ago)
```

構成ステータスがSUCCESSの場合は、ブローカ構成全体が正常に動作しています。

一方、WARNINGまたはERRORステータスが表示される場合は、構成の一部に問題があることを意味します。追加のエラー・メッセージは、現在の問題を特定するのに使用するWARNINGステータスまたはERRORステータスを伴います。

次のステップでは、構成内の各データベースを調べて、特定のエラーに関連する内容を絞り込みます。

プライマリ・データベースの警告を確認するには、SHOW DATABASEコマンドを使用してステータスを取得します。

```
DGMGRL> show database tin
Database – tin
  Role:               PRIMARY
  Intended State:     TRANSPORT-ON
  Instance(s):
```

```
    tin1
    tin2
Database Status:
SUCCESS
```

データベース・ステータスがSUCCESSの場合、データベースは正常に動作しています。

一方、WARNINGまたはERRORステータスが表示される場合は、データベースの一部に問題があることを意味します。追加のエラー・メッセージは、現在の問題を特定するのに使用するWARNINGステータスまたはERRORステータスを伴います。

スタンバイ・データベースで同じSHOW DATABASEコマンドを繰り返し、エラー・メッセージを評価します。

前述のコマンドに加えて、ブローカにはVALIDATE DATABASEコマンドもあります。

```
DGMGRL> validate database tin
  Database Role:    Primary database
  Ready for Switchover:  Yes
DGMGRL> validate database can;
  Database Role:     Physical standby database
  Primary Database:  tin
  Ready for Switchover:  No
  Ready for Failover:    Yes (Primary Running)
  Capacity Information:
    Database  Instances        Threads
    tin       2                2
    can       1                2
    Warning: the target standby has fewer instances than the
    primary database, this may impact application performance
  Standby Apply-Related Information:
    Apply State:      Not Running
    Apply Lag:        Unknown
    Apply Delay:      0 minutes
```

VALIDATE DATABASEは、SUCCESSステータスまたはWARNINGステータスを提示しないため、アクションを実行する必要があるかどうかを判断するための調査が必要になります。

ブローカ構成の作成後、およびロール・トランジション操作の前後に、VALIDATE DATABASEコマンドを実行することをお薦めします。

VALIDATE DATABASEコマンドにより、次のチェックが実行されます。

- スタンバイ・データベースに、失われたREDOデータがあるかどうか
- フラッシュバックが有効かどうか
- 構成されている一時表領域ファイルの数
- オンライン・データ・ファイルの移動が進行中かどうか
- フィジカル・スタンバイ・データベースの場合、オンラインREDOログがクリアされているかどうか
- プライマリ・データベースの場合、スタンバイREDOログがクリアされているかどうか
- オンライン・ログ・ファイル構成
- スタンバイ・ログ・ファイル構成
- 適用関連のプロパティ設定
- 転送関連のプロパティ設定
- 自動診断リポジトリにエラーがあるかどうか(制御ファイルの破損、システム・データ・ファイルの問題、ユーザー・データ・ファイルの問題など)

Oracle Data Guard Brokerを使用した転送または適用ラグの検出

十分なリソース(特にネットワーク帯域幅)があれば、Oracle Data Guardスタンバイは非常に高いワークロードを維持できます。

リソースが制約されている場合、スタンバイが遅れ始め、転送または適用ラグが発生する可能性があります。

転送ラグとは、スタンバイがプライマリから受信していない、時間で測定されたデータ量です。

適用ラグとは、最後に適用された変更がスタンバイで表示可能になってから、同じ変更がプライマリで最初に表示可能になるまでの時間の差(経過時間)です。

Data Guard Brokerを使用する場合、次に示すように、SHOW DATABASEコマンドを使用してスタンバイ・データベースを参照することで、転送または適用ラグを表示できます。

```
DGMGRL> show database orclsb
Database – orclsb
  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 0 seconds ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 792.00 KByte/s
  Real Time Query:    ON
  Instance(s):
    orclsb1 (apply instance)
    orclsb2
Database Status:
SUCCESS
```

ブローカのTransportDisconnectedThresholdデータベース・プロパティ(Oracle Database 11.2ではデフォルトは0、Oracle Database 12.1以降のリリースでは30秒)は、プライマリ・データベースからの最後の通信がプロパティによって指定された値を上回ったときに、スタンバイについて警告ステータスを生成するために使用できます。プロパティ値を表す単位は秒です。

切断が発生した場合の警告の例を次に示します。

```
DGMGRL> show database orclsb;
Database – orclsb
  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 981 seconds ago)
  Apply Lag:          0 seconds (computed 981 seconds ago)
  Average Apply Rate: 12.00 KByte/s
  Real Time Query:    OFF
  Instance(s):
    orclsb1 (apply instance)
    orclsb2
  Database Warning(s):
    ORA-16857: member disconnected from redo source for longer than specified
threshold
```

ブローカには、転送または適用ラグがユーザー定義の値を上回ったっときに警告を生成するために使用できる、次の構成可能なデータベース・プロパティもあります。

- ApplyLagThresholdプロパティは、データベースの適用ラグがプロパティで指定された値を上回ったときに、ロジカルまたはフィジカルのスタンバイについて警告ステータスを生成します。

  プロパティ値を表す単位は秒です。値が0秒の場合、適用ラグが存在しても警告を生成しません。ベスト・プラクティスとして、ApplyLagThresholdを15分以上に設定することをお薦めします。

- TransportLagThresholdプロパティは、データベースの転送ラグがプロパティで指定された値を上回ったときに、ロジカル、フィジカルまたはスナップショットのスタンバイについて警告ステータスを生成するために使用できます。

  プロパティ値を表す単位は秒です。値が0秒の場合、転送ラグが存在しても警告を生成しません。ベスト・プラクティスとして、TransportLagThresholdを15分以上に設定することをお薦めします。

SQLを使用したOracle Data Guard構成ヘルスの監視

次の表の問合せを使用すると、プライマリ・データベースおよびスタンバイ・データベースのData Guard構成全体のヘルスを評価できます。

表16-1 プライマリ・データベースの問合せ

目標	問合せ	予想される結果
リモート・スタンバイ・アーカイブ先でエラーが発生していないかどうかをチェックします すべてのリモート・スタンバイ・アーカイブ先が有効(VALID)かどうかをチェックします	`select sysdate,status,error from gv$archive_dest_status where type='PHYSICAL' and status!='VALID' or error is not null;`	正常なヘルス = 行が返されません 問合せで行が返された場合は、返されたデータを使用してアラートを生成します。
前日にプライマリ・データベースでNOLOGGING アクティビティが発生したかどうかをチェックします	`select file#, name, unrecoverable_change#, unrecoverable_time from v$datafile where unrecoverable_time > (sysdate - 1);`	正常なヘルス = 行が返されません 問合せで行が返された場合は、スタンバイ・データベースが脆弱であり、出力にリストされているファイルをスタンバイでリフレッシュする必要があります。
スタンバイ・データベースのギャップを検出します	`select sysdate,database_mode,recovery_mode, gap_status from v$archive_dest_status where type='PHYSICAL' and gap_status !='NO GAP';`	正常なヘルス = 行が返されません 問合せで行が返された場合は、プライマリ・データベースとスタンバイ・データベースの間にギャップがすでに存するため、スタンバイ・データベースで同じ問合せを実行する必要があります。 プライマリとスタンバイの出力が同一の場合、アクションは不要です。 スタンバイの出力がプライマリの出力と一致しない場合は、スタンバイのデータファイルをリフレッシュする必要があります。
前日に重大な Data Guard イベントが発生したかどうかを評価します	`select * from v$dataguard_status where severity in ('Error','Fatal') and timestamp > (sysdate -1);`	正常なヘルス = 行が返されません 問合せで行が返された場合は、返された出力を使用してアラートを生成します。

目標	問合せ	予想される結果
同期環境の場合のみ: 最大可用性モードで実行されており、構成が同期しているかどうかを評価します	`select sysdate,protection_mode, synchronized, synchronization_status from v$archive_dest_status where type='PHYSICAL' and synchronization_status !='OK ';`	正常なヘルス = 行が返されません 問合せで行が返された場合は、返された出力を使用してアラートを生成します。

表16-2 フィジカル・スタンバイ・データベースの問合せ

目標	問合せ	予想される結果
転送ラグがあるかどうかを確認します	`select name,value,time_computed,datum_time from v$dataguard_stats where name='transport lag' and value > '+00 00:01:00';`	正常なヘルス = 行が返されません 行が返されない場合は、転送ラグがないことを意味します
適用ラグがあるかどうかを確認します	`select name,value,time_computed,datum_time from v$dataguard_stats where name='apply lag' and value > '+00 00:01:00';`	正常なヘルス = 行が返されません 行が返されない場合は、適用ラグがないことを意味します
スタンバイ・データ・ファイル・チェック(オフライン・ファイルまたはアクセスできないファイル)	`select * from v$datafile_header where status ='OFFLINE' or ERROR is not null;`	正常なヘルス = 行が返されません 返される行には、I/O またはリカバリの問題があるファイルがリストされます
メディア・リカバリ・プロセスが現在実行中であることを確認します	`select * from v$managed_standby where process like 'MRP%';`	正常なヘルス = 行が返されます 行が返されない場合、MRP プロセスは実行されていません
前日に重大な Data Guard イベントが発生したかどうかを評価します	`select * from v$dataguard_status where severity in ('Error','Fatal') and timestamp > (sysdate -1);`	正常なヘルス = 行が返されません 問合せで行が返された場合は、返された出力を使用してアラートを生成します

Oracle Data Guard Brokerの診断情報

Oracle Data Guard Brokerのアクティビティに関する情報は、いくつかの形式で提供されます。

- データベース・ステータスの情報
- Oracleアラート・ログ・ファイル

  ブローカでは、ブローカ構成に含まれる各データベースのインスタンスごとのアラート・ログ・ファイルに重要情報が記録されます。
- Oracle Data Guard Brokerのログ・ファイル

  ブローカ構成内の各データベースのインスタンスごとに、ブローカのDMONプロセスにより、重要な動作およびステータス情報がブローカ・ログ・ファイルに記録され、このファイルは、Oracle Data Guardの障害を診断する際に役立ちます。`TraceLevel`構成プロパティを設定して、ブローカ・ログ・ファイルでレポートされる診断情報のレベルを指定します。ブローカ・ログ・ファイルは、アラート・ログと同じディレクトリに作成され、`drc<$ORACLE_SID>.log`という名前が付けられます。
- Oracle Data Guardコマンドライン(DGMGRL)の`logfile`オプション

  DGMGRLコマンドライン・インタフェースを、`-logfile`オプション・パラメータを付けて開始すると、結果として生じるログ・ファイルに、過去の操作およびエラー条件に関する、役に立つ記録が含まれる場合があります。

データ破損の検出および監視

破損データがディスクに書き込まれた場合、または正常なデータの書込み後にコンポーネント障害によってそのデータが破損した場合は、破損したブロックをできるだけ早く検出することが重要です。

データベースのエラーおよびアラートを監視するには、次のことを行います。

- ブロック破損が検出または修復されると自動的に更新される、`V$DATABASE_BLOCK_CORRUPTION`ビューを問い合せます。
- データ障害の自動診断、適切な修復オプションの判別と提供、およびリクエストに応じた修復操作の実行が行われるように、データ・リカバリ・アドバイザを構成します。

  データ・リカバリ・アドバイザはOracle Enterprise Manager Support Workbench (サポート・ワークベンチ)、ヘルス・モニターおよびRMANと統合されています。
- Data Guardを使用して、物理破損を検出し、書込みの欠落を検出します。

  Data Guardは、REDOストリームで破損ブロックのために適用プロセスが停止した場合、または書込みの欠落を検出した場合に、物理破損を検出します。

  Data Guard構成の管理および監視にはEnterprise Managerを使用します。

  自動ブロック・メディア・リカバリを利用することで、プライマリ・データベースまたはフィジカル・スタンバイ・データベースのどちらかで検出された破損ブロックは、Active Data Guardオプションが使用されていると自動的に修正できます。
- SQL*Plusを使用して、データファイル破損およびブロック間破損を検出します。

  次のSQL*Plus文を実行します。

```
sqlplus> ANALYZE TABLE table_name VALIDATE STRUCTURE CASCADE;
```

破損の検出後は、表を再作成するか、他のアクションを行うことができます。

- Recovery Manager (RMAN)のバックアップおよびリカバリ計画では、物理ブロック破損を検出できます。

  次のコマンドを使用したさらに徹底的なRMANチェックでは、論理ブロック破損を検出できます。

```
RMAN> BACKUP VALIDATE CHECK LOGICAL;
```

第V部 MAA PlatinumおよびOracle GoldenGateのベスト・プラクティス

17 MAA Platinum参照アーキテクチャの概要

MAA PlatinumまたはNever-Down Architectureは、ほぼゼロのリカバリ時間目標(RTO、停止中に発生した停止時間)と、潜在的にゼロまたはほぼゼロのリカバリ・ポイント目標(RPO、データ損失の可能性)を達成します。

MAA Platinumリファレンス・アーキテクチャにより、次の事項が保証されます。

- RTO = Oracle RAC、フルスタック冗長性およびフェイルオーバー機能を備えたOracle Exadata Database Machineプラットフォーム使用することで、すべてのローカル障害に対してゼロまたはほぼゼロ
- RTO = データベース、クラスタ、サイトの障害などのディザスタに対してゼロまたはほぼゼロ。アクティブなOracle GoldenGateソースまたはターゲットにアプリケーションをリダイレクトすることで実現します
- Oracle RACおよびExadata Database Machineプラットフォームを使用することで、ソフトウェアおよびハードウェアの更新のためのメンテナンスの停止時間ゼロ
- アップグレード済のOracle GoldenGateソース・データベースまたはターゲット・データベースにアプリケーションをリダイレクトすることで、データベース・アップグレードまたはアプリケーション・アップグレードの停止時間ゼロ
- RPO = REDO転送(SYNC、FAR SYNCまたはASYNC)を指定するOracle Data Guard保護モード設定に応じて、ゼロまたはほぼゼロのデータ損失
- 障害発生後のOracle GoldenGateソースおよびターゲット・データベース間の高速再同期とゼロまたはほぼゼロのRPO。

  データベース障害の発生後、そのスタンバイ・データベースへの自動フェイルオーバーが自動的に実行されます。その後、Oracle GoldenGateソースとターゲット・データベースの間の自動再同期が再開されます。SYNC転送の場合、データ損失は最終的にゼロになります。

表17-1 MAA Platinum停止マトリックス

イベント	**RTO/RPOサービス・レベル目標[1]**
計画外停止	
リカバリ可能なノードまたはインスタンスの障害	ゼロまたは 1 桁秒 [2、3]
破損とサイトの障害のあるディザスタ	ゼロ [3]
計画メンテナンス	
最も一般的なソフトウェアおよびハードウェアの更新	ゼロ [2]
データベースまたはアプリケーションのメジャー・アップグレード	ゼロ [3]

[1]RPO=0 (特に明示されていない場合)

[2]オンライン処理の停止時間ゼロまたは最小限の影響を達成するには、MAAアプリケーション高可用性のベスト・プラクティス

(チェックリストとも呼ばれる)を適用します。バッチ操作などの長時間実行トランザクションについては、計画メンテナンス・ウィンドウ外に延期することをお薦めします。

[3]アプリケーション・フェイルオーバーは、Global Data Servicesでカスタマイズまたは管理されます。

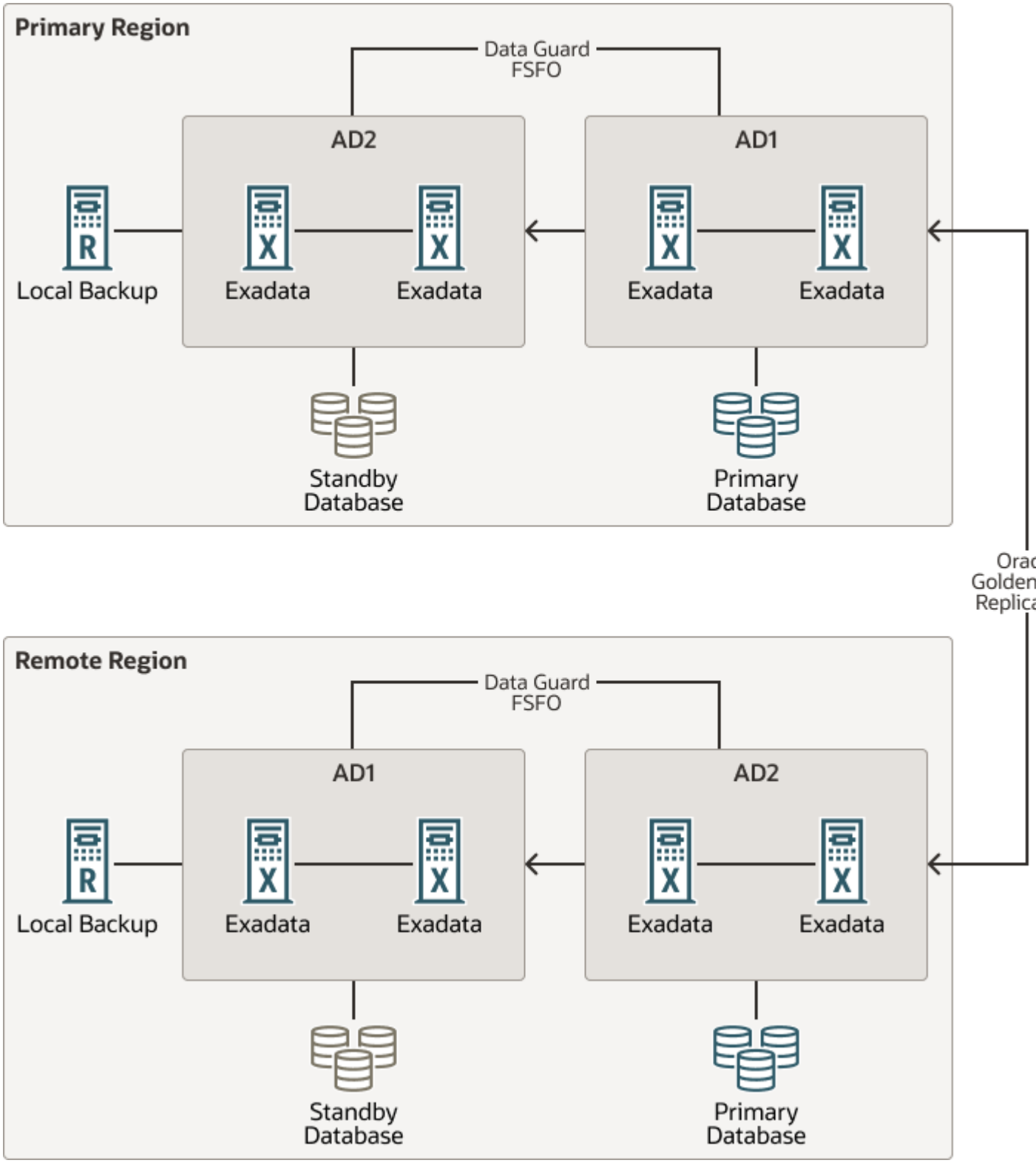
Platinum MAAソリューションでは次の機能を使用できます。

- Oracle Real Application Clusters。ソースおよびターゲットの推奨データベースはExadata Database Machine Platform, Exadata Database Service on Dedicated Infrastructure (ExaDB-D), or Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C)です
- Oracle Active Data Guard。ファスト・スタート・フェイルオーバーを使用することで、データベース、クラスタまたはデータ・センターの障害が発生した場合に、データ損失を抑制し、自動的にスタンバイにフェイルオーバーします。通常、スタンバイ・データベースは、独立した電源装置を備えた個別のフォルト・ドメイン(FD)、個別のデータ・センター、または独立した電源とネットワークを備えた可用性ドメイン(AD)にあります。また、スタンバイは、一般に障害分離が最大になるようにリージョンをまたいで存在することもできます。
- Oracle GoldenGate。データベース、クラスタ、サイトの障害または計画停止(データベースやアプリケーションのアップグレードなど)の直後にアプリケーションをフェイルオーバーするために利用できる2つのアクティブな読取り/書込みデータベース・システムが有効になります。Oracle GoldenGateレプリケーションが発生しているソースおよびターゲット・データベースは、同じリージョンに存在することも、複数のADや複数のリージョンにまたがって存在することもできます。

次の図に示すMAA Platinumアーキテクチャでは、2つのアクティブな読取り/書込みプライマリ・データベース(ソースまたはターゲット・データベース)が個別のリージョンに存在します。Oracle GoldenGateレプリケーションは、プライマリとリモートのリージョン間のソースおよびターゲット・データベース間で発生します。それぞれのプライマリ・データベースは、同じリージョン内の別のADのスタンバイ・データベースによって保護されます。

Data Guardファスト・スタート・フェイルオーバー(FSFO)を使用すると、プライマリ・データベース、クラスタまたはADの障害後に、スタンバイ・データベースが自動的に新しいプライマリ・データベースになります。MAA Oracle GoldenGate構成のベスト・プラクティスを実装すると、Data Guardロールのトランジション後にソースとターゲットのデータベース間でレプリケーションが自動的に再開されます。各データベースは、Exadataプラットフォーム上に存在しています。このプラットフォームは、固有の組込みReal Application Cluster、システムとストレージの冗長性および短時間のブラウンアウト・フェイルオーバー機能を備えています。

図17-1 MAA Platinum参照アーキテクチャ



MAAプラチナ・アーキテクチャのバリアント

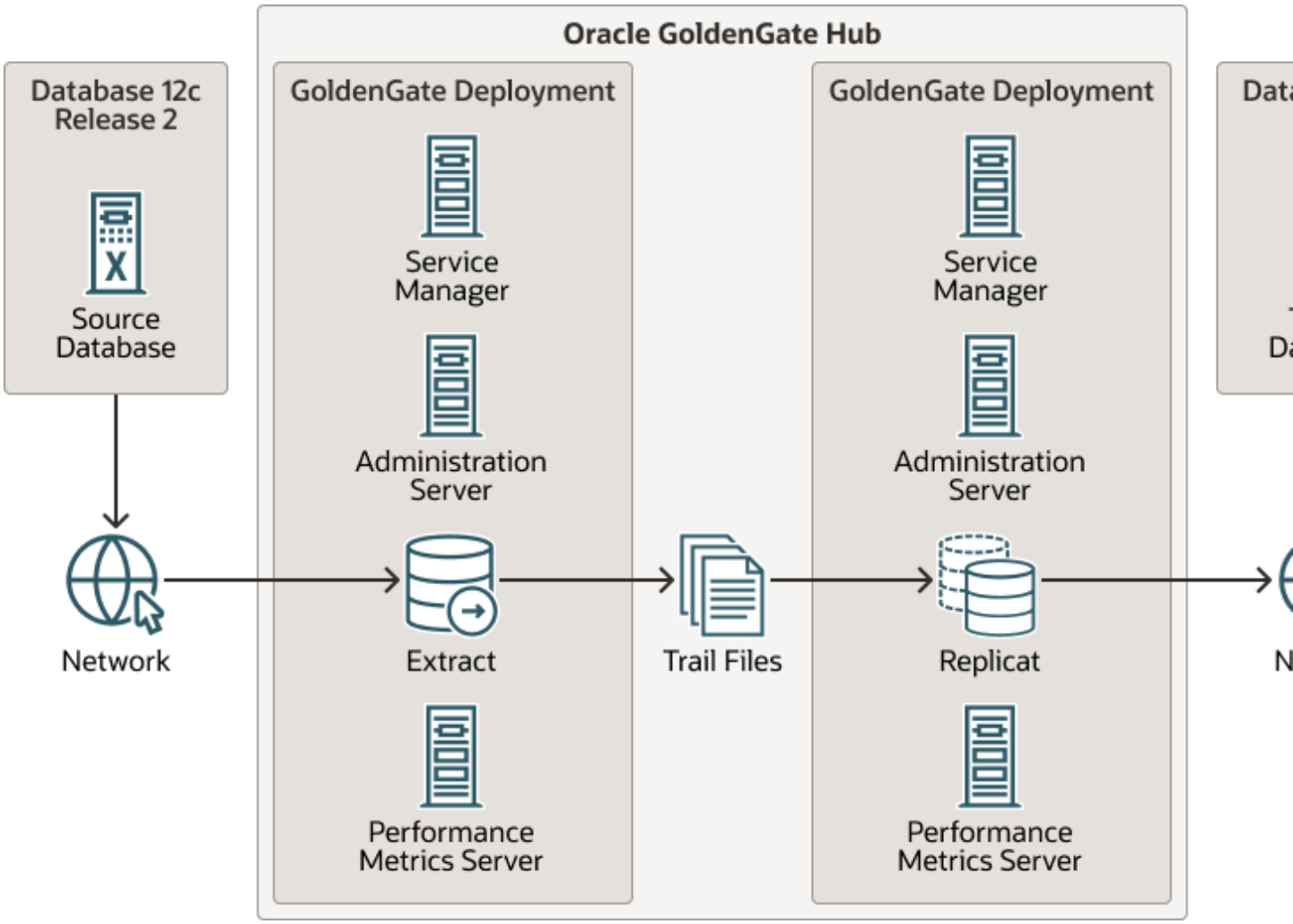
MAA Platinumアーキテクチャを設定する場合、管理者は、潜在的なソースまたはターゲットのデータベースごとにOracle GoldenGateを設定するか、データベース・サーバーから独立したOracle GoldenGateハブを作成するかを決定できます。

次の図に示すMAA Oracle GoldenGateハブには、次の利点があります。

- ソースとターゲットのExadataデータベース・システムから、Oracle GoldenGateソフトウェアのインストール、構成およびライフサイクル管理をオフロードします。
- ソースおよびターゲットのデータベース・システムに対するOracle GoldenGateリソースの影響を軽減します。

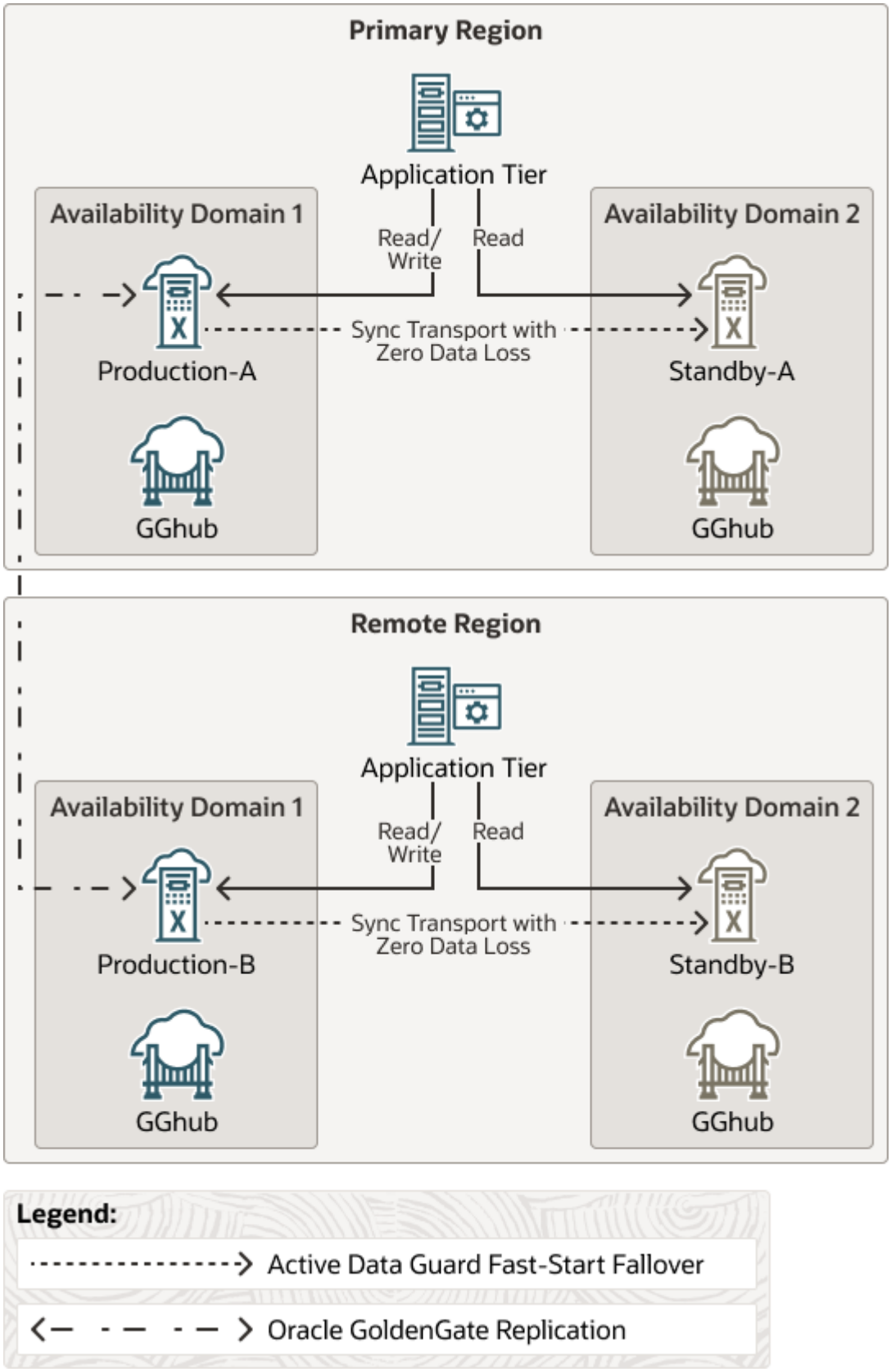
- 高速で簡単なフェイルオーバーのための2ノード・クラスタ・サーバーを構成することで高可用性を提供し、個別の2ノード・クラスタ・サーバー上の片方の同一GoldenGateハブ・サーバーへのACFSレプリケーションを利用することで障害時リカバリを提供します。
- 複数の独立したMAA PlatinumまたはOracle GoldenGateアーキテクチャ用のOracle GoldenGate構成およびソフトウェア・デプロイメントを統合します。

図17-2 MAA Oracle GoldenGate Hub



次の図に、MAA Platinumアーキテクチャによる MAA Oracle GoldenGateハブの例を示します。各ハブは、ローカルの高可用性を提供する2ノード・クラスタであり、追加の保護のために、別のハブ(通常は可用性ドメイン(AD)間またはリージョン間にデプロイされたハブ)へのACFSレプリケーションを使用します。

図17-3 Oracle GoldenGate Hubを使用したMAA Platinum



MAA Platinumソリューションの実装方法

MAA Platinumソリューションを実現するには、次のステップで参照されている技術資料とドキュメントを確認および活用します。

1. Oracle ExadataのOracle MAA Platinum層を確認して、MAA Platinumの利点とユースケースについて理解してください。
2. MAA Oracle GoldenGateハブ・ソリューションを実装するか、Oracle GoldenGateをデータベース・サーバーに直

接設定するかを決定します。

- オプション1: (推奨) MAA Oracle GoldenGate Hubの構成
  - Oracle Cloud構成については、Cloud: Oracle GoldenGate Hubの構成を参照してください。または
  - オンプレミス構成については、Oracle Maximum Availability Architecture (MAA) GoldenGate Hubを参照してください。
- オプション2: Exadata DatabaseサーバーのOracle GoldenGateの構成
  - Oracle Cloud Serviceについては、次を参照してください
    a. 「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」、および
    b. Cloud MAA Platinum: Active Data Guardと統合されたOracle GoldenGate Microservices Architecture
  - オンプレミス・システムについては、次を参照してください
    a. 「オンプレミス: Oracle Real Application Clustersを使用したOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」、および
    b. オンプレミスMAA Platinum: Active Data Guardと統合されたOracle GoldenGate Microservices Architecture

3. 双方向レプリケーションと自動競合検出および解決。Oracle Cloud Infrastructure GoldenGateのドキュメントまたは最新のOracle GoldenGateのドキュメントを参照してください。
4. 次のようなアプリケーション・フェイルオーバー・オプションを構成します
   - Global Data Services (Global Data Servicesのドキュメントを参照)、および
   - アプリケーションの継続的な可用性の構成のMAAアプリケーション高可用性構成(「チェックリスト」とも呼ばれます)。

18 Oracle GoldenGateのベスト・プラクティスの概要

Oracle MAAベスト・プラクティスを使用してOracle GoldenGateを構成し、Oracle GoldenGateデプロイメントから最高の可用性とパフォーマンスを引き出します。

Oracle GoldenGateには次のような利点があります。

- 単方向または双方向のレプリケーションで、任意のレプリケートされたデータベースの読取りおよび更新が可能です。
- データの移動がリアルタイムで行われ、レイテンシが低減されます。
- レプリケートされたデータベースは、異なるハードウェア・プラットフォーム、データベース・バージョンおよび異なるデータベースまたはアプリケーション構成で実行できるため、オンライン移行が可能です。この柔軟性により、オンラインでのデータベースおよびアプリケーションのアップグレードも可能になります。
- ソースおよびターゲットのレプリケート・データベースはオンラインであるため、停止中および計画メンテナンス・アクティビティ中に、アプリケーションの停止時間ゼロのスイッチオーバーが可能です。透過的アプリケーション・コンティニュイティなどの組込み機能を使用するのではなく、アプリケーションのスイッチオーバーをカスタマイズする必要があることに注意してください。

様々なOracle GoldenGate構成のベスト・プラクティスとMAA Platinumのベスト・プラクティスを、次の表に示します。

表18-1 Oracle GoldenGateのユースケースおよびベスト・プラクティス

ユースケース	Oracle GoldenGateのベスト・プラクティス
Oracle Cloud または Exadata プラットフォームへのデータベースの移行	GoldenGate を使用した停止時間ゼロの移行(ZDM) (論理移行) Oracle Zero Downtime Migration - 論理移行のパフォーマンス・ガイドライン
停止時間が最小またはゼロであることを必要とするデータベース移行 クロス・プラットフォームまたは異なるデータベース・バージョンが含まれるデータベース移行	Oracle GoldenGate Hub 構成による Oracle Database の移行
ハブ構成のデータベース・サーバーからの Oracle GoldenGate のデプロイ	Oracle Maximum Availability Architecture (MAA)の GoldenGate Hub
Oracle RAC データベース・サーバーまたは Exadata データベース・システムへの Oracle GoldenGate の直接インストールと構成	Cloud: Oracle Exadata Database Service での Oracle GoldenGate Microservices Architecture の構成のベスト・プラクティス オンプレミス: Oracle Real Application Clusters を使用した Oracle GoldenGate Microservices Architecture の構成のベスト・プラクティス
MAA Platinum の実装または Oracle Active	Oracle Cloud Service の場合: Cloud: Oracle Exadata

ユースケース	Oracle GoldenGateのベスト・プラクティス
Data Guard を使用した Oracle RAC データベース・サーバーへの Oracle GoldenGate の直接インストールと構成	Database Service での Oracle GoldenGate Microservices Architecture の構成のベスト・プラクティス、Cloud MAA Platinum: Active Data Guard と統合された Oracle GoldenGate Microservices Architecture オンプレミスの場合: オンプレミス: Oracle Real Application Clusters を使用した Oracle GoldenGate Microservices Architecture の構成のベスト・プラクティス、オンプレミス MAA Platinum: Active Data Guard と統合された Oracle GoldenGate Microservices Architecture
Oracle GoldenGate のアプリケーション・フェイルオーバー・オプション	『Global Data Services 概要および管理ガイド』、「アプリケーションの継続的な可用性の構成」
Oracle GoldenGate 双方向レプリケーションと自動競合検出および解決	Oracle Cloud Service の場合: Oracle Cloud Infrastructure GoldenGate ドキュメント オンプレミスの場合: 最新の Oracle GoldenGate ドキュメント

Oracle GoldenGateのドキュメント(https://docs.oracle.com/en/middleware/goldengate/core/21.1/)も参照してください

Oracle GoldenGateとサポート・テクノロジの概要

データベース間でデータをレプリケートするために必要なテクノロジは、Oracle GoldenGateと、様々な障害が発生した後に確実にレプリケーションを再開するためのサポート・テクノロジ(Oracle Grid Infrastructure Agent、ACFSまたはDBFS、GoldenGateハブなど)です。ここでは、Oracle GoldenGateとサポート・テクノロジの概要について説明します。

Oracle GoldenGate

Oracle GoldenGateは、同種システムと異種システム間で、リアルタイムのログベースの変更データの取得と配信を提供します。このテクノロジにより、コスト効率が高く影響の少ないリアルタイムのデータ統合と継続的な可用性ソリューションを構築できます。

Oracle GoldenGateはトランザクションの整合性を維持し、既存のインフラストラクチャのオーバーヘッドを最小限に抑えて、コミットされたトランザクションからデータをレプリケートします。このアーキテクチャは、1対多、多対多、カスケード、双方向など、複数のデータ・レプリケーション・トポロジをサポートしています。その幅広いユースケースには、リアルタイムのビジネス・インテリジェンス(クエリのオフロード、停止時間ゼロのアップグレードと移行、データ分散、データ同期、高可用性のためのアクティブ/アクティブ・データベース)が含まれます。

Oracle GoldenGate Microservices ArchitectureはREST対応サービスを提供します。REST対応サービスでは、HTML5 Webページ、コマンドライン・インタフェースおよびAPIを介して、リモート構成、管理および監視を提供します。

推奨されるOracle GoldenGate 21c (および、それ以降のリリース)では統合ビルド・サポートが導入されているため、単一のソフトウェア・インストールで、レプリケートされたデータの取得と複数の主要なOracle Databaseバージョン(11gリリース2から21c)への適用がサポートされます。これが可能になるのは、Oracle GoldenGateインストールに必要なOracle Database

クライアント・ライブラリが含まれており、別途データベースORACLE_HOMEをインストールする必要がないためです。

Oracle Grid Infrastructure Agent

Oracle Grid Infrastructure Agent (XAG)は、エージェント管理インタフェースであるAGCTLを介して管理されるアプリケーション・リソースとリソース・タイプに高可用性(HA)フレームワークを提供する、Oracle Grid Infrastructureコンポーネントです。このフレームワークは、完全なアプリケーションHAのためにアプリケーションを統合するための、事前定義されたOracle Grid Infrastructureリソース構成とエージェントを含む、すぐに使用できる完全なソリューションを提供します。

Oracle Grid Infrastructure Agentは、Oracle GoldenGate、Siebel、Oracle PeopleSoft、JD Edwards、Oracle WebLogic Server、ApacheおよびMySQLアプリケーション用に事前定義されたOracle Clusterwareリソースを提供します。Oracle GoldenGateのエージェントを使用すると、ソース・データベースとターゲット・データベース、アプリケーションVIPおよびファイル・システム(ACFSまたはDBFS)マウント・ポイントへの依存性の作成が簡素化されます。エージェント・コマンドライン・ユーティリティ(AGCTL)は、Oracle GoldenGateの起動と停止に使用します。また、クラスタ内のノード間でOracle GoldenGateを再配置するために使用することもできます。

Oracle Database File System(DBFS)

Oracle DBFSを使用すると、Oracle GoldenGateファイルを格納できます。

Oracle Database File System (DBFS)は、データベースに格納されるファイルへのファイル・システム・インタフェースを作成します。DBFSはローカルのファイル・システムのように見える共有ネットワーク・ファイル・システムを提供するという点でNFSと似ています。データはデータベースに格納されるため、ファイル・システムはOracle Databaseが提供する高可用性とディザスタ・リカバリ機能をすべて継承します。

DBFSでは、サーバーはOracle Databaseです。ファイルはSecureFiles LOBとして格納されます。PL/SQLプロシージャは、作成、オープン、読取り、書込み、リスト・ディレクトリなど、ファイル・システム・アクセスのプリミティブを実装します。データベース内のファイル・システムの実装は、DBFS SecureFilesストアと呼ばれます。DBFS SecureFilesストアを使用すると、ユーザーはクライアントがマウントできるファイル・システムを作成できます。各ファイル・システムには、ファイル・システム・コンテンツが保持されるシステム専用の表があります。

Oracle Advanced Cluster File System (ACFS)

Oracle ACFSを使用すると、Oracle GoldenGateファイルを格納できます。

Oracle Advanced Cluster File System (Oracle ACFS)は、マルチプラットフォームのスケーラブルなファイル・システムであり、Oracle Automatic Storage Management (Oracle ASM)の機能を拡張して、すべてのカスタマ・ファイルをサポートするストレージ管理テクノロジです。

Oracle ACFSは、クラスタ・メンバーシップの状態遷移とリソースベースの高可用性にOracle Clusterwareを利用します。Oracle ACFSはOracle Grid Infrastructure (GI)にバンドルされているため、データベース、リソース、ボリュームおよびファイル・システムの統合され最適化された管理を可能にします。

19 Cloud: Oracle GoldenGate Hubの構成

提供されている計画上の考慮事項、タスク、管理およびトラブルシューティング情報を使用して、Oracle Cloud上でMAA Oracle GoldenGate Hubアーキテクチャを構成およびデプロイします。

次の項を参照してください。

- MAA GoldenGate Hubの概要
- Platinum MAAアーキテクチャでのGGHub配置の計画
- タスク1: Oracle GoldenGateのソースおよびターゲット・データベースの構成
- タスク2: GGHubのプライマリおよびスタンバイ・ベース・システムの準備
- タスク3: プライマリおよびスタンバイGGHub向けのOracle GoldenGateの構成
- タスク4: Oracle GoldenGate環境の構成
- Oracle GoldenGate Hub向けの計画および計画外停止の管理
- MAA GoldenGate Hubのトラブルシューティング

MAA GoldenGate Hubの概要

最高レベルの可用性を実現し、計画外停止と計画メンテナンス・アクティビティの両方の停止時間をゼロまたはほぼゼロにするために、顧客は高頻度でExadata Database Service on Dedicated Infrastructure (ExaDB-D)またはOracle Exadata Database Service on Cloud@Customer (ExaDB-C@C)、Oracle Active Data GuardとOracle GoldenGateの組合せを使用します。

このアーキテクチャ(通常はPlatinum MAAまたはNever Down Architecture)は、ゼロに近いリカバリ時間目標(RTOまたは停止中に発生する停止時間)と、ゼロまたはゼロに近いリカバリ・ポイント目標(RPOまたはデータ損失の可能性)を実現します。

以前から、Oracle GoldenGateはGoldenGateプロセスが接続するデータベース・サーバーにインストールされ、ローカルに実行されていました。Oracle Grid Infrastructure Standalone Agent (XAG)と併用すると、Oracle GoldenGateプロセスは、Oracle RACノード間でシームレスに再配置またはフェイルオーバーし、Oracle Active Data Guardのスイッチオーバーとフェイルオーバーに従うように構成できます。

MAA Oracle GoldenGate (MAA GGHub)を使用すると、GoldenGateソフトウェアとプロセスがExadataデータベース・サーバーから移動され、複雑さとシステム・リソースの使用率が軽減されます。MAA GGHubはOracle GoldenGate管理を一元化し、Oracle GoldenGate処理の大部分と関連するCPUおよびストレージ・リソースの使用率を、Exadataシステム・リソースからオフロードします。GoldenGateプロセスとそれらが操作するデータベースの間の接続は、Oracle Net Servicesで管理されます。

全体的に、Oracle CloudでPlatinum MAAソリューションを実現するには、次に示す大まかなステップに従う必要があります。

1. Oracle ExadataのOracle MAA Platinum層を確認して、Platinum MAAの利点とユースケースについて理解してください。
2. データベースを、Exadata Cloud ServiceまたはExadata Cloud@Customerにデプロイまたは移行します。
3. Oracle Cloud Control Planまたはクラウドの自動化を使用して、Oracle Cloudに対称スタンバイ・データベースを追加します。
4. 「ファスト・スタート・フェイルオーバーの構成」に記載されたOracle MAAのベスト・プラクティスの推奨事項を使用して、Oracle Data Guardファスト・スタート・フェイルオーバーを手動で構成し、デプロイします。
5. MAA GGHubを設定します。
6. 双方向レプリケーションと自動競合検出および解決。詳細は、Oracle Cloud Infrastructure GoldenGateドキュメントを参照してください。
7. Global Data Servicesなどのアプリケーション・フェイルオーバー・オプションを決定します。Global Data Servicesの概要を参照してください

Platinum MAAアーキテクチャでのGGHub配置の計画

停止時間をゼロ(RTO=0またはほぼゼロ)にし、データ損失をゼロまたはほぼゼロ(RPO=0またはほぼゼロ)にする極限の可用性を実現するには、通常、次に示すPlatinum MAAアーキテクチャが必要です。

1. Oracle GoldenGateアーキテクチャには、障害(データベース、クラスタまたはサイト障害)の発生時にアプリケーションをただちにフェイルオーバーできるようにして、データベースまたはアプリケーションのアップグレード時にスイッチオーバーできるようにするためのソース・データベースとターゲット・データベースがあります。このアーキテクチャにより、障害シナリオやデータベースおよびアプリケーションのアップグレードの保守に対して、潜在的なRTOをゼロまたはほぼゼロにできます。
2. 各ソースおよびターゲット・データベースは、Exadataクラウド・システムにデプロイされるため、ローカルの障害は許容されるかほぼ瞬時にリカバリされます。
3. 各ソースおよびターゲット・データベースは、Data Guardファスト・スタート・フェイルオーバーを備えたスタンバイ・データベースとともに構成されているため、データベースに障害が発生すると数秒から数分で新しいプライマリ・データベースがアクティブ化されます。最大可用性保護モードのSYNC転送を利用すると、データ損失ゼロのData Guardフェイルオーバーが実現されます。
4. ソースおよびターゲット・データベースの間にMAA GGhubを使用するGoldenGateレプリケーションを使用して構成されます。
5. Data Guardのスイッチオーバーまたはフェイルオーバーによってプライマリ・データベースになるスタンバイが、自動的にターゲットのGoldenGateデータベースと再同期するように構成されます。データ損失ゼロのData Guardスイッチオーバーまたはフェイルオーバーが発生した場合は、GoldenGateの再同期により、分散データベース環境全体のデータ損失がゼロになります。
6. GoldenGateの自動競合検出および解決を使用して構成されます。これは、Data Guardのフェイルオーバー操作の発生後に必要です。

MAAプライマリGGHubとスタンバイGGHubを配置する場所

1. GGHubペア(プライマリとスタンバイのGGhub)は、各プライマリおよびスタンバイ・データベースと同じOCIリージョンに存在する必要があります。たとえば、
    a. プライマリ・データベースがAD1、リージョンAにあり、スタンバイ・データベースがAD2、リージョンAにある場合、GGHubペアはリージョンAに存在します。
    b. プライマリ・データベースがリージョンAにあり、スタンバイ・データベースがリージョンBにある場合、GGHubペアはリージョンAとリージョンBの間で分割します。プライマリまたはアクティブなGGhubは、ターゲットのプライマリ・データベースと同じOCIリージョンにコロケートされている必要があります。
2. パフォーマンスへの影響:
    a. プライマリまたはアクティブなGGhubは、ターゲット・データベースと同じOCIリージョンに存在している必要があります。
    b. プライマリまたはアクティブなGGhubは、GoldenGateのパフォーマンス(Extractのパフォーマンス)低下が発生することのない、ソース・データベースから90ミリ秒未満に収める必要があります。同じ国内のいずれの可用性ドメインまたはOCIリージョンについても、Oracleクラウドはその要件を常に満たします。
3. GoldenGate分散パス
    a. ソースおよびターゲットのGGhubが異なるリージョンにあり、OCIリージョン間のレイテンシが90ミリ秒を超える場合は、GoldenGate分散パスが必要になります。
    b. Oracle Cloudでは、Oracle GoldenGateのソースとターゲットのデータベースが同じリージョン、または同じ国内の異なるリージョンに存在する場合は、レイテンシが常に90ミリ秒未満になるため、GoldenGate分散パ

スを設定する必要はありません。

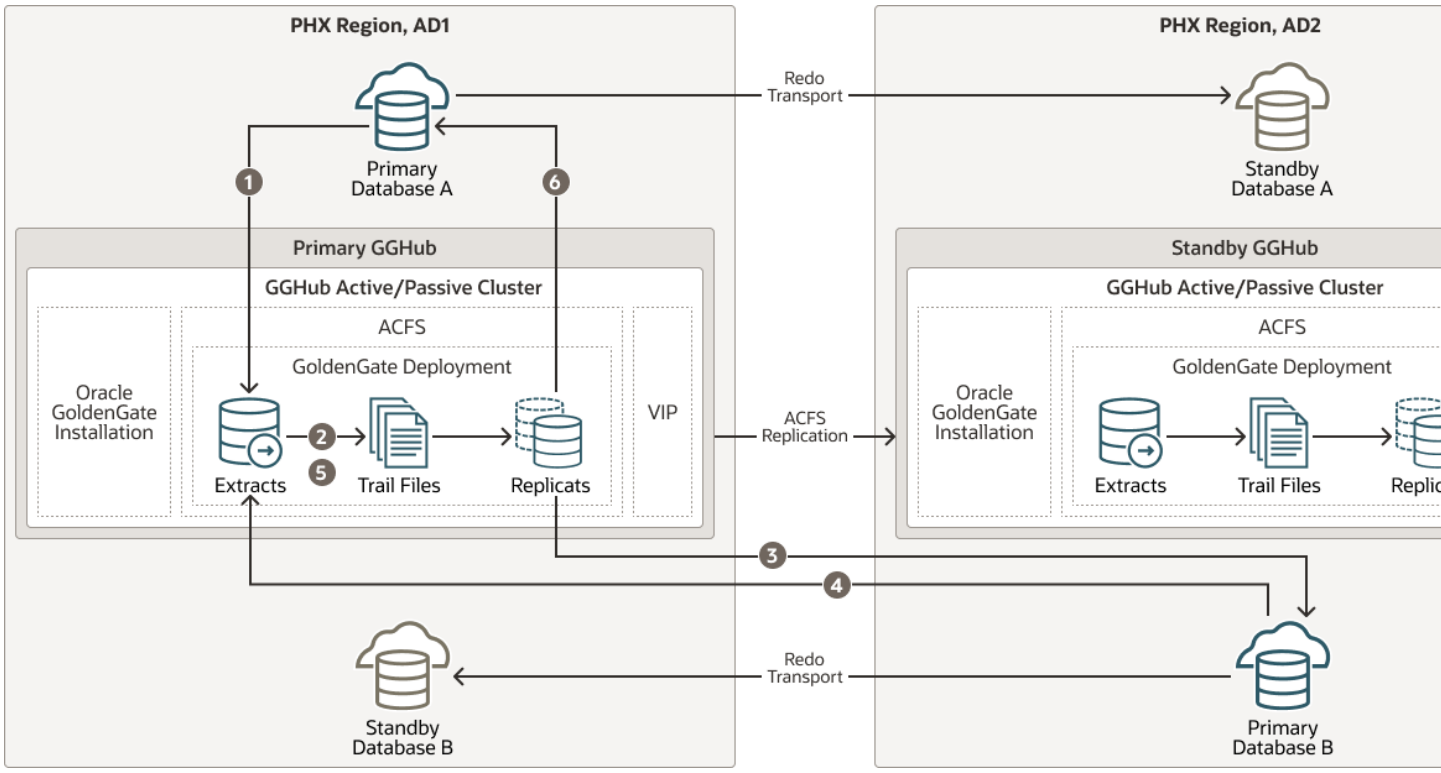
同じOCIリージョン内に配置されたMAA GGHub

このシナリオでは、プライマリとスタンバイのデータベースは同じOCIリージョンに配置されるため、プライマリ(アクティブ)GGHubとスタンバイGGHubも同じリージョンに配置されます。

図1に示すように、次のアーキテクチャ・コンポーネントがあります。

1. プライマリ・データベースおよび関連付けられたスタンバイ・データベースは、Oracle Active Data Guardファスト・スタート・フェイルオーバー(FSFO)によって構成されています。FSFOは、データ損失の最大許容度に応じて、ASYNCまたはSYNC REDO転送を使用するData Guard保護モードで構成できます。
2. プライマリGGHubアクティブ/パッシブ・クラスタ: 2ノード・クラスタに1つのみのGGHubソフトウェア・デプロイメントと構成があります。このクラスタには、11.2.0.4以降のデータベース・バージョンのサポートが可能な、21cのOracle GoldenGateソフトウェア・デプロイメントが含まれています。このGGHubは、多数のプライマリ・データベースをサポートし、ソース・データベースからトランザクションをマイニングするGoldenGate Extractとターゲット・データベースに同じ変更を適用するGoldenGate ReplicatのGoldenGateプロセスをカプセル化できます。GoldenGate証跡とチェックポイント・ファイルは、GGhub ACFSファイル・システム内にも存在します。HAフェイルオーバー・ソリューションはGGhubに組み込まれています。このソリューションには、同じクラスタ内のパッシブ・ノードへの自動フェイルオーバーが含まれていて、ノード障害後にGoldenGateプロセスとアクティビティを再起動します。
3. スタンバイGGHubアクティブ/パッシブ・クラスタ: 対称スタンバイGGhubが構成されています。ACFSレプリケーションは、すべてのGoldenGateファイルを保持するために、プライマリとスタンバイのGGHubの間に設定されています。ACFSフェイルオーバーを含む手動によるGGhubフェイルオーバーは、プライマリGGhub全体が損なわれるまれなケースで実行することになります。

図19-1 同じOCIリージョン内のプライマリおよびスタンバイGGHub



上の図は、次のステップによって、プライマリ・データベースAからプライマリ・データベースBにデータをレプリケートして、プライマリBか

らプライマリAに戻している様子を示しています。

1. プライマリ・データベースA: プライマリAのLogminerサーバーは、プライマリGGHub Extractプロセスに REDO変更を送信します。
2. プライマリGGHub: Extractプロセスによって、変更が証跡ファイルに書き込まれます。
3. プライマリGGHubからプライマリ・データベースB: プライマリGGHubのReplicatプロセスは、該当する変更をターゲット・データベース(プライマリB)に適用します。
4. プライマリ・データベースB: プライマリBのLogminerサーバーは、プライマリGGHub Extractプロセスに REDOを送信します。
5. プライマリGGHub: プライマリGGHubのExtractプロセスによって、変更が証跡ファイルに書き込まれます。
6. プライマリGGHubからプライマリ・データベースA: プライマリGGHubのReplicatプロセスは、該当する変更をターゲット・データベース(プライマリA)に適用します。

ソースとターゲットのデータベースが異なるOracle Databaseリリースであっても、1つのGGHubで複数のソース・データベースとターゲット・データベースをサポートできます。

表19-1 同じOCIリージョン内のGGHubの停止シナリオ、修復および冗長性のリストア

停止シナリオ	アプリケーションの可用性と修復	冗長性と初期状態の復元
プライマリ・データベース A (またはデータベース B)の障害	影響: アプリケーション停止時間は、ほぼゼロです。GoldenGate レプリケーションは、新しいプライマリ・データベースの起動時に再開されます。 1. 1 つのプライマリ・データベースが引き続き使用できます。すべてのアクティビティは、アプリケーションの停止時間がゼロになるように、使用可能な既存のプライマリ・データベースにルーティングされます。たとえば、アプリケーション・サービスの A から F はデータベース A にルーティングされ、アプリケーション・サービスの G から J はデータベース B にルーティングされているとします。データベース A に障害が発生すると、すべてのアプリケーション・サービスは一時的にデータベース B に移動します。 2. スタンバイは、Data Guard FSFO によって自動的に新しいプライマリになります。Oracle GoldenGate レプリケーションが再開され、プライマリ・データベースが再同期されます。データ損失は、Data Guard 保護レベルによって制限されます。最大可用性または最大保護が構成されている場合は、データ損失がゼロになります。すべてのコミットされたトランザクションは、一方または両方のデータベースにあります。ワークロードは、プライマリ・データベース A とデータベース B が使用可能であり、同期状態にあるときには「リバランス」できます。たとえば、データベース A が稼働中で同期状態にある場合、サービス A から F はデータベース A に戻すことができます。	1. 元のプライマリ・データベースは、冗長性をリストアするために新しいスタンバイ・データベースとして回復されます。 2. オプションで、Data Guard スイッチオーバーを実行して元の構成に戻すと、1 つ以上のプライマリ・データベースが個別の AD 内に存在するようにします。

停止シナリオ	アプリケーションの可用性と修復	冗長性と初期状態の復元
プライマリまたはスタンバイ GGHub の単一ノード障害	影響: アプリケーションに影響はありません。GoldenGate レプリケーションは、数分後に自動的に再開されます。 処置は必要ありません。GGHub に組み込まれた HA フェイルオーバー・ソリューションには、GoldenGate プロセスとアクティビティの自動フェイルオーバーと再起動が含まれます。レプリケーション・アクティビティは、GoldenGate プロセスが再度アクティブになるまでブロックされます。GoldenGate レプリケーションのブラックアウトは数分間続くことがあります。	ノードが再起動すると、アクティブ/パッシブ構成が再確立されます。
プライマリ GGHub クラスタがクラッシュしてリカバリできない	影響: アプリケーションに影響はありません。GoldenGate レプリケーションは、既存の GGHub の再起動後、または手動による GGHub フェイルオーバー操作の実行後に再開されます。 1. GGHub クラスタの再起動が可能な場合は、それが最も単純なソリューションです。これは分単位で制限する必要があります。GGHub クラスタの再起動を試行する方が簡単です。 2. プライマリ GGHub がリカバリ不可能な場合は、スタンバイ GGHub への手動による GGHub フェイルオーバー(ACFS フェイルオーバーを含む)を実行します。これには通常、数分かかります。 3. GoldenGate レプリケーションは、新しいプライマリ GGhub が使用可能になるまで停止するため、ステップ 1 またはステップ 2 は迅速に実行する必要があります。	前の GGHub が最終的に再起動されると、ACFS レプリケーションは別の方向に自動的に再開されます。GGHub クラスタが損なわれた場合やリカバリ不能になった場合は、新しいスタンバイ GGHub を再構築する必要があります。
スタンバイ GGHub クラスタがクラッシュしてリカバリできない	影響: アプリケーションまたはレプリケーションに影響はありません。 1. GGHub クラスタが再起動可能な場合は、それが最も単純なソリューションであり、ACFS レプリケーションを再開できます。 2. スタンバイ GGHub がリカバリ不可能な場合は、新しいスタンバイ GGHub を再構築することになります。	N/A
データ・センターまたは可用性ドメイン(AD1 または AD2)の全体的な障害	影響: アプリケーション停止時間は、ほぼゼロです。GoldenGate レプリケーションは、新しいプライマリ・データベースの起動時に再開されます。 1. 1 つのプライマリ・データベースが引き続き使用できます。すべてのアクティビティは、アプリケーションの停止時間がゼロになるように、使用可能な既存のプライマリ・データベースにルーティングされます。たとえば、アプリケーション・サービスの	1. データ・センター/AD の復帰時に、スタンバイの回復などの構成を再確立します。前の GGHub が最終的に再起動されると、ACFS レプリケーションは別の方向に自動的に再

停止シナリオ	アプリケーションの可用性と修復	冗長性と初期状態の復元
	A から F はデータベース A にルーティングされ、アプリケーション・サービスの G から J はデータベース B にルーティングされているとします。データベース A に障害が発生すると、すべてのサービスは一時的にデータベース B に移動します。 2. プライマリ GGHub がまだ機能している場合、GoldenGate レプリケーションは続行されます。可用性ドメイン(AD)の障害が原因でプライマリ GGHub が損なわれた場合は、手動による GGhub フェイルオーバーが必要です。GoldenGate レプリケーションが再開され、プライマリ・データベースが再同期されます。データ損失は、Data Guard 保護レベルによって制限されます。最大可用性または最大保護が構成されている場合は、データ損失がゼロになります。すべてのコミットされたトランザクションは、一方または両方のデータベースにあります。ワークロードは、プライマリ・データベース A とデータベース B が使用可能であり同期状態にあるときにはリバランスできます。データベース A が稼働中で同期状態にある場合、サービス A から F はデータベース A に戻すことができます。	開されます。 2. 可能な場合は、Data Guard スイッチオーバー(フェイルバック)を実行して、各 AD に 1 つのプライマリ・データベースが存在する元の状態に戻します。

異なるOCIリージョン内に配置されたMAA GGHub

このシナリオでは、プライマリとスタンバイのデータベースが異なるOCIリージョン内に配置されるため、プライマリ(アクティブ)GGHubはプライマリ・データベースと同じリージョンに配置され、スタンバイGGHubはスタンバイ・データベースと同じリージョンに配置されます。

図2に示すように、次のアーキテクチャ・コンポーネントがあります。

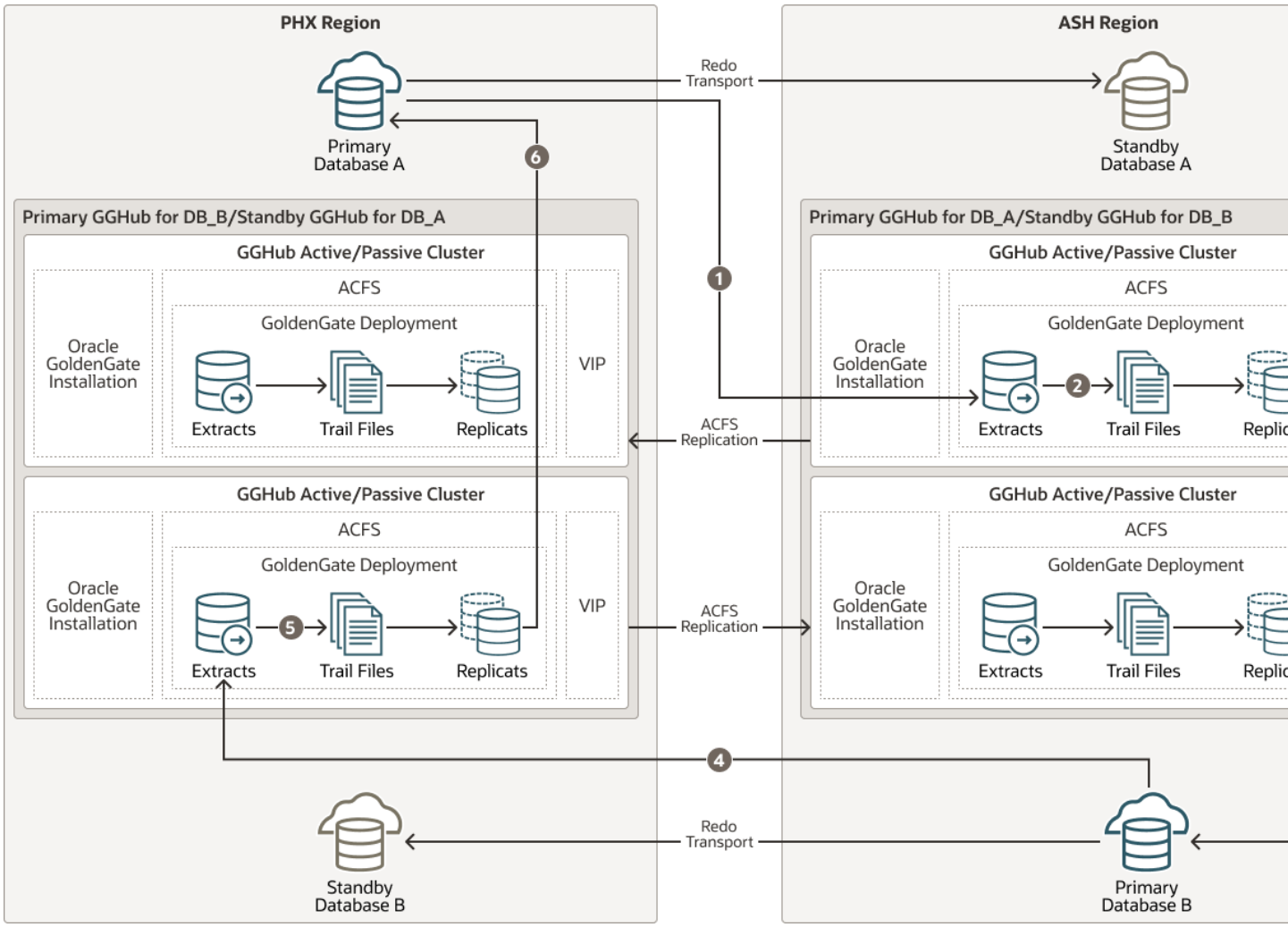
1. プライマリ・データベースおよび関連付けられたスタンバイ・データベースは、Oracle Active Data Guardファスト・スタート・フェイルオーバー(FSFO)によって構成されています。FSFOは、データ損失の最大許容度に応じて、ASYNCまたはSYNC REDO転送を使用するData Guard保護モードで構成できます。
2. プライマリGGHubアクティブ/パッシブ・クラスタ: この構成には、2つのOracle GoldenGateソフトウェア構成を備えた2ノード・クラスタがあります。プライマリGGHubはターゲット・データベースから4ミリ秒以下にする必要がありますが、2つのリージョン(PHXとASH)のネットワーク・レイテンシは5ミリ秒を超えるため、GGHubクラスタごとに2つのGGhub構成が作成されています。基本的に、プライマリGGHub構成は常にターゲット・データベースと同じリージョン内にあります。GGHubは、11g以降のOracle Databaseリリースのサポートが可能なOracle GoldenGate 21cソフトウェア・デプロイメントで構成されています。このGGHubは、多数のプライマリ・データベースをサポートし、ソース・データベースからトランザクションをマイニングするExtractとターゲット・データベースに該当する変更を適用するReplicatのGoldenGateプロセスをカプセル化できます。GoldenGate証跡とチェックポイント・ファイルは、ACFSファイル・システム内にも存在します。GGhubクラスタに組み込まれたHAフェイルオーバー・ソリューションには、ノード障害後のGoldenGateプロセスとアクティビティの自動フェイルオーバーと再起動が含まれています。

   それぞれのGGhub構成に、GoldenGateサービス・マネージャとデプロイメント、ACFSレプリケーションによるACFSファ

イル・システムおよび個別のアプリケーションVIPが含まれています。

3. スタンバイGGHubアクティブ/パッシブ・クラスタ: 対称スタンバイGGhubが構成されています。ACFSレプリケーションは、すべてのGoldenGateファイルを保持するために、プライマリとスタンバイのGGHubの間に設定されています。ACFSフェイルオーバーを含む手動によるGGhubフェイルオーバーは、プライマリGGhub全体が損なわれた場合に実行することになります。

図19-2 異なるOCIリージョン内のプライマリおよびスタンバイGGHub



上の図は、次のステップによって、プライマリ・データベースAからプライマリ・データベースBにデータをレプリケートして、プライマリBからプライマリAに戻している様子を示しています。

1. プライマリ・データベースA: プライマリAのLogminerサーバーは、データベースAのプライマリGGHubにあるASHリージョンのGGHub Extractプロセスにredo変更を送信します。
2. プライマリGGHub: Extractプロセスによって、証跡ファイルに変更が書き込まれます。
3. プライマリGGHubからプライマリ・データベースB: ASHリージョンのGoldenGate Replicatプロセスは、該当する変更をターゲット・データベース(プライマリB)に適用します。
4. プライマリ・データベースB: プライマリBのLogminerサーバーは、データベースBのプライマリGGHubにあるPHXリージョンのGGHub Extractプロセスにredoを送信します。
5. プライマリGGHub: Extractプロセスによって、証跡ファイルに変更が書き込まれます。
6. プライマリGGHubからプライマリ・データベースA: PHXリージョンのGoldenGate Replicatプロセスは、該当する変更をターゲット・データベース(プライマリA)に適用します。

表19-2 異なるOCIリージョン内のGGHubの停止シナリオ、修復および冗長性のリストア

停止シナリオ	アプリケーションの可用性と修復	冗長性と初期状態の復元
プライマリ・データベース A (またはデータベース B)の障害	影響: アプリケーション停止時間は、ほぼゼロです。GoldenGate レプリケーションは、新しいプライマリ・データベースの起動時に再開されます。 1. 1 つのプライマリ・データベースが引き続き使用できます。すべてのアクティビティは、アプリケーションの停止時間がゼロになるように、使用可能な既存のプライマリ・データベースにルーティングされます。たとえば、アプリケーション・サービスの A から F はデータベース A にルーティングされ、アプリケーション・サービスの G から J はデータベース B にルーティングされているとします。データベース A に障害が発生すると、すべてのサービスは一時的にデータベース B に移動します。 2. スタンバイは、Data Guard FSFO によって自動的に新しいプライマリになります。GoldenGate レプリケーションが再開され、プライマリ・データベースが再同期されます。データ損失は、Data Guard 保護レベルによって制限されます。最大可用性または最大保護が構成されている場合は、データ損失がゼロになります。すべてのコミットされたトランザクションは、一方または両方のデータベースにあります。ワークロードは、プライマリ・データベース A とデータベース B が使用可能であり同期状態にあるときにはリバランスできます。たとえば、データベース A が稼働中で同期状態にある場合、サービス A から F はデータベース A に戻すことができます。 3. Replicat のパフォーマンスは、プライマリ GGHub がターゲット・データベースと同じリージョンに存在していないと低下するようになります。ACFS レプリケーション・スイッチオーバーによる GGHub スイッチオーバーをスケジュールして、ターゲット・データベースに対する最適な Replicat パフォーマンスを再開します。そのようにすると、同じ GGHub クラスタで 2 つのアクティブな GGhub 構成が発生する可能性があります。	1. 元のプライマリ・データベースは、冗長性をリストアするために新しいスタンバイ・データベースとして回復されます。 2. オプションで、Data Guard スイッチオーバーを実行し、元の構成に戻すことで 1 つ以上のプライマリ・データベースが個別の AD 内に存在するようにします。ACFS レプリケーション・スイッチオーバーによる GGHub スイッチオーバーをスケジュールして、ターゲット・データベースに対する最適な Replicat パフォーマンスを再開します。
プライマリまたはスタンバイ GGHub の単一ノード障害	影響: アプリケーションに影響はありません。GoldenGate レプリケーションは、数分後に自動的に再開されます。 処置は必要ありません。GGHub に組み込まれた HA フェイ	ノードが再起動すると、アクティブ/パッシブ構成が再確立されます。

停止シナリオ	アプリケーションの可用性と修復	冗長性と初期状態の復元
	ルオーバー・ソリューションには、GoldenGate プロセスとアクティビティの自動フェイルオーバーおよび再起動が含まれます。レプリケーション・アクティビティは、GoldenGate プロセスが再度アクティブになるまでブロックされます。GoldenGate レプリケーションのブラックアウトは数分間続くことがあります。	
プライマリ GGHub クラスタがクラッシュしてリカバリできない	影響：アプリケーションに影響はありません。GoldenGate レプリケーションは、既存のプライマリ GGHub の再起動後または手動による GGHub フェイルオーバーの完了後に再開されます。 1. GGHub クラスタの再起動が可能な場合は、それが最も単純なソリューションです。これは分単位で制限する必要があります。GGHub クラスタの再起動を試行する方が簡単です。 2. プライマリ GGHub がリカバリ不可能な場合は、スタンバイ GGHub への手動による GGHub フェイルオーバー(ACFS フェイルオーバーを含む)を実行します。これには通常、数分かかります。 3. レプリケーションは、新しいプライマリ GGhub が起動されるまで停止するため、ステップ 1 またはステップ 2 は迅速に実行する必要があります。オーケストレーションが存在する場合、これは自動化する必要があります。	1. 前の GGHub が最終的に再起動されると、ACFS レプリケーションは別の方向に自動的に再開されます。GGHub クラスタが損なわれた場合やリカバリ不能になった場合は、新しいスタンバイ GGHub を再構築する必要があります。 2. Replicat のパフォーマンスは、プライマリ GGHub がターゲット・データベースと同じリージョンに存在していないと低下します。ACFS レプリケーション・スイッチオーバーによる GGHub スイッチオーバーをスケジュールして、ターゲット・データベースに対する最適な Replicat パフォーマンスを再開します。
スタンバイ GGHub クラスタがクラッシュしてリカバリできない	影響：アプリケーションまたはレプリケーションに影響はありません。 1. GGHub クラスタが再起動可能な場合は、それが最も単純なソリューションであり、それにより ACFS レプリケーションが再開されます。 2. スタンバイ GGHub がリカバリ不可能な場合は、新しいスタンバイ GGHub を再構築することになります。	N/A
リージョン全体の障害	影響：アプリケーションの停止時間は、ほぼゼロです。GoldenGate レプリケーションは、新しいプライマリ・データベースが起動すると再開されます。 1. 1 つのプライマリ・データベースが引き続き使用できます。すべてのアクティビティは、アプリケーションの停止時間がゼロになるように、使用可能な既存のプライマ	1. OCI リージョンの復帰時に、スタンバイの回復などの構成を再確立します。前の GGHub が最終的に再起動されると、ACFS レプリケーションは別の方向に自動的に再開されます。 2. 可能な場合は、Data Guard

停止シナリオ	アプリケーションの可用性と修復	冗長性と初期状態の復元
	リ・データベースにルーティングされます。たとえば、アプリケーション・サービスの A から F はデータベース A にルーティングされていて、アプリケーション・サービスの G から J はデータベース B にルーティングされているとします。データベース A に障害が発生すると、すべてのサービスは一時的にデータベース B に移動するようになります。 2. プライマリ GGHub がまだ機能している場合は、GoldenGate レプリケーションが続行されます。リージョンの障害が原因でプライマリ GGHub が損なわれた場合は、手動による GGhub フェイルオーバーが必要です。GoldenGate レプリケーションが再開され、プライマリ・データベースが再同期されます。データ損失は、Data Guard 保護レベルによって制限されます。最大の可用性または保護が構成されている場合は、データ損失がゼロになります。すべてのコミットされたトランザクションは、一方または両方のデータベースにあります。ワークロードは、プライマリ・データベース A とデータベース B が使用可能であり同期状態にあるときにはリバランスできます。データベース A が稼働中で同期状態にある場合、サービス A から F はデータベース A に戻すことができます。	スイッチオーバー(フェイルバック)を実行して、各リージョンに 1 つのプライマリ・データベースが存在する元の状態に戻します。 3. Replicat のパフォーマンスは、プライマリ GGHub がターゲット・データベースと同じリージョンに存在していないと低下します。ACFS レプリケーション・スイッチオーバーによる GGHub スイッチオーバーをスケジュールして、ターゲット・データベースに対する最適な Replicat パフォーマンスを再開します。

タスク1: Oracle GoldenGateのソースおよびターゲット・データベースの構成

ソースおよびターゲットのOracle GoldenGateデータベースは、次に示す推奨事項を使用して構成する必要があります。

このタスクを完了するには、次のステップを実行します。

- ステップ1.1 - データベース構成
- ステップ1.2 - データベース・レプリケーション管理者ユーザーの作成
- ステップ1.3 - データベース・サービスの作成

ステップ1.1 - データベース構成

ソースおよびターゲットのOracle GoldenGateデータベースは、次に示す推奨事項を使用して構成する必要があります。

構成	有効範囲	例
ARCHIVELOG モードを有効にする	ソース、ターゲット	`SQL> ARCHIVE LOG LIST Database log mode Archive Mode Automatic archival Enabled Archive destination USE_DB_RECOVERY_FILE_DEST Oldest online log sequence 110 Next log sequence to archive 113 Current log sequence 113`
`FORCE LOGGING` を有効にする	ソース、ターゲット	`ALTER DATABASE FORCE LOGGING;`
`ENABLE_GOLDENGATE_REPLICATION`	ソース、ターゲット、スタンバイ	`ALTER SYSTEM SET ENABLE_GOLDENGATE_REPLICATION=TRUE SCOPE=BOTH SID='*';`
サプリメンタル・ロギング	ソース レプリケーションの逆転の場合にターゲットで必要	`ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;`
レプリケートされたオブジェクトのスキーマまたは表レベルのログを追加する	ソース レプリケーションの逆転の場合にターゲットで必要	`ADD SCHEMATRANDATA` または `ADD T`
`STREAMS_POOL_SIZE`	ソース レプリケーションの逆転の場合にターゲットで必要	`STREAMS_POOL_SIZE` の値は、次に示す値に設定する必要があります。 `STREAMS_POOL_SIZE = (((#Extracts + #Integrated Replicats) * 1GB) * 1.25)` たとえば、2 つの Extract と 2 つの統合 Replicat があるデータ

構成	有効範囲	例

		ベースの場合は、次のようになります。
		
		`STREAMS_POOL_SIZE = 4GB * 1.25 = 5GB`
		
		`ALTER SYSTEM SET STREAMS_POOL_SIZE=5G SCOPE=BOTH SID='*';`

Oracle GoldenGate用にデータベースを準備するステップは、Oracle GoldenGate用のデータベースの準備を参照してください。

ステップ1.2 - データベース・レプリケーション管理者ユーザーの作成

ソースおよびターゲット・データベースには、次のように適切な権限が割り当てられたGoldenGate管理者ユーザーを作成する必要があります。

- マルチテナント・コンテナ・データベース(CDB)の場合:
  - ソース・データベースであるGoldenGate Extractは、c##を使用してルート・コンテナ・データベース内のユーザーに接続するように構成する必要があります
  - ターゲット・データベースには、プラガブル・データベース(PDB)ごとに個別のGoldenGate管理者ユーザーが必要です。
  - Oracleマルチテナント・データベースでのGoldenGate管理者の作成の詳細は、マルチテナント・コンテナ・データベースでのOracle GoldenGateの構成を参照してください。
- 非CDBデータベースについては、Oracle GoldenGate資格証明の確立を参照してください

ソース・データベース・システムのoracle OSユーザーとして、次のSQL命令を実行してOracle GoldenGateのデータベース・ユーザーを作成し、必要な権限を割り当てます。

```
[opc@exadb1_node1 ~]$ sudo su - oracle
[oracle@exadb1_node1 ~]$ source dbName.env
[oracle@exadb1_node1 ~]$ sqlplus / as sysdba
# Source CDB
SQL>
alter session set container=cdb$root;
create user c##ggadmin identified by "ggadmin_password" container=all default
tablespace USERS temporary tablespace temp;
alter user c##ggadmin quota unlimited on users;
grant set container to c##ggadmin container=all;
grant alter system to c##ggadmin container=all;
grant create session to c##ggadmin container=all;
grant alter any table to c##ggadmin container=all;
grant resource to c##ggadmin container=all;
exec dbms_goldengate_auth.grant_admin_privilege('c##ggadmin',container=>'all');
# Source PDB
SQL>
alter session set container=pdbName;
create user ggadmin identified by "ggadmin_password" container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

ターゲット・システムのoracle OSユーザーとして、次のSQL命令を実行してOracle GoldenGateのデータベース・ユーザーを作成し、必要な権限を割り当てます。

```
[opc@exadb2_node1 ~]$ sudo su - oracle
[oracle@exadb2_node1 ~]$ source dbName.env
```

```
[oracle@exadb2_node1 ~]$ sqlplus / as sysdba
# Target PDB
SQL>
alter session set container=pdbName;
create user ggadmin identified by "ggadmin_password" container=current;
grant alter system to ggadmin container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
grant dv_goldengate_admin, dv_goldengate_redo_access to ggadmin container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

ステップ1.3 - データベース・サービスの作成

ソース・データベースとターゲット・データベースがOracle Data Guardを使用するOracle RACクラスタで推奨構成を実行している場合は、ExtractまたはReplicatプロセスが適切なData Guardプライマリ・データベース・インスタンスに接続できるロールベースのサービスを作成する必要があります。

ソース・マルチテナント・データベースを使用する場合は、ルート・コンテナ・データベース(CDB)とレプリケートされるスキーマを含むプラガブル・データベース(PDB)に、個別のサービスが必要です。ターゲット・マルチテナント・データベースの場合は、PDBに単一のサービスが必要です。

プライマリ・データベース・システムのoracle OSユーザーとして、次に示すように、dbaascliを使用してCDBおよびPDB名を検索します。

```
[opc@exadb1_node1 ~]$ sudo su - oracle
[oracle@exadb1_node1 ~]$ source dbName.env
[oracle@exadb1_node1 ~]$ dbaascli database getDetails
 --dbname dbName |egrep 'dbName|pdbName'
  "dbName" : "dbName",
    "pdbName" : "pdbName",
```

プライマリおよびスタンバイ・データベース・システムのoracle OSユーザーとして、次のコマンドを使用してCDBデータベース・サービスを作成および開始します。

```
[opc@exadb1_node1 ~]$ sudo su - oracle
[oracle@exadb1_node1 ~]$ source dbName.env
[oracle@exadb1_node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service dbName.goldengate.com -preferred ORACLE_SID1
 -available ORACLE_SID2 -role PRIMARY
```

プライマリおよびスタンバイ・データベース・システムのoracle OSユーザーとして、次のコマンドを使用してPDBデータベース・サービスを作成および開始します。

```
[oracle@exadb1_node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service dbName.pdbName.goldengate.com -preferred ORACLE_SID1
 -available ORACLE_SID2 -pdb pdbName -role PRIMARY
```

プライマリおよびスタンバイ・データベース・システムのoracle OSユーザーとして、次に示すように、開始してサービスを実行していることを確認します。

```
[oracle@exadb1_node1 ~]$ srvctl start service -db $ORACLE_UNQNAME -role
[oracle@exadb1_node1 ~]$ srvctl status service -d $ORACLE_UNQNAME |grep goldengate
Service dbName.goldengate.com  is running on instance(s) SID1
Service dbName.pdbName.goldengate.com is running on instance(s) SID1
```

ノート:

ソースおよびターゲット・データベース・システムでステップ 1.3 を繰り返します。

タスク2: GGHubのプライマリおよびスタンバイ・ベース・システムの準備

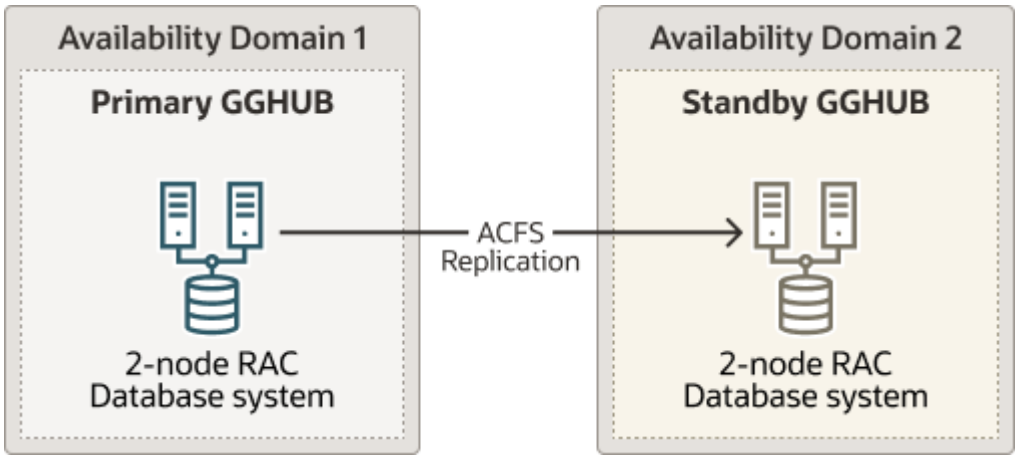
このタスクを完了するには、次のステップを実行します。

- ステップ2.1 - Oracle 2ノード・クラスタ・システムのデプロイ
- ステップ2.2 - 標準データベースの削除とディスクグループ・レイアウトの再配置
- ステップ2.3 - 必要なソフトウェアのダウンロード
- ステップ2.4 - Oracle Public YUMリポジトリの使用に向けたOracle Linuxの構成

ステップ2.1 - Oracle 2ノード・クラスタ・システムのデプロイ

リージョンごとに少なくとも2つのGGHubをデプロイします(プライマリおよびスタンバイ)。それぞれのGGHubは、Oracle Base Database Serviceの説明に従って、2ノードのOracle RACデータベース・システムとしてデプロイする必要があります。

図19-3 Oracle GoldenGate Hubハードウェア・アーキテクチャ



ステップ2.2 - 標準データベースの削除とディスクグループ・レイアウトの再配置

1. 最初のGGHubノードのoracle OSユーザーとして、標準データベースを削除します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghubN-node1 ~]$ dbca -deleteDatabase -silent -sourceDB $ORACLE_UNQNAME
Enter SYS user password: ###############
[WARNING] [DBT-19202] The Database Configuration Assistant will delete the
Oracle instances and datafiles for your database. All information in the
database will be destroyed.
Prepare for db operation
32% complete
Connecting to database
39% complete
...
100% complete
Database deletion completed.
Look at the log file
"/u01/app/oracle/cfgtoollogs/dbca/DB0502_fra2pr/DB0502_fra2pr.log" for further
details.
```

2. 2番目のGGHubノードのgrid OSユーザーとして、RECOディスクグループをディスマウントします。

```
[opc@gghub_prim2 ~]$ sudo su - grid
[grid@gghub_prim2 ~]$ sqlplus / as sysasm
```

```
SQL> alter diskgroup RECO dismount;
```

3. 最初のgghubノードのgrid OSユーザーとして、RECOディスク・グループを削除し、ディスクをDATAディスクグループに割り当てます。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ sqlplus / as sysasm
SQL>
drop diskgroup RECO INCLUDING CONTENTS;
alter diskgroup DATA add disk '/dev/RECODISK1';
alter diskgroup DATA add disk '/dev/RECODISK2';
alter diskgroup DATA add disk '/dev/RECODISK3';
alter diskgroup DATA add disk '/dev/RECODISK4';
```

4. すべてのGGhubノードのroot OSユーザーとして、ノードを再起動します。

```
[opc@gghub_prim1 ~]$ sudo reboot
```

ノート:

プライマリおよびスタンバイの GGHub でこのステップを繰り返します。

ステップ2.3 - 必要なソフトウェアのダウンロード

1. すべてのGGHubノードのopc OSユーザーとして、ステージング・ディレクトリとスクリプト・ディレクトリを作成します。

```
[opc@gghub_prim1 ~]$
sudo mkdir -p /u01/oracle/stage
sudo mkdir /u01/oracle/scripts
sudo chown -R oracle:oinstall /u01/oracle
sudo chmod -R g+w /u01/oracle
sudo chmod -R o+w /u01/oracle/stage
```

2. すべてのGGHubノードでopc OSユーザーとして、/u01/oracle/stageディレクトリに次のソフトウェアをダウンロードします。

- Oracle GoldenGateのダウンロードから、Oracle GoldenGate 21c (またはそれ以降のリリース) Microservicesソフトウェアをダウンロードします。
- My Oracle Supportの「パッチと更新版」タブから、それ以降のパッチをベース・リリースにダウンロードします。
  - 詳細は、「Oracle GoldenGate Microservices Architecture用のパッチのインストール」を参照してください。
  - 最低限必要なバージョンは、パッチ35214851: Oracle GoldenGate 21.9.0.0.2 Microservices for Oracle
- My Oracle Supportドキュメント2542082.1から、Oracle Database 21c (21.0.0.0.0)用の最新のOPatchリリース(パッチ6880880)をダウンロードします。
- Oracle Clusterware用Oracle Grid Infrastructure Standalone Agentから、Oracle Clusterware 19cのリリース10.2以上用のOracle Grid Infrastructure Standalone Agentをダウンロードします。
- My Oracle Supportのドキュメント2826001.1からpythonスクリプト(secureServices.py)をダウンロードします
- My Oracle Supportのドキュメント2951572.1からOracle GGHUBスクリプトをダウンロードします

3. すべてのGGHubノードでgrid OSユーザーとして、My Oracle Supportのドキュメント2951572.1からダウンロードしたGGhubスクリプト・ファイルを/u01/oracle/scriptsディレクトリに解凍します。

   このスクリプトは、すべてのプライマリおよびスタンバイGGhubノード上で同じ場所に配置します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip -q /u01/oracle/stage/gghub_scripts_YYYYYMMDD.zip -d
/u01/oracle/scripts/
```

ステップ2.4 - Oracle Public YUMリポジトリの使用に向けたOracle Linuxの構成

Oracle Linux yumサーバーは、Oracle Linuxおよび互換性のあるディストリビューション用のソフトウェアをホストします。これらの手順は、LinuxシステムをOracle Linux yumサーバー用に構成し、yumを介してソフトウェアをインストールする作業を開始するのに役立ちます。

- すべてのGGHubシステムのroot OSユーザーとして、次の内容のファイル/etc/yum.repos.d/oracle-public-yum-ol7.repoを作成します。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]#
cat > /etc/yum.repos.d/oracle-public-yum-ol7.repo <<EOF
[ol7_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://yum$ociregion.oracle.com/repo/OracleLinux/OL7/latest/¥$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

タスク3: プライマリおよびスタンバイGGHubのためのOracle GoldenGateの構成

このタスクを完了するには、次のステップを実行します。

- ステップ3.1 - Oracle GoldenGateソフトウェアのインストールとパッチ適用
- ステップ3.2 - Oracle GoldenGate Hubアーキテクチャ・ネットワーク構成の設定
- ステップ3.3 - 同じリージョン内のGGHUB間のACFSファイル・システム・レプリケーションの構成

ステップ3.1 - Oracle GoldenGateソフトウェアのインストールとパッチ適用

Oracle GoldenGateソフトウェアのインストールとパッチ適用は、GoldenGate構成に含めるプライマリおよびスタンバイGGHub構成のすべてのノードで実施します。インストール・ディレクトリは、すべてのノードで同一にしてください。

次のサブステップを実行して、このステップを完了します。

- ステップ3.1.1 ソフトウェアの解凍とインストール用レスポンス・ファイルの作成
- ステップ3.1.2 Oracle GoldenGateソフトウェアのインストール
- ステップ3.1.3 Oracle GoldenGate Microservices Architecture用のパッチのインストール

ステップ3.1.1 ソフトウェアの解凍とインストール用レスポンス・ファイルの作成

すべてのGGHubノードのoracle OSユーザーとして、Oracle GoldenGateソフトウェアを解凍します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ unzip
 /u01/oracle/stage/213000_fbo_ggs_Linux_x64_Oracle_services_shiphome.zip
 -d /u01/oracle/stage
```

このソフトウェアには、Oracle Database 21c以前のサポート対象バージョンに対応するレスポンス・ファイルの例が含まれています。レスポンス・ファイルは、すべてのデータベース・ノードへのOracle GoldenGateのインストールに同じファイルを使用できるように共有ファイル・システムにコピーして、次のパラメータを編集します。

- INSTALL_OPTION=ora21c
- SOFTWARE_LOCATION=/u01/app/oracle/goldengate/gg21c (推奨の場所)

すべてのGGHubノードのoracle OSユーザーとして、インストール用のレスポンス・ファイルをコピーして編集します。

```
[oracle@gghub_prim1 ~]$ cp

/u01/oracle/stage/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/response/oggcore.r
sp
 /u01/oracle/stage
[oracle@gghub_prim1 ~]$ vi /u01/oracle/stage/oggcore.rsp
# Before
INSTALL_OPTION=
SOFTWARE_LOCATION=
# After
INSTALL_OPTION=ora21c
SOFTWARE_LOCATION=/u01/app/oracle/goldengate/gg21c
```

ステップ3.1.2 Oracle GoldenGateソフトウェアのインストール

すべてのGGHubノードでoracle OSユーザーとして、runInstallerを実行してOracle GoldenGateをインストールします。

```
[oracle@gghub_prim1 ~]$ cd
```

```
 /u01/oracle/stage/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/
[oracle@gghub_prim1 ~]$ ./runInstaller -silent -nowait
 -responseFile /u01/oracle/stage/oggcore.rsp
Starting Oracle Universal Installer...
Checking Temp space: must be greater than 120 MB.   Actual 32755 MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 16383 MB    Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_02-54-51PM.
 Please wait ...
You can find the log of this install session at:
 /u01/app/oraInventory/logs/installActions2022-07-08_02-54-51PM.log
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check
 '/u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log'
 for more details.
[oracle@gghub_prim1 ~]$ cat
 /u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log
The installation of Oracle GoldenGate Services was successful.
```

ステップ3.1.3 Oracle GoldenGate Microservices Architecture用のパッチのインストール

すべてのGGHubノードのoracle OSユーザーとして、最新のOPatchをインストールします。

```
[oracle@gghub_prim1 ~]$ unzip -oq -d
 /u01/app/oracle/goldengate/gg21c
 /u01/oracle/stage/p6880880_210000_Linux-x86-64.zip
[oracle@gghub_prim1 ~]$ cat >> ~/.bashrc <<EOF
export ORACLE_HOME=/u01/app/oracle/goldengate/gg21c
export PATH=$ORACLE_HOME/OPatch:$PATH
EOF
[oracle@gghub_prim1 ~]$ . ~/.bashrc
[oracle@gghub_prim1 ~]$ opatch lsinventory |grep
 'Oracle GoldenGate Services'
Oracle GoldenGate Services                    21.1.0.0.0
[oracle@gghub_prim1 Disk1]$ opatch version
OPatch Version: 12.2.0.1.37
OPatch succeeded.
```

すべてのGGHubノードでoracle OSユーザーとして、OPatchのprereqを実行して、パッチの適用前に競合があるかどうかを検証します。

```
[oracle@gghub_prim1 ~]$ unzip -oq -d /u01/oracle/stage/
 /u01/oracle/stage/p35214851_219000OGGRU_Linux-x86-64.zip
[oracle@gghub_prim1 ~]$ cd /u01/oracle/stage/35214851/
[oracle@gghub_prim1 35214851]$ opatch prereq
 CheckConflictAgainstOHWithDetail -ph ./
Oracle Interim Patch Installer version 12.2.0.1.26
Copyright (c) 2023, Oracle Corporation. All rights reserved.
PREREQ session
Oracle Home    : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
 from      : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version  : 12.2.0.1.26
OUI version    : 12.2.0.9.0
Log file location :
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/opatch2023-04-21_13-44-
16PM_1.log
Invoking prereq "checkconflictagainstohwithdetail"
Prereq "checkConflictAgainstOHWithDetail" passed.
OPatch succeeded.
```

すべてのGGHubノードでoracle OSユーザーとして、OPatchを使用してOracle GoldenGate Microservices Architechtureにパッチを適用します。

```
[oracle@gghub_prim1 35214851]$ opatch apply
Oracle Interim Patch Installer version 12.2.0.1.37
Copyright (c) 2023, Oracle Corporation. All rights reserved.
Oracle Home    : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
 from      : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version  : 12.2.0.1.37
OUI version    : 12.2.0.9.0
Log file location :
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/opatch2023-04-21_19-40-
41PM_1.log
Verifying environment and performing prerequisite checks...
OPatch continues with these patches:  35214851
Do you want to proceed? [y|n]
y
User Responded with: Y
All checks passed.
Please shutdown Oracle instances running out of this ORACLE_HOME on
 the local system.
(Oracle Home = '/u01/app/oracle/goldengate/gg21c'
Is the local system ready for patching? [y|n]
y
User Responded with: Y
Backing up files...
Applying interim patch '35214851' to OH '/u01/app/oracle/goldengate/gg21c'
Patching component oracle.oggcore.services.ora21c, 21.1.0.0.0...
Patch 35214851 successfully applied.
Log file location:
 /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/opatch2023-04-21_19-40-
41PM_1.log
OPatch succeeded.
[oracle@gghub_prim1 35214851]$ opatch lspatches
35214851;
OPatch succeeded.
```

ノート:

プライマリおよびスタンバイの GGHub システムについて、ステップ 3.1 のすべての手順を繰り返します。

ステップ3.2 - クラウド・ネットワークの構成

Oracle GoldenGateが正しく機能するように、プライベートDNSゾーン、VIP、要塞、セキュリティ・リスト、ファイアウォールなどの仮想クラウド・ネットワーク(VCN)のコンポーネントを構成する必要があります。

VCNとセキュリティ・リストの詳細と作成手順は、Oracle Cloud Infrastructure Networkingのドキュメントを参照してください。

次のサブステップを実行して、このステップを完了します。

- ステップ3.2.1 - GGhub用のアプリケーション仮想IPアドレス(VIP)の作成
- ステップ3.2.2 - ポート443のイングレス・ルールの追加
- ステップ3.2.3 - GGhubファイアウォールでポート443を開く
- ステップ3.2.4 - プライマリおよびスタンバイGGHUBシステム間のネットワーク接続の構成
- ステップ3.2.5 - プライベートDNSゾーンのビューとリゾルバの構成

ステップ3.2.1 - GGhub用のアプリケーション仮想IPアドレス(VIP)の作成

専用のアプリケーションVIPは、クラスタのどのノードでサービスがホストされていても同じホスト名を使用してGoldenGate

Microservicesにアクセスできるようにするために必要です。VIPはGGHUBシステムに割り当てられ、ノード障害が発生すると自動的に別のノードに移行されます。2つのVIPが必要になります。その1つはプライマリ用、もう1つはスタンバイGGHUB用です。

すべてのGGhubノードのgrid OSユーザーとして、次のコマンドを実行し、リソースora.net1.networkの同じサブネット内にあるプライベート・エンドポイントのvnicIdを取得します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl status resource -p -attr NAME,USR_ORA_SUBNET
 -w "TYPE = ora.network.type" |sort | uniq
NAME=ora.net1.network
USR_ORA_SUBNET=10.60.2.0
[grid@gghub_prim1 ~]$ curl 169.254.169.254/opc/v1/vnics
[
  {
    "macAddr": "02:00:17:04:70:AF",
    "privateIp": "10.60.2.120",
    "subnetCidrBlock": "10.60.2.0/24",
    "virtualRouterIp": "10.60.2.1",
    "vlanTag": 3085,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  },
  {
    "macAddr": "02:00:17:08:69:6E",
    "privateIp": "192.168.16.18",
    "subnetCidrBlock": "192.168.16.16/28",
    "virtualRouterIp": "192.168.16.17",
    "vlanTag": 879,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  }
[grid@gghub_prim2 ~]$ curl 169.254.169.254/opc/v1/vnics
[
  {
    "macAddr": "00:00:17:00:C9:19",
    "privateIp": "10.60.2.148",
    "subnetCidrBlock": "10.60.2.0/24",
    "virtualRouterIp": "10.60.2.1",
    "vlanTag": 572,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  },
  {
    "macAddr": "02:00:17:00:84:B5",
    "privateIp": "192.168.16.19",
    "subnetCidrBlock": "192.168.16.16/28",
    "virtualRouterIp": "192.168.16.17",
    "vlanTag": 3352,
    "vnicId": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
  }
```

ノート:

次のステップでは、GGHUB ノードにプライベート IP を割り当てるために、Cloud Shell を使用する必要があります。詳細は、Cloud Shell の使用を参照してください。

Cloud Shellのユーザーとして、次のコマンドを実行し、GGHUBノードにプライベートIPを割り当てます。

```
username@cloudshell:~ (eu-frankfurt-1)$ export node1_vnic=
'ocid1.vnic.oc1.eu-frankfurt-
1.abtheljrl5udtgryrscypy5btmlfncawqkjlcql3kkpj64e2lb5xbmbrehkq'
username@cloudshell:~ (eu-frankfurt-1)$ export node2_vnic=
'ocid1.vnic.oc1.eu-frankfurt-
1.abtheljre6rf3xoxtgl2gam3lav4vcyftz5fppm2ciin4wzjxucalzj7b2bq'
username@cloudshell:~ (eu-frankfurt-1)$ export ip_address='10.60.2.65'
username@cloudshell:~ (eu-frankfurt-1)$ oci network vnic assign-private-ip
```

```
 --unassign-if-already-assigned --vnic-id $node1_vnic --ip-address $ip_address
username@cloudshell:~ (eu-frankfurt-1)$ oci network vnic assign-private-ip
 --unassign-if-already-assigned --vnic-id $node2_vnic --ip-address $ip_address
Example of the output:
{
 "data": {
  "availability-domain": null,
   "compartment-id": "ocid1.compartment.oc1..ocid_value",
  "defined-tags": {},
   "display-name": "privateip20230292502117",
  "freeform-tags": {},
  "hostname-label": null,
  "id": "ocid1.privateip.oc1.eu-frankfurt-1.ocid_value",
   "ip-address": "10.60.2.65",
  "is-primary": false,
  "subnet-id": "ocid1.subnet.oc1.eu-frankfurt-1.ocid_value",
  "time-created": "2023-07-27T10:21:17.851000+00:00",
  "vlan-id": null,
  "vnic-id": "ocid1.vnic.oc1.eu-frankfurt-1.ocid_value"
 },
  "etag": "da972988"
}
```

最初のGGhubノードでroot OSユーザーとして、次のコマンドを実行して、Oracle Clusterwareで管理されるアプリケーションVIPを作成します。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_appvip.sh
Application VIP Name: gghub_prim_vip
Application VIP Address: 10.60.2.65
Using configuration parameter file:
 /u01/app/19.0.0.0/grid/crs/install/crsconfig_params
The log of current session can be found at:
 /u01/app/grid/crsdata/gghublb1/scripts/appvipcfg.log
```

ノート:

プライマリおよびスタンバイの GGHUB システムについて、ステップ 3.2.1 のすべての手順を繰り返します。

ステップ3.2.2 - イングレス・セキュリティ・リスト・ルールの追加

Cloudコンソールを使用して、そのGGhubに割り当てられたVirtual Cloud Network (VCN)に、2つのイングレス・セキュリティ・リスト・ルールを追加します。

一方のイングレス・ルールは、リバース・プロキシとしてNGINXを使用してOracle GoldenGateサービスに接続するための、認可されたソースIPアドレスと任意のソース・ポートからの宛先ポート443でのTCPトラフィック用であり、他方は、ACFSレプリケーションの有効化に必要な、プライマリとスタンバイのGGhubの間でのICMP TYPE 8 (ECHO)を許可するためのものです。詳細は、セキュリティ・リストの使用およびMy Oracle Supportのドキュメント2584309.1を参照してください。

セキュリティ・リストの更新後、リストには次のような値のエントリが登録されます。

1. NGINX - TCP 443
    - ソース・タイプ: CIDR
    - ソースCIDR: 0.0.0.0/0
    - IPプロトコル: TCP
    - ソース・ポート範囲: All
    - 宛先ポート範囲: 443

- 次のポートに対してTCPトラフィックを許可: 443 HTTPS
- 説明: Oracle GoldenGate 443

2. ACFS - ICMP TYPE 8 (ECHO)

- ソース・タイプ: CIDR
- ソースCIDR: 0.0.0.0/0
- IPプロトコル: ICMP
- 許可: 次の場合のICMPトラフィック: 8エコー
- 説明: ACFSレプリケーションに必要

ステップ3.2.3 - GGhubファイアウォールでポート443を開く

プライマリ・システムとスタンバイ・システムのすべてのGGhubノードのopc OSユーザーとして、IPTablesに必要なルールを追加します。

```
[opc@gghub_prim1 ~]$ sudo vi /etc/sysconfig/iptables
-A INPUT -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
 -m comment --comment "Required for access to GoldenGate, Do not remove
 or modify. "
-A INPUT -p tcp -m state --state NEW -m tcp --match multiport
 --dports 9100:9105 -j ACCEPT -m comment --comment "Required for access
 to GoldenGate, Do not remove or modify. "
[opc@gghub_prim1 ~]$ sudo systemctl restart iptables
```

ノート:

詳細は、Oracle Linux セキュリティの実装を参照してください。

ステップ3.2.4 - プライマリおよびスタンバイGGHUBシステム間のネットワーク接続の構成

Oracle ACFSスナップショットベースのレプリケーションでは、sshをプライマリ・クラスタとスタンバイ・クラスタ間のトランスポートとして使用します。ACFSをサポートするには、クラスタ間のどちらの方向(プライマリ・クラスタからスタンバイ・クラスタ、スタンバイからプライマリ)でもsshを使用できることが必要です。『*Oracle Automatic Storage Management*管理者ガイド』のOracle ACFSレプリケーションに使用するためのsshの構成を参照してください。

サブネットがパブリックとプライベートのどちらであるかの詳細や接続の作成手順は、Oracle Cloud Infrastructure Networkingドキュメントの接続の選択肢を参照してください。

ステップ3.2.5 - プライベートDNSゾーンのビューとリゾルバの構成

プライベートDNSゾーンのビューとレコードは、アプリケーションVIPごとに作成する必要があります。これは、プライマリGGHUBがスタンバイGGHUBデプロイメントのVIPホスト名に到達するために必要です。

「プライベートDNSゾーンのビューとリゾルバの構成」のステップに従って、ステップ3.2.1で作成した専用のGGHUBアプリケーション仮想IPアドレス(VIP)ごとに、プライベートDNSゾーンとレコード・エントリを作成します。

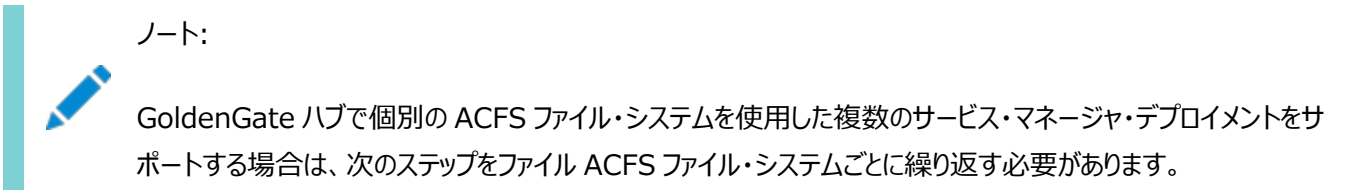
任意のGGhubノードでopc OSユーザーとして、すべてのアプリケーションVIPを解決できることを検証します。

```
[opc@gghub_prim1 ~]$ nslookup
 gghub_prim_vip.frankfurt.goldengate.com |tail -2
Address: 10.60.2.120
[opc@gghub_prim1 ~]$ nslookup
 gghub_stby_vip.frankfurt.goldengate.com |tail -2
Address: 10.60.0.185
```

ステップ3.3 - 同じリージョン内のGGHUB間のACFSファイル・システム・レプリケーションの構成

Oracle GoldenGate Microservices Architectureの設計では、インストールとデプロイメントのディレクトリ構造が簡略化されています。インストール・ディレクトリ: ソフトウェア・パッチ適用中の停止時間を最小限に抑えるために、各データベース・ノードのローカル記憶域に配置することが必要です。デプロイメント・ディレクトリ: デプロイメントの作成中にOracle GoldenGate Configuration Assistant (oggca.sh)を使用して作成されます。このディレクトリは、共有ファイル・システムに配置する必要があります。デプロイメント・ディレクトリには、構成、セキュリティ、ログ、パラメータ、証跡およびチェックポイントのファイルが含まれます。Oracle Automatic Storage Management Cluster File System (ACFS)にデプロイメントを配置すると、システム障害が発生した際に、最高のリカバリ性とフェイルオーバー機能が提供されます。クラスタ全体でチェックポイント・ファイルの可用性を確保することは、障害発生後にGoldenGateプロセスが最後に認識された位置から実行を継続できるようにするために不可欠です。

証跡ファイルのディスク領域には、12時間以上の証跡ファイルに対して十分な量を割り当てることをお薦めします。そうすることで、新しい証跡ファイルを受信できないターゲット環境で問題が発生した場合に、証跡ファイルの生成に十分な領域を確保します。12時間に必要な領域の量は、実際の本番データで証跡ファイルの生成率をテストすることでのみ決定できます。
GoldenGateプライマリ・データベースまたはシステムのいずれかの長期計画メンテナンス・イベントの緊急対応策を確立する場合は、2日間分の十分なACFS領域を割り当てるようにします。領域使用率の監視は、どれだけの領域が割り当てられていても、常にお薦めします。



ノート:

GoldenGate ハブで個別の ACFS ファイル・システムを使用した複数のサービス・マネージャ・デプロイメントをサポートする場合は、次のステップをファイル ACFS ファイル・システムごとに繰り返す必要があります。

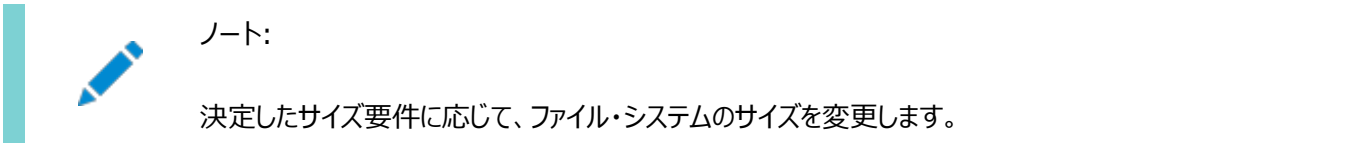
次のサブステップを実行して、このステップを完了します。

- ステップ3.3.1 - ASMファイル・システムの作成
- ステップ3.3.2 - Cluster Ready Services (CRS)リソースの作成
- ステップ3.3.3 - 現在構成されているACFSファイル・システムの確認
- ステップ3.3.4 - ACFSリソースの起動とステータスの確認
- ステップ3.3.5 - ACFSとアプリケーションVIP間のCRS依存関係の作成
- ステップ3.3.6 - SSHデーモンCRSリソースの作成
- ステップ3.3.7 - ACFSレプリケーションの有効化
- ステップ3.3.8 - ACFSレプリケーションCRSアクション・スクリプトの作成

ステップ3.3.1 - ASMファイル・システムの作成

最初のGGHUBノードでgrid OSユーザーとして、asmcmdを使用してACFSボリュームを作成します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ asmcmd volcreate -G DATA -s 120G ACFS_GG1
```



ノート:

決定したサイズ要件に応じて、ファイル・システムのサイズを変更します。

最初のGGHUBノードでgrid OSユーザーとして、asmcmdを使用して「ボリューム・デバイス」を確認します。

```
[grid@gghub_prim1 ~]$ asmcmd volinfo -G DATA ACFS_GG1
Diskgroup Name: DATA
  Volume Name: ACFS_GG1
  Volume Device: /dev/asm/acfs_gg1-256
 State: ENABLED
```

```
 Size (MB): 1228800
 Resize Unit (MB): 64
 Redundancy: UNPROT
 Stripe Columns: 8
 Stripe Width (K): 1024
 Usage:
 Mountpath:
```

最初のGGHUBノードのgrid OSユーザーとして、次のmkfsコマンドでパーティションをフォーマットします。

```
[grid@gghub_prim1 ~]$ /sbin/mkfs -t acfs /dev/asm/acfs_gg1-256
mkfs.acfs: version          = 19.0.0.0.0
mkfs.acfs: on-disk version      = 46.0
mkfs.acfs: volume          = /dev/asm/acfs_gg1-256
mkfs.acfs: volume size        = 128849018880 ( 120.00 GB )
mkfs.acfs: Format complete.
```

ステップ3.3.2 - Cluster Ready Services (CRS)リソースの作成

すべてのGGHUBノードのopc OSユーザーとして、ACFSマウント・ポイントを作成します。

```
[opc@gghub_prim1 ~]$ sudo mkdir -p /mnt/acfs_gg1
[opc@gghub_prim1 ~]$ sudo chown oracle:oinstall /mnt/acfs_gg1
```

rootユーザーとして、ファイル・システム・リソースを作成します。ACFSの分散ファイル・ロックの実装により、DBFSとは異なり、一度に複数のGGhubノードにACFSをマウントできます。

最初のGGHUBノードでroot OSユーザーとして、新しいACFSファイル・システム用のCRSリソースを作成します。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]#
cat > /u01/oracle/scripts/add_asm_filesystem.sh <<EOF
# Run as ROOT
$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/srvctl
 add filesystem ¥
-device /dev/asm/<acfs_volume> ¥
-volume ACFS_GG1 ¥
-diskgroup DATA ¥
-path /mnt/acfs_gg1 -user oracle ¥
-node gghub_prim1,gghub_prim2 ¥
-autostart NEVER ¥
-mountowner oracle ¥
-mountgroup oinstall ¥
-mountperm 755
EOF
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_asm_filesystem.sh
```

ステップ3.3.3 - 現在構成されているACFSファイル・システムの確認

最初のGGHUBノードのgrid OSユーザーとして、次のコマンドを使用してファイル・システムの詳細を確認します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ srvctl config filesystem -volume ACFS_GG1
 -diskgroup DATA
Volume device: /dev/asm/acfs_gg1-256
Diskgroup name: data
Volume name: acfs_gg1
Canonical volume device: /dev/asm/acfs_gg1-256
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg1
Mount point owner: oracle
Mount point group: oinstall
Mount permissions: owner:oracle:rwx,pgrp:oinstall:r-x,other::r-x
Mount users: grid
Type: ACFS
```

```
Mount options:
Description:
Nodes: gghub_prim1 gghub_prim2
Server pools: *
Application ID:
ACFS file system is enabled
ACFS file system is individually enabled on nodes:
ACFS file system is individually disabled on nodes:
```

ステップ3.3.4 - ACFSリソースの起動とステータスの確認

最初のgghubノードのgrid OSユーザーとして、次のコマンドを使用して、ファイル・システムを起動および確認します。

```
[grid@gghub_prim1 ~]$ srvctl start  filesystem -volume ACFS_GG1
 -diskgroup DATA -node `hostname`
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
```

作成したCRSリソースには、ora.diskgroup_name.volume_name.acfsという形式を使用した名前が付けられます。前述のファイル・システムの例を使用すると、CRSリソースの名前はora.data.acfs_gg.acfsになります。

最初のgghubノードのgrid OSユーザーとして、次のコマンドを使用してCRSのACFSリソースを確認します。

```
[grid@gghub_prim1 ~]$ crsctl stat res ora.data.acfs_gg1.acfs
NAME=ora.data.acfs_gg1.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

ステップ3.3.5 - ACFSとアプリケーションVIP間のCRS依存関係の作成

ファイル・システムが確実にVIPと同じOracle GGhubノードにマウントされるようにするには、次のコマンド例を使用して、VIP CRSリソースをACFSリソースへの依存関係として追加します。個別にレプリケートしたACFSファイル・システムごとに、それぞれ専用のVIPを使用します。

最初のGGHubノードでgrid OSユーザーとして、次のコマンドを使用して、VIPリソースの現在の起動および停止の依存関係を判別します。

```
[grid@gghub_prim1 ~]$ export appvip='gghub_prim_vip'
[grid@gghub_prim1 ~]$ crsctl stat res $appvip -f|grep _DEPENDENCIES
START_DEPENDENCIES=hard(ora.net1.network) pullup(ora.net1.network)
STOP_DEPENDENCIES=hard(intermediate:ora.net1.network)
```

最初のgghubノードのgrid OSユーザーとして、ACFSファイル・システム名を決定します。

```
[grid@gghub_prim1 ~]$ crsctl stat res -w "NAME co acfs_gg1"
 |grep NAME
NAME=ora.data.acfs_gg.acfs
```

最初のGGHUBノードのroot OSユーザーとして、VIPリソースの起動および停止の依存関係を変更します。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# export appvip='gghub_prim_vip'
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl modify res $appvip -attr
"START_DEPENDENCIES='hard(ora.net1.network,ora.data.acfs_gg1.acfs)
 pullup(ora.net1.network) pullup:always(ora.data.acfs_gg1.acfs)'"
[root@gghub_prim1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl modify res $appvip -attr
 "STOP_DEPENDENCIES='hard(intermediate:ora.net1.network,ora.data.acfs_gg1.acfs)'"
```

ノート:

acfs_gg1 は、正しい ACFS ボリューム名に置き換えます。

最初のGGHUBノードのgrid OSユーザーとして、VIPリソースを起動します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ export appvip='gghub_prim_vip'
[grid@gghub_prim1 ~]$ crsctl start resource $appvip
CRS-2672: Attempting to start 'gghub_prim_vip' on 'gghub_prim1'
CRS-2676: Start of 'gghub_prim_vip' on 'gghub_prim1' succeeded
```

ノート:

次のステップに進む前に、両方の GoldenGate Hub ノードに VIP をマウントできることを確認しておくことが重要です。

最初のGGHUBノードのgrid OSユーザーとして、VIPリソースを再配置します。

```
[grid@gghub_prim1 ~]$ crsctl relocate resource $appvip -f
CRS-2673: Attempting to stop 'gghub_prim_vip' on 'gghub_prim1'
CRS-2677: Stop of 'gghub_prim_vip' on 'gghub_prim1' succeeded
CRS-2673: Attempting to stop 'ora.data.acfs_gg1.acfs' on 'gghub_prim1'
CRS-2677: Stop of 'ora.data.acfs_gg1.acfs' on 'gghub_prim1' succeeded
CRS-2672: Attempting to start 'ora.data.acfs_gg1.acfs' on 'gghub_prim2'
CRS-2676: Start of 'ora.data.acfs_gg1.acfs' on 'gghub_prim2' succeeded
CRS-2672: Attempting to start 'gghub_prim_vip' on 'gghub_prim2'
CRS-2676: Start of 'gghub_prim_vip' on 'gghub_prim2' succeeded
[grid@gghub_prim1 ~]$ crsctl status resource $appvip
NAME=gghub_prim_vip
TYPE=app.appviptypex2.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim2
[grid@gghub_prim1 ~]$ crsctl relocate resource $appvip -f
CRS-2673: Attempting to stop 'gghub_prim_vip' on 'gghub_prim2'
CRS-2677: Stop of 'gghub_prim_vip' on 'gghub_prim2' succeeded
CRS-2673: Attempting to stop 'ora.data.acfs_gg1.acfs' on 'gghub_prim2'
CRS-2677: Stop of 'ora.data.acfs_gg1.acfs' on 'gghub_prim2' succeeded
CRS-2672: Attempting to start 'ora.data.acfs_gg1.acfs' on 'gghub_prim1'
CRS-2676: Start of 'ora.data.acfs_gg1.acfs' on 'gghub_prim1' succeeded
CRS-2672: Attempting to start 'gghub_prim_vip' on 'gghub_prim1'
CRS-2676: Start of 'gghub_prim_vip' on 'gghub_prim1' succeeded
```

最初のGGHUBノードのgrid OSユーザーとして、ACFSファイル・システムのステータスを確認します。

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
```

ステップ3.3.6 - SSHデーモンCRSリソースの作成

ACFSレプリケーションでは、前に作成した仮想IPアドレスを使用する、プライマリ・ファイル・システムとスタンバイ・ファイル・システムとの通信にセキュア・シェル(ssh)が使用されます。サーバーの再起動時に、VIP CRSリソースの前にsshデーモンが起動されて、VIPを使用したクラスタへのアクセスが防止されます。次の手順では、仮想IPリソースの起動後にsshデーモンを再起動する、ssh再起動CRSリソースを作成します。レプリケートされたファイル・システムごとに、個別のssh再起動CRSリソースが必要です。

すべてのGGHUBノードでgrid OSユーザーとして、sshデーモンを再起動するためのCRSアクション・スクリプトをコピーします。このスクリプトは、すべてのプライマリおよびスタンバイGGHUBノードの同じ場所に配置します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip /u01/oracle/stage/gghub_scripts_<YYYYMMDD>.zip
 -d /u01/oracle/scripts/
Archive: /u01/oracle/stage/gghub_scripts_<YYYYMMDD>.zip
 inflating: /u01/oracle/scripts/acfs_primary.scr
 inflating: /u01/oracle/scripts/acfs_standby.scr
 inflating: /u01/oracle/scripts/sshd_restart.scr
 inflating: /u01/oracle/scripts/add_acfs_primary.sh
 inflating: /u01/oracle/scripts/add_acfs_standby.sh
 inflating: /u01/oracle/scripts/add_nginx.sh
  inflating: /u01/oracle/scripts/add_sshd_restart.sh
 inflating: /u01/oracle/scripts/reverse_proxy_settings.sh
 inflating: /u01/oracle/scripts/secureServices.py
```

最初のGGHUBノードのroot OSユーザーとして、次のコマンドを使用してCRSリソースを作成します。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]#sh /u01/oracle/scripts/add_sshd_restart.sh
Application VIP Name: gghub_prim_vip
```

最初のGGHUBノードのgrid OSユーザーとして、CRSリソースを起動してテストします。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res sshd_restart
NAME=sshd_restart
TYPE=cluster_resource
TARGET=OFFLINE
STATE=OFFLINE
[grid@gghub_prim1 ~]$ crsctl start res sshd_restart
CRS-2672: Attempting to start 'sshd_restart' on 'gghub_prim1'
CRS-2676: Start of 'sshd_restart' on 'gghub_prim1' succeeded
[grid@gghub_prim1 ~]$ cat /tmp/sshd_restarted
STARTED
[grid@gghubtest1 ~]$ crsctl stop res sshd_restart
CRS-2673: Attempting to stop 'sshd_restart' on 'gghub_prim1'
CRS-2677: Stop of 'sshd_restart' on 'gghub_prim1' succeeded
[grid@gghub1 ~]$ cat /tmp/sshd_restarted
STOPPED
[grid@gghub1 ~]$ crsctl start res sshd_restart
CRS-2672: Attempting to start 'sshd_restart' on 'gghub_prim1'
CRS-2676: Start of 'sshd_restart' on 'gghub_prim1' succeeded
[grid@gghub1 ~]$ crsctl stat res sshd_restart
NAME=sshd_restart
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
```

ステップ3.3.7 - ACFSレプリケーションの有効化

ACFSスナップショットベース・レプリケーションでは、指定されたレプリケーション・ユーザー(通常はgrid ユーザー)を使用して、プライマリ・ホストとスタンバイ・ホストの間でスナップショットを転送するためにopensshが使用されます。

プライマリおよびスタンバイ・ハブ・システムのgrid OSユーザーとして、プライマリ・ノードとスタンバイ・ノードの間のssh接続を構成します。構成手順については、Oracle ACFSレプリケーションに使用するためのsshの構成を参照してください。

すべてのプライマリおよびスタンバイGGHUBノードのgrid OSユーザーとして、プライマリおよびスタンバイ・システムのすべてのVIPを、すべてのGGHUBノードのレプリケーション・ユーザーのknown_hostsファイルに追加します。

```
[grid@gghub_prim1 ~]$ cat ~/.ssh/known_hosts
## Example
# <hostnanme>,<host_ip>,<vip_name.domain>,<vip> ssh-rsa <ssh-key>
gghub_prim1,10.60.2.65,gghub_prim_vip.frankfurt.goldengate.com,10.60.2.120
 ssh-rsa <ssh-key>
gghub_prim2,10.60.2.65,gghub_prim_vip.frankfurt.goldengate.com,10.60.2.148
```

```
 ssh-rsa <ssh-key>
gghub_stby1,10.60.0.75,gghub_stby_vip.frankfurt.goldengate.com,10.60.0.185
 ssh-rsa <ssh-key>
gghub_stby2,10.60.0.75,gghub_stby_vip.frankfurt.goldengate.com,10.60.0.165
 ssh-rsa <ssh-key>
[grid@gghub_prim1 ~]$ cat ~/.ssh/known_hosts
ssh-rsa <ssh-key> grid@gghub_prim1
ssh-rsa <ssh-key> grid@gghub_prim2
ssh-rsa <ssh-key> grid@gghub_stby1
ssh-rsa <ssh-key> grid@gghub_stby2
```

すべてのプライマリおよびスタンバイのGGhubノードのgrid OSユーザーとして、sshを使用して、すべてのプライマリ・ノードからスタンバイ・ノードへの接続をテストし、レプリケーション・ユーザーとしてsshを使用して逆方向の接続をテストします。

```
# On the Primary GGhub
[grid@gghub_prim1 ~]$ ssh gghub_stby_vip.frankfurt.goldengate.com hostname
gghub_stby1
[grid@gghub_prim2 ~]$ ssh gghub_stby_vip.frankfurt.goldengate.com hostname
gghub_stby1
# On the Standby GGhub
[grid@gghub_stby1 ~]$ ssh gghub_prim_vip.frankfurt.goldengate.com hostname
gghub_prim1
[grid@gghub_stby2 ~]$ ssh gghub_prim_vip.frankfurt.goldengate.com hostname
gghub_prim1
```

ACFSがマウントされているプライマリおよびスタンバイGGHUBノードのgrid OSユーザーとして、acfsutilを使用してプライマリ・ノードとスタンバイ・ノードの間の接続をテストします。

```
# On the Primary GGhub
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
[grid@gghub_prim1 ~]$ acfsutil repl info -c -u grid
 gghub_prim_vip.frankfurt.goldengate.com
 gghub_stby_vip.frankfurt.goldengate.com /mnt/acfs_gg1
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip.frankfurt.goldengate.com as user grid.
# On the Standby GGhub
[grid@gghub_stby1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_stby1
[grid@gghub_stby1 ~]$ acfsutil repl info -c -u grid
 gghub_prim_vip.frankfurt.goldengate.com gghub_stby_vip.frankfurt.goldengate.com
 /mnt/acfs_gg
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip.frankfurt.goldengate.com as user grid.
```

ACFSがマウントされていないGGHUBノードからacfsutilコマンドを実行すると、予想されるようにエラーACFS-05518が表示されます。ACFSがマウントされているGGHUBを見つけるためにsrvctl status filesytemを使用し、コマンドを再実行します。

```
[grid@gghub_prim1 ~]$ acfsutil repl info -c -u
 grid gghub_stby_vip.frankfurt.goldengate.com
 gghub_stby_vip.frankfurt.goldengate.com /mnt/acfs_gg1
acfsutil repl info: ACFS-05518: /mnt/acfs_gg1 is not an ACFS mount point
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim2
[grid@gghub_prim1 ~]$ ssh gghub_prim2
[grid@gghub_prim2 ~]$ acfsutil repl info -c -u grid
```

```
 gghub_prim_vip.frankfurt.goldengate.com
 gghub_stby_vip.frankfurt.goldengate.com /mnt/acfs_gg1
A valid 'ssh' connection was detected for standby node
 gghub_prim_vip.frankfurt.goldengate.com as user grid.
A valid 'ssh' connection was detected for standby node
 gghub_stby_vip.frankfurt.goldengate.com as user grid.
```

ノート:

すべてのプライマリ・ノードからすべてのスタンバイ・ノードへの接続と、その逆方向の接続を検証してください。この接続テストにエラーがない場合にのみ続行します。

ACFSが現在マウントされているスタンバイGGhubノードでgrid OSユーザーとして、ACFSレプリケーションを初期化します。

```
[grid@gghub_stby1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_stby1
[grid@gghub_stby1 ~]$ /sbin/acfsutil repl init standby -u grid
 /mnt/acfs_gg1
```

ACFSが現在マウントされているプライマリGGhubノードでgrid OSユーザーとして、ACFSレプリケーションを初期化します。

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg is mounted on nodes gghub_prim1
[grid@gghub_prim1 ~]$ /sbin/acfsutil repl init primary -C
 -p grid@gghub_prim_vip.frankfurt.goldengate.com
 -s grid@gghub_stby_vip.frankfurt.goldengate.com
 -m /mnt/acfs_gg1 /mnt/acfs_gg1
```

プライマリおよびスタンバイのGGhubノードのgrid OSユーザーとして、初期化の進行状況が「送信が完了しました」のステータスに変化するまでモニターします。このステータスは、プライマリ・ファイル・システムの初期コピーが完了し、プライマリ・ファイル・システムがスタンバイ・ホストにレプリケートされたことを意味します。

```
# On the Primary GGhub
[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |
 grep Status
Status:               Send Completed
# On the Standby GGhub
[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |
 grep Status
Status:               Receive Completed
```

プライマリおよびスタンバイのGGhubノードのgrid OSユーザーとして、レプリケートされたACFSファイル・システムを確認およびモニターします。

```
# On the Primary GGhub
[grid@gghub_prim1 ~]$ acfsutil repl util verifystandby /mnt/acfs_gg1
verifystandby returned: 0
# On the Standby GGhub
[grid@gghubtest31 ~]$ acfsutil repl util verifyprimary /mnt/acfs_gg1
verifyprimary returned: 0
```

ノート:

問題が検出されていない場合は、どちらのコマンドも 0 (ゼロ)の値を返します。作業の続行前に、ACFS レプリケーションに関する一般的な問題のモニタリング、診断および解決について、「ACFS レプリケーションのトラブルシューティ

ング」を参照してください。

プライマリGGhubノードのgrid OSユーザーとして、次のコマンドを使用してACFSレプリケーションのステータスをモニターします。

```
[grid@gghub_prim1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1
Site:             Primary
Primary hostname:       gghub_prim_vip.frankfurt.goldengate.com
Primary path:         /mnt/acfs_gg1
Primary status:        Running
Background Resources:     Active
Standby connect string:    grid@gghub_stby_vip.frankfurt.goldengate.com
Standby path:         /mnt/acfs_gg1
Replication interval:     0 days, 0 hours, 0 minutes, 0 seconds
Sending primary as of:     Fri May 05 12:37:02 2023
Status:            Send Completed
Lag Time:           00:00:00
Retries made:         0
Last send started at:     Fri May 05 12:37:02 2023
Last send completed at:    Fri May 05 12:37:12 2023
Elapsed time for last send:  0 days, 0 hours, 0 minutes, 10 seconds
Next send starts at:      now
Replicated tags:
Data transfer compression:   Off
ssh strict host key checking: On
Debug log level:        3
Replication ID:        0x4d7d34a
```

ACFSが現在マウントされているスタンバイGGhubノードでgrid OSユーザーとして、次のコマンドを使用してACFSレプリケーションのステータスをモニターします。

```
[grid@gghub_stby1 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1
Site:             Standby
Primary hostname:       gghub_prim_vip.frankfurt.goldengate.com
Primary path:         /mnt/acfs_gg1
Standby connect string:    grid@gghub_stby_vip.frankfurt.goldengate.com
Standby path:          /mnt/acfs_gg1
Replication interval:     0 days, 0 hours, 0 minutes, 0 seconds
Last sync time with primary:  Fri May 05 12:37:02 2023
Receiving primary as of:    Fri May 05 12:37:02 2023
Status:            Receive Completed
Last receive started at:    Fri May 05 12:37:02 2023
Last receive completed at:   Fri May 05 12:37:07 2023
Elapsed time for last receive: 0 days, 0 hours, 0 minutes, 5 seconds
Data transfer compression:   Off
ssh strict host key checking: On
Debug log level:        3
Replication ID:        0x4d7d34a
```

ステップ3.3.8 - ACFSレプリケーションCRSアクション・スクリプトの作成

プライマリとスタンバイのACFSファイル・システムの状態を判別するには、CRSアクション・スクリプトを使用します。アクション・スクリプトでは、事前定義された間隔で、ファイル・システムの状態がCRSトレース・ファイルcrsd_scriptagent_grid.trcにレポートされます。このファイルは、GGhubノードの各プライマリ・ファイル・システム上および各スタンバイ・ファイル・システム上のGrid Infrastructureトレース・ファイル・ディレクトリ/u01/app/grid/diag/crs/<node_name>/crs/traceにあります。

プライマリ・ファイル・システムとスタンバイ・ファイル・システムの両方のクラスタで、2つのスクリプトが必要です。その1つはローカル・プライマリ・ファイル・システムをモニターし、リモート・スタンバイ・ファイル・システムが使用可能かどうかを確認するためのものであり、もう1つはローカル・スタンバイ・ファイル・システムをモニターし、リモート・プライマリ・ファイル・システムの可用性をチェックするためのものです。ACFSのモニタリングを実装するためのスクリプト例が提供されていますが、そのスクリプトは環境に応じて編集する必要があります。

レプリケートされたファイル・システムごとに、専用のアクション・スクリプトacfs_primaryとacfs_standbyが必要になります。

ステップ3.3.8.1 - アクション・スクリプトacfs_primary.scr

acfs_primary CRSリソースでは、現在のACFSマウントがプライマリ・ファイル・システムであるかどうかをチェックし、スタンバイ・ファイル・システムがアクセス可能なことと、レプリケートされたデータを受信していることを確認します。このリソースは、Oracle GoldenGateがプライマリOracle GoldenGateハブでプロセスを開始できるかどうかを自動的に判断するために使用します。プライマリからスタンバイ・ファイル・システムにアクセスできない場合、このスクリプト例ではスタンバイ・ファイル・システムの検証を複数回試行します。

acfs_primary CRSリソースは、プライマリとスタンバイの両方のホストで実行されますが、現在のファイル・システムがプライマリ・ファイル・システムであり、スタンバイ・ファイル・システムがアクセス可能な場合にのみ成功を返します。このスクリプトは、すべてのプライマリおよびスタンバイ・ファイル・システム・ノードの同じ場所に配置する必要があります。

次のパラメータには、推奨のデフォルト設定が使用されています。この設定の値は、変更前にテストする必要があります。

- ```
  MOUNT_POINT=/mnt/acfs_gg1
  # The replicated ACFS mount point
  ```

- ```
  PATH_NAME=$MOUNT_POINT/status/acfs_primary
  # Must be unique from other mount files
  ```

- ```
  ATTEMPTS=3
  # Number of attempts to check the remote standby file system
  ```

- ```
  INTERVAL=10
  # Number of seconds between each attempt
  ```

すべてのプライマリおよびスタンバイGGHUBノードのgrid OSユーザーとして、acfs_primary.scrスクリプトを環境に適合するように編集します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/acfs_primary.scr
```

ACFSが現在マウントされているプライマリGGhubノードでoracle OSユーザーとして、次のコマンドを実行してstatusディレクトリを作成します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ mkdir /mnt/acfs_gg1/status
[oracle@gghub_prim1 ~]$ chmod g+w /mnt/acfs_gg1/status
```

ACFSが現在マウントされているプライマリおよびスタンバイGGHUBノードでgrid OSユーザーとして、次のコマンドを実行して、プライマリおよびスタンバイ・ファイル・システムをモニタリングするためにacfs_primaryアクション・スクリプトを登録します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/add_acfs_primary.sh
crsctl add resource acfs_primary ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=/u01/oracle/scripts/acfs_primary.scr, ¥
   CHECK_INTERVAL=60, ¥
   START_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs)
 pullup:always(ora.data.acfs_gg1.acfs)', ¥
   STOP_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs)', ¥
   HOSTING_MEMBERS=<hostname>, ¥
   PLACEMENT=favored, ¥
   INSTANCE_FAILOVER=0, ¥
   SERVER_POOLS=*, ¥
   SCRIPT_TIMEOUT=80, ¥
   OFFLINE_CHECK_INTERVAL=0, ¥
   RESTART_ATTEMPTS=0"
```

```
[grid@gghub_prim1 ~]$ sh /u01/oracle/scripts/add_acfs_primary.sh
```

ノート:

HOSTING_MEMBERS パラメータを優先 GGhub ノードの名前に設定することで、プライマリ・ファイル・システムをマウントします。優先ノードが実行時に使用できない場合は、自動的に別のいずれかのノードで実行されます。

ACFSが現在マウントされているプライマリGGhubノードでgrid OSユーザーとして、acfs_primaryリソースを起動しそのステータスをチェックします。

```
[grid@gghub_prim1 ~]$ crsctl start resource acfs_primary
CRS-2672: Attempting to start 'acfs_primary' on 'gghub_prim1'
CRS-2676: Start of 'acfs_primary' on 'gghub_prim1' succeeded
[grid@gghub_prim1 ~]$ crsctl stat resource acfs_primary
NAME=acfs_primary
TYPE=cluster_resource
TARGET=ONLINE
STATE=ONLINE on gghub_prim1
[grid@gghub_prim1 ~]$ grep acfs_primary
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |grep check
2023-05-05 12:57:40.372 :CLSDYNAM:2725328640: [acfs_primary]{1:33562:34377}
 [check] Executing action script:
 /u01/oracle/scripts/acfs_primary.scr[check]
2023-05-05 12:57:42.376 :CLSDYNAM:2725328640: [acfs_primary]{1:33562:34377}
 [check] SUCCESS: STANDBY file system /mnt/acfs_gg1 is ONLINE
```

ACFSが現在マウントされているスタンバイGGhubノードでgrid OSユーザーとして、acfs_primaryリソースを起動しそのステータスをチェックします。acfs_primaryは、プライマリGGhubでのみオンラインにする必要があるため、このステップは失敗します。

```
[grid@gghub_stby1 ~]$ crsctl start res acfs_primary -n `hostname`
CRS-2672: Attempting to start 'acfs_primary' on 'gghub_stby1'
CRS-2674: Start of 'acfs_primary' on 'gghub_stby1' succeeded
CRS-2679: Attempting to clean 'acfs_primary' on 'gghub_stby1'
CRS-2681: Clean of 'acfs_primary' on 'gghub_stby1' succeeded
CRS-4000: Command Start failed, or completed with errors.
[grid@gghub_stby1 ~]$ crsctl stat res acfs_primary
NAME=acfs_primary
TYPE=cluster_resource
TARGET=ONLINE
STATE=OFFLINE
[grid@gghub_stby1 trace]$ grep acfs_primary
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |grep check
2023-05-05 13:09:53.343 :CLSDYNAM:3598239488: [acfs_primary]{1:8532:2106}
 [check] Executing action script: /u01/oracle/scripts/acfs_primary.scr[check]
2023-05-05 13:09:53.394 :CLSDYNAM:3598239488: [acfs_primary]{1:8532:2106}
 [check] Detected local standby file system
2023-05-05 13:09:53.493 :CLSDYNAM:1626130176: [acfs_primary]{1:8532:2106}
 [clean] Clean/Abort -- Stopping ACFS file system type checking...
```

ノート:

acfs_primary リソースのステータスは、ACFS ファイル・システムがプライマリ・ファイル・システムの場合にのみ ONLINE になります。このリソースは、現在プライマリ・クラスタにないノードで起動すると、そのリソースがスタンバイ・ファイル・システムであるために起動に失敗し、エラーが報告されます。このエラーは無視しても問題ありません。このリ

ソースは、ACFS スタンバイ・クラスタでは OFFLINE ステータスになります。

ステップ3.3.8.2 - アクション・スクリプトacfs_standby.scr

acfs_standbyリソースでは、ローカル・ファイル・システムがスタンバイ・ファイル・システムであることをチェックして、リモートのプライマリ・ファイル・システムのステータスを確認します。プライマリ・ファイル・システムの検証が複数回失敗すると(回数はアクション・スクリプトの変数によって制御)、CRSトレース・ファイルcrsd_scriptagent_grid.trcに警告が出力されます。このファイルは、Grid Infrastructureトレース・ファイル・ディレクトリ/u01/app/grid/diag/CRS/<node_name>/CRS/traceに配置されています。

このリソースは、プライマリとスタンバイの両方のホストで実行されますが、現在のファイル・システムがスタンバイ・ファイル・システムであり、プライマリ・ファイル・システムがアクセス可能な場合にのみ成功を返します。

次のパラメータには、推奨のデフォルト設定が使用されています。この設定の値は、変更前にテストする必要があります。

- MOUNT_POINT=/mnt/acfs_gg
  # This is the replicated ACFS mount point

- ATTEMPTS=3
  # Number of tries to check the remote primary file system

- INTERVAL=10
  # Number of seconds between each attempt

すべてのプライマリおよびスタンバイGGHUBノードのgrid OSユーザーとして、acfs_standby.scrスクリプトを環境に適合するように編集します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/acfs_standby.scr
```

ACFSが現在マウントされているプライマリGGHUBノードでgrid OSユーザーとして、次のコマンドを実行して、プライマリおよびスタンバイ・ファイル・システムをモニタリングするためにacfs_standbyアクション・スクリプトを登録します。

```
[grid@gghub_prim1 ~]$ crsctl stat res -w "TYPE co appvip"
 |grep NAME
NAME=gghub_prim_vip
[grid@gghub_prim1 ~]$ vi /u01/oracle/scripts/add_acfs_standby.sh
crsctl add resource acfs_standby ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=/u01/oracle/scripts/acfs_standby.scr, ¥
   CHECK_INTERVAL=150, ¥
   CHECK_TIMEOUT=140, ¥
   START_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_prim_vip)
 pullup:always(ora.data.acfs_gg1.acfs,gghub_prim_vip)', ¥
  STOP_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_prim_vip)' ¥
   OFFLINE_CHECK_INTERVAL=300, ¥
   RESTART_ATTEMPTS=0, ¥
   INSTANCE_FAILOVER=0"
[grid@gghub_prim1 ~]$ sh /u01/oracle/scripts/add_acfs_standby.sh
```

ACFSが現在マウントされているプライマリGGHUBノードでgrid OSユーザーとして、acfs_standbyリソースを起動しステータスをチェックします。

```
[grid@gghub_prim1 ~]$ crsctl start res acfs_standby
CRS-2672: Attempting to start 'acfs_standby' on 'gghub_prim1'
CRS-2676: Start of 'acfs_standby' on 'gghub_prim1' succeeded
[grid@gghub_prim1 ~]$ grep acfs_standby
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |egrep 'check|INFO'
2023-05-05 13:22:09.612 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
 [start] acfs_standby.scr starting to check ACFS remote primary at
```

```
 /mnt/acfs_gg1
2023-05-05 13:22:09.612 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
 [check] Executing action script: /u01/oracle/scripts/acfs_standby.scr[check]
2023-05-05 13:22:09.663 :CLSDYNAM:2725328640: [acfs_standby]{1:33562:34709}
 [check] Local PRIMARY file system /mnt/acfs_gg1
```

ACFSが現在マウントされているスタンバイGGHUBノードでgrid OSユーザーとして、次のコマンドを実行して、プライマリおよびスタンバイ・ファイル・システムをモニタリングするためにacfs_standbyアクション・スクリプトを登録します。

```
[grid@gghub_stby1 ~]$ crsctl stat res -w "TYPE co appvip"
 |grep NAME
NAME=gghub_stby_vip
[grid@gghub_stby1 ~]$ vi /u01/oracle/scripts/add_acfs_standby.sh
crsctl add resource acfs_standby ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=/u01/oracle/scripts/acfs_standby.scr, ¥
   CHECK_INTERVAL=150, ¥
   CHECK_TIMEOUT=140, ¥
   START_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_stby_vip)
 pullup:always(ora.data.acfs_gg1.acfs,gghub_stby_vip)', ¥
   STOP_DEPENDENCIES='hard(ora.data.acfs_gg1.acfs,gghub_stby_vip)' ¥
   OFFLINE_CHECK_INTERVAL=300, ¥
   RESTART_ATTEMPTS=0, ¥
   INSTANCE_FAILOVER=0"
[grid@gghub_stby1 ~]$ sh /u01/oracle/scripts/add_acfs_standby.sh
```

ACFSが現在マウントされているプライマリGGHUBノードでgrid OSユーザーとして、acfs_standbyリソースを起動しステータスをチェックします。

```
[grid@gghub_stby1 ~]$ crsctl start res acfs_standby
CRS-2672: Attempting to start 'acfs_standby' on 'gghub_stby1'
CRS-2676: Start of 'acfs_standby' on 'gghub_stby1' succeeded
[grid@gghub_stby1 ~]$ grep acfs_standby
 /u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc
 |egrep 'check|INFO'
2023-05-05 13:25:20.699 :CLSDYNAM:1427187456: [acfs_standby]{1:8532:2281}
 [check] SUCCESS: PRIMARY file system /mnt/acfs_gg1 is ONLINE
2023-05-05 13:25:20.699 :    AGFW:1425086208: [     INFO] {1:8532:2281}
 acfs_standby 1 1 state changed from: STARTING to: ONLINE
2023-05-05 13:25:20.699 :    AGFW:1425086208: [     INFO] {1:8532:2281}
 Started implicit monitor for [acfs_standby 1 1]
 interval=150000 delay=150000
2023-05-05 13:25:20.699 :    AGFW:1425086208: [     INFO] {1:8532:2281}
 Agent sending last reply for: RESOURCE_START[acfs_standby 1 1]
 ID 4098:8346
```

ステップ3.3.9 - ACFS GGhubノードの再配置のテスト

Oracle GoldenGateの構成前に、計画的および計画外のACFS GGhubノードの再配置とサーバー・ロール・トランジションをテストすることは、非常に重要です。

プライマリおよびスタンバイGGHUBノードのgrid OSユーザーとして、次のコマンドを実行して、GGHUBノード間でACFSを再配置します。

```
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim1
[grid@gghub_prim1 ~]$ srvctl relocate filesystem -diskgroup DATA
 -volume acfs_gg1 -force
[grid@gghub_prim1 ~]$ srvctl status filesystem -volume ACFS_GG1
 -diskgroup DATA
ACFS file system /mnt/acfs_gg1 is mounted on nodes gghub_prim2
```

プライマリおよびスタンバイGGHUBノードのgrid OSユーザーとして、次のサンプル・コマンドを使用して、ファイル・システムがVIP、sshd_restartおよび2つのACFSリソース(acfs_primaryおよびacfs_standby)とともに別のノードにマウントされていることを確認します。

```
[grid@gghub_prim1 ~]$ crsctl stat res sshd_restart acfs_primary
 acfs_standby ora.data.acfs_gg1.acfs sshd_restart -t
--------------------------------------------------------------------------------
Name      Target State    Server          State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
acfs_primary
    1    ONLINE ONLINE    gghub_prim2        STABLE
acfs_standby
    1    ONLINE ONLINE                  STABLE
gghubfad2
   1    ONLINE ONLINE    gghub_prim2        STABLE
ora.data.acfs_gg1.acfs
   1    ONLINE ONLINE    gghub_prim2        mounted on /mnt/acfs
                              _gg1,STABLE
sshd_restart
   1    ONLINE ONLINE    gghub_prim2        STABLE
--------------------------------------------------------------------------------
[grid@gghub_stby1 ~]$ crsctl stat res sshd_restart acfs_primary acfs_standby
 ora.data.acfs_gg1.acfs sshd_restart -t
--------------------------------------------------------------------------------
Name      Target State    Server          State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
acfs_primary
   1    ONLINE OFFLINE                STABLE
acfs_standby
    1    ONLINE ONLINE    gghub_stby2       STABLE
ora.data.acfs_gg1.acfs
    1    ONLINE ONLINE    gghub_stby2        mounted on /mnt/acfs
                            _gg1,STABLE
sshd_restart
    1    ONLINE ONLINE    gghub_stby2       STABLE
--------------------------------------------------------------------------------
```

ステップ3.3.10 - プライマリとスタンバイGGhub間のACFSスイッチオーバーのテスト

スタンバイGGHUBノードのgrid OSユーザーとして、次のコマンドを実行して、プライマリとスタンバイのGGHUB間でのACFSスイッチオーバー(ロール・リバーサル)を発行します。

```
[grid@gghub_stby2 ~]$ crsctl stat res ora.data.acfs_gg1.acfs
NAME=ora.data.acfs_gg.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_stby2
[grid@gghub_stby2 ~]$ acfsutil repl failover /mnt/acfs_gg1
[grid@gghub_stby2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1
Site:                 Primary
Primary hostname:          gghub_stby_vip.frankfurt.goldengate.com
Primary path:             /mnt/acfs_gg1
Primary status:            Running
Background Resources:         Active
Standby connect string:        gghub_prim_vip.frankfurt.goldengate.com
Standby path:             /mnt/acfs_gg1
Replication interval:         0 days, 0 hours, 0 minutes, 0 seconds
Sending primary as of:         Fri May 05 13:51:37 2023
Status:                Send Completed
Lag Time:               00:00:00
Retries made:             0
```

```
Last send started at:         Fri May 05 13:51:37 2023
Last send completed at:        Fri May 05 13:51:48 2023
Elapsed time for last send:     0 days, 0 hours, 0 minutes, 11 seconds
Next send starts at:          now
Replicated tags:
Data transfer compression:       Off
ssh strict host key checking:     On
Debug log level:            3
Replication ID:            0x4d7d34a
```

新しいスタンバイGGHUBノード(元のプライマリ)のgrid OSユーザーとして、次のコマンドを実行して、プライマリとスタンバイのGGHUB間でのACFSスイッチオーバー(ロール・リバーサル)を発行します。このステップはオプションですが、サイトを元のロールに戻すためにお薦めします。

```
[grid@gghub_prim2 ~]$ crsctl stat res ora.data.acfs_gg1.acfs
NAME=ora.data.acfs_gg1.acfs
TYPE=ora.acfs_cluster.type
TARGET=ONLINE
STATE=ONLINE on gghub_prim2
[grid@gghub_prim2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |grep Site
Site:                 Standby
[grid@gghub_prim2 ~]$ acfsutil repl failover /mnt/acfs_gg1
[grid@gghub_prim2 ~]$ /sbin/acfsutil repl info -c -v /mnt/acfs_gg1 |grep Site
Site:                 Primary
```

ステップ3.4 - Oracle GoldenGateデプロイメントの作成

GGHUBにOracle GoldenGateソフトウェアをインストールした後のステップでは、Oracle GoldenGate Configuration Assistantを使用して、GoldenGateデプロイメントを作成するためのレスポンス・ファイルを作成します。

Oracle GoldenGate 21cに導入された統合ビルド機能により、単一のデプロイメントで、異なるOracle DatabaseバージョンにアタッチするextractとReplicatのプロセスを管理できるようになりました。各デプロイメントは、管理サーバーと(オプションの)パフォーマンス・メトリック・サーバーによって作成されます。GoldenGate証跡ファイルを別のハブまたはGoldenGate環境に転送する必要がない場合は、分散サーバーまたは受信サーバーの作成は不要です。

現時点では、Oracle GoldenGateとXAGには2つの制限があります。

1. XAGに登録されているサービス・マネージャごとに、単一のデプロイメントのみを管理できます。複数のデプロイメントが必要な場合は、それぞれのデプロイメントに専用のサービス・マネージャを使用する必要があります。Oracle GoldenGateリリース21cでは、様々なバージョンのOracle Databaseに接続するExtractプロセスおよびReplicatプロセスをサポートする単一のデプロイメントが使用されるため、この要件が簡素化されています。
2. XAGに登録されている各サービス・マネージャは、別々のOGG_HOMEソフトウェア・インストール・ディレクトリに属している必要があります。Oracle GoldenGateを複数回インストールするのではなく、Oracle GoldenGateを1回インストールし、その後はサービス・マネージャのOGG_HOMEごとにシンボリック・リンクを作成することをお薦めします。シンボリック・リンクおよびOGG_HOME環境変数は、すべてのOracle RACノードで、Oracle GoldenGate Configuration Assistantを実行する前に構成する必要があります。

レスポンス・ファイルの作成

サイレント構成の場合は、次のサンプル・ファイルをコピーして、oracleユーザーがアクセスできる場所に貼り付けます。次の値を適切に編集します。

- CONFIGURATION_OPTION
- DEPLOYMENT_NAME
- ADMINISTRATOR_USER
- SERVICEMANAGER_DEPLOYMENT_HOME
- OGG_SOFTWARE_HOME

- OGG_DEPLOYMENT_HOME
- ENV_TNS_ADMIN
- OGG_SCHEMA

レスポンス・ファイル例(oggca.rsp):

ACFSが現在マウントされているプライマリGGHUBノードでoracle OSユーザーとして、Oracle GoldenGateデプロイメントを作成するためのレスポンス・ファイルoggca.rspを作成および編集します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ vi /u01/oracle/stage/oggca.rsp
oracle.install.responseFileVersion=/oracle/install/rspfmt_oggca_response_schema_v21_1
_0
CONFIGURATION_OPTION=ADD
DEPLOYMENT_NAME=gghub1
ADMINISTRATOR_USER=oggadmin
ADMINISTRATOR_PASSWORD=<password_for_oggadmin>
SERVICEMANAGER_DEPLOYMENT_HOME=/mnt/acfs_gg1/deployments/ggsm01
HOST_SERVICEMANAGER=localhost
PORT_SERVICEMANAGER=9100
SECURITY_ENABLED=false
STRONG_PWD_POLICY_ENABLED=true
CREATE_NEW_SERVICEMANAGER=true
REGISTER_SERVICEMANAGER_AS_A_SERVICE=false
INTEGRATE_SERVICEMANAGER_WITH_XAG=true
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=false
OGG_SOFTWARE_HOME=/u01/app/oracle/goldengate/gg21c
OGG_DEPLOYMENT_HOME=/mnt/acfs_gg1/deployments/gg01
ENV_LD_LIBRARY_PATH=${OGG_HOME}/lib/instantclient:${OGG_HOME}/lib
ENV_TNS_ADMIN=/u01/app/oracle/goldengate/network/admin
FIPS_ENABLED=false
SHARDING_ENABLED=false
ADMINISTRATION_SERVER_ENABLED=true
PORT_ADMINSRVR=9101
DISTRIBUTION_SERVER_ENABLED=true
PORT_DISTSRVR=9102
NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=false
RECEIVER_SERVER_ENABLED=true
PORT_RCVRSRVR=9103
METRICS_SERVER_ENABLED=true
METRICS_SERVER_IS_CRITICAL=false
PORT_PMSRVR=9104
UDP_PORT_PMSRVR=9105
PMSRVR_DATASTORE_TYPE=BDB
PMSRVR_DATASTORE_HOME=/u01/app/oracle/goldengate/datastores/gghub1
OGG_SCHEMA=ggadmin
```

Oracle GoldenGateデプロイメントの作成

ACFSが現在マウントされているプライマリGGHUBノードでoracle OSユーザーとして、oggca.shを実行してGoldenGateデプロイメントを作成します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ $OGG_HOME/bin/oggca.sh -silent
 -responseFile /u01/oracle/stage/oggca.rsp
Successfully Setup Software.
```

Oracle GoldenGateデータストアとTNS_ADMINディレクトリの作成

プライマリ・システムとスタンバイ・システムのすべてのGGHUBノードでoracle OSユーザーとして、次のコマンドを実行して、Oracle GoldenGateデータストアとTNS_ADMINディレクトリを作成します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
```

```
[oracle@gghub_prim1 ~]$ mkdir -p /u01/app/oracle/goldengate/network/admin
[oracle@gghub_prim1 ~]$ mkdir -p /u01/app/oracle/goldengate/datastores/gghub1
```

ステップ3.5 - Oracle Grid Infrastructure Agent (XAG)の構成

次のステップごとの手順では、Oracle Grid Infrastructure Standalone Agent (XAG)を使用してGoldenGateを管理するようにOracle Clusterwareを構成する方法を示します。XAGを使用することで、Oracle GGhubノード間での再配置時に、ACFSファイル・システムのマウントとGoldenGateデプロイメントの停止および起動を自動化します。

ステップ3.5.1 - Oracle Grid Infrastructure Standalone Agentのインストール

XAGソフトウェアは、スタンドアロン・エージェントとしてGrid InfrastructureのORACLE_HOMEの外部にインストールすることをお薦めします。これにより、入手可能な最新のXAGリリースを使用できるようになり、Grid Infrastructureに影響を与えることなくソフトウェアを更新できます。

Oracle Grid Infrastructureホーム・ディレクトリの外部に、XAGスタンドアロン・エージェントをインストールします。XAGは、GoldenGateがインストールされているシステム内のすべてのGGhubノードで同じディレクトリにインストールする必要があります。

プライマリ・システムとスタンバイ・システムの最初のGGHubノードでgrid OSユーザーとして、ソフトウェアを解凍し、xagsetup.shを実行します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ unzip /u01/oracle/stage/p31215432_190000_Generic.zip
 -d /u01/oracle/stage
[grid@gghub_prim1 ~]$ /u01/oracle/stage/xag/xagsetup.sh --install
 --directory /u01/app/grid/xag --all_nodes
Installing Oracle Grid Infrastructure Agents on: gghub_prim1
Installing Oracle Grid Infrastructure Agents on: gghub_prim2
Updating XAG resources.
Successfully updated XAG resources.
```

プライマリ・システムとスタンバイ・システムのすべてのGGHUBノードでgrid OSユーザーとして、新しくインストールしたXAGソフトウェアの場所をPATH変数に追加して、gridユーザーがマシンにログオンしたときにagctlの場所が認識されるようにします。

```
[grid@gghub_prim1 ~]$ vi ~/.bashrc
PATH=/u01/app/grid/xag/bin:$PATH:/u01/app/19.0.0.0/grid/bin; export PATH
```

ノート:

正しい agctl バイナリが見つかるように、Grid Infrastructure の bin ディレクトリの前に XAG の bin ディレクトリが指定されていることを必ず確認してください。これは、Bash シェル使用時の.bashrc ファイルなど、ログオン時に有効になる grid ユーザー環境で設定する必要があります。 ステップ 3.5.2 - プライマリおよびスタンバイ GGhub への Oracle Grid Infrastructure Agent の登録

次の手順では、Oracle Grid Infrastructure Standalone Agent (XAG)を使用して Oracle GoldenGate を管理するように Oracle Clusterware を構成する方法を示します。XAG を使用することで、Oracle GGhub ノード間での再配置時に、共有ファイル・システムのマウントと Oracle GoldenGate デプロイメントの停止および起動を自動化します。

Oracle GoldenGateは、データベースの起動時およびファイル・システムのマウント時に自動的にデプロイメントが起動および停止されるように、XAGに登録しておく必要があります。

Oracle GoldenGate Microservices ArchitectureをXAGに登録するには、次のコマンド形式を使用します。

```
agctl add goldengate <instance_name>
--gg_home <GoldenGate_Home>
--service_manager
--config_home <GoldenGate_SvcMgr_Config>
--var_home <GoldenGate_SvcMgr_Var Dir>
--port <port number>
--oracle_home <$OGG_HOME/lib/instantclient>
--adminuser <OGG admin user>
--user <GG instance user>
--group <GG instance group>
--file systems <CRS_resource_name>
--db_services <service_name>
--use_local_services
--attribute START_TIMEOUT=60
```

説明:

- `--gg_home`では、GoldenGateソフトウェアの場所を指定します。
- `--service_manager`では、これがGoldenGate Microservicesインスタンスであることを示します。
- `--config_home`では、GoldenGateデプロイメント構成のホーム・ディレクトリを指定します。
- `--var_home`では、GoldenGateデプロイメント変数のホーム・ディレクトリを指定します。
- `--oracle_home`では、Oracle Instant Clientホームを指定します
- `--port`では、デプロイメント・サービス・マネージャのポート番号を指定します。
- `--adminuser`では、GoldenGate Microservices管理者のアカウント名を指定します。
- `--user`では、GoldenGateデプロイメント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します。
- `--group`では、GoldenGateデプロイメントを所有するオペレーティング・システム・グループの名前を指定します。
- `--filesystems`では、そのデプロイメントの起動前にオンラインになっている必要があるCRSファイル・システム・リソースを指定します。これは、前のステップで作成したacfs_primaryリソースになります。
- `--filesystem_verify`では、`config_home`パラメータと`var_home`パラメータで指定したディレクトリの存在をXAGで確認するかどうかを指定します。アクティブなACFSプライマリ・ファイル・システムについてはこれを`yes`に設定する必要があります。スタンバイ・クラスタにGoldenGateインスタンスを追加するときは、noを指定します。
- `--filesystems_always`では、`--filesystems`パラメータで指定したファイル・システムCRSリソースと同じGGhubノードでXAGによってGoldenGateサービス・マネージャが起動されるようにすることを指定します。
- `--attributes`では、リソースのターゲット・ステータスがオンラインであることを示します。これは、`acfs_primary`リソースの起動時に自動的にGoldenGateデプロイメントを起動するために必要です。

GoldenGateデプロイメントは、読取り/書込みモードまたは読取り専用モードでACFSがマウントされているプライマリおよびスタンバイGGHUBに登録する必要があります。

プライマリ・システムとスタンバイ・システムの最初のGGHUBノードで`grid` OSユーザーとして、次のコマンドを実行して、ファイル・システムがマウントされているクラスタのノードを調べます。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res acfs_standby |grep STATE
STATE=ONLINE on gghub_prim1
```

ステップ3.5.2.1 - プライマリOracle GoldenGate Microservices ArchitectureのXAGへの登録

プライマリGGHUBの最初のノードで`root` OSユーザーとして、次のコマンド形式を使用して、XAGにOracle GoldenGate Microservices Architectureを登録します。

```
[opc@gghub_prim1 ~]$ sudo su - root
[root@gghub_prim1 ~]# vi /u01/oracle/scripts/add_xag_goldengate.sh
# Run as ROOT:
```

```
/u01/app/grid/xag/bin/agctl add goldengate gghub1 ¥
--gg_home /u01/app/oracle/goldengate/gg21c ¥
--service_manager ¥
--config_home /mnt/acfs_gg1/deployments/ggsm01/etc/conf ¥
--var_home /mnt/acfs_gg1/deployments/ggsm01/var ¥
--oracle_home /u01/app/oracle/goldengate/gg21c/lib/instantclient ¥
--port 9100 ¥
--adminuser oggadmin ¥
--user oracle ¥
--group oinstall ¥
--filesystems acfs_primary ¥
--filesystems_always yes ¥
--filesystem_verify yes ¥
--attribute TARGET_DEFAULT=online
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_xag_goldengate.sh
Enter password for 'oggadmin' : ##########
```

プライマリGGHUBの最初のノードでgrid OSユーザーとして、Oracle GoldenGate Microservices ArchitectureがXAGに登録されていることを確認します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl status goldengate
Goldengate instance 'gghub1' is not running
```

ステップ3.5.2.2 - スタンバイOracle GoldenGate Microservices ArchitectureのXAGへの登録

スタンバイGGHUBの最初のノードでroot OSユーザーとして、次のコマンド形式を使用して、XAGにOracle GoldenGate Microservices Architectureを登録します。

```
[opc@gghub_stby1 ~]$ sudo su - root
[root@gghub_stby1 ~]# vi /u01/oracle/scripts/add_xag_goldengate.sh
# Run as ROOT:
/u01/app/grid/xag/bin/agctl add goldengate gghub1 ¥
--gg_home /u01/app/oracle/goldengate/gg21c ¥
--service_manager ¥
--config_home /mnt/acfs_gg1/deployments/ggsm01/etc/conf ¥
--var_home /mnt/acfs_gg1/deployments/ggsm01/var ¥
--oracle_home /u01/app/oracle/goldengate/gg21c/lib/instantclient ¥
--port 9100 --adminuser oggadmin --user oracle --group oinstall ¥
--filesystems acfs_primary ¥
--filesystems_always yes ¥
--filesystem_verify no ¥
--attribute TARGET_DEFAULT=online
[root@gghub_stby1 ~]# sh /u01/oracle/scripts/add_xag_goldengate.sh
Enter password for 'oggadmin' : ##########
```

ノート:

スタンバイ・クラスタに GoldenGate インスタンスを追加するときは、--filesystem_verify no を指定します。

スタンバイGGHUBの最初のノードでgrid OSユーザーとして、Oracle GoldenGate Microservices ArchitectureがXAGに登録されていることを確認します。

```
[opc@gghub_stby1 ~]$ sudo su - grid
[grid@gghub_stby1 ~]$ agctl status goldengate
Goldengate instance 'gghub1' is not running
```

ステップ3.5.3 - Oracle GoldenGateデプロイメントの起動

次に、XAGでGoldenGateデプロイメントを管理するために使用するいくつかのagctlコマンドの例を示します。

プライマリGGHUBの最初のノードでgrid OSユーザーとして、次のコマンドを実行して、Oracle GoldenGateデプロイメントを起動し確認します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl start goldengate gghub1
[grid@gghub_prim1 ~]$ agctl status goldengate
Goldengate instance 'gghub1' is running on gghub_prim1
```

最初のGGHUBノードでgrid OSユーザーとして、次のコマンドを実行して、Oracle GoldenGateリソースの構成パラメータを検証します。

```
[grid@gghub_prim1 ~]$ agctl config goldengate gghub1
Instance name: gghub1
Application GoldenGate location is: /u01/app/oracle/goldengate/gg21c
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
 /mnt/acfs_gg1/deployments/ggsm01/etc/conf
Goldengate Service Manager var directory:
 /mnt/acfs_gg1/deployments/ggsm01/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: no
ORACLE_HOME location is:
 /u01/app/oracle/goldengate/gg21c/lib/instantclient
File System resources needed: acfs_primary
CRS additional attributes set: TARGET_DEFAULT=online
```

詳細は、Oracle Grid Infrastructure Bundled Agentを参照してください。

ステップ3.6 - NGINXリバース・プロキシの構成

GoldenGateリバース・プロキシ機能を使用すると、GoldenGateデプロイメントに関連付けられたすべてのGoldenGateマイクロサービスに対応する単一のアクセス先を使用できます。リバース・プロキシを使用しないと、GoldenGateデプロイメント・マイクロサービスは、ホスト名またはIPアドレスと個別のポート番号で構成されるサービスごとに1つずつのURLを使用してアクセスされます。たとえば、サービス・マネージャにアクセスするためにhttp://gghub.example.com:9100を使用し、管理サーバーにはhttp://gghub.example.com:9101、2番目のサービス・マネージャにはhttp://gghub.example.com:9110を使用してアクセスするようになります。

Grid Infrastructure Agent (XAG)を使用してOracle Exadata Database Serviceで高可用性(HA)構成のOracle GoldenGateを実行する場合、GoldenGateサービス・マネージャでは複数のデプロイメントを管理できないとう制限があります。この制限のため、サービス・マネージャ/デプロイメントのペアごとに、個別の仮想IPアドレス(VIP)を作成するようにお薦めします。このようにすると、VIPを使用してマイクロサービスに直接アクセスできます。

リバース・プロキシにより、ポート番号はデプロイメント名とホスト名のVIPに置き換えられるため、マイクロサービスに接続するためのポート番号は不要になります。たとえば、コンソールにWebブラウザ経由で接続するには、次のURLを使用します。

GoldenGateサービス	URL
サービス・マネージャ	https://localhost:localPort
管理サーバー	https://localhost:localPort/instance_name/adminsrvr
分散サーバー	https://localhost:localPort/instance_name/distsrvr
パフォーマンス・メトリック・サーバー	https://localhost:localPort/instance_name/pmsr

GoldenGateサービス	URL
	vr
レシーバ・サーバー	https://localhost:localPort/instance_name/recvsrvr

ノート:

OCI で Oracle GoldenGate に接続する場合は、要塞と SSH ポート転送セッションを作成する必要があります(ステップ 6.1 を参照)。この後は、https://locahost:localPort を使用すると Oracle GoldenGate サービスに接続できます。

リバース・プロキシは、マイクロサービスに簡単にアクセスし、セキュリティと管理性を強化するために必須です。

複数のサービス・マネージャを実行する場合は、次の手順によって、サービス・マネージャごとに個別のVIPを使用する構成を用意します。NGINXは、VIPを使用してHTTPS接続リクエストのルーティング先になるサービス・マネージャを決定します。

SSL証明書は、クライアントがNGINX経由で接続するサーバーを認証するために必要です。続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

ノート:

CA 署名付き証明書の共通名は、NGINX で使用されるターゲット・ホスト名/VIP と一致している必要があります。

手順に従って、SSL接続のNGINXリバース・プロキシをインストールおよび構成し、すべての外部通信をセキュアにします。

ステップ3.6.1 - セキュアなデプロイメントの要件(証明書)

セキュアなデプロイメントでは、RESTful APIコールを発行し、証跡データをSSL/TLSを介してDistribution ServerとReceiver Server間で転送します。認証局(CA)から取得した既存のビジネス証明書を使用するか、独自の証明書を作成できます。続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

ステップ3.6.2 - NGINXリバース・プロキシ・サーバーのインストール

すべてのGGHUBノードで`root` OSユーザーとして、次の内容でファイル`/etc/yum.repos.d/nginx.repo`を作成することでyumリポジトリを設定します。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# cat > /etc/yum.repos.d/nginx.repo <<EOF
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/rhel/7/¥$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
EOF
```

すべてのGGHUBノードで`root` OSユーザーとして、次のコマンドを実行してNGINXをインストール、有効化および起動します。

```
[root@gghub_prim1 ~]# yum install -y python-requests python-urllib3 nginx
```

```
[root@gghub_prim1 ~]# systemctl enable nginx
```

すべてのGGHUBノードでroot OSユーザーとして、ソフトウェアのインストール後にNGINXリポジトリを無効にします。

```
[root@gghub_prim1 ~]# yum-config-manager --disable nginx-stable
```

ステップ3.6.3 - NGINX構成ファイルの作成

Oracle GoldenGate Microservices Architectureは、リバース・プロキスを使用するように構成できます。Oracle GoldenGate MAには、ReverseProxySettingsというスクリプトが含まれています。このスクリプトにより、NGINXリバース・プロキシ・サーバー専用の構成ファイルが生成されます。

このスクリプトには次のパラメータが必要です。

- --userパラメータは、初期デプロイメントの作成で指定したGoldenGate管理者アカウントの写しにする必要があります。
- GoldenGate管理者パスワードの入力が求められます。
- --portパラメータで指定したリバース・プロキシ・ポート番号は、デフォルトのHTTPSポート番号(443)にする必要があります。ただし、同じ--hostを使用して複数のGoldenGateサービス・マネージャを実行している場合を除きます。その場合は、前のサービス・マネージャ・リバース・プロキシ構成と競合しないHTTPSポート番号を指定します。たとえば、同じホスト名/VIPを使用して2つのサービス・マネージャを実行する場合、最初のリバース・プロキシ構成は'--port 443 --host hostvip01'で作成し、2番目は'--port 444 --host hostvip01'で作成します。個別のホスト名/VIPを使用する場合は、'--port 443 --host hostvip01'と'--port 443 --host hostvip02'を使用して、2つのサービス・マネージャ・リバース・プロキシ構成を作成します。
- 最後に、HTTPポート番号(9100)は、デプロイメントの作成時に指定したサービス・マネージャのポート番号と一致する必要があります。

このステップは、追加のGoldenGateサービス・マネージャごとに繰り返します。

最初のGGHUBノードでoracle OSユーザーとして、次のコマンドを使用してOracle GoldenGate NGINX構成ファイルを作成します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ export PATH=$PATH:$OGG_HOME/bin
[oracle@gghub_prim1 ~]$ cd /u01/oracle/scripts
[oracle@gghub_prim1 ~]$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings
 --user oggadmin --port 443 --output ogg_<gghub1>.conf http://localhost:9100
 --host <VIP hostname>
Password: <oggadmin_password>
```

ステップ3.6.4 - NGINX構成ファイルの変更

複数のGoldenGateサービス・マネージャが同じHTTPS 443ポートのIP/VIPを使用するように構成されている場合は、前のステップで生成したNGINXリバース・プロキシ構成ファイルに小さな変更が必要です。同じポート番号を共有するすべてのサービス・マネージャは、--hostパラメータで指定したそれぞれのVIP/IPを使用して個別にアクセスされます。

最初のGGHUBノードでoracle OSユーザーとして、リバース・プロキシ構成ファイルにリストされている、このサービス・マネージャで管理するデプロイメントの名前を調べて、"_ServiceManager"の出現箇所すべてを、アンダースコアの前にデプロイメント名を付加することで変更します。

```
[oracle@gghub_prim1 ~]$ cd /u01/oracle/scripts
[oracle@gghub_prim1 ~]$ grep "Upstream Servers" ogg_<gghub1>.conf
## Upstream Servers for Deployment 'gghub1'
[oracle@gghub_prim1 ~]$ sed -i 's/_ServiceManager/<gghub1>_ServiceManager/'
 ogg_<gghub1>.conf
```

ステップ3.6.5 - NGINXのサーバー証明書のインストール

最初のGGHUBノードでroot OSユーザーとして、ファイル権限400 (-r--------)があるrootが所有する/etc/nginx/sslディレクトリにサーバー証明書とキー・ファイルをコピーします。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# mkdir /etc/nginx/ssl
[root@gghub_prim1 ~]# cp <ssl_keys> /etc/nginx/ssl/.
[root@gghub_prim1 ~]# chmod 400 /etc/nginx/ssl
[root@gghub_prim1 ~]# ll /etc/nginx/ssl
-r-------- 1 root root 2750 May 17 06:12 gghub1.chained.crt
-r-------- 1 root root 1675 May 17 06:12 gghub1.key
```

最初のGGHUBノードでoracle OSユーザーとして、リバース・プロキシ構成ファイルごとに、その証明書およびキー・ファイルの正しいファイル名を設定します。

```
[root@gghub_prim1 ~]$ vi /u01/oracle/scripts/ogg_<gghub1>.conf
# Before
 ssl_certificate /etc/nginx/ogg.pem;
 ssl_certificate_key /etc/nginx/ogg.pem;
# After
  ssl_certificate /etc/nginx/ssl/gghub1.chained.crt;
  ssl_certificate_key /etc/nginx/ssl/gghub1.key;
```

CA署名付き証明書を使用する場合、ssl_certificate NGINXパラメータで指定した証明書には、1) CA署名付き証明書、2)中間証明書および3)ルート証明書を単一ファイルに含める必要があります。この順序は重要です。そうしていないと、NGINXの起動は失敗して、エラー・メッセージが表示されます。

```
(SSL: error:0B080074:x509 certificate routines:
  X509_check_private_key:key values mismatch)
```

ルート証明書と中間証明書は、CA署名付き証明書プロバイダからダウンロードできます。

最初のGGHUBノードでroot OSユーザーとして、次のコマンド例を使用してSSL証明書の単一ファイルを生成します。

```
[root@gghub_prim1 ~]# cd /etc/nginx/ssl
[root@gghub_prim1 ~]# cat CA_signed_cert.crt
 intermediate.crt root.crt > gghub1.chained.crt
```

ssl_certificate_keyファイルは、CA署名付き証明書をリクエストする際に必要になる証明書署名リクエスト(CSR)の作成時に生成されます。

ステップ3.6.6 - NGINX構成ファイルのインストール

最初のGGhubノードでroot OSユーザーとして、デプロイメント構成ファイルを/etc/nginx/conf.dディレクトリにコピーし、デフォルトの構成ファイルを削除します。

```
[root@gghub_prim1 ~]# cp /u01/oracle/scripts/ogg_<gghub1>.conf
 /etc/nginx/conf.d
[root@gghub_prim1 ~]# rm /etc/nginx/conf.d/default.conf
```

最初のGGHUBノードでroot OSユーザーとして、NGINX構成ファイルを検証します。ファイルにエラーがある場合は、次のコマンドでエラーが報告されます。

```
[root@gghub_prim1 ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

最初のGGHUBノードでroot OSユーザーとして、NGINXを再起動して新しい構成をロードします。

```
[root@gghub_prim1 ~]# systemctl restart nginx
```

ステップ3.6.7 - GoldenGate Microservicesの接続のテスト

最初のGGHUBノードでroot OSユーザーとして、デプロイメントのユーザー名とパスワードが含まれているcurl構成ファイル(access.cfg)を作成します。

```
[root@gghub_prim1 ~]# vi access.cfg
user = "oggadmin:<password>"
[root@gghub_prim1 ~]# curl -svf
 -K access.cfg https://<VIP hostname>:<port#>/services/v2/config/health
 -XGET && echo -e "¥n*** Success"
Sample output:
* About to connect() to gghub_prim_vip.frankfurt.goldengate.com port 443 (#0)
*  Trying 10.40.0.75...
* Connected to gghub_prim_vip.frankfurt.goldengate.com (10.40.0.75) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
*  CAfile: /etc/pki/tls/certs/ca-bundle.crt
 CApath: none
* skipping SSL peer certificate verification
* NSS: client certificate not found (nickname not specified)
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
* Server certificate:
*    subject: CN=gghub_prim_vip.frankfurt.goldengate.com,OU=Oracle MAA,
O=Oracle,L=Frankfurt,ST=Frankfurt,C=GE
*    start date: Jul 27 15:59:00 2023 GMT
*    expire date: Jul 26 15:59:00 2024 GMT
*    common name: gghub_prim_vip.frankfurt.goldengate.com
*    issuer: OID.2.5.29.19=CA:true,
CN=gghub_prim_vip.frankfurt.goldengate.com,OU=Oracle MAA,O=Oracle,L=Frankfurt,C=EU
* Server auth using Basic with user 'oggadmin'
> GET /services/v2/config/health HTTP/1.1
> Authorization: Basic b2dnYWRtaW46V0VsY29tZTEyM19fXw==
> User-Agent: curl/7.29.0
> Host: gghub_prim_vip.frankfurt.goldengate.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.24.0
< Date: Thu, 27 Jul 2023 16:25:26 GMT
< Content-Type: application/json
< Content-Length: 941
< Connection: keep-alive
< Set-Cookie:
 ogg.sca.mS+pRfBERzqE+RTFZPPoVw=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJv
Z2cuc2NhIiwiZXhwIjozNjAwLCJ0eXAiOiJ4LVNDQS1BdXRob3JpemF0aW9uIiwic3ViIjoib2dnYWRta
W4iLCJhdWQiOiJvZ2cuc2NhIiwiaWF0IjoxNjkwNDc1MTI2LCJob3N0IjoiZ2dodWJsYV92aXAubG9uZG
9uLmdvbGRlbmdhdGUuY29tIiwicm9sZSI6IlNlY3VyaXR5IiwiYXV0aFR5cGUiOiJCYXNpYyIsImNyZWQ
iOiJFd3VqV0hOdzlGWDNHai9FN1RYU3A1N1dVRjBheUd4OFpCUTdiZDlKOU9RPSIsInNlcnZlcklEIjoi
ZmFkNWVkN2MtZThlYi00YmE2LTg4Y2EtNmQxYjk3ZjdiMGQ3IiwiZGVwbG95bWVudElEIjoiOTkyZmE5N
DUtZjA0NC00NzNhLTg0ZjktMTRjNTY0ZjNlODU3In0=.knACABXPmZE4BEyux7lZQ5GnrSCCh4x1zBVBL
aX3Flo=; Domain=gghub_prim_vip.frankfurt.goldengate.com; Path=/; HttpOnly; Secure;
 SameSite=strict
< Set-Cookie:
 ogg.csrf.mS+pRfBERzqE+RTFZPPoVw=1ae439e625798ee02f8f7498438f27c7bad036b270d6bfc9
5aee60fcee111d35ea7e8dc5fb5d61a38d49cac51ca53ed9307f9cbe08fab812181cf163a743bfc7;
 Domain=gghub_prim_vip.frankfurt.goldengate.com; Path=/; Secure; SameSite=strict
< Cache-Control: max-age=0, no-cache, no-store, must-revalidate
< Expires: 0
< Pragma: no-cache
< Content-Security-Policy: default-src 'self' 'unsafe-eval'
 'unsafe-inline';img-src 'self' data:;frame-ancestors
 https://gghub_prim_vip.frankfurt.goldengate.com;child-src
 https://gghub_prim_vip.frankfurt.goldengate.com blob:;
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 1; mode=block
< X-OGG-Proxy-Version: v1
< Strict-Transport-Security: max-age=31536000 ; includeSubDomains
```

```
<
* Connection #0 to host gghub_prim_vip.frankfurt.goldengate.com left intact
{"$schema":"api:standardResponse","links":[{"rel":"canonical",
"href":"https://gghub_prim_vip.frankfurt.goldengate.com/services/v2/config/health",
"mediaType":"application/json"},{"rel":"self",
"href":"https://gghub_prim_vip.frankfurt.goldengate.com/services/v2/config/health",
"mediaType":"application/json"},{"rel":"describedby",
"href":"https://gghub_prim_vip.frankfurt.goldengate.com/services/ServiceManager/v2/me
tadata-catalog/health",
"mediaType":"application/schema+json"}],"messages":[],
"response":{"$schema":"ogg:health","deploymentName":"ServiceManager",
"serviceName":"ServiceManager","started":"2023-07-27T15:39:41.867Z","healthy":true,
"criticalResources":[{"deploymentName":"gghubl1","name":"adminsrvr","type":"service",
"status":"running","healthy":true},{"deploymentName":"gghub1","name":"distsrvr",
"type":"service","status":"running","healthy":true},{"deploymentName":"gghub1",
"name":"recvsrvr","type":"service","status":"running","healthy":true}]}}
*** Success
[root@gghub_prim1 ~]# rm access.cfg
```

ノート:

自己署名 SSL 証明書を使用している環境では、curl コマンドにフラグ--insecure を追加することで、エラー "NSS error -8172 (SEC_ERROR_UNTRUSTED_ISSUER)"を回避します。

ステップ3.6.8 - NGINX default.conf構成ファイルの削除

すべてのGGHUBでroot OSユーザーとして、/etc/nginx/conf.dに作成されているデフォルト構成ファイル(default.conf)を削除します。

```
[opc@gghub_prim1 ~]$ sudo rm -f /etc/nginx/conf.d/default.conf
[opc@gghub_prim1 ~]$ sudo nginx -s reload
```

ステップ3.6.9 - GoldenGate NGINX構成ファイルの配布

GoldenGateサービス・マネージャ用のすべてのリバース・プロキシ構成ファイルを作成した後に、そのファイルを2番目のGoldenGate Hubノードにコピーする必要があります。

最初のGGHUBノードでopc OSユーザーとして、すべてのデータベース・ノードにNGINX構成ファイルを配布します。

```
[opc@gghub_prim1 ~]$ sudo tar fczP /tmp/nginx_conf.tar /etc/nginx/conf.d/
 /etc/nginx/ssl/
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ scp /tmp/nginx_conf.tar gghub_prim2:/tmp/.
```

2番目のGGHUBノードでopc OSユーザーとして、NGINX構成ファイルを展開し、デフォルトの構成ファイルを削除します。

```
[opc@gghub_prim2 ~]$ sudo tar fxzP /tmp/nginx_conf.tar
[opc@gghub_prim2 ~]$ sudo rm /etc/nginx/conf.d/default.conf
```

2番目のGGHUBノードでopc OSユーザーとして、NGINXを再起動します。

```
[opc@gghub_prim2 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[root@gghub_prim2 ~]$ sudo systemctl restart nginx
```

ノート:

プライマリおよびスタンバイの GGHUB システムで、セクション 3.6 のすべてのステップを繰り返します。

ステップ3.7 - GoldenGate Microservicesの保護によるセキュアでない直接アクセスの制限

セキュアでないOracle GoldenGate MicroservicesデプロイメントにNGINXリバース・プロキシを構成していると、そのマイクロサービスは構成したマイクロサービスのポート番号を使用してHTTP (非セキュア)のアクセスができる状態が続きます。たとえば、http://vip-name:9101というセキュアでないURLを使用して管理サーバーにアクセスできます。

Oracle GoldenGate Microservicesの各サーバー(サービス・マネージャ、adminserver、pmsrvr、distsrvrおよびrecsrvr)に対するデフォルトの動作では、構成されたポート番号を使用してすべてのネットワーク・インタフェースでリスニングします。これは、マイクロサービスへのHTTPを使用した直接アクセスを無効にして、NGINX HTTPSを使用したアクセスのみ許可する必要がある、よりセキュアなインストールには望ましくありません。

次のコマンドを使用して、localhostアドレスのみを使用するようにサービス・マネージャとデプロイメント・サービスのリスナー・アドレスを変更します。Oracle GoldenGate Microservicesへのアクセスはlocalhostからのみ許可されるようになり、localhost外部のアクセスはNGINX HTTPSポートを使用してのみ成功するようになります。

ステップ3.7.1 - サービス・マネージャの停止

最初のGGHUBノードでgrid OSユーザーとして、GoldenGateデプロイメントを停止します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl stop goldengate gghub1
[grid@gghub_prim1 ~]$ agctl status goldengate
Goldengate instance 'gghub1' is not running
```

ステップ3.7.2 - サービス・マネージャ・リスナー・アドレスの変更

最初のGGHUBノードでoracle OSユーザーとして、次のコマンドでリスナー・アドレスを変更します。変更するサービス・マネージャに、適切なポート番号を使用してください。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ export OGG_HOME=/u01/app/oracle/goldengate/gg21c
[oracle@gghub_prim1 ~]$ export OGG_VAR_HOME=/mnt/acfs_gg1/deployments/ggsm01/var
[oracle@gghub_prim1 ~]$ export OGG_ETC_HOME=/mnt/acfs_gg1/deployments/ggsm01/etc
[oracle@gghub_prim1 ~]$ $OGG_HOME/bin/ServiceManager
 --prop=/config/network/serviceListeningPort
 --value='{"port":9100,"address":"127.0.0.1"}' --type=array --persist --exit
```

ステップ3.7.3 - サービス・マネージャとデプロイメントの再起動

最初のGGHUBノードでgrid OSユーザーとして、GoldenGateデプロイメントを再起動します。

```
[opc@gghub_prim1 ~]$ sudo su - grid[grid@gghub_prim1 ~]$ agctl start
    goldengate gghub1[grid@gghub_prim1 ~]$ agctl status goldengateGoldengate
    instance 'gghub1' is running on exadb-node1
```

ステップ3.7.4 - GoldenGate Microservicesリスナー・アドレスの変更

最初のGGHUBノードでoracle OSユーザーとして、次のコマンドを使用して、サービス・マネージャで管理されるデプロイメントのGoldenGateマイクロサービス(adminsrvr、pmsrvr、distsrvr、recvsrvr)のリスニング・アドレスをすべてlocalhostに変更します。

```
[opc@gghub_prim1 ~]$ sudo chmod g+x /u01/oracle/scripts/secureServices.py
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$ /u01/oracle/scripts/secureServices.py http://localhost:9100
 --user oggadmin
Password for 'oggadmin': <oggadmin_password>
*** Securing deployment - gghub1
Current value of "/network/serviceListeningPort" for "gghub1/adminsrvr" is 9101
```

```
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/adminsrvr" is
{
  "address": "127.0.0.1",
  "port": 9101
}.
Current value of "/network/serviceListeningPort" for "gghub1/distsrvr" is 9102
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/distsrvr" is
{
  "address": "127.0.0.1",
  "port": 9102
}.
Current value of "/network/serviceListeningPort" for "gghub1/pmsrvr" is 9104
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/pmsrvr" is
{
  "address": "127.0.0.1",
  "port": 9104
}.
Current value of "/network/serviceListeningPort" for "gghub1/recvsrvr" is 9103
Setting new value and restarting service.
New value of "/network/serviceListeningPort" for "gghub1/recvsrvr" is
{
  "address": "127.0.0.1",
  "port": 9103
}.
```

ノート:

単一のデプロイメント(adminsrvr、pmsrvr、distsrvr、recvsrvr)を変更する場合は、フラグ"--deployment <instance_name>"を追加します

ステップ3.8 - NGINXを管理するClusterwareリソースの作成

Oracle Clusterwareは、NGINXリバース・プロキシの起動を制御して、GoldenGateデプロイメントの起動前に自動的にリバース・プロキシを起動できるようにする必要があります。

最初のGGHUBノードで`grid` OSユーザーとして、次のコマンドを使用して、基盤となるネットワークCRSリソースに依存するNGINXリソースの作成に必要なアプリケーションVIPリソース名を取得します。

```
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ crsctl stat res -w "TYPE = app.appviptypex2.type" |grep NAME
NAME=gghub_prim_vip
```

最初のGGHUBノードで`root` OSユーザーとして、次のコマンドを使用して、NGINXを管理するClusterwareリソースを作成します。`HOSTING_MEMBERS`値と`CARDINALITY`値は、ご使用の環境に合わせて置き換えます。

```
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# vi /u01/oracle/scripts/add_nginx.sh
# Run as ROOT
$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/crsctl add resource nginx
 -type generic_application
 -attr "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-x,
user:oracle:rwx',EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl
 start -f nginx',STOP_PROGRAM='/bin/systemctl stop
 -f nginx',CHECK_PROGRAMS='/bin/systemctl status nginx'
 ,START_DEPENDENCIES='hard(<gghub_prim_vip>)
 pullup(<gghub_prim_vip>)', STOP_DEPENDENCIES='hard(intermediate:<gghub_prim_vip>)',
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='<gghub_prim1>,<gghub_prim2>', CARDINALITY=2"
[root@gghub_prim1 ~]# sh /u01/oracle/scripts/add_nginx.sh
```

この例で作成したNGINXリソースは、指定のデータベース・ノード(HOSTING_MEMBERSで指定)で同時に実行されます。これは、複数のGoldenGateサービス・マネージャのデプロイメントが構成されていて、それらがデータベース・ノード間で独立して移動できる場合にお薦めします。

NGINX Clusterwareリソースの作成後には、GoldenGateデプロイメントの起動前にNGINXを起動するように、GoldenGate XAGリソースを変更する必要があります。

最初のGGHUBノードでroot OSユーザーとして、次のコマンド例を使用してXAGリソースを変更します。

```
# Determine the current --file systems parameter:
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl config goldengate gghub1 |grep -i "file system"
File System resources needed: acfs_primary
# Modify the --file systems parameter:
[opc@gghub_prim1 ~]$ sudo su -
[root@gghub_prim1 ~]# /u01/app/grid/xag/bin/agctl modify goldengate gghub1
 --filesystems acfs_primary,nginx
[opc@gghub_prim1 ~]$ sudo su - grid
[grid@gghub_prim1 ~]$ agctl config goldengate gghub1 |grep -i "File system"
File System resources needed: acfs_primary,nginx
```

ノート:

NGINX に依存する XAG GoldenGate の登録ごとに、前述のコマンドを繰り返します。

プライマリおよびスタンバイの GGHUB システムで、セクション 3.8 のすべてのステップを繰り返します。

ステップ3.9 - Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成

ノード間の切替え時にOracle GoldenGateプロセスにローカル・データベース接続を提供するために、Oracle GoldenGateが起動される可能性のあるクラスタのすべてのノードに対するTNS別名を作成します。TNS別名は、デプロイメントの作成時に指定したTNS_ADMINディレクトリ内のtnsnames.oraファイルで作成します。

ソース・データベースがマルチテナント・データベースの場合は、2つのTNS別名エントリが必要になります。その1つはコンテナ・データベース(CDB)用、もう1つはレプリケートされるプラガブル・データベース(PDB)用です。ターゲットのマルチテナント・データベースでは、TNS別名によってレプリケートしたデータが適用されている場所にPDBを接続します。プラガブル・データベースSERVICE_NAMEを、前のステップで作成したデータベース・サービスに設定する必要があります(「タスク2: GGHubのプライマリおよびスタンバイ・ベース・システムの準備」の「ステップ2.3: データベース・サービスの作成」を参照)。

プライマリおよびスタンバイのデータベース・システムの任意のデータベース・ノードでoracle OSユーザーとして、dbaascliを使用してデータベース・ドメイン名とSCAN名を検索します。

```
# Primary DB
[opc@exadb1_node1]$ sudo su - oracle
[oracle@exadb1_node1]$ source db_name.env
[oracle@exadb1_node1]$ dbaascli database getDetails --dbname <db_name>
 |grep 'connectString'
     "connectString" : "<primary_scan_name>:1521/<service_name>"
# Standby DB
[opc@exadb2_node1]$ sudo su - oracle
[oracle@exadb2_node1]$ source db_name.env
[oracle@exadb2_node1]$ dbaascli database getDetails --dbname <db_name>
 |grep 'connectString'
    "connectString" : "<standby_scan_name>:1521/<service_name>"
```

プライマリおよびスタンバイGGHUBのすべてのノードでoracle OSユーザーとして、sqlnet.oraファイルにOracle

GoldenGateの推奨パラメータを追加します。

```
[opc@gghub_prim1]$ sudo su - oracle
[oracle@gghub_prim1]$ mkdir -p /u01/app/oracle/goldengate/network/admin
[oracle@gghub_prim1]$
cat > /u01/app/oracle/goldengate/network/admin/sqlnet.ora <<EOF
DEFAULT_SDU_SIZE = 2097152
EOF
```

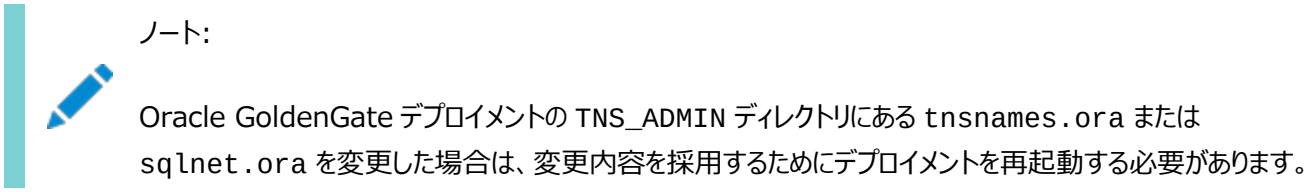
プライマリおよびスタンバイGGHUBのすべてのノードでoracle OSユーザーとして、ステップに従ってTNS別名の定義を作成します。

```
[opc@gghub_prim1 ~]$ sudo su - oracle
[oracle@gghub_prim1 ~]$
cat > /u01/app/oracle/goldengate/network/admin/tnsnames.ora <<EOF
# Source
<source_cbd_service_name>=
        (DESCRIPTION =

(CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)(RECV_TIMEOUT=30)
                (ADDRESS_LIST =
                        (ADDRESS = (PROTOCOL =
TCP)(HOST=<primary_scan_name>)(PORT=1521)))
                (ADDRESS_LIST =
                        (ADDRESS = (PROTOCOL =
TCP)(HOST=<standby_scan_name>)(PORT=1521)))
                (CONNECT_DATA=(SERVICE_NAME =
<source_cbd_service_name>.goldengate.com)))
<source_pdb_service_name>=
        (DESCRIPTION =

(CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)(RECV_TIMEOUT=30)
                (ADDRESS_LIST =
                        (ADDRESS = (PROTOCOL =
TCP)(HOST=<primary_scan_name>)(PORT=1521)))
                (ADDRESS_LIST =
                        (ADDRESS = (PROTOCOL =
TCP)(HOST=<standby_scan_name>)(PORT=1521)))
                (CONNECT_DATA=(SERVICE_NAME =
<source_pdb_service_name>.goldengate.com)))
# Target
<target_pdb_service_name>=
        (DESCRIPTION =

(CONNECT_TIMEOUT=3)(RETRY_COUNT=2)(LOAD_BALANCE=off)(FAILOVER=on)(RECV_TIMEOUT=30)
                (ADDRESS_LIST =
                        (ADDRESS = (PROTOCOL =
TCP)(HOST=<primary_scan_name>)(PORT=1521)))
                (ADDRESS_LIST =
                        (ADDRESS = (PROTOCOL =
TCP)(HOST=<standby_scan_name>)(PORT=1521)))
                (CONNECT_DATA=(SERVICE_NAME =
<target_pdb_service_name>.goldengate.com)))
EOF
[oracle@gghub_prim1 ~]$ scp /u01/app/oracle/goldengate/network/admin/*.ora
 gghub_prim2:/u01/app/oracle/goldengate/network/admin
```



ノート:

Oracle GoldenGate デプロイメントの TNS_ADMIN ディレクトリにある tnsnames.ora または sqlnet.ora を変更した場合は、変更内容を採用するためにデプロイメントを再起動する必要があります。

タスク4: Oracle GoldenGate環境の構成

このタスクを完了するには、次のステップを実行します。

- ステップ4.1 - データベース資格証明の作成
- ステップ4.2 - 自動起動プロファイルの作成
- ステップ4.3 - Oracle GoldenGateプロセスの構成

ステップ4.1 - データベース資格証明の作成

Oracle GoldenGateデプロイメントを作成したら、Oracle GoldenGate管理サービスのホーム・ページを使用して、前述のTNS別名を使用してデータベース資格証明を作成します。TNS別名を使用したデータベース資格証明の作成例は、次の図4を参照してください。

GGHUBにアクセスできるクライアント・マシンから、Oracle GoldenGate管理サービスに接続するsshトンネルを作成します。

```
$ ssh -N -L <local_port>:<vip>:443 -p 22 <gghub-node>
```

oggadminユーザーとして、データベース資格証明を作成します。

1. 管理サービスにログインします: https://localhost:<localPort>/<instance_name>/adminsrvr。
2. 「管理サービス」の「構成」をクリックします。
3. 「データベース」タブの「資格証明の追加」のプラス・ボタンをクリックします。
   ソースおよびターゲットのCDBとPDBに必要な情報を追加します。

リージョン	コンテナ	ドメイン	別名	ユーザーID
Region 1	CDB	GoldenGate	Reg1_CDB	c##ggadmin@<tns_alias>
Region 1	PDB	GoldenGate	Reg1_PDB	ggadmin@<tns_alias>
Region 2	CDB	GoldenGate	Reg2_CDB	c##ggadmin@<tns_alias>
Region 2	PDB	GoldenGate	Reg2_PDB	ggadmin@<tns_alias>

ステップ4.2 - 自動起動プロファイルの作成

Oracle GoldenGate Administration Serverの起動時にExtractプロセスとReplicatプロセスを自動的に起動する、新しいプロファイルを作成します。その後、ExtractプロセスまたはReplicatプロセスが中止された場合は再起動します。GoldenGate Microservicesでは、自動起動と再起動はプロファイルによって管理されます。

Oracle GoldenGate Administration Server GUIを使用して、各Oracle GoldenGateプロセスに割り当てることができる新しいプロファイルを作成します。

1. ソースおよびターゲットGoldenGateで、管理サービスにログインします。
2. 「管理サービス」の「プロファイル」をクリックします。
3. 「管理対象プロセスの設定」で「プロファイル」の横にあるプラス(+)記号をクリックします。
4. 次に示すように詳細を入力します。

- プロファイル名: Start_Default
- 説明: デフォルトの自動起動/再起動プロファイル
- デフォルト・プロファイル: はい
- 自動開始: はい
- 自動開始オプション
  - 開始の遅延: 1分
  - 自動再起動: はい
- 自動再起動オプション
  - 最大再試行回数: 5
  - 再試行の遅延: 30秒
  - 再試行期間: 30分
  - 失敗時にのみ再起動: はい
  - 試行回数に達したらタスクを無効化: はい

5. 「発行」をクリックします

ステップ4.3 - Oracle GoldenGateプロセスの構成

Oracle GoldenGate Microservices ArchitectureによるExtract、分散パスおよびReplicatプロセスの作成時に、GGHubノード間で共有する必要があるすべてのファイルは、すでに共有ファイル・システムに格納されているデプロイメント・ファイルによって共有されています。

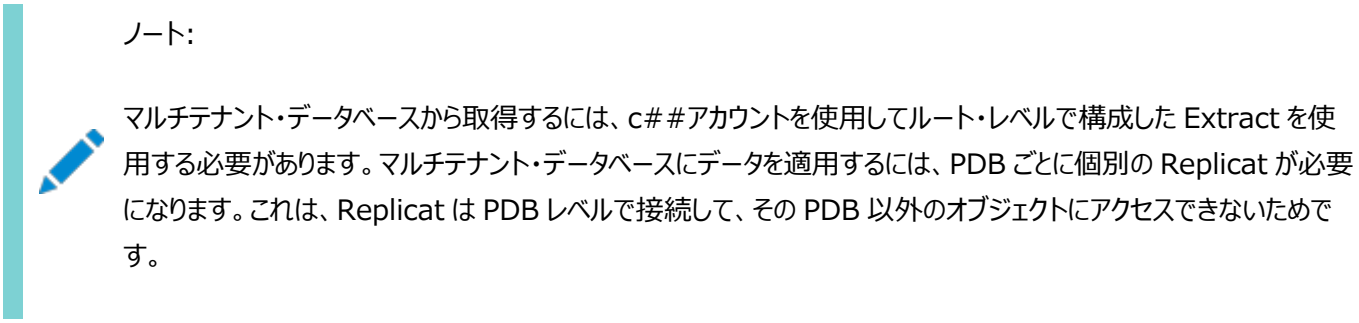
次に示す基本的な構成の詳細は、Extract、分散パスおよびReplicatプロセスのために GGhubでOracle GoldenGate Microservicesを実行する際にお薦めします。

次のサブステップを実行して、このステップを完了します。

- ステップ4.3.1 - データベース資格証明の作成
- ステップ4.3.2 - Replicatの構成
- ステップ4.3.3 - 分散パスの構成

ステップ4.3.1 - Extractの構成

Oracle GoldenGate管理サービスのGUIインタフェースを使用してExtractを作成するときには、「トレイルのサブディレクトリ」パラメータを空白のままにして、証跡ファイルが自動的に共有ファイル・システムに格納されているデプロイメント・ディレクトリに作成されるようにします。証跡ファイルのデフォルトの場所は、/<デプロイメント・ディレクトリ>/var/lib/dataディレクトリです。



ノート:

マルチテナント・データベースから取得するには、c##アカウントを使用してルート・レベルで構成した Extract を使用する必要があります。マルチテナント・データベースにデータを適用するには、PDB ごとに個別の Replicat が必要になります。これは、Replicat は PDB レベルで接続して、その PDB 以外のオブジェクトにアクセスできないためです。

REDO転送の最大パフォーマンス・モードまたは最大可用性モードを使用しているData Guard構成を使用するGoldenGate Extractプロセスでは、トランザクションが失われて論理データの不整合が発生しないように、プライマリ・システムのExtractプロセス・パラメータ・ファイルに次のパラメータを追加する必要があります。

```
TRANLOGOPTIONS HANDLEDLFAILOVER
```

このパラメータにより、まだData Guardスタンバイ・データベースに適用されていないREDOからExtractがトランザクション・データを抽出しないようにします。これは、Oracle GoldenGateがソース・スタンバイ・データベースに存在しないデータをターゲット・データベースにレプリケートしないようにするために重要です。

このパラメータが指定されていないと、ソース・データベースのデータ損失フェイルオーバー後に、ソース・データベースに存在していないデータがターゲット・データベースに保持される可能性があります。それにより、論理データの不整合が発生します。

デフォルトでは、スタンバイ・データベースで適用されたSCN情報を問い合せることができないためにExtractが停止すると、その60秒後に、警告メッセージがExtractレポート・ファイルに書き込まれます。たとえば:

```
WARNING OGG-02721 Extract has been waiting for the standby database for 60 seconds.
```

警告メッセージがExtractレポート・ファイルに書き込まれるまでの時間は、Extractパラメータの"TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNING"を使用して調整できます。

Extractが30分(デフォルト)後にスタンバイ・データベースによって適用されたSCN情報を問い合せることができない場合、Extractプロセスは異常終了して、次のメッセージがExtractレポート・ファイルに記録されます。

```
ERROR   OGG-02722  Extract abended waiting for 1,800 seconds for the
    standby database to be accessible or caught up with the primary database.
```

デフォルトの30のタイムアウトが満了する前にスタンバイ・データベースが使用可能になると、Extractはソース・データベースからデータのマイニングを続行して、次のメッセージをレポート・ファイルに報告します。

```
INFO    OGG-02723  Extract resumed from stalled state and started
    processing LCRs.
```

タイムアウト値の30分は、Extractパラメータの"TRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND <value>"を使用して調整できます。このvalueは、使用できないスタンバイを異常終了にするまでの秒数です。

計画メンテナンスの停止時など、スタンバイ・データベースが長期間使用できなくなったときに、Extractでプライマリ・データベースからデータの抽出を継続する場合は、ExtractパラメータからTRANLOGOPTIONS HANDLEDLFAILOVERパラメータを削除してExtractを再起動します(次の図4から6の例を参照)。このパラメータは、スタンバイが使用可能になったら必ず設定してください。

ノート:

スタンバイが使用できない間にプライマリ・データベースからの抽出が続行されていると、スタンバイが使用可能になったときにデータ損失フェイルオーバーが発生し、フェイルオーバーの前にすべてのプライマリ REDO が適用されていない可能性もあります。GoldenGate ターゲット・データベースには、ソース・データベースに存在しないデータが含まれるようになります。

「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」で説明したように、Extractプロセスに自動再起動プロファイルが割り当てられている場合は、Data Guardのロール・トランジション後にExtractプロセスが自動的に再起動されます。Extractは、デフォルトの5分タイムアウト期間が満了するまで、新しいスタンバイ・データベースの現在の状態を無視して、新しいプライマリ・データベースからのREDOデータのマイニングを続行します。この期間の経過後にスタンバイが使用できない場合、Extractは次のエラーで異常終了します。

```
INFO OGG-25053 Timeout waiting for 300 seconds for standby database
 reinstatement. Now enforcing HANDLEDLFAILOVER.
ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_XXXXX.
ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
 abending.
```

Extractは、GoldenGate Microservicesの自動再起動プロファイルに基づいて、再試行回数に達するか新しいスタンバイ・データベースが使用可能になるまで自動的に再起動し続け、HANDLEDLFAILOVERタイムアウトに達すると失敗します。

データベース・ロール・トランジション後のタイムアウト期間中、HANDLEDLFAILOVERパラメータは自動的に一時停止されるため、ソース・スタンバイ・データベースが最新の状態に維持されていないことの考慮なしに、データがOracle GoldenGateレプリカ・データベースにレプリケートされます。Extractの異常終了前に起動するスタンバイ・データベースのタイムアウト期間は、ExtractパラメータのTRANLOGOPTIONS DLFAILOVER_TIMEOUTを使用すると調整できます。

DLFAILOVER_TIMEOUTをデフォルトの5分のままにして、古いプライマリがスタンバイに変換できるようにすることをお薦めします。新しいスタンバイ・データベースが長期間使用できなくなるか完全に消失した場合に、Extractを起動して実行状態を維持するには、Extractパラメータ・ファイルからHANDLEDLFAILOVERパラメータを削除する必要があります。このパラメータを削除すると、ExtractはREDOがスタンバイ・データベースに適用されるまで待機することなくデータを抽出します。

スタンバイ・データベースがオンラインに復帰して、プライマリ・データベースからすべてのREDOを適用するまでは、スタンバイ・データベースとOracle GoldenGateレプリカ・データベースの間にデータの相違があります。これは、スタンバイ・データベースが最新状態になると解決されます。その時点で、統合Extractプロセス・パラメータ・ファイルにHANDLEDLFAILOVERパラメータを再追加し、Extractを停止してから再起動します。

プライマリ・データベースが損なわれたときにはブローカがスタンバイ・データベースに自動的にフェイルオーバーできるなど、Oracle Data Guardファスト・スタート・フェイルオーバーが無効になっている場合は、次に示す追加の統合Extractパラメータを指定する必要があります。

```
TRANLOGOPTIONS FAILOVERTARGETDESTID n
```

このパラメータでは、スタンバイ・データベースにまだ適用されていないREDOデータを抽出しないことに関して、Oracle GoldenGate Extractプロセスを遅らせたままにしておく必要があるスタンバイ・データベースを指定します。

Oracle Data Guardファスト・スタート・フェイルオーバーが無効のときに、追加の統合ExtractパラメータFAILOVERTARGETDESTIDを指定していないと、抽出は次のエラーで異常終了します。

```
ERROR OGG-06219 Unable to extract data from the Logmining server OGG$CAP_XXXXX.
ERROR OGG-02078 Extract encountered a fatal error in a processing thread and is
 abending.
```

FAILOVERTARGETDESTIDの正しい値を判断するには、ソース・スタンバイ・データベースへのREDOの送信に使用されるGoldenGateソース・データベースのLOG_ARCHIVE_DEST_Nパラメータを使用します。たとえば、LOG_ARCHIVE_DEST_2がスタンバイ・データベースを指している場合は2の値を使用します。

プライマリ・データベース・システムでoracleユーザーとして、次のコマンドを実行します。

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ sqlplus / as sysdba
SQL> show parameters log_archive_dest
NAME TYPE VALUE
-------------------- ----------- ------------------------------------------------------
log_archive_dest_1 string location=USE_DB_RECOVERY_FILE_DEST,
 valid_for=(ALL_LOGFILES, ALL_ROLES)
log_archive_dest_2 string service="<db_name>", SYNC AFFIRM delay=0
 optional compression=disable max_failure=0 reopen=300
 db_unique_name="<db_name>" net_timeout=30,
 valid_for=(online_logfile,all_roles)
```

この例では、次のようにExtractパラメータを設定します。

```
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
```

Extractの作成:

1. Oracle GoldenGate管理サーバーにログインします
2. 「管理サービス」の「概要」をクリックします
3. 「Extractの追加」のプラス・ボタンをクリックします。
4. 「統合Extract」を選択します
5. 次のように必要な情報を追加します。
    - プロセス名: EXT_1
    - 説明: リージョン1 CDBのExtract
    - 目的: 一方向
    - 開始: 今すぐ
    - トレイル名: aa
    - 資格証明ドメイン: GoldenGate
    - 資格証明別名: Reg1_CDB
    - PDBに登録: PDB名
6. 「次」をクリックします
7. PDBからのCDBルート取得を使用する場合は、PDB名を指定してSOURCECATALOGパラメータを追加します。
8. 「Create and Run」をクリックします

ステップ4.3.2 - Replicatの構成

通常、GGHUBがターゲットのOracle GoldenGateデータベースと同じリージョンにある場合は、ほとんどのワークロードに対する適用パフォーマンスが優れている統合パラレルReplicatを使用するようにお薦めします。

GGHUBとターゲット・データベースの間のネットワーク・レイテンシが可能なかぎり小さい場合に、最適な適用パフォーマンスを実現できます。Oracle GGHUBで実行しているリモートReplicatには、次の構成をお薦めします。

- APPLY_PARALLELISM – 自動並列度を無効化します。MAX_APPLY_PARALLELISMとMIN_APPLY_PARALLELISMを使用するかわりに、ターゲット・データベスに最大量の並行性を許可します。これは、ハブとターゲット・データベース・サーバーの使用可能なCPUに基づいて、できるかぎり高く設定することをお薦めします。
- MAP_PARALLELISM – 2から5の値を設定する必要があります。アプライヤが多数ある場合は、マッパーを増やすことでアプライヤに作業を渡す能力が向上します。
- BATCHSQL - 配列処理を使用してDMLを適用します。これにより、レイテンシが高いネットワークでネットワーク・オーバーヘッドの量が削減されます。多数のデータ競合がある場合は、BATCHSQLによってパフォーマンスが低下することに注意してください。これは、バッチ操作のロールバック後に非バッチ・モードで適用するために証跡ファイルからの再読取りがあるためです。

ステップ4.3.2.1 - チェックポイント表の作成

チェックポイント表は、Oracle GoldenGate Replicatプロセスに必須のコンポーネントです。Administration Serviceの資格証明ページからデータベースに接続した後、チェックポイント表を作成できます。

ターゲット・デプロイメントにチェックポイント表を作成します。

1. Oracle GoldenGate管理サーバーにログインします
2. 「管理サービス」の「構成」をクリックします。
3. 「データベース」をクリックして、ターゲット・データベースまたはPDBへの「接続」をクリックします。
4. 「チェックポイント」の横のプラス(+)記号をクリックします。「チェックポイントの追加」ページが表示されます。
5. 次に示すように詳細を入力します。

- チェックポイント表: ggadmin.chkp_table

6. 「発行」をクリックします

チェックポイント表の詳細は、Oracle DatabaseでのOracle GoldenGateのガイドを参照してください。

ステップ4.3.2.2 - Replicatの追加

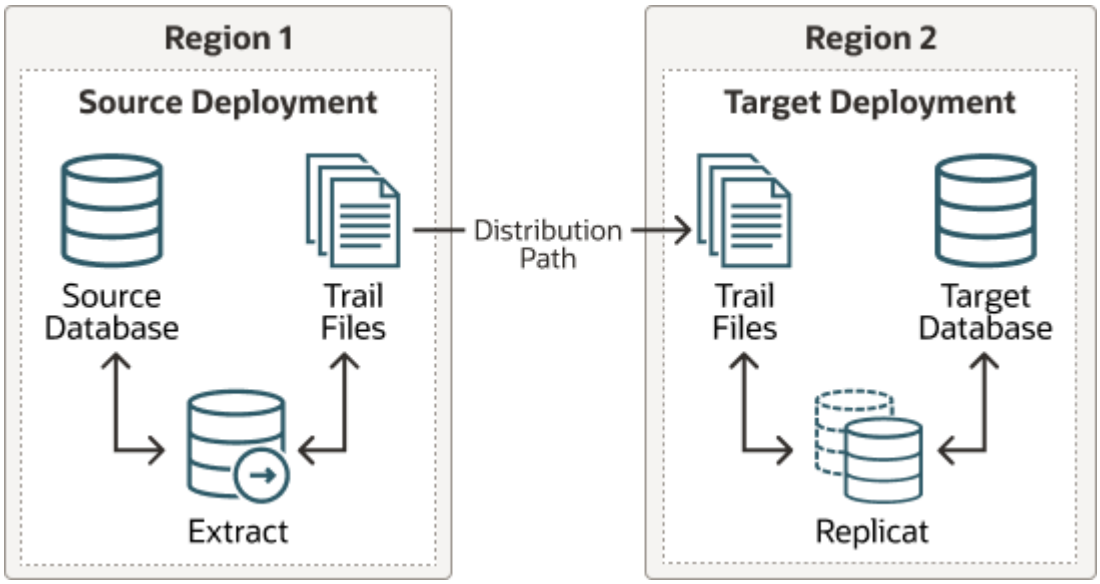
データベース接続を設定して検証した後に、次のステップを実行するとデプロイメントのReplicatを追加できます。

1. Oracle GoldenGate管理サーバーにログインします
2. 「管理サービス」ホーム・ページで、「Replicat」の横にあるプラス(+)記号をクリックします。Replicatの追加ページが表示されます。
3. Replicatのタイプを選択して、「次」をクリックします。
4. 次に示すように詳細を入力します。
    - プロセス名: REP_1
    - 説明: リージョン2 PDBのReplicat
    - 目的: 一方向
    - 資格証明ドメイン: GoldenGate
    - 資格証明別名: Reg2_PDB
    - ソース: トレイル
    - トレイル名: aa
    - 開始: ログでの位置
    - チェックポイント表: "GGADMIN"."CHKP_TABLE"
5. 「次」をクリックします
6. 「アクション・メニュー」から、「開始」をクリックします。

ステップ4.3.3 - 分散パスの構成

分散パスは、次の図に示すように、証跡ファイルを別の(または同じ)リージョン内の追加のOracle GoldenGate Hubに送信する必要がある場合にのみ必要になります。

図19-4 Oracle GoldenGate分散パス



NGINXリバース・プロキシでOracle GoldenGate Distributionパスを使用する場合は、パスのクライアントとサーバーの証

明書が構成されるように、追加のステップを実行する必要があります。

分散パスの作成の詳細は、『Oracle GoldenGate Microservices Architectureの使用』を参照してください。証明書を正しく構成するためのステップバイステップの例は、NGINXを使用したオンプレミスのOracle GoldenGateからOCI GoldenGateへの接続についてのビデオを視聴してください。

このサブステップで実行するステップは次のとおりです。

- ステップ4.3.3.1 - ターゲット・サーバーのルート証明書のダウンロードとソースOracle GoldenGateへのアップロード
- ステップ4.3.3.2 - ソースOracle GoldenGateが使用するターゲット・デプロイメントのユーザーの作成
- ステップ4.3.3.3 - ソースOracle GoldenGateの資格証明の作成
- ステップ4.3.3.4 - ソースOracle GoldenGateからターゲット・デプロイメントへの分散パスの作成
- ステップ4.3.3.5 - ターゲット・デプロイメント・コンソールのレシーバ・サービスでの接続の確認

ステップ4.3.3.1 - ターゲット・サーバーのルート証明書のダウンロードとソースOracle GoldenGateへのアップロード

ターゲット・デプロイメント・サーバーのルート証明書をダウンロードし、ソース・デプロイメントのサービス・マネージャにCA証明書を追加します。

1. ターゲットGoldenGateで、管理サービスにログインします。
2. NGINXを使用したオンプレミスのOracle GoldenGateからOCI GoldenGateへの接続についてのビデオの"ステップ2 - ターゲット・サーバーのルート証明書のダウンロード"に関する説明に従います。

ステップ4.3.3.2 - ソースOracle GoldenGateが使用するターゲット・デプロイメントのユーザーの作成

ターゲット・デプロイメントで、次に接続する分散パスのユーザーを作成します。

1. ターゲットGoldenGateで、管理サービスにログインします。
2. 「管理サービス」の「管理者」をクリックします。
3. 「ユーザー」の横のプラス(+)記号をクリックします。
4. 次に示すように詳細を入力します。
    - ユーザー名: ggnet
    - ロール: オペレータ
    - タイプ: パスワード
5. 「発行」をクリックします

ステップ4.3.3.3 - ソースOracle GoldenGateの資格証明の作成

前のステップで作成したユーザーでターゲット・デプロイメントを接続するソース・デプロイメントに資格証明を作成します。たとえば、OP2CのドメインとWSSNETの別名です。

1. ソースOracle GoldenGateで、管理サービスにログインします。
2. 「管理サービス」の「構成」をクリックします。
3. 「データベース」ホーム・ページで「資格証明」の横にあるプラス(+)記号をクリックします。
4. 次に示すように詳細を入力します。
    - 資格証明ドメイン: OP2C
    - 資格証明別名: wssnet
    - ユーザーID: ggnet
5. 「発行」をクリックします

ステップ4.3.3.4 - ソースOracle GoldenGateからターゲット・デプロイメントへの分散パスの作成

分散サーバーから受信サーバーに証跡ファイルを送信するパスが作成されます。パスはDistribution Serviceで作成できます。

ソース・デプロイメントのパスを追加するには:

1. ソースOracle Goldengateで、分散サービスにログインします。
2. Distribution Serviceホームページで「パス」の横のプラス(+)記号をクリックします。「パスの追加」ページが表示されます。
3. 次に示すように詳細を入力します。

オプション	説明
パス名	パスの名前を選択します
ソース: トレイル名	ドロップダウン・リストから Extract の名前を選択します(証跡名が自動的に入力されます)。表示されない場合は、Extract の追加時に指定した証跡名を入力します。
生成されたソース URI	サーバーの名前に localhost を指定します。これにより、任意の Oracle RAC のノードで分散パスを起動できます。
ターゲット認証方式	「ユーザーID 別名」を使用します
ターゲット	「ターゲット」の転送プロトコルを wss (セキュア Web ソケット)に設定します。「ターゲット・ホスト」を、NGINX が構成された「ポート番号」とともにターゲット・システムへの接続に使用されるターゲット・ホスト名/VIP に設定します(デフォルトは 443)。
ドメイン	「ドメイン」は、前のステップ 11.3.3 で作成した資格証明ドメイン(OP2C など)に設定します。
別名	「別名」は、ステップ 11.3.3 で作成した資格証明別名 wssnet に設定します。
自動再起動オプション	分散サーバーが自動的に起動されたときに分散パスを再起動するように設定します。これは、分散サーバーの RAC ノードの再配置後に、手動操作を不要にするために必要です。「再試行」の回数は 10 に設定することをお薦めします。「遅延」を 1 に設定します。これは、再起動の試行間に一時停止する分数です。

4. 「パスの作成」をクリックします。
5. 「アクション・メニュー」から、「開始」をクリックします。

Oracle GoldenGate Hub向けの計画および計画外停止の管理

ハブでプライマリまたはスタンバイ・ファイル・システム・クラスタの計画停止または計画外停止が発生した場合、検討する必要がある考慮事項が多数あります。

計画停止の管理

GoldenGateハブで計画メンテナンスを実行する必要がある場合は、一部のCRSリソースを停止して無効にして、再起動、ファイル・システムのフェイルオーバーの誤った開始、GoldenGateの実行の停止などから望ましくない結果が生じないようにする必要があります。プライマリまたはスタンバイ・ハブ・クラスタの計画停止のイベントでは、次の推奨事項を使用します。

すべての計画メンテナンス・イベントの場合:

- オペレーティング・システムのソフトウェアまたはハードウェアの更新とパッチ
- Oracle Grid Infrastructureの個別または診断パッチ
- クリティカル・パッチ・アップデート(CPU)プログラムのOracle Grid Infrastructureの四半期ごとの更新、またはOracle Grid Infrastructureリリースのアップグレード
- GGHubソフトウェア・ライフサイクル(次のものを含む):
    - Oracle GoldenGate
    - Oracle Grid Infrastructure Agent
    - NGINX

高可用性ソリューションとターゲットの停止時間:

GoldenGateレプリケーションが一時停止される数秒から数分

ステップ1: アイドル状態のGGHubノードのソフトウェア更新

ステップ2: GGHubノードの再配置

ステップ3: 残りの非アクティブなGGHubノードのソフトウェア更新

GGHubノードの再配置

プライマリGGHubシステムのgrid OSユーザーとして、Oracle GoldenGateインスタンスを再配置します。

```
[grid@gghubad11 ~]$ agctl status goldengate
Goldengate instance 'gghub' is running on gghubad12
[grid@gghubad11 ~]$ time agctl relocate goldengate gghubfs
real  0m43.984s
user  0m0.156s
sys   0m0.049s
```

プライマリGGHubシステムのgrid OSユーザーとして、Oracle GoldenGateインスタンスのステータスを確認します。

```
[grid@gghubad11 ~]$ agctl status goldengate
Goldengate instance 'gghub' is running on gghubad11
```

DRイベントまたはターゲット・データベースと同じリージョンでGGHubを移動するためのGGHubロール・リバーサル

GGHUBロール・リバーサルでは、スタンバイが新しいプライマリになるように、ACFSロール・リバーサルを実行します。プライマリ・

ファイル・システムとスタンバイ・ファイル・システムの両方がオンラインの場合は、acfsutil repl failoverコマンドにより、ロール・リバーサルが完了する前に、未処理のプライマリ・ファイル・システムの変更がすべて転送され、スタンバイに適用されます。

GGHUBロール・リバーサルを使用する必要がある場合:

- レプリケーションのパフォーマンスを向上させるために、GGHUBデプロイメントをターゲット・データベースの近くに移動する。
- サイトの停止をサポートする
- サイトのメンテナンスをサポートする

現在のスタンバイGGHubノードのgrid OSユーザーとして、ACFSロール・リバーサルを実行するスクリプトを作成します。

```
[grid@gghub_stby1]$ vi /u01/oracle/scripts/acfs_role_reversal.sh
acfs_mount_point=/mnt/acfs_gg1  # Specify the correct mount point
gghub_name=gghub1           # Specify the correct Goldengate instance name
log_file=`date +${gghub_name}_role_reversal_%Y-%m-%d_%H:%M:%S.log`
echo "`date` - Begin Role Reversal" | tee -a ${log_file}
/sbin/acfsutil repl failover ${acfs_mount_point}
echo "`date` - End Role Reversal" | tee -a ${log_file}
echo "`date` - Begin Start GG ${gghub_name}" | tee -a ${log_file}
agctl start goldengate ${gghub_name}
echo "`date` - End Start GG ${gghub_name}" | tee -a ${log_file}
```

現在のスタンバイ・ファイル・システムのGGHubノードのgrid OSユーザーとして、スクリプトを実行してACFSロール・リバーサルを実行します。

```
[grid@gghub_stby1]$ sh /u01/oracle/scripts/acfs_role_reversal.sh
[grid@gghub_stby1]$ cat /tmp/gghubfs_role_reversal_2023-06-16_08:21:06.log
Fri Jun 16 08:21:06 UTC 2023 - Begin Role Reversal
Fri Jun 16 08:22:27 UTC 2023 - End  Role Reversal
Fri Jun 16 08:22:27 UTC 2023 - Begin Start GG gghubfs
Fri Jun 16 08:23:15 UTC 2023 - End  Start GG gghubfs
```

現在の新しいプライマリ・ファイル・システムのGGHubノードのgrid OSユーザーとして、Oracle GoldenGateデプロイメントのステータスを確認します。

```
[grid@gghub_stby1]$ agctl status goldengate
Goldengate instance 'gghub1' is running on gghub_stby1
```

計画外停止の管理

計画外停止に伴う予想される影響

プライマリまたはスタンバイのGGHubクラスタで計画外停止が発生した場合、GoldenGateの継続的稼働を保証するための手順がいくつかあります。プライマリおよびスタンバイGGHUBシステムの計画外停止が発生した場合のガイドとして、次のGGHUB障害のユースケースを使用してください。

ユース・ケース#1 - スタンバイ・ハブの障害またはプライマリGGHubがスタンバイGGHubと通信できない

プライマリGGHubがスタンバイGGHubと通信できない場合は、次のメッセージがアクティブなクラスタ・ノードのプライマリCRSトレース・ファイル(crsd_scriptagent_grid.trc)に出力されます。

```
2023-06-21 12:06:59.506 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141} [check]
Executing action script: /u01/oracle/scripts/acfs_primary.scr[check]
2023-06-21 12:07:05.666 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141} [check]
WARNING: STANDBY not accessible (attempt 1 of 3))
2023-06-21 12:07:18.683 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141} [check]
WARNING: STANDBY not accessible (attempt 2 of 3))
2023-06-21 12:07:31.751 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141} [check]
WARNING: STANDBY not accessible (attempt 3 of 3))
```

```
2023-06-21 12:07:31.751 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141} [check]
WARNING: Problem with STANDBY file system (error: 222)
```

この時点で、スタンバイ・ファイル・システムはプライマリ・ファイル・システムの変更を受信しなくなります。プライマリ・ファイル・システムとOracle GoldenGateは障害なく機能し続けます。

このシナリオでは、次のアクション・プランを使用します。

- コマンド'acfsutil repl util verifystandby /mnt/acfs_gg –v'を使用して、スタンバイ・ファイル・システムを確認し、スタンバイ・ハブにアクセスできない理由を特定します。
- 通信エラーの原因を修正すると、スタンバイは未処理のプライマリ・ファイル・システムの変更の適用を自動的に取り入れます。警告メッセージはCRSトレース・ファイルに報告されなくなり、次に示すメッセージに置き換えられます。

```
2023-06-21 12:15:01.720 :CLSDYNAM:1427187456: [acfs_primary]{1:8532:12141} [check]
SUCCESS: STANDBY file system /mnt/acfs_gg is ONLINE
```

ユース・ケース#2 - プライマリGGHubの障害またはスタンバイGGHubがプライマリGGHubと通信できない

スタンバイGGHubがプライマリGGHubと通信できない場合は、次のメッセージがアクティブなクラスタ・ノードのスタンバイCRSトレース・ファイル(crsd_scriptagent_grid.trc)に出力されます。

```
2023-06-21 12:24:03.823 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2} [check]
Executing action script: /u01/oracle/scripts/acfs_standby.scr[check]
2023-06-21 12:24:06.928 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2} [check]
WARNING: PRIMARY not accessible (attempt 1 of 3)
2023-06-21 12:24:19.945 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2} [check]
WARNING: PRIMARY not accessible (attempt 2 of 3)
2023-06-21 12:24:32.962 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2} [check]
WARNING: PRIMARY not accessible (attempt 3 of 3)
2023-06-21 12:24:32.962 :CLSDYNAM:4156544768: [acfs_standby]{1:10141:2} [check]
WARNING: Problem with PRIMARY file system (error: 222)
```

この時点では、スタンバイ・ファイル・システムがプライマリ・ファイル・システムからファイル・システムの変更を受信している可能性はほとんどありません。

このシナリオでは、次のアクション・プランを使用します。

- コマンド'acfsutil repl util verifyprimary /mnt/acfs_gg -v'を使用して、プライマリ・ファイル・システムを確認し、プライマリ・ハブにアクセスできない理由を特定します。
- プライマリ・ファイル・システム・クラスタが停止して再起動できない場合は、スタンバイGGHubでACFSフェイルオーバーを発行します。

```
[grid@gghub_stby1]$ /sbin/acfsutil repl failover /mnt/acfs_gg # Specify the
correct mount point
[grid@gghub_stby1]$ acfsutil repl info -c -v /mnt/acfs_gg |egrep 'Site:|Primary
status|Background Resources:'
Site:                 Primary
Primary status:           Running
Background Resources:        Active
```

- 次のコマンドを実行して、新しいプライマリ・ハブで開始するacfs_primaryリソースを準備し、GoldenGateを再起動します。

```
[grid@gghub_stby1]$ echo "RESTART" > /mnt/acfs_gg/status/acfs_primary
[grid@gghub_stby1]$ agctl start goldengate <instance_name>
# Specify the GoldenGate instance name
[grid@gghub_stby1]$ agctl status goldengate
Goldengate instance '<instance_name>' is running on gghubstby-node1
```

- 古いプライマリ・ファイル・システムがオンラインに戻ったときに、新しいプライマリと古いプライマリとの接続が再開されると、古いプライマリ・ファイル・システムは自動的にスタンバイに変換されます。
- 古いプライマリ・ファイル・システムがオンラインに戻っても、プライマリ・ファイル・システムとスタンバイ・ファイル・システム間の接続が確立できない場合、acfs_primaryリソースはノードがクラッシュしたことを検出し、スタンバイへの接続を確認できないため、GoldenGateは起動されません。2つのファイル・システムが相互に通信できないため、それらの両方がプライマリであると認識される「スプリット・ブレイン」を回避できます。

ユース・ケース#3 - 二重障害のケース: プライマリGGHubの障害とスタンバイGGHubの接続失敗

プライマリGGHubがクラッシュし、オンラインに戻ったときにスタンバイ・ファイル・システムとの通信を確立できない場合は、次に示すのメッセージがアクティブ・クラスタ・ノードのプライマリCRSトレース・ファイル(crsd_scriptagent_grid.trc)に出力されます。

```
2023-06-21 17:08:52.621:[acfs_primary]{1:40360:36312} [start] WARNING: PRIMARY file
system /mnt/acfs_gg previously crashed
2023-06-21 17:08:55.678:[acfs_primary]{1:40360:36312} [start] WARNING: STANDBY not
accessible - disabling acfs_primary
```

プライマリ・ファイル・システムを手動で再起動しようとすると、CRSトレース・ファイルに、さらにメッセージが出力されます。

```
2023-06-21 17:25:54.224:[acfs_primary]{1:40360:37687} [start] WARNING:
    PRIMARY /mnt/acfs_gg disabled to prevent split brain
```

このシナリオでは、次のアクション・プランを使用します。

- コマンド'acfsutil repl util verifystandby /mnt/acfs_gg -v'を使用して、スタンバイ・ファイル・システムを確認し、スタンバイ・ハブにアクセスできない理由を特定します。
- スタンバイ・ファイル・システムとの通信を再確立できる場合は、プライマリ・ハブでGoldenGateを再起動します。

```
[grid@gghub_prim1]$ agctl start goldengate <instance_name> # Specify the
GoldenGate instance name
[grid@gghub_prim1]$ agctl status goldengate
Goldengate instance '<instance_name>' is running on gghub_prim1
```

- スタンバイ・ファイル・システムとの通信を再確立できない場合は、次のコマンドを使用して、プライマリ・ハブでGoldenGateを再起動します。

```
[grid@gghub_prim1]$ echo "RESTART" > /mnt/acfs_gg/status/acfs_primary
[grid@gghub_prim1]$ agctl start goldengate <instance_name>
# Specify the GoldenGate instance name
[grid@gghub_prim1]$ agctl status goldengate
Goldengate instance '<instance_name>' is running on gghub_prim1
```

- スタンバイ・ファイル・システムとの通信がリストアされると、ACFSレプリケーションはプライマリ・ファイル・システムの変更を引き続きレプリケートします。

20 Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス

これらのベスト・プラクティスを使用して、Oracle GoldenGate Microservices Architectureを構成し、Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D)またはOracle Exadata Database Service on Cloud@Customer (ExaDB-C@C)、およびOracle Database File System (DBFS)またはOracle ASM Cluster File System (ACFS)と連動するようにします。

次の項を参照してください。

Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成の概要

Oracle GoldenGate Microservices Architectureをホストするターゲット Oracle Exadata Database Serviceは、ソース・データベース、ターゲット・データベース、または場合によってOracle GoldenGateのソース・データベースとターゲット・データベースの両方として機能できます。これらのベスト・プラクティスは、Oracle Exadata Database Service on Dedicated InfrastructureまたはCloud@Customerを使用したOracle GoldenGate Microservices Architectureの構成に適用できます。

このロードマップに従って、Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D)または Oracle Exadata Database Service on Cloud@CustomerでOracle GoldenGateを構成します。

- タスク1 - 始める前に: Oracle Exadata Cloud InfrastructureまたはCloud@CustomerでOracle GoldenGateを構成するには、ExaDB-DまたはExaDB-C@Cシステム、CA証明書および追加のソフトウェアの構成が必要です。
- タスク2 - GoldenGateに向けたOracle Databaseの構成: ベスト・プラクティスを使用して、Oracle GoldenGateのレプリケート環境でソース・データベースとターゲット・データベースを構成します。
- タスク3 - Oracle GoldenGateデプロイメントを格納する共有ファイル・システムの作成: Oracle GoldenGateを使用してOracle Cloud InfrastructureでHAを構成するために、Oracle DBFSまたはOracle ACFSを設定します。アーキテクチャのGoldenGateレプリカ・データベースがクラウド・フィジカル・スタンバイ・データベース(Oracle Data Guard)によって保護されている場合は、Oracle DBFSを使用します。それ以外の場合はACFSを使用します。
- タスク4 - Oracle GoldenGateのインストール: ベスト・プラクティスを使用して、Oracle Cloud InfrastructureにOracle GoldenGateコンポーネントをインストールし、構成します。
- タスク5 - Oracle GoldenGateデプロイメントの作成: Oracle GoldenGate Configuration Assistantを使用してGoldenGateデプロイメントを作成するための、レスポンス・ファイルを作成します。
- タスク6 - ネットワークの構成: Oracle GoldenGateが正しく機能するように、プライベートDNSゾーン、VIP、要塞、セキュリティ・リスト、ファイアウォールなどの仮想クラウド・ネットワーク(VCN)のコンポーネントを構成します。
- タスク7 - Oracle Grid Infrastructure Agentの構成: Oracle Cloud InfrastructureでHA用にOracle GoldenGateを構成します。
- タスク8 - NGINXリバース・プロキシの構成: NGINXを使用して、リバース・プロキシとHAを構成します。
- タスク9 - Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成: TNS別名を作成して、Oracle RACノード間を切り替えるときのOracle GoldenGateプロセスのデータベース接続を簡素化します。
- タスク10 - 新規プロファイルの作成: Oracle GoldenGate Administration Serverの起動時にExtractプロセスとReplicatプロセスを自動的に起動する、新しいプロファイルを作成します。
- タスク11 - Oracle GoldenGateプロセスの構成: データ・レプリケーションに必要なOracle GoldenGate Extract、ReplicatおよびPathプロセスを作成し、構成します。

タスク1 - 始める前に

このタスクを完了するには、次のステップを実行します。

- ステップ1.1 - Oracle Cloud Infrastructure DBシステムの設定
- ステップ1.2 - 必要なソフトウェアのダウンロード
- ステップ1.3 - Oracle Linux Yum Serverからソフトウェアをインストールするためのシステムの構成
- ステップ1.4 - セキュアなデプロイメントの要件(証明書)

ステップ1.1 - Oracle Cloud Infrastructure DBシステムの設定

開始するには、Oracle GoldenGateデプロイメントの Oracle Exadata Database Service on Dedicated InfrastructureまたはCloud@Customerが必要です。

ビジネスのニーズに応じて、既存のExaDB-D/ExaDB-C@Cシステムを使用してOracle GoldenGateをデプロイすることも、新しいシステムを起動することもできます。

ExaDB-Dシステムの起動と管理の手順については、Oracle Exadata Database Service on Dedicated Infrastructureを参照してください。また、ExaDB-C@Cの場合は、Oracle Exadata Database Service on Cloud@Customerを参照してください。

ステップ1.2 - 必要なソフトウェアのダウンロード

1. 必要なソフトウェアをすべてダウンロードするためのステージング・ディレクトリを作成します

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# mkdir /u02/app_acfs/goldengate
[root@exadb-node1 ~]# chown oracle:oinstall /u02/app_acfs/goldengate
[root@exadb-node1 ~]# chmod g+w /u02/app_acfs/goldengate
```

2. それ以降のパッチをベース・リリースにダウンロードして、My Oracle Supportの「パッチと更新版」タブに移動します
    - 詳細は、「Oracle GoldenGate Microservices Architecture用のパッチのインストール」を参照してください。
    - 最低限必要なバージョンは、パッチ35214851: Oracle GoldenGate 21.9.0.0.2 Microservices for Oracle
3. My Oracle Supportドキュメント2542082.1から、Oracle Database 21c (21.0.0.0.0)用の最新のOPatchリリース(パッチ6880880)をダウンロードします。
4. Oracle GoldenGateダウンロードから、Oracle GoldenGate 21c Microservicesソフトウェア以上をダウンロードします。
5. Oracle Clusterware用Oracle Grid Infrastructure Standalone Agentから、Oracle Clusterware 19c、バージョン10.2以上用のOracle Grid Infrastructure Standalone Agentをダウンロードします。
6. My Oracle Supportのドキュメント1054431.1から、`mount-dbfs.sh`および`mount-dbfs.conf`を含む`mount-dbfs-version.zip`ファイルをダウンロードします。
7. My Oracle Supportのドキュメント2826001.1からpythonスクリプト(`secureServices.py`)をダウンロードします。

ステップ1.3 - Oracle Linux Yum Serverからソフトウェアをインストールするためのシステムの構成

Oracle Linux yumサーバーは、Oracle Linuxおよび互換性のあるディストリビューション用のソフトウェアをホストします。これらの手順は、LinuxシステムをOracle Linux yumサーバー用に構成し、yumを介してソフトウェアをインストールする作業を開始するのに役立ちます。

1. root OSユーザーとして、次の内容のファイル/etc/yum.repos.d/oracle-public-yum-ol7.repoを作成します。

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]#
cat > /etc/yum.repos.d/oracle-public-yum-ol7.repo <<EOF
[ol7_latest]
name=Oracle Linux $releasever Latest ($basearch)
baseurl=http://yum$ociregion.oracle.com/repo/OracleLinux/OL7/latest/¥$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
EOF
```

2. root OSユーザーとして、ドキュメントID 2397264.1に従って構成ファイル/etc/yum.confを変更し、ソフトウェア・リポジトリが有効になっていることを確認します。

```
[root@exadb-node1 ~]# yum repolist
repo idrepo name       status
!public_ol7_latestOracle Linux 7.9-6.0.1.el7_9 Latest (x86_64)19,712+4,957
repolist: 19,992
```

ステップ1.4 - セキュアなデプロイメントの要件(証明書)

セキュアなデプロイメントでは、RESTful APIコールを発行し、証跡データをSSLまたはTLSを介して分散サーバーとレシーバ・サーバー間で転送します。

認証局(CA)から取得した既存のビジネス証明書を使用するか、独自の証明書を作成できます。

続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

タスク2 - GoldenGateに向けたOracle Databaseの構成

ソースおよびターゲットのOracle GoldenGateデータベースは、次に示す推奨事項を使用して構成する必要があります。

このタスクを完了するには、次のステップを実行します。

- ステップ2.1 - データベース構成
- ステップ2.2 - データベース・レプリケーション管理者ユーザーの作成
- ステップ2.3 - データベース・サービスの作成

ステップ2.1 - データベース構成

ソースおよびターゲットのOracle GoldenGateデータベースは、次に示す推奨事項を使用して構成する必要があります。

1. データベース初期化パラメータを設定してOracle GoldenGateレプリケーションを有効にします。
2. ソースOracle GoldenGateデータベースの場合:
    - データベースをARCHIVELOGモードで実行します
    - FORCE LOGGINGモードを有効にします
    - 最小サプリメンタル・ロギングを有効にしてください
    - さらに、すべてのレプリケートされたオブジェクトのスキーマまたは表レベルのログを追加します
3. STREAMS_POOL_SIZE初期化パラメータを使用して、ソース・データベースのシステム・グローバル領域(SGA)でストリーム・プールを構成します。統合Replicatを使用する場合は、ターゲット・データベースでストリーム・プールのみが必要です。

Oracle GoldenGate用にデータベースを準備するステップは、『Oracle DatabaseでのOracle GoldenGateクラシック・アーキテクチャの使用』を参照してください。

1. ソースおよびターゲット・システムのoracle OSユーザーとして、次のSQL命令を発行してデータベースを構成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
SQL> alter system set ENABLE_GOLDENGATE_REPLICATION=true scope=both sid='*';
SQL> alter system set STREAMS_POOL_SIZE=<SIZE_IN_GB> scope=both sid='*';
```

2. ソース・システムのoracle OSユーザーとして、次のSQL命令を発行してデータベースを構成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
SQL> ALTER DATABASE FORCE LOGGING;
SQL> ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
SQL> ARCHIVE LOG LIST
Database log mode Archive Mode
Automatic archival Enabled
Archive destination USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence 110
Next log sequence to archive 113
Current log sequence 113
```

ステップ2.2 - データベース・レプリケーション管理者ユーザーの作成

ソースおよびターゲットのOracleデータベースには、次のように適切な権限が割り当てられたGoldenGate管理者ユーザーを作成する必要があります。

- マルチテナント・コンテナ・データベース(CDB)の場合:

- ソース・データベースであるGoldenGate Extractは、c##を使用してルート・コンテナ・データベース内のユーザーに接続するように構成する必要があります
- ターゲット・データベースには、プラガブル・データベース(PDB)ごとに個別のGoldenGate管理者ユーザーが必要です。Oracleマルチテナント・データベースでのGoldenGate管理者の作成の詳細は、マルチテナント・コンテナ・データベースでのOracle GoldenGateの構成を参照してください。

- 非CDBデータベースについては、Oracle GoldenGate資格証明の確立を参照してください。

1. ソース・システムのoracle OSユーザーとして、次のSQL命令を発行してOracle GoldenGateのデータベース・ユーザーを作成し、必要な権限を割り当てます。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
# CDB
alter session set container=cdb$root;
create user c##ggadmin identified by "<ggadmin_password>" container=all default
 tablespace USERS temporary tablespace temp;
alter user c##ggadmin quota unlimited on users;
grant set container to c##ggadmin container=all;
grant alter system to c##ggadmin container=all;
grant create session to c##ggadmin container=all;
grant alter any table to c##ggadmin container=all;
grant resource to c##ggadmin container=all;
exec dbms_goldengate_auth.grant_admin_privilege('c##ggadmin',container=>'all');
# Source PDB
alter session set container=<PDB_name>;
create user ggadmin identified by "<ggadmin_password>" container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

2. ターゲット・システムのoracle OSユーザーとして、次のSQL命令を発行してOracle GoldenGateのデータベース・ユーザーを作成し、必要な権限を割り当てます。

```
# Target PDB
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ sqlplus / as sysdba
alter session set container=<PDB_name>;
create user ggadmin identified by "<ggadmin_password>" container=current;
grant alter system to ggadmin container=current;
grant create session to ggadmin container=current;
grant alter any table to ggadmin container=current;
grant resource to ggadmin container=current;
grant dv_goldengate_admin, dv_goldengate_redo_access to ggadmin
container=current;
exec dbms_goldengate_auth.grant_admin_privilege('ggadmin');
```

ステップ2.3 - データベース・サービスの作成

データベースを開いたときにOracle Grid Infrastructure AgentがOracle GoldenGateのデプロイメントを自動的に開始するには、データベースサービスが必要です。DBFSを共有ファイル・システムに使用すると、データベース・サービスを使用してDBFSを正しいRACインスタンスにマウントすることもできます。

ソース・マルチテナント・データベースを使用する場合は、ルート・コンテナ・データベース(CDB)とレプリケートされるスキーマを含むプラガブル・データベース(PDB)に、個別のサービスが必要です。ターゲット・マルチテナント・データベースの場合は、PDBに単一のサービスが必要です。

1. oracle OSユーザーとして、次のコマンドを使用してCDBデータベース・サービスを作成および開始します。

```
[oracle@exadb-node1 ~]$ source <db_name>.env
[oracle@exadb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service `echo $ORACLE_UNQNAME`_ogg -preferred <SID1> -available <SID2>
 -role PRIMARY
[oracle@exadb-node1 ~]$ srvctl start service -db $ORACLE_UNQNAME
 -service `echo $ORACLE_UNQNAME`_ogg
```

データベースがマルチテナント環境の一部である場合は、必ずプラガブル・データベース(PDB)でサービスを作成してください。

2. oracle OSユーザーとして、次のコマンドを使用してPDBデータベース・サービスを作成および開始します。

```
[oracle@exadb-node1 ~]$ dbaascli database getDetails
 --dbname <db_name> |grep pdbName
 "pdbName" : "<PDB_NAME>",
[oracle@exadb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service <PDB_NAME>_ogg -preferred <SID1>,<SID2> -pdb <PDB_NAME> -role PRIMARY
[oracle@exadb-node1 ~]$ srvctl start service -db $ORACLE_UNQNAME
 -service <PDB_NAME>_ogg
```

3. oracle OSユーザーとして、サービスが実行されていることを確認します。

```
[oracle@exadb-node1 ~]$ srvctl status service -d $ORACLE_UNQNAME |grep _ogg
Service <ORACLE_UNQNAME>_ogg is running on instance(s) <SID1>
Service <PDB_NAME>_ogg is running on instance(s) <SID1>
```

データベース・サービスの作成の詳細は、『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』のサーバー制御ユーティリティ・リファレンスを参照してください。

タスク3 - Oracle GoldenGateデプロイメントを格納する共有ファイル・システムの作成

Oracle GoldenGate Microservices Architectureの設計では、インストールとデプロイメントのディレクトリ構造が簡略化されています。

- インストール・ディレクトリは、ソフトウェア・パッチ適用時の停止時間を最小限に抑えるために、各データベース・ノードのローカル記憶域に配置する必要があります。
- デプロイメント・ディレクトリは、Oracle GoldenGate Configuration Assistant (oggca.sh)を使用したデプロイメントの作成時に作成されます。このディレクトリは、共有ファイル・システムに配置する必要があります。デプロイメント・ディレクトリには、構成、セキュリティ、ログ、パラメータ、証跡およびチェックポイントのファイルが含まれます。

DBFSまたはOracle Automatic Storage Management Cluster File System (ACFS)にデプロイメントを配置すると、システム障害の発生時に最高のリカバリ性とフェイルオーバー機能が得られます。クラスタ全体でチェックポイント・ファイルの可用性を確保することは、障害発生後にGoldenGateプロセスが最後に認識された位置から実行を継続できるようにするために不可欠です。

Oracle Data GuardとともにOracle GoldenGateを構成する場合、推奨のファイル・システムはDBFSです。DBFSはData Guardによって保護されるデータベースに格納され、XAGと完全に統合できます。Data Guardのロール・トランジションが発生した場合、このファイル・システムは新しいプライマリ・サーバーに自動的にマウントされ、その後でOracle GoldenGateが自動起動されます。ACFSはOracle Data Guard構成の一部ではないため、これは現時点ではできません。

ノート:

このドキュメントには、Oracle Data Guard とともに Oracle GoldenGate を構成するステップは含まれていません。

Oracle Data Guardが存在しない場合、推奨のファイル・システムはACFSです。Oracle ACFSは、複数のプラットフォームに対応するスケーラブルなファイル・システムおよびストレージ管理のテクノロジです。このテクノロジにより、Oracle Automatic Storage Management (Oracle ASM)機能は、Oracle Databaseの外部に維持されているカスタマ・ファイルをサポートするように拡張されます。

ファイル・システムの要件に基づいて、次のいずれかのステップを実行して、このタスクを完了します。

- ステップ3a - Oracle Database File System (DBFS)
- ステップ3b - Oracle ASM Cluster File System (ACFS)

ステップ3a - Oracle Database File System (DBFS)

Oracle GoldenGateプロセスが接続されるデータベースと同じデータベース内にDBFS表領域を作成する必要があります。たとえば、Oracle GoldenGate統合ExtractプロセスがGGDBというデータベースから抽出される場合は、同じGGDBデータベースにDBFS表領域を配置します。

Oracle GoldenGateのデプロイメント・ファイルを格納するためのファイル・システムを作成します。最大12時間の証跡ファイルの格納を可能にするには、十分な証跡ファイルのディスク領域を割り当てる必要があります。そうすることで、新しい証跡ファイルを受信できないターゲット環境で問題が発生した場合に、証跡ファイルの生成に十分な領域を確保します。12時間に必要な

領域の量は、実際の本番データで証跡ファイルの生成率をテストすることでのみ決定できます。

次のサブステップを実行して、このステップを完了します。

- ステップ3a.1 - Oracle Exadata Database ServiceのDBFSの構成
- ステップ3a.2 - DBFSリポジトリの作成
- ステップ3a.3 - (CDBのみ) TNSNAMES内のエントリの作成
- ステップ3a.4 - mount-dbfsスクリプトのダウンロードと編集
- ステップ3a.5 - DBFSリソースのOracle Clusterwareへの登録
- ステップ3a.6 - DBFSリソースの起動

ステップ3a.1 - Oracle Exadata Database ServiceのDBFSの構成

1. opc OSユーザーとして、fuseグループにgridユーザーを追加します。

```
[opc@exadb-node1]$ sudo -u grid $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
-f2)/bin/olsnodes > ~/dbs_group
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc sudo usermod -a -G fuse grid
```

2. opc OSユーザーとして、ファイル/etc/fuse.confが存在していることと、user_allow_otherオプションが含まれていることを確認します。

```
[opc@exadb-node1]$ cat /etc/fuse.conf
# mount_max = 1000
# user_allow_other
```

3. このステップは、すでにオプションuser_allow_otherが/etc/fuse.confファイル内にある場合はスキップします。

   それ以外の場合は、opc OSユーザーとして次のコマンドを実行してオプションを追加します。

```
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc "echo user_allow_other | sudo tee
-a /etc/fuse.conf"
```

4. opc OSユーザーとして、DBFSファイル・システムのマウント・ポイントとして使用する空のディレクトリを作成します。

```
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc sudo mkdir -p /mnt/dbfs
```

5. opc OSユーザーとして、マウント・ポイント・ディレクトリの所有権を変更して、grid OSユーザーがアクセスできるようにします。

```
[opc@exadb-node1]$ dcli -g ~/dbs_group -l opc sudo chown oracle:oinstall
/mnt/dbfs
```

ステップ3a.2 - DBFSリポジトリの作成

ターゲット・データベース内にDBFSリポジトリを作成します。リポジトリを作成するには、DBFSオブジェクトを保持するターゲットPDB内に新しい表領域を作成して、そのオブジェクトを所有するデータベース・ユーザーを作成します。

ノート:

Oracle マルチテナント・データベースを使用する場合は、プラガブル・データベース(PDB)内に DBFS 表領域を作成する必要があります。GoldenGate Extract または Replicat プロセスの接続先と同じ PDB を使用することをお薦めします。これにより、DBFS がデータベースの依存関係に前のステップで作成したものと同じデータベース・サービスを使用できるようになります。

1. oracle OSユーザーとして、データベースに表領域を作成します。

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ source DB_NAME.env
[oracle@exadb-node1]$ sqlplus / as sysdba
SQL> alter session set container=<pdb_name>;
SQL> create bigfile tablespace dbfstb1 datafile size 32g autoextend on next 8g
 maxsize 300g NOLOGGING EXTENT MANAGEMENT LOCAL AUTOALLOCATE SEGMENT SPACE
 MANAGEMENT AUTO;
SQL> create user dbfs_user identified by "<dbfs_user_password>"
 default tablespace dbfstb1 quota unlimited on dbfstb1;
SQL> grant connect, create table, create view, create procedure,
 dbfs_role to dbfs_user;
```

2. oracle OSユーザーとして、DBFSを保持するデータベース・オブジェクトを作成します。このスクリプトには、2つの引数を指定します。
   - dbfstb1: DBFSデータベース・オブジェクトの表領域
   - goldengate: ファイル・システム名 - 任意の文字列を指定可能で、マウント・ポイントの下のディレクトリとして表示されます

```
 [oracle@exadb-node1]$ sqlplus dbfs_user/"<dbfs_user_password>"@<db_name>_dbfs
SQL> start $ORACLE_HOME/rdbms/admin/dbfs_create_filesystem dbfstb1 goldengate
```

ステップ3a.3 - (CDBのみ) TNSNAMES内のエントリの作成

1. oracle OSユーザーとして、データベース・ドメイン名を検索します。

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ source DB_NAME.env
[oracle@exadb-node1]$ sqlplus / as sysdba
SQL> show parameter db_domain
NAME TYPE VALUE
------------------------------------ ----------- ------------------------------
db_domain string <db_domain_name>
```

2. oracle OSユーザーとして、$TNS_ADMIN/tnsnames.oraファイルに接続エントリを追加します。

```
[oracle@exadb-node1]$ vi $TNS_ADMIN/tnsnames.ora
dbfs =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
 (CONNECT_DATA =
 (SERVICE_NAME = <pdb_service_name>.<db_domain_name> )
 )
 )
```

3. oracle OSユーザーとして、残りのノードに$TNS_ADMIN/tnsnames.oraファイルを配布します。

```
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f $TNS_ADMIN/tnsnames.ora -d $TNS_ADMIN/
```

ステップ3a.4 - mount-dbfsスクリプトの編集

1. zipファイルを解凍し、mount-dbfs.conf内の変数設定を環境に応じて編集します。

   ファイルのコメントは、該当する変数の値を確認するために役立ちます。

   - DBNAME: echo $ORACLE_UNQNAME
   - MOUNT_POINT: /mnt/dbfs/goldengate
   - ORACLE_HOME (RDBMS ORACLE_HOMEディレクトリ): echo $ORACLE_HOME
   - GRID_HOME (GRID INFRASTRUCTURE HOMEディレクトリ): echo $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)

- DBFS_PASSWD (WALLET=falseの場合にのみ使用します)
- DBFS_PWDFILE_BASE (WALET=falseの場合にのみ使用します)
- WALLET (trueまたはfalseにします)
- TNS_ADMIN (WALLET=trueまたはPDBの場合にのみ使用します): echo $TNS_ADMIN
- DBFS_LOCAL_TNSALIAS (WALLET=trueの場合にのみ使用します)
- IS_PDB (PDBを使用する場合はtrueに設定します)
- PDB (該当する場合はPDB名): PDB name
- PDB_SERVICE (該当する場合はステップ2.3で作成したデータベース・サービス): PDB_SERVICE_NAME
- MOUNT_OPTIONS: allow_other,direct_io,failover,nolock
    - failoverオプションは、すべてのファイル書込みをIMMEDIATE WAITモードでDBFSデータベースにコミットすることを強制します。これにより、データベースまたはノードの障害時にデータがdbfs_clientキャッシュに書き込まれていても、まだデータベースに書き込まれていない場合に、データの損失を防止できます。
    - nolockマウント・オプションは、Oracle Database 18c以降のバージョンを使用する場合に必要になります。これは、ファイルが現在ロックされているときにOracle RACノードの障害が発生するとGoldenGateプロセスに問題が発生する可能性があるDBFSファイルのロックが変更されたためです。

2. grid OSユーザーとして、mount-dbfs-<version>.zipを解凍して、構成ファイルmount-dbfs.confを編集します。

```
[opc@exadb-node1]$ sudo su - grid
[grid@exadb-node1]$ cd /u02/app_acfs/goldengate
[grid@exadb-node1]$ unzip mount-dbfs-<version>.zip
[grid@exadb-node1]$ vi mount-dbfs.conf
```

mount-dbfs.confの例:

```
DBNAME=<DB_UNIQUE_NAME>
MOUNT_POINT=/mnt/dbfs/goldengate
DBFS_USER=dbfs_user
GRID_HOME=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)
if [ -z "${GRID_HOME}" ]; then
 echo "GRID_HOME is unset or set to the empty string"
fi
ORACLE_HOME=$($GRID_HOME/bin/srvctl config database -d $DBNAME |grep 'Oracle
home:' | cut -d: -f2 |sed 's/ //g')
if [ -z "${ORACLE_HOME}" ]; then
 echo "ORACLE_HOME is unset or set to the empty string"
fi
LOGGER_FACILITY=user
MOUNT_OPTIONS=allow_other,direct_io,failover,nolock
PERL_ALARM_TIMEOUT=14
DBFS_PASSWD=<DBFS_USER_PASSWORD>
DBFS_PWDFILE_BASE=/tmp/.dbfs-passwd.txt
WALLET=false
TNS_ADMIN=$ORACLE_HOME/network/admin/<DB_NAME>
IS_PDB=true
PDB=<PDB_NAME>
PDB_SERVICE=<PDB_SERVICE_NAME>
```

3. grid OSユーザーとして、CRSリソースが停止したときにDBFSが強制的にアンマウントされるようにmount-dbfs.shスクリプトを変更します。

```
[grid@exadb-node1]$ vi /u02/app_acfs/goldengate/mount-dbfs.sh
# Change two occurrences of:
$FUSERMOUNT -u $MOUNT_POINT
# To the following:
```

```
$FUSERMOUNT -uz $MOUNT_POINT
```

4. opc OSユーザーとして、mount-dbfs.confをデータベース・ノードの/etc/oracleディレクトリにコピー(必要に応じて名前を変更)して、適切な権限を設定します。

```
[opc@exadb-node1]$ sudo -u grid $(grep ^crs_home /etc/oracle/olr.loc | cut
 -d= -f2)/bin/olsnodes > ~/dbs_group
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc -d /tmp
 -f /u02/app_acfs/goldengate/mount-dbfs.conf
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 cp /u02/app_acfs/goldengate/mount-dbfs.conf /etc/oracle
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chown grid:oinstall /etc/oracle/mount-dbfs.conf
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chmod 660 /etc/oracle/mount-dbfs.conf
```

5. opc OSユーザーとして、mount-dbfs.shをデータベース・ノードの適切なディレクトリ($GI_HOME/crs/script)にコピー(必要に応じて名前を変更)し、適切な権限を設定します。

```
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 mkdir $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo chown
 grid:oinstall $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l grid
 -d $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
 -f /u02/app_acfs/goldengate/mount-dbfs.sh
[opc@exadb-node1]$ /usr/local/bin/dcli -g ~/dbs_group -l grid chmod 770
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script/mount-dbfs.sh
```

ステップ3a.5 - DBFSリソースのOracle Clusterwareへの登録

このリソースは、Oracle Clusterwareに登録するときにはcluster_resourceとして作成します。

cluster_resourceを使用する理由は、ファイル・システムを一度に1つのノードにのみマウントできるようにして、同時ファイル書込みの可能性が生じ、ファイル破損問題の原因になるDBFSの同時ノードのマウントを防止するためです。

1. grid OSユーザーとして、前のDBFSサービス依存関係についてのステップで作成したデータベース・サービスのリソース名を検索します。

```
[opc@exadb-node1]$ sudo su - grid
[grid@exadb-node1]$ crsctl stat res |grep <PDB_NAME>
NAME=ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.svc
```

2. oracle OSユーザーとして、次のスクリプトを実行してClusterwareリソースを登録します。

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ vi /u02/app_acfs/goldengate/add-dbfs-resource.sh
##### start script add-dbfs-resource.sh
#!/bin/bash
ACTION_SCRIPT=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -
f2)/crs/script/mount-dbfs.sh
RESNAME=dbfs_mount
DEPNAME=ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.svc
ORACLE_HOME=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=$ACTION_SCRIPT, ¥
 CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, ¥
 START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',¥
 STOP_DEPENDENCIES='hard($DEPNAME)',¥
 SCRIPT_TIMEOUT=300"
```

```
##### end script add-dbfs-resource.sh
[oracle@exadb-node1]$ sh /u02/app_acfs/goldengate/add-dbfs-resource.sh
```

ノート:

$RESNAME リソースの作成後、$RESNAME リソースが ONLINE のときに$DBNAME データベースを停止するには、srvctl の使用時に force フラグを指定します。

例: srvctl stop database -d fsdb -f

ステップ3a.6 - DBFSリソースの起動

grid OSユーザーとして、リソースを起動します。

```
 [opc@exadb-node1]$ sudo su - grid
[grid@exadb-node1]$ crsctl start res dbfs_mount -n `hostname`
CRS-2672: Attempting to start 'dbfs_mount' on 'exadb-node1'
CRS-2676: Start of 'dbfs_mount' on 'exadb-node1' succeeded
[grid@exadb-node1]$ crsctl stat res dbfs_mount -t
--------------------------------------------------------------------------------
Name Target State Server State details
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
dbfs_mount
 1 ONLINE ONLINE exadb-node1 STABLE
--------------------------------------------------------------------------------
```

ノート:

共有ファイル・システムは、マウントされたままにします。これは、今後のステップで Oracle GoldenGate デプロイメントを作成するために必要です。

ステップ 3b - Oracle ASM Cluster File System (ACFS)

Oracle ACFS は、Oracle RAC 構成の共有 Oracle GoldenGate ファイルのための DBFS の代替です。Oracle のデプロイメント・ファイルを格納するための単一の ACFS ファイル・システムを作成します。

最大12時間の証跡ファイルの格納を可能にするために、十分な証跡ファイルのディスク領域を割り当てることをお薦めします。そうすることで、新しい証跡ファイルを受信できないターゲット環境で問題が発生した場合に、証跡ファイルの生成に十分な領域を確保します。12時間に必要な領域の量は、実際の本番データで証跡ファイルの生成率をテストすることでのみ決定できます。

次のサブステップを実行して、このステップを完了します。

- ステップ3b.1 - ASMファイル・システムの作成
- ステップ3b.2 - ファイル・システムの作成
- ステップ3b.3 - Cluster Ready Services (CRS)リソースの作成
- ステップ3b.4 - 現在構成されているACFSファイル・システムの検証
- ステップ3b.5 - ACFSリソースの起動とステータスの確認
- ステップ3b.6- GoldenGate ACFSディレクトリの作成

ステップ3b.1 - ASMファイル・システムの作成

grid OSユーザーとして、asmcmdを使用してボリュームを作成します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ asmcmd volcreate -G DATAC1 -s 1200G ACFS_GG
```

ノート:

決定したサイズ要件に応じて、ファイル・システムのサイズを変更します。

ステップ3b.2 - ファイル・システムの作成

1. grid OSユーザーとして、asmcmdを使用して"ボリューム・デバイス"を確認します。

```
[grid@exadb-node1 ~]$ asmcmd volinfo -G DATAC1 ACFS_GG
```

次に、ACFSボリューム・デバイスの出力例を示します。

```
Diskgroup Name: DATAC1
 Volume Name: ACFS_GG
 Volume Device: /dev/asm/acfs_gg-151
 State: ENABLED
 Size (MB): 1228800
 Resize Unit (MB): 64
 Redundancy: MIRROR
 Stripe Columns: 8
 Stripe Width (K): 1024
 Usage:
 Mountpath:
```

2. grid OSユーザーとして、次のmkfsコマンドを使用して、ファイル・システムを作成します。

```
[grid@exadb-node1 ~]$ /sbin/mkfs -t acfs /dev/asm/acfs_gg-151
```

ステップ3b.3 - Cluster Ready Services (CRS)リソースの作成

1. opc OSユーザーとして、ACFSマウント・ポイントを作成します。

```
[opc@exadb-node1 ~]$ dcli -l opc -g ~/dbs_group sudo mkdir -p /mnt/acfs_gg
[opc@exadb-node1 ~]$ dcli -l opc -g ~/dbs_group sudo chown
 oracle:oinstall /mnt/acfs_gg
```

2. rootユーザーとして、ファイル・システム・リソースを作成します。

ACFSの分散ファイル・ロックの実装により、DBFSとは異なり、一度に複数のOracle RACノードにACFSをマウントできます。

3. root OSユーザーとして、新しいACFSファイル・システムのためのACFSリソースを作成します。

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -
f2)/bin/srvctl
 add filesystem -device /dev/asm/acfs_gg-151 -volume ACFS_GG -diskgroup DATAC1
 -path /mnt/acfs_gg -user oracle
```

ステップ3b.4 - 現在構成されているACFSファイル・システムの検証

grid OSユーザーとして、次のコマンドを使用して、ファイル・システムの詳細を表示します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ srvctl config filesystem -volume ACFS_GG -diskgroup DATAC1
```

```
Volume device: /dev/asm/acfs_gg-151
Diskgroup name: datac1
Volume name: acfs_gg
Canonical volume device: /dev/asm/acfs_gg-151
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg
Mount point owner: oracle
Mount point group: oinstall
Mount permissions: owner:oracle:rwx,pgrp:oinstall:r-x,other::r-x
Mount users: grid
Type: ACFS
Mount options:
Description:
ACFS file system is enabled
ACFS file system is individually enabled on nodes:
ACFS file system is individually disabled on nodes:
```

ステップ3b.5 - ACFSリソースの起動とステータスの確認

grid OSユーザーとして、次のコマンドを使用して、ファイル・システムを起動および確認します。

```
[grid@exadb-node1 ~]$ srvctl start filesystem -volume ACFS_GG
 -diskgroup DATAC1 -node `hostname`
[grid@exadb-node1 ~]$ srvctl status filesystem -volume ACFS_GG -diskgroup DATAC1
```

ACFSファイル・システム/mnt/acfs_ggは、ノードexadb-node1にマウントされます

作成したCRSリソースには、ora.diskgroup_name.volume_name.acfsという形式を使用した名前が付けられます。前述のファイル・システムの例を使用すると、CRSリソースの名前はora.datac1.acfs_gg.acfsになります。

現在存在しているすべてのACFSファイル・システムのCRSリソースを表示するには、次のコマンドを使用します。

```
[grid@exadb-node1 ~]$ crsctl stat res -w "((TYPE = ora.acfs.type) OR (TYPE =
ora.acfs_cluster.type))"
NAME=ora.datac1.acfs_gg.acfs
TYPE=ora.acfs.type
TARGET=ONLINE , OFFLINE
STATE=ONLINE on exadb-node1, OFFLINE
NAME=ora.datac1.acfsvol01.acfs
TYPE=ora.acfs.type
TARGET=ONLINE , ONLINE
STATE=ONLINE on exadb-node1, ONLINE on exadb-node2
```

ステップ3b.6- GoldenGate ACFSディレクトリの作成

grid OSユーザーとして、Oracle GoldenGateデプロイメントを格納するディレクトリを作成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ mkdir -p /mnt/acfs_gg/deployments
```

ACFSの詳細は、『Oracle Automatic Storage Management Cluster File System管理者ガイド』を参照してください。

ノート:

共有ファイル・システムは、マウントされたままにします。これは、今後のステップで Oracle GoldenGate デプロイメントを作成するために必要です。

タスク4 - Oracle GoldenGateのインストール

Oracle GoldenGate構成の一部となるOracle Exadata Database Service構成のすべてのノードに、Oracle GoldenGateソフトウェアをローカルにインストールします。すべてのノードでインストール・ディレクトリが同一であることを確認します。

このタスクを完了するには、次のステップを実行します。

- ステップ4.1 - ソフトウェアの解凍とインストール用のレスポンス・ファイルの作成
- ステップ4.2 - Oracle GoldenGateのインストール
- ステップ4-3 - Oracle GoldenGateへのパッチ適用

ステップ4.1 - ソフトウェアの解凍とインストール用のレスポンス・ファイルの作成

1. 最初のデータベース・ノードのoracle OSユーザーとして、ソフトウェアを解凍します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ unzip
 /u02/app_acfs/goldengate/213000_fbo_ggs_Linux_x64_Oracle_services_shiphome.zip
 -d /u02/app_acfs/goldengate
```

このソフトウェアには、Oracle Databaseリリース21c以前のサポートされているリリースのレスポンス・ファイルの例が含まれています。レスポンス・ファイルは、すべてのデータベース・ノードへのOracle GoldenGateのインストールに同じファイルを使用できるように共有ファイル・システムにコピーして、次のパラメータを編集します。

- INSTALL_OPTION=ora21c
- SOFTWARE_LOCATION=/u02/app/oracle/goldengate/gg21c (recommended location)

2. 最初のデータベース・ノードのoracle OSユーザーとして、インストール用のレスポンス・ファイルをコピーおよび編集します。

```
[oracle@exadb-node1 ~]$ cp

/u02/app_acfs/goldengate/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/respo
nse/oggcore.rsp
 /u02/app_acfs/goldengate
[oracle@exadb-node1 ~]$ vi /u02/app_acfs/goldengate/oggcore.rsp
# Before edit
INSTALL_OPTION=
SOFTWARE_LOCATION=
# After edit
INSTALL_OPTION=ora21c
SOFTWARE_LOCATION=/u02/app/oracle/goldengate/gg21c
```

ステップ4.2 - Oracle GoldenGateのインストール

すべてのデータベース・ノードのoracle OSユーザーとして、Oracle GoldenGateをインストールします。

```
[oracle@exadb-node1 ~]$ cd
 /u02/app_acfs/goldengate/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1/
[oracle@exadb-node1 ~]$ ./runInstaller -silent -nowait
 -responseFile /u02/app_acfs/goldengate/oggcore.rsp
Starting Oracle Universal Installer...
Checking Temp space: must be greater than 120 MB. Actual 32755 MB Passed
Checking swap space: must be greater than 150 MB. Actual 16383 MB Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_02-54-51PM. Please wait ...
You can find the log of this install session at:
/u01/app/oraInventory/logs/installActions2022-07-08_02-54-51PM.log
```

```
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check '/u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log'
 for more details.
[oracle@exadb-node1 ~]$ cat
 /u01/app/oraInventory/logs/silentInstall2022-07-08_02-54-51PM.log
The installation of Oracle GoldenGate Services was successful.
[oracle@exadb-node1 ~]$ ssh exadb-node2
[oracle@exadb-node2 ~]$ cd
 /u02/app_acfs/goldengate/fbo_ggs_Linux_x64_Oracle_services_shiphome/Disk1
[oracle@exadb-node2 ~]$ ./runInstaller -silent -nowait
 -responseFile /u02/app_acfs/goldengate/oggcore.rsp
Starting Oracle Universal Installer...
Checking Temp space: must be greater than 120 MB. Actual 32755 MB Passed
Checking swap space: must be greater than 150 MB. Actual 16383 MB Passed
Preparing to launch Oracle Universal Installer from
 /tmp/OraInstall2022-07-08_03-54-51PM. Please wait ...
You can find the log of this install session at:
/u01/app/oraInventory/logs/installActions2022-07-08_03-54-51PM.log
Successfully Setup Software.
The installation of Oracle GoldenGate Services was successful.
Please check '/u01/app/oraInventory/logs/silentInstall2022-07-08_03-54-51PM.log'
 for more details.
[oracle@exadb-node1 ~]$ cat
 /u01/app/oraInventory/logs/silentInstall2022-07-08_03-54-51PM.log
The installation of Oracle GoldenGate Services was successful.
```

Oracle Goldengateへのパッチ適用

すべてのデータベース・ノードでoracle OSユーザーとして、最新のOPatchをインストールします。

```
[oracle@exadb-node1 ~]$ unzip -oq -d /u01/app/oracle/goldengate/gg21c
 /u02/app_acfs/goldengate /p6880880_210000_Linux-x86-64.zip
[oracle@exadb-node1 ~]$ cat >> ~/.bashrc <<EOF
export ORACLE_HOME=/u01/app/oracle/goldengate/gg21c
export PATH=$ORACLE_HOME/OPatch:$PATH
EOF
[oracle@exadb-node1 ~]$. ~/.bashrc
[oracle@exadb-node1 ~]$ opatch lsinventory |grep 'Oracle GoldenGate Services'
Oracle GoldenGate Services                                        21.1.0.0.0
[oracle@gghub_prim1 Disk1]$ opatch version
OPatch Version: 12.2.0.1.37
```

oracle OSユーザーとして、OPatchのprereqを実行して、パッチの適用前に競合があるかどうかを検証します。

```
[oracle@exadb-node1 ~]$ unzip -oq -d /u02/app_acfs/goldengate
 /u02/app_acfs/goldengate /p35214851_219000OGGRU_Linux-x86-64.zip
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate/35214851/
[oracle@exadb-node1 35214851]$ opatch prereq CheckConflictAgainstOHWithDetail -ph ./
Oracle Interim Patch Installer version 12.2.0.1.26
Copyright (c) 2023, Oracle Corporation.  All rights reserved.
PREREQ session
Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.26
OUI version       : 12.2.0.9.0
Log file location : /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/opatch2023-
04-21_13-44-16PM_1.log
Invoking prereq "checkconflictagainstohwithdetail"
Prereq "checkConflictAgainstOHWithDetail" passed.
```

すべてのデータベース・ノードでoracle OSユーザーとして、OPatchを使用してOracle GoldenGate Microservices Architechtureにパッチを適用します。

```
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate/35214851/
[oracle@exadb-node1 35214851]$ opatch apply
Oracle Interim Patch Installer version 12.2.0.1.37
Copyright (c) 2023, Oracle Corporation.  All rights reserved.
Oracle Home       : /u01/app/oracle/goldengate/gg21c
Central Inventory : /u01/app/oraInventory
   from           : /u01/app/oracle/goldengate/gg21c/oraInst.loc
OPatch version    : 12.2.0.1.37
OUI version       : 12.2.0.9.0
Log file location : /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/opatch2023-
04-21_19-40-41PM_1.log
Verifying environment and performing prerequisite checks...
OPatch continues with these patches:   35214851
Do you want to proceed? [y|n]
y
User Responded with: Y
All checks passed.
Please shutdown Oracle instances running out of this ORACLE_HOME on the local system.
(Oracle Home = '/u01/app/oracle/goldengate/gg21c')
Is the local system ready for patching? [y|n]
y
User Responded with: Y
Backing up files...
Applying interim patch '35214851' to OH '/u01/app/oracle/goldengate/gg21c'
Patching component oracle.oggcore.services.ora21c, 21.1.0.0.0...
Patch 35214851 successfully applied.
Log file location: /u01/app/oracle/goldengate/gg21c/cfgtoollogs/opatch/opatch2023-04-
21_19-40-41PM_1.log
OPatch succeeded.
[oracle@exadb-node1 35214851]$ opatch lspatches
35214851;
```

タスク5 - Oracle GoldenGateデプロイメントの作成

Oracle GoldenGateソフトウェアがインストール済の場合、次のステップは、Oracle GoldenGate Configuration Assistantを使用してOracle GoldenGateデプロイメントを作成するためのレスポンス・ファイルを作成することです。

このタスクを完了するには、次のステップを実行します。

- ステップ5.1 - レスポンス・ファイルの作成
- ステップ5.2 - GoldenGateデプロイメントの作成
- ステップ5.3 - (DBFSを使用している場合のみ) GoldenGateデプロイメントの一時ディレクトリの移動

ステップ5.1 - レスポンス・ファイルの作成

サイレント構成の場合、`oracle` OSユーザーとしてレスポンス・ファイル`oggca.rsp`を作成および編集し、Oracle GoldenGateデプロイメントを作成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ vi /u02/app_acfs/goldengate/oggca.rsp
oracle.install.responseFileersion=/oracle/install/rspfmt_oggca_response_schema_v21_1_
0
CONFIGURATION_OPTION=ADD
DEPLOYMENT_NAME=<ggNN>
ADMINISTRATOR_USER=oggadmin
ADMINISTRATOR_PASSWORD=<password_for_oggadmin>
SERVICEMANAGER_DEPLOYMENT_HOME=<ACFS or DBFS mount point>/deployments/<ggsmNN>
HOST_SERVICEMANAGER=localhost
PORT_SERVICEMANAGER=9100
SECURITY_ENABLED=false
STRONG_PWD_POLICY_ENABLED=true
CREATE_NEW_SERVICEMANAGER=true
REGISTER_SERVICEMANAGER_AS_A_SERVICE=false
INTEGRATE_SERVICEMANAGER_WITH_XAG=true
EXISTING_SERVICEMANAGER_IS_XAG_ENABLED=false
OGG_SOFTWARE_HOME=/u02/app/oracle/goldengate/gg21c
OGG_DEPLOYMENT_HOME=<ACFS or DBFS mount point>/deployments/<ggNN>
ENV_LD_LIBRARY_PATH=${OGG_HOME}/lib/instantclient:${OGG_HOME}/lib
ENV_TNS_ADMIN=/u02/app/oracle/goldengate/network/admin
FIPS_ENABLED=false
SHARDING_ENABLED=false
ADMINISTRATION_SERVER_ENABLED=true
PORT_ADMINSRVR=9101
DISTRIBUTION_SERVER_ENABLED=true
PORT_DISTSRVR=9102
NON_SECURE_DISTSRVR_CONNECTS_TO_SECURE_RCVRSRVR=false
RECEIVER_SERVER_ENABLED=true
PORT_RCVRSRVR=9103
METRICS_SERVER_ENABLED=true
METRICS_SERVER_IS_CRITICAL=false
PORT_PMSRVR=9104
UDP_PORT_PMSRVR=9105
PMSRVR_DATASTORE_TYPE=BDB
PMSRVR_DATASTORE_HOME=/u02/app/oracle/goldengate/datastores/<instance_name>
OGG_SCHEMA=<goldengate_database_schema>
```

レスポンス・ファイルで、次に示す値を適切に編集します。

- `CONFIGURATION_OPTION`
- `DEPLOYMENT_NAME`
- `ADMINISTRATOR_USER`
- `SERVICEMANAGER_DEPLOYMENT_HOME`
- `OGG_SOFTWARE_HOME`

- OGG_DEPLOYMENT_HOME
- ENV_TNS_ADMIN
- OGG_SCHEMA

ステップ5.2 - GoldenGateデプロイメントの作成

最初のデータベース・ノードのoracle OSユーザーとして、oggca.shを実行し、Oracle GoldenGateデプロイメントを作成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ export OGG_HOME=/u02/app/oracle/goldengate/gg21c
[oracle@exadb-node1 ~]$ $OGG_HOME/bin/oggca.sh -silent
 -responseFile /u02/app_acfs/goldengate/oggca.rsp
Successfully Setup Software.
```

ステップ5.3 - (DBFSを使用している場合のみ) GoldenGateデプロイメントの一時ディレクトリの移動

デプロイメントの作成後、共有ファイル・システムにDBFSを使用する場合は、次のコマンドを実行してGoldenGateデプロイメントの一時ディレクトリをDBFSからローカル・ストレージに移動します。

1. 最初のデータベース・ノードのoracle OSユーザーとして、GoldenGateデプロイメントの一時ディレクトリをローカル・ストレージに移動します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/deployments/<instance_name>
[oracle@exadb-node1 ~]$ mv
 /mnt/dbfs/goldengate/deployments/<instance_name>/var/temp
 /u02/app/oracle/goldengate/datastores/<instance_name>
[oracle@exadb-node1 ~]$ ln -s
 /u02/app/oracle/goldengate/deployments/<instance_name>/temp
 /mnt/dbfs/goldengate/deployments/<instance_name>/var/temp
```

2. 残りのデータベース・ノードのoracle OSユーザーとして、ローカル記憶域にディレクトリを作成します。

```
[oracle@exadb-node2 ~]$ mkdir
 /u02/app/oracle/goldengate/deployments/<instance_name>
```

タスク6 ネットワークの構成

ネットワークの構成方法はExadataプラットフォームによって異なります。ステップ6aで説明されている最初のメソッドはExaDB-Dのみに適用され、ステップ6bで説明されている2番目のメソッドはExaDB-C@Cのみに適用されます。

このタスクを完了するには、次のステップの1つを実行します。

- ステップ6a - (ExaDB-Dのみ) Oracle Cloud Infrastructure Networkingの構成
- ステップ6b - (ExaDB-C@Cのみ) アプリケーションの仮想IPアドレス作成の準備

ステップ6a - (ExaDB-Dのみ) Oracle Cloud Infrastructure Networkingの構成

Oracle GoldenGateが正しく機能するように、プライベートDNSゾーン、VIP、要塞、セキュリティ・リスト、ファイアウォールなどの仮想クラウド・ネットワーク(VCN)のコンポーネントを構成する必要があります。

VCNとセキュリティ・リストの詳細と作成手順は、Oracle Cloud Infrastructure Networkingを参照してください。

次のサブステップを実行して、このステップを完了します。

- ステップ6a.1 - プライベートIPを使用したGoldenGate Microservices Webインタフェースへの接続
- ステップ6a.2 - アプリケーションの仮想IPアドレス(VIP)の作成
- ステップ6a.3 - イングレス・ルールの追加
- ステップ6a.4 - ファイアウォールでポート443を開く
- ステップ6a.5 - ソースVIPとターゲットVIPの接続
- ステップ6a.5 - GoldenGateソースとターゲット間のネットワーク接続の構成
- ステップ6a.6 - プライベートDNSゾーンのビューとリゾルバの構成

ステップ6a.1 - プライベートIPを使用したGoldenGate Microservices Webインタフェースへの接続

GoldenGate Microservices Webインタフェースには、OCIネットワーク内から、またはOCIリソースへのアクセスを保護する要塞ホストを介して、プライベート・エンドポイントを使用する方法でのみアクセスできます。

リージョンでOCI Bastionサービスを使用できない場合は、OCIコンピュート・インスタンスを要塞として使用できます。OCI Bastion As A Serviceの手順に従って要塞を作成します。Oracle GoldenGate Microservicesが実行されているリージョンごとに、1つの要塞が必要です。

> ノート:
>
> 要塞を作成した後やコンピュート・インスタンスを要塞として使用した後は、`https://localhost:local_port` を使用して Oracle GoldenGate Microservices に接続するための、SSH ポート転送セッションを作成する必要があります。

ステップ6a.2 - アプリケーションの仮想IPアドレス(VIP)の作成

専用のアプリケーションVIPは、どのOracle RACノードがサービスをホストしていても、同じホスト名を使用してOracle GoldenGate Microservicesにアクセスできるようにするために必要です。また、アプリケーションVIPは、Oracle GoldenGate Distribution Serverが現在のOracle RACノードを実行しているDistribution Receiverと通信できるようにします。

VIPは、Oracle Clusterwareが管理するクラスタ・リソースです。VIPはデータベース・ノードに割り当てられ、ノード障害が発

生すると自動的に別のノードに移行されます。

コンソールを使用して、VIPをOracle Exadata Database Serviceに割り当てます。

1. ナビゲーション・メニューを開きます。「Oracle Database」をクリックして、「Oracle Public Cloud上のExadata」をクリックします。
2. コンパートメントを選択します。
3. Oracle Exadata Database Service on Dedicated Infrastructureで、「Exadata VMクラスタ」をクリックします。
4. 新しいVIPを作成するExadata VMクラスタに移動します。
5. 「リソース」で、「仮想IPアドレス」をクリックします。
6. 「仮想IPアドレスのアタッチ」をクリックします。
7. 「仮想IPアドレスのアタッチ」ダイアログで、次の必須情報を入力します。
    - サブネット: クライアントのサブネット
    - 仮想IPアドレス・ホスト名: 「DNS名のスキャン」を使用して、Oracle GoldenGateのSCANワードを置き換えます(例: `exadb-xxxx-ggN`)。
8. 「作成」をクリックします。

仮想IPアドレスの作成が完了すると、ステータスが「プロビジョニング中」から「使用可能」に変わり、割り当てられたIPが仮想IPアドレスに表示されます。完全修飾ドメイン名をメモしておきます。これは、ソースをターゲットOracle GoldenGateデプロイメントに接続するために必要なホスト名です。

ノート:

ほとんどのテナンシで新しい VIP を追加できます。問題がある場合はサービス・リクエストを記録してください。

ステップ6a.3 - イングレス・ルールの追加

コンソールを使用してイングレス・ポート443を開き、NGINXをリバース・プロキシとして使用して、Oracle GoldenGateサービスを接続します。詳細は、セキュリティ・リストの使用を参照してください。

セキュリティ・リストの更新後、リストには次のような値のエントリが登録されます。

- ソース・タイプ: CIDR
- ソースCIDR: 0.0.0.0/0
- IPプロトコル: TCP
- ソース・ポート範囲: All
- 宛先ポート範囲: 443
- 次のポートに対してTCPトラフィックを許可: 443 HTTPS
- 説明: Oracle GoldenGate 443

ステップ6a.4 - ファイアウォールでポート443を開く

opc OSユーザーとして、チェーンが現在トラフィックを受け入れるように設定されているかどうかを検証します。

```
[opc@exadb-node1 ~]$ sudo iptables --list |grep policy

Chain INPUT (policy ACCEPT)
Chain FORWARD (policy ACCEPT)
Chain OUTPUT (policy ACCEPT)
```

ポリシーがACCEPTの場合は、このステップをスキップしてタスク7に進むことができます。それ以外の場合は、ネットワーク管理者に連絡して、イングレス・アクティビティ用にポート443を開くようにファイアウォールを更新してください。

ステップ6a.5 - GoldenGateソースとターゲット間のネットワーク接続の構成

必要に応じて、インターネットにアクセスするようにVCNを設定できます。VCNをオブジェクト・ストレージなどのパブリックOracle Cloud Infrastructureサービス、オンプレミス・ネットワークまたは別のVCNにプライベートで接続することもできます。

サブネットがパブリックとプライベートのどちらであるかの詳細や接続の作成手順は、Oracle Cloud Infrastructure Networkingドキュメントの接続の選択肢を参照してください。

ステップ6a.6 - プライベートDNSゾーンのビューとリゾルバの構成

ソースとターゲットのOracle GoldenGateデプロイメントが異なるリージョンにある場合は、プライベート・ゾーンを使用して、ソース・リージョンにプライベートDNSビューを作成する必要があります。これは、ソースのOracle GoldenGate分散パスがターゲットのOracle GoldenGateデプロイメントVIPホスト名に到達するために必要です。

「プライベートDNSゾーンのビューとリゾルバの構成」のステップに従って、プライベートDNSビューとゾーンを作成します。

ソース・システムのopc OSユーザーとして、コマンドnslookupを使用して、ターゲットのOracle GoldenGateデプロイメントの完全修飾ドメイン名(ステップ6.2から)を解決します。

```
[opc@exadb-node1 ~]$ nslookup <target_vip_fully_qualified_domain_name>
Server: <DNS_IP>
Address: <DNS_IP>#53
Non-authoritative answer:
Name: <target_vip_fully_qualified_domain_name>
Address: <target_vip_ip>
```

ステップ6b - (ExaDB-C@Cのみ) アプリケーションの仮想IPアドレス作成の準備

専用のアプリケーションVIPは、どのOracle RACノードがサービスをホストしていても、同じホスト名を使用してOracle GoldenGate Microservicesにアクセスできるようにするために必要です。また、アプリケーションVIPは、Oracle GoldenGate Distribution Serverが現在のOracle RACノードを実行しているDistribution Receiverと通信できるようにします。

VIPは、Oracle Clusterwareが管理するクラスタ・リソースです。VIPはデータベース・ノードに割り当てられ、ノード障害が発生すると自動的に別のノードに移行されます。

システム管理者は、新しいアプリケーションVIPのIPアドレスを指定する必要があります。このIPアドレスは、前述のシステム環境と同じサブネット内にある必要があります。

VIPは、Oracle Grid Infrastructure Agentを構成するときに、次のタスクで作成されます。

タスク7 - Oracle Grid Infrastructure Agentの構成

次の手順では、Oracle Grid Infrastructureスタンドアロン・エージェント(XAG)を使用してOracle GoldenGateを管理するようにOracle Clusterwareを構成する方法を示します。

XAGを使用することで、Oracle RACノード間での再配置時に、共有ファイル・システム(DBFSまたはACFS)のマウントとOracle GoldenGateデプロイメントの停止および起動を自動化します。

このタスクを完了するには、次のステップを実行します。

- ステップ7.1 - Oracle Grid Infrastructureスタンドアロン・エージェントのインストール
- ステップ7.2 - Oracle Grid Infrastructure Agentの構成
- ステップ7.2 - Oracle GoldenGateデプロイメントの起動

ステップ7.1 - Oracle Grid Infrastructureスタンドアロン・エージェントのインストール

XAGソフトウェアは、スタンドアロン・エージェントとしてGrid Infrastructure `ORACLE_HOME`の外部にインストールすることをお薦めします。これにより、入手可能な最新のXAGリリースを使用できるようになり、Grid Infrastructureに影響を与えることなくソフトウェアを更新できます。

XAGは、Oracle GoldenGateがインストールされているシステム内のすべてのOracle RACデータベース・ノードで同じディレクトリにインストールする必要があります。

1. 最初のデータベース・ノードの`grid` OSユーザーとして、ソフトウェアを解凍し、`sagsetup.sh`を実行します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ unzip
/u02/app_acfs/goldengate/p31215432_190000_Generic.zip
 -d /u02/app_acfs/goldengate
[grid@exadb-node1 ~]$ /u02/app_acfs/goldengate/xag/xagsetup.sh --install
 --directory /u01/app/grid/xag --all_nodes
Installing Oracle Grid Infrastructure Agents on: exadb-node1
Installing Oracle Grid Infrastructure Agents on: exadb-node2
Updating XAG resources.
Successfully updated XAG resources.
```

2. 新しくインストールされたXAGソフトウェアの場所をPATH変数に追加して、`grid`ユーザーがマシンにログオンしたときに`agctl`の場所がわかるようにします。

```
[grid@exadb-node1 ~]$ grep PATH ~/.bashrc
PATH=
/u01/app/grid/xag/bin:/sbin:/bin:/usr/sbin:/usr/bin:/u01/app/19.0.0.0/grid/bin:
/u01/app/19.0.0.0/grid/OPatch;
 export PATH
```

ノート:

適切な `agctl` バイナリが見つかるように、Grid Infrastructure の `bin` ディレクトリの前に XAG の `bin` ディレクトリが指定されていることを必ず確認してください。これは、Bash シェル使用時の `.bashrc` ファイルなど、ログオン時に有効になる `grid` ユーザー環境で設定する必要があります。

ステップ7.2 - Oracle Grid Infrastructure Agentの構成

次の手順では、Oracle Grid Infrastructureスタンドアロン・エージェント(XAG)を使用してOracle GoldenGateを管理

するようにOracle Clusterwareを構成する方法を示します。

XAGを使用することで、Oracle RACノード間での再配置時に、共有ファイル・システム(DBFSまたはACFS)のマウントとOracle GoldenGateデプロイメントの停止および起動を自動化します。

Oracle GoldenGateは、データベースの起動時およびファイル・システムのマウント時に自動的にデプロイメントが起動および停止されるように、XAGに登録しておく必要があります。

Oracle GoldenGate Microservices ArchitectureをXAGに登録するには、次のコマンド形式を使用します。

```
agctl add goldengate <instance_name>
--gg_home <GoldenGate_Home>
--service_manager
--config_home <GoldenGate_SvcMgr_Config>
--var_home <GoldenGate_SvcMgr_Var Dir>
--port <port number>
--oracle_home <$OGG_HOME/lib/instantclient>
--adminuser <OGG admin user>
--user <GG instance user>
--group <GG instance group>
--network <network_number>
--ip <ip_address>
--vip_name <vip_name>
--filesystems <CRS_resource_name>
--db_services <service_name>
--use_local_services
--attribute START_TIMEOUT=60
--nodes <node1, node2, ... ,nodeN>
```

説明:

- `--gg_home`では、GoldenGateソフトウェアの場所を指定します。
- `--service_manager`では、これがGoldenGate Microservicesインスタンスであることを示します。
- `--config_home`では、GoldenGateサービス・マネージャのデプロイメント構成のホーム・ディレクトリを指定します。
- `--var_home`では、GoldenGateサービス・マネージャのデプロイメント変数のホーム・ディレクトリを指定します。
- `--port`では、デプロイメント・サービス・マネージャのポート番号を指定します。
- `--oracle_home`では、Oracle GoldenGate 21c以降のリリースの一部として含まれるOracleデータベース・ライブラリの場所を指定します。

  例: `$OGG_HOME/lib/instantclient`

- `--adminuser`では、Oracle GoldenGate Microservices管理者アカウント名を指定します。
- `--user`では、Oracle GoldenGateデプロイメント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します。
- `--group`では、Oracle GoldenGateデプロイメントを所有するオペレーティング・システム・グループの名前を指定します。
- `--network`では、VIPのネットワーク・サブネットを指定します。
- `--ip`では、VIPのIPアドレスを指定します。

  すでにVIPを作成している場合は、`--network`と`--ip`のかわりに`--vip_name vip_name`パラメータを使用して指定します。

- `--vip_name`では、以前に作成したアプリケーションVIPのCRSリソース名を指定します。

  このパラメータは、`--network`と`--ip` (オプション)を置き換えます。

- `--filesystems`では、デプロイメントの開始前にマウントする必要があるDBFSまたはACFS CRSファイル・システ

ム・リソースを指定します。

- `--db_services`では、前のステップで作成した`ora.<database>.<service_name>.svc`サービス名を指定します。

  Oracleマルチテナント・データベースを使用している場合、ReplicatにはPDBデータベース・サービス、ExtractにはCDBデータベース・サービスを指定します。ReplicatとExtractを使用する場合は、両方のサービス名をカンマで区切って指定します。

- `--use_local_services`では、Oracle GoldenGateインスタンスが、`db_services`サービスが実行されているOracle RACノードと同じノードに共存している必要があることを指定します。
- `--attribute name=value`は、適用可能な属性を指定します。

  属性`START_TIMEOUT=60`を変更して、データベースがクラッシュして再起動した後のブラックアウトを最適化することをお薦めします。

- `--nodes`では、このGoldenGateインスタンスを実行できるOracle RACノードを指定します。

  Oracle GoldenGateがクラスタ内のいずれかのOracle RACノードで実行されるように構成されている場合は、このパラメータを使用して、Oracle GoldenGateを実行するノードの優先順序を決定する必要があります。

このタスクを完了するには、次のステップの1つを実行します。

- ステップ7.2a - DBFSでのGoldenGateデプロイメント
- ステップ7.2b - ACFSでのGoldenGateデプロイメント

ステップ7.2a - DBFSでのGoldenGateデプロイメント

1. 最初のデータベース・ノードで`grid` OSユーザーとして、次に示すコマンドを実行してネットワーク番号を識別します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ srvctl config network
Network 1 exists
Subnet IPv4: 10.1.0.0/255.255.255.0/bondeth0, static
Subnet IPv6:
Ping Targets: 10.1.0.1
Network is enabled
Network is individually enabled on nodes:
Network is individually disabled on nodes:
```

2. 最初のデータベース・ノードの`root` OSユーザーとして、次のコマンド形式を使用して、XAGにOracle GoldenGate Microservices Architectureを登録します。

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# /u01/app/grid/xag/bin/agctl add goldengate
<instance_name> ¥
--gg_home /u02/app/oracle/goldengate/gg21c   ¥
--service_manager   ¥
--config_home /mnt/dbfs/deployments/ggsm01/etc/conf   ¥
--var_home /mnt/dbfs/deployments/ggsm01/var    ¥
--port 9100   ¥
--oracle_home /u02/app/oracle/goldengate/gg21c/lib/instantclient  ¥
--adminuser oggadmin   ¥
--user oracle   ¥
--group oinstall   ¥
--network 1 --ip <virtual_IP_address>  ¥
--filesystems <dbfs_mount_name>  ¥
--db_services ora.<db_service_name>.svc , ora.<pdb_service_name>.svc   ¥
--use_local_services   ¥
--attribute START_TIMEOUT=60  ¥
--nodes <exadb-node1>, <exadb-node2>
```

```
Enter password for 'oggadmin' : <oggadmin_password>
```

ステップ7.2b - ACFSでのGoldenGateデプロイメント

1. 最初のデータベース・ノードでgrid OSユーザーとして、次に示すコマンドを実行してネットワーク番号を識別します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ srvctl config network
Network 1 exists
Subnet IPv4: 10.1.0.0/255.255.255.0/bondeth0, static
Subnet IPv6:
Ping Targets: 10.1.0.1
Network is enabled
Network is individually enabled on nodes:
Network is individually disabled on nodes:
```

2. 最初のデータベース・ノードのroot OSユーザーとして、次のコマンド形式を使用して、XAGにOracle GoldenGate Microservices Architectureを登録します。

```
[root@exadb-node1 ~]# /u01/app/grid/xag/bin/agctl add goldengate
<instance_name>¥
--gg_home /u02/app/oracle/goldengate/gg21c  ¥
--service_manager  ¥
--config_home /mnt/acfs_gg/deployments/ggsm01/etc/conf   ¥
--var_home /mnt/acfs_gg/deployments/ggsm01/var   ¥
--port 9100  ¥
--oracle_home /u02/app/oracle/goldengate/gg21c/lib/instantclient ¥
--adminuser oggadmin  ¥
--user oracle  ¥
--group oinstall  ¥
--network 1 --ip <virtual_IP_address>  ¥
--filesystems ora.<acfs_name>.acfs   ¥
--db_services ora.<db_service_name>.svc      ¥
--use_local_services   ¥
--attribute START_TIMEOUT=60  ¥
--nodes <exadb-node1>,<exadb-node2>
```

ステップ7.3 - Oracle GoldenGateデプロイメントの起動

次に、XAGでGoldenGateデプロイメントを管理するために使用するいくつかのagctlコマンドの例を示します。

1. grid OSユーザーとして、次に示すコマンドを実行して、Oracle GoldenGateデプロイメントを開始します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ agctl start goldengate <instance_name>
```

2. grid OSユーザーとして、次に示すコマンドを実行して、Oracle GoldenGateデプロイメントのステータスを確認します。

```
[grid@exadb-node1 ~]$ agctl status goldengate
Goldengate instance <instance_name> is running on exadb-node1
```

3. grid OSユーザーとして、次のコマンドを実行してOracle GoldenGateリソースの構成パラメータを表示します。

```
[grid@exadb-node1 ~]$ agctl config goldengate <instance_name>
Instance name: <instance_name>
Application GoldenGate location is: /u02/app/oracle/goldengate/gg21c_MS
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
 /mnt/acfs_gg/deployments/ggsm01/etc/conf
Goldengate Service Manager var directory: /mnt/acfs_gg/deployments/ggsm01/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: no
```

```
Configured to run on Nodes: exadb-node1 exadb-node2
ORACLE_HOME location is: /u02/app/oracle/goldengate/gg21c/lib/instantclient
Database Services needed: ora.<db_unique_name>.<service_name>.svc
[use_local_services]
File System resources needed: ora.datac1.acfs_gg.acfs
Network: 1, IP:NN.NN.NN.NN, User:oracle, Group:oinstall
```

Oracle Grid Infrastructure Bundled Agentの詳細は、Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 11gリリース2、12c、18cおよび19cを参照してください。

タスク8 - NGINXリバース・プロキシの構成

GoldenGateのリバース・プロキシ機能を使用すると、Oracle GoldenGateデプロイメントに関連付けられたすべてのOracle GoldenGate Microservicesに対応する単一のアクセス先を使用できます。

リバース・プロキシを使用しないと、Oracle GoldenGateデプロイメント・マイクロサービスは、ホスト名またはIPアドレスと個別のポート番号で構成されるサービスごとに1つずつのURLを使用してアクセスされます。

たとえば、サービス・マネージャにアクセスするためにhttp://gghub.example.com:9100を使用し、管理サーバーにはhttp://gghub.example.com:9101、2番目のサービス・マネージャにはhttp://gghub.example.com:9110を使用してアクセスするようになります。

Grid Infrastructure Agent (XAG)を使用してOracle Exadata Database Serviceで高可用性(HA)構成のOracle GoldenGateを実行する場合、GoldenGateサービス・マネージャでは複数のデプロイメントを管理できないとう制限があります。この制限のため、サービス・マネージャとデプロイメントのペアごとに、個別の仮想IPアドレス(VIP)を作成するようにお薦めします。このようにすると、VIPを使用してマイクロサービスに直接アクセスできます。

リバース・プロキシにより、ポート番号はデプロイメント名とホスト名のVIPに置き換えられるため、マイクロサービスに接続するためのポート番号は不要になります。たとえば、Webブラウザでコンソールに接続するには、次のURLを使用します。

サービス	URL
サービス・マネージャ	https://localhost:localPort
管理サーバー	https://localhost:localPort/instance_name/adminsrvr
分散サーバー	https://localhost:localPort/instance_name/distsrvr
パフォーマンス・メトリック・サーバー	https://localhost:localPort/instance_name/pmsrvr
レシーバ・サーバー	https://localhost:localPort/instance_name/recvsrvr

ノート:

OCI で Oracle GoldenGate に接続する場合は、要塞と SSH ポート転送セッションを作成する必要があります(ステップ 6.1 を参照)。その後で、https://locahost:<localPort>を使用すると、Oracle GoldenGate サービスに接続できます。

リバース・プロキシは、マイクロサービスに簡単にアクセスし、セキュリティと管理性を強化するために必須です。

手順に従って、SSL接続のNGINXリバース・プロキシをインストールおよび構成し、すべての外部通信をセキュアにします。

ノート:

NGINX で CA 署名付き証明書を使用する場合は、NGINX `ssl_certificate` パラメータが、CA 署名付

き証明書、中間証明書、ルート証明書という正しい順序で証明書が含まれている証明書ファイルを指していることを確認します。

このタスクを完了するには、次のステップを実行します。

- ステップ8.1 - NGINXのインストール
- ステップ8.2 - NGINXリバース・プロキシの構成
- ステップ8.3 - GoldenGate Microservicesの保護によるセキュアでない直接アクセスの制限
- ステップ8.4 - NGINXを管理するClusterwareリソースの作成

ステップ8.1 - NGINXリバース・プロキシ・サーバーのインストール

1. すべてのノードでroot OSユーザーとして、次の内容でファイル/etc/yum.repos.d/nginx.repoを作成することでYUMリポジトリを設定します。

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# cat > /etc/yum.repos.d/nginx.repo <<EOF
[nginx-stable]
name=nginx stable repo
baseurl=http://nginx.org/packages/rhel/7/¥$basearch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key
module_hotfixes=true
EOF
```

2. root OSユーザーとして、次のコマンドを実行して、NGINXをインストール、有効化および起動します。

```
[root@exadb-node1 ~]# yum install -y python-requests python-urllib3 nginx
[root@exadb-node1 ~]# systemctl enable nginx
```

3. root OSユーザーとして、ソフトウェアのインストール後に、NGINXリポジトリを無効にします。

```
[root@exadb-node1 ~]# yum-config-manager --disable nginx-stable
```

ステップ8.2 - NGINXリバース・プロキシの構成

Oracle GoldenGateホームごとに、個別のリバース・プロキシ構成が必要です。

複数のサービス・マネージャを実行する場合は、次の手順によって、サービス・マネージャごとに個別のVIPを使用する構成を用意します。NGINXは、VIPを使用してHTTPS接続リクエストのルーティング先になるサービス・マネージャを決定します。

SSL証明書は、クライアントがNGINX経由で接続するサーバーを認証するために必要です。続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

ノート:

CA 署名付き証明書の共通名は、NGINX で使用されるターゲット・ホスト名/VIP と一致している必要があります。

次のサブステップを実行して、このステップを完了します。

- ステップ8.2.1 - NGINX構成ファイルの作成
- ステップ8.2.2 - NGINX構成ファイルの変更
- ステップ8.2.3 - NGINXのサーバー証明書のインストール
- ステップ8.2.4 - NGINX構成ファイルのインストール

- ステップ8.2.5 - 新しいNGINX構成のテスト
- ステップ8.2.6 - NGINXと新しい構成のリロード
- ステップ8.2.7 - GoldenGate Microservicesの接続のテスト
- ステップ8.2.8 - GoldenGate NGINX構成ファイルの配布

ステップ8.2.1 - NGINX構成ファイルの作成

Oracle GoldenGate Microservices Architectureは、リバース・プロキスを使用するように構成できます。Oracle GoldenGate Microservices Architectureには、ReverseProxySettingsというスクリプトが含まれています。このスクリプトによって、NGINXリバース・プロキシ・サーバーのみの構成ファイルを作成します。

このスクリプトには次のパラメータが必要です。

- --userパラメータは、初期デプロイメントの作成で指定したGoldenGate管理者アカウントの写しにする必要があります。
- GoldenGate管理者パスワードの入力が求められます。
- --portパラメータで指定したリバース・プロキシ・ポート番号は、デフォルトのHTTPSポート番号(443)にする必要があります。ただし、同じ--hostを使用して複数のGoldenGateサービス・マネージャを実行している場合を除きます。その場合は、前のサービス・マネージャ・リバース・プロキシ構成と競合しないHTTPSポート番号を指定します。

  たとえば、同じホスト名/VIPを使用して2つのサービス・マネージャを実行する場合、最初のリバース・プロキシ構成は--port 443 --host hostvip01で作成し、2番目は--port 444 --host hostvip01で作成します。

  個別のホスト名/VIPを使用する場合は、--port 443 --host hostvip01と--port 443 --host hostvip02を使用して、2つのサービス・マネージャ・リバース・プロキシ構成を作成します。

- 最後に、HTTPポート番号(9100)は、デプロイメントの作成時に指定したサービス・マネージャのポート番号と一致する必要があります。

このステップは、追加のGoldenGateサービス・マネージャごとに繰り返します。

oracle OSユーザーとして、次のコマンドを使用して、Oracle GoldenGate NGINX構成ファイルを作成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ export OGG_HOME=/u02/app/oracle/goldengate/gg21c
[oracle@exadb-node1 ~]$ export PATH=$PATH:$OGG_HOME/bin
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate
[oracle@exadb-node1 ~]$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings
 --user oggadmin --port 443 --output ogg_<instance_name>.conf http://localhost:9100
 --host <VIP hostname/IP>
Password: <oggadmin_password>
```

ステップ8.2.2 - NGINX構成ファイルの変更

複数のGoldenGateサービス・マネージャが同じHTTPS 443ポートのIP/VIPを使用するように構成されている場合は、前のステップで生成したNGINXリバース・プロキシ構成ファイルに小さな変更が必要です。

同じポート番号を共有するすべてのサービス・マネージャは、--hostパラメータで指定したそれぞれのVIP/IPを使用して個別にアクセスされます。

1. oracle OSユーザーとして、このサービス・マネージャによって管理されるデプロイメント名を調べます。まだわからない場合、デプロイメント名はリバース・プロキシ構成ファイルにリストされています。

   ```
   [opc@exadb-node1 ~]$ sudo su - oracle
   [oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate
   [oracle@exadb-node1 ~]$ grep "Upstream Servers" ogg_<instance_name>.conf
   ## Upstream Servers for Deployment '<instance_name>'
   ```

この例では、デプロイメントはSOURCEと呼ばれます。

2. oracle OSユーザーとして、アンダースコアの前にデプロイメント名を追加することで、_ServiceManagerのすべての出現箇所を変更します。

```
$ sed -i 's/_ServiceManager/<instance_name>_ServiceManager/'
        ogg_<instance_name>.conf
```

ステップ8.2.3 - NGINXのサーバー証明書のインストール

1. root OSユーザーとして、ファイル権限400 (-r--------)のrootが所有する/etc/nginx/sslディレクトリでサーバー証明書とキー・ファイルをコピーします。

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# mkdir /etc/nginx/ssl
[root@exadb-node1 ~]# chmod 400 /etc/nginx/ssl
```

2. root OSユーザーとして、ステップ8.2.1で生成したリバース・プロキシ構成ファイルごとに、証明書とキー・ファイルの正しいファイル名を設定します。

```
[oracle@exadb-node1 ~]$ vi /u02/app_acfs/goldengate/ogg_<instance_name>.conf
# Before
    ssl_certificate         /etc/nginx/ogg.pem;
    ssl_certificate_key     /etc/nginx/ogg.pem;
# After
    ssl_certificate         /etc/nginx/ssl/server.chained.crt;
    ssl_certificate_key     /etc/nginx/ssl/server.key;
```

CA署名付き証明書を使用する場合、ssl_certificate NGINXパラメータで指定した証明書には、1) CA署名付き証明書、2)中間証明書および3)ルート証明書を単一ファイルに含める必要があります。この順序は重要です。そうしていないと、NGINXの起動は失敗して、エラー・メッセージが表示されます。

```
(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key
values mismatch)
```

ルート証明書と中間証明書は、CA署名付き証明書プロバイダからダウンロードできます。

SSL証明書の単一ファイルは、次のコマンド例を使用して生成できます。

```
[root@exadb-node1 ~]# cat
 CA_signed_cert.crt intermediate.crt root.crt > server.chained.crt
```

ssl_certificate_keyファイルは、CA署名付き証明書をリクエストする際に必要になる証明書署名リクエスト(CSR)の作成時に生成されます。

ステップ8.2.4 - NGINX構成ファイルのインストール

root OSユーザーとして、デプロイメント構成ファイル(ステップ8.2.1で複数のファイルを作成した場合はそれらのファイル)を/etc/nginx/conf.dディレクトリにコピーします。

```
[root@exadb-node1 ~]# mv /u02/app_acfs/goldengate/ogg_<instance_name>.conf
 /etc/nginx/conf.d
```

ステップ8.2.5 - 新しいNGINX構成のテスト

root OSユーザーとして、NGINX構成ファイルを検証します。

ファイルにエラーがある場合は、次のコマンドでエラーが報告されます。

```
[root@exadb-node1 ~]# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
```

```
nginx: configuration file /etc/nginx/nginxconf test is successful
```

ステップ8.2.6 - NGINXと新しい構成のリロード

root OSユーザーとして、新しい構成をロードするためにNGINXを再起動します。

```
[root@exadb-node1 ~]# systemctl restart nginx
```

ステップ8.2.7 - GoldenGate Microservicesの接続のテスト

1. root OSユーザーとして、デプロイメントのユーザー名とパスワードが含まれるcurl構成ファイル(access.cfg)を作成します。

```
[root@exadb-node1 ~]# vi access.cfg
user = "oggadmin:<password>"
```

2. root OSユーザーとして、次のコマンドを使用してデプロイメントのヘルスを問い合せます。

```
[root@exadb-node1 ~]# curl -svf
 -K access.cfg https://<VIP hostname/IP>:<port#>/services/v2/config/health
 -XGET && echo -e "¥n*** Success"
```

サンプル出力:

```
{"$schema":"api:standardResponse","links":
[{"rel":"canonical","href":"https://gg-prmy-vip1/services/v2/config/health",
"mediaType":"application/json"},
{"rel":"self","href":"https://gg-prmy-vip1/services/v2/config/health",
"mediaType":"application/json"},{"rel":"describedby",
"href":"https://gg-prmy-vip1/services/ServiceManager/v2/metadata-
catalog/health",
"mediaType":"application/schema+json"}],"messages":[],
"response":{"$schema":"ogg:health","deploymentName":"ServiceManager",
"serviceName":"ServiceManager","started":"2021-12-
09T23:33:03.425Z","healthy":true,
"criticalResources":[{"deploymentName":"SOURCE","name":"adminsrvr","type":"serv
ice",
"status":"running","healthy":true},{"deploymentName":"SOURCE","name":"distsrvr",
"type":"service","status":"running","healthy":true},
{"deploymentName":"SOURCE","name":"recvsrvr","type":"service","status":"running
",
"healthy":true}]}}
*** Success ***
```

3. root OSユーザーとして、デプロイメントのユーザー名とパスワードが含まれるcurl構成ファイル(access.cfg)を削除します。

```
[root@exadb-node1 ~]# rm access.cfg
rm: remove regular file ‘access.cfg’? y
```

ステップ8.2.8 - GoldenGate NGINX構成ファイルの配布

GoldenGateサービス・マネージャ用のすべてのリバース・プロキシ構成ファイルを作成した後に、そのファイルをすべてのデータベース・ノードにコピーする必要があります。

1. opc OSユーザーとして、すべてのデータベース・ノードにNGINX構成ファイルを配布します。

```
[opc@exadb-node1 ~]$ sudo tar fczP nginx_conf.tar
 /etc/nginx/conf.d/ /etc/nginx/ssl/
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc -d /tmp
 -f nginx_conf.tar
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo tar fxzP
 /tmp/nginx_conf.tar
```

2. opc OSユーザーとして、新しい構成ファイルのコピー先にしたすべてのノードで新しいNGINX構成をテストします。

```
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo nginx -t
exadb-node1: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
exadb-node1: nginx: configuration file /etc/nginx/nginx.conf test is successful
exadb-node2: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
exadb-node2: nginx: configuration file /etc/nginx/nginx.conf test is successful
```

3. opc OSユーザーとして、新しい構成をロードするために、すべてのノードでNGINXを再起動します。

```
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo systemctl
 restart nginx
```

ステップ8.3 - GoldenGate Microservicesの保護によるセキュアでない直接アクセスの制限

セキュアでないOracle GoldenGate Microservicesデプロイメントに NGINXリバース・プロキシを構成していると、そのマイクロサービスは構成したマイクロサービスのポート番号を使用してHTTP (非セキュア)のアクセスができる状態が続きます。

たとえば、セキュアでないURLの`http://<vip-name>:9101`を使用して管理サーバーにアクセスできます。

Oracle GoldenGate Microservicesの各サーバー(サービス・マネージャ、`adminserver`、`pmsrvr`、`distsrvr`および`recsrvr`)に対するデフォルトの動作では、構成されたポート番号を使用してすべてのネットワーク・インタフェースでリスニングします。これは、マイクロサービスへのHTTPを使用した直接アクセスを無効にして、NGINX HTTPSを使用したアクセスのみ許可する必要がある、よりセキュアなインストールには望ましくありません。

次のコマンドを使用して、localhostアドレスのみを使用するようにサービス・マネージャとデプロイメント・サービスのリスナー・アドレスを変更します。Oracle GoldenGate Microservicesへのアクセスはlocalhostからのみ許可されるようになり、localhost外部のアクセスはNGINX HTTPSポートを使用してのみ成功するようになります。

ステップ8.3.1 - サービス・マネージャの停止

`grid` OSユーザーとして、サービス・マネージャを停止します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ agctl stop goldengate <instance_name>
[grid@exadb-node1 ~]$ agctl status goldengate
Goldengate  instance '<instance_name>' is not running
```

ステップ8.3.2 - サービス・マネージャ・リスナー・アドレスの変更

`oracle` OSユーザーとして、次のコマンドでリスナー・アドレスを変更します。

変更するサービス・マネージャに、適切なポート番号を使用してください。サーバーの起動に失敗しますが、エラーを無視して次のステップに進みます。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ export OGG_HOME=/u02/app/oracle/goldengate/gg21c
[oracle@exadb-node1 ~]$ export
 OGG_VAR_HOME=<acfs or dbfs mount point>/deployments/ggsm01/var
[oracle@exadb-node1 ~]$ export OGG_ETC_HOME=<acfs or
 dbfs mount point>/deployments/ggsm01/etc
[oracle@exadb-node1 ~]$ $OGG_HOME/bin/ServiceManager
 --prop=/config/network/serviceListeningPort
 --value='{"port":9100,"address":"127.0.0.1"}'
 --type=array --persist --exit
[oracle@exadb-node1 ~]$
```

ステップ8.3.3 - サービス・マネージャとデプロイメントの再起動

`grid` OSユーザーとして、サービス・マネージャとデプロイメントを再起動します。

```
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ agctl start goldengate <instance_name>
[grid@exadb-node1 ~]$ agctl status goldengate
Goldengate  instance '<instance_name>' is running on exadb-node1
```

ステップ8.3.4 - GoldenGate Microservicesリスナー・アドレスの変更

oracle OSユーザーとして、次のコマンドを使用して、サービス・マネージャによって管理されるデプロイメントのすべてのGoldenGateマイクロサービス(adminsrvr、pmsrvr、distsrvr、recvsrvr)のリスニング・アドレスをlocalhostに変更します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ cd /u02/app_acfs/goldengate
[oracle@exadb-node1 ~]$ chmod u+x secureServices.py
[oracle@exadb-node1 ~]$ ./secureServices.py http://localhost:9100 --user oggadmin
Password for 'oggadmin': <oggadmin_password>
*** Securing deployment - ogg_deployment
Current value of "/network/serviceListeningPort" for "<instance_name>/adminsrvr" is
{
    "address": "127.0.0.1",
    "port": 9101
}
Current value of "/network/serviceListeningPort" for "<instance_name>/distsrvr" is
{
    "address": "127.0.0.1",
    "port": 9102
}
Current value of "/network/serviceListeningPort" for "<instance_name>/pmsrvr" is
{
    "address": "127.0.0.1",
    "port": 9104
}
Current value of "/network/serviceListeningPort" for "<instance_name>/recvsrvr" is
{
    "address": "127.0.0.1",
    "port": 9103
}
```

ノート:

単一のデプロイメント(adminsrvr、pmsrvr、distsrvr、recvsrvr)を変更する場合は、フラグ--deployment instance_name を追加します

ステップ8.3.5 - NGINX default.conf構成ファイルの削除

opc OSユーザーとして、/etc/nginx/conf.dに作成されたデフォルト構成ファイル(default.conf)を削除します。

```
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo rm
 -f /etc/nginx/conf.d/default.conf
[opc@exadb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo nginx -s reload
```

ステップ8.4 - NGINXを管理するClusterwareリソースの作成

Oracle Clusterwareは、NGINXリバース・プロキシの起動を制御して、Oracle GoldenGateデプロイメントの起動前に自動的にリバース・プロキシを起動できるようにする必要があります。

1. grid OSユーザーとして、次のコマンドを使用して、基盤となるネットワークCRSリソースに依存するNGINXリソースの作成に必要なネットワークCRSリソース名を取得します。

   ```
   [opc@exadb-node1 ~]$ sudo su - grid
   ```

```
[grid@exadb-node1 ~]$ crsctl stat res -w "TYPE == ora.network.type"|grep NAME
NAME=ora.net1.network
```

2. root OSユーザーとして、次のサンプル・コマンドを使用して、NGINXを管理するClusterwareリソースを作成します。HOSTING_MEMBERSおよびCARDINALITYは環境に合致するように置き換えます。

```
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -
f2)/bin/crsctl
 add resource nginx -type generic_application -attr
 "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
x,user:oracle:rwx',
EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl start
 -f nginx',STOP_PROGRAM='/bin/systemctl stop
 -f nginx',CHECK_PROGRAMS='/bin/systemctl status nginx'
 ,START_DEPENDENCIES='hard(ora.net1.network) pullup(ora.net1.network)',
 STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='<exadb-node1, exadb-node2>',
CARDINALITY=2"
```

この例で作成したNGINXリソースは、指定のデータベース・ノード(HOSTING_MEMBERSで指定)で同時に実行されます。これは、複数のGoldenGateサービス・マネージャのデプロイメントが構成されていて、それらがデータベース・ノード間で独立して移動できる場合にお薦めします。

NGINX Clusterwareリソースの作成後には、GoldenGateデプロイメントの起動前にNGINXを起動するように、GoldenGate XAGリソースを変更する必要があります。

3. root OSユーザーとして、次のコマンド例を使用してXAGリソースを変更します。

```
# Determine the current --filesystems parameter:
[opc@exadb-node1 ~]$ sudo su - grid
[grid@exadb-node1 ~]$ agctl config goldengate <instance_name> |grep "File
System"
File System resources needed: <file_system_resource_name>
# Modify the --filesystems parameter:
[opc@exadb-node1 ~]$ sudo su -
[root@exadb-node1 ~]# /u01/app/grid/xag/bin/agctl modify goldengate
<instance_name>
 --filesystems <file_system_resource_name>,nginx
```

4. NGINXに依存するXAG GoldenGateの登録ごとに、前述のコマンドを繰り返します。

タスク9 - Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成

Oracle RAC ノード間の切替え時にOracle GoldenGateプロセスにローカル・データベース接続を提供するために、Oracle GoldenGateが起動される可能性のあるOracle RACノードのすべてに対するTNS別名を作成します。

TNS別名は、デプロイメントの作成時に指定したTNS_ADMINディレクトリ内のtnsnames.oraファイルで作成します。

このタスクを完了するには、次のステップを実行します。

- ステップ9.1 - TNS別名の定義の作成
- ステップ9.2 - データベース資格証明の作成

ステップ9.1 - TNS別名の定義の作成

ソース・データベースがマルチテナント・データベースの場合は、2つのTNS別名エントリが必要になります。その1つはコンテナ・データベース(CDB)用、もう1つはレプリケートされるプラガブル・データベース(PDB)用です。

ターゲットのマルチテナント・データベースでは、TNS別名によってレプリケートしたデータが適用されている場所にPDBを接続します。プラガブル・データベースSERVICE_NAMEは、前のステップ(ステップ2.3: データベース・サービスの作成)で作成したデータベース・サービスに設定する必要があります。

1. oracle OSユーザーとして、データベース・ドメイン名を検索します。

```
[opc@exadb-node1]$ sudo su - oracle
[oracle@exadb-node1]$ source DB_NAME.env
[oracle@exadb-node1]$ sqlplus / as sysdba
SQL> show parameter db_domain
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_domain                            string      <db_domain_name>
```

2. 最初のデータベース・ノードのoracle OSユーザーとして、ステップに従ってTNS別名の定義を作成し、すべてのデータベース・ノードにそれらを分散します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ dcli -l oracle -g ~/dbs_group mkdir -p
/u02/app/oracle/goldengate/network/admin
[oracle@exadb-node1 ~]$ vi
/u02/app/oracle/goldengate/network/admin/tnsnames.ora
OGGSRV_CDB =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <cdb_service_name>.<db_domain_name>)
       )
    )
OGGSRV_<PDB_NAME> =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <pdb_service_name>.<db_domain_name>)
       )
    )
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group -f
/u02/app/oracle/goldengate/network/admin/*.ora -d
/u02/app/oracle/goldengate/network/admin
```

ノート:

Oracle GoldenGate デプロイメントの TNS_ADMIN ディレクトリにある `tnsnames.ora` または `sqlnet.ora` を変更した場合は、変更内容を採用するためにデプロイメントを再起動する必要があります。

ステップ 9.2 - データベース資格証明の作成

Oracle GoldenGate デプロイメントを作成したら、Oracle GoldenGate 管理サービスのホーム・ページを使用して、前述の TNS 別名を使用してデータベース資格証明を作成します。

`oggadmin`ユーザーとして、データベース資格証明を作成します。

1. 管理サービスにログインします: https://localhost:<localPort>/<instance_name>/adminsrvr
2. 管理サービスの「構成」をクリックします。
3. 「資格証明の追加」のプラス・ボタンをクリックします。
4. 必要な情報を追加します。

   ソース・データベースがマルチテナント・データベースの場合は、CDBとPDBのデータベース資格証明を作成します。ターゲット・データベースがマルチテナント・データベースの場合は、PDBに対して単一の資格証明を作成します。

タスク10 - 新規プロファイルの作成

Oracle GoldenGate Administration Serverの起動時にExtractプロセスとReplicatプロセスを自動的に起動する、新しいプロファイルを作成します。

その後、ExtractプロセスまたはReplicatプロセスが中止された場合は再起動します。GoldenGate Microservicesでは、自動起動と再起動はプロファイルによって管理されます。

Oracle GoldenGate Administration Server GUIを使用して、各Oracle GoldenGateプロセスに割り当てることができる新しいプロファイルを作成します。

1. ソースおよびターゲットGoldenGateで、管理サービスにログインします。
2. 管理サービスの「プロファイル」をクリックします。
3. 管理対象プロセスの設定ホームページで、「プロファイル」の横にあるプラス(+)記号をクリックします。「プロファイルの追加」ページが表示されます。
4. 詳細を入力します。
5. 「発行」をクリックします。

タスク11 - Oracle GoldenGateプロセスの構成

Oracle GoldenGate Microservices Architectureを使用してExtract、分散パスおよびReplicatプロセスを作成する場合、Oracle RACノード間で共有する必要があるファイルはすべて、共有ファイル・システム(DBFSまたはACFS)に格納されているデプロイメント・ファイルですでに共有されています。

次に示す基本的な構成の詳細は、Extract、分散パスおよびReplicatプロセスのために Oracle RACでOracle GoldenGate Microservicesを実行する際にお薦めします。

このタスクを完了するには、次のステップを実行します。

- ステップ11.1 - Extractの構成
- ステップ11.2 - (DBFSのみ)共有ストレージへの一時キャッシュ・ファイルの配置
- ステップ11.3 - 分散パスの構成
- ステップ11.4 - Replicatの構成

ステップ11.1 - Extractの構成

Oracle GoldenGate管理サービスのGUIインタフェースを使用してExtractを作成するときには、`Trail SubDirectory`パラメータを空白のままにして、証跡ファイルが自動的に共有ファイル・システムに格納されているデプロイメント・ディレクトリに作成されるようにします。証跡ファイルのデフォルトの場所は、`/<deployment directory>/var/lib/data`ディレクトリです。

> ノート:
>
> マルチテナント・データベースから取得するには、c##アカウントを使用してルート・レベルで構成した Extract を使用する必要があります。マルチテナント・データベースにデータを適用するには、PDB ごとに個別の Replicat が必要になります。これは、Replicat は PDB レベルで接続して、その PDB 以外のオブジェクトにアクセスできないためです

データベース資格証明を作成します。

1. ソースOracle GoldenGateのOracle GoldenGate管理サーバーにログインします。
2. 「管理サービス」の「概要」をクリックします。
3. 「Extractの追加」のプラス・ボタンをクリックします。
4. 「統合Extract」を選択します。
5. 必要な情報を追加します。
6. 「次」をクリックします。
7. PDBからのCDBルート取得を使用する場合は、PDB名を指定したSOURCATALOGパラメータを追加します。
8. 「作成」をクリックします。

ステップ11.2 - (DBFSのみ)共有ストレージへの一時キャッシュ・ファイルの配置

共有ストレージにDBFSを使用しているときに、デプロイメントの`var/temp`ディレクトリが「タスク5 - Oracle GoldenGateデプロイメントの作成」の説明に従ってローカル・ストレージに移動された場合は、ExtractのCACHEMGRパラメータを使用して一時キャッシュ・ファイルを共有ストレージに配置することをお薦めします。

1. `oracle` OSユーザーとして、DBFSデプロイメントのマウント・ポイントの下位に新しいディレクトリを作成します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ mkdir
```

```
 /mnt/dbfs/goldengate/deployments/<instance_name>/temp_cache
```

2. Extractパラメータを新しいディレクトリに設定します。

```
CACHEMGR CACHEDIRECTORY
 /mnt/dbfs/goldengate/deployments/<instance_name>/temp_cache
```

Extractプロセスの作成の詳細は、Oracle DatabaseでのOracle GoldenGate Classicアーキテクチャの使用を参照してください。

ステップ11.3 - 分散パスの構成

NGINXリバース・プロキシでOracle GoldenGate Distributionパスを使用する場合は、パスのクライアントとサーバーの証明書が構成されるように、追加のステップを実行する必要があります。

分散パスの作成の詳細は、Oracle GoldenGate Microservicesのドキュメントを参照してください。証明書を正しく構成するためのステップバイステップの例は、NGINXを使用したオンプレミスのOracle GoldenGateからOCI GoldenGateへの接続についてのビデオを視聴してください。

次のサブステップを実行して、このステップを完了します。

- ステップ11.3.1 - ターゲット・サーバーのルート証明書のダウンロードとソースOracle GoldenGateへのアップロード
- ステップ11.3.2 - ソースOracle GoldenGateが使用するターゲット・デプロイメントのユーザーの作成
- ステップ11.3.3 - ソースOracle GoldenGateの資格証明の作成
- ステップ11.3.4 - ソースOracle GoldenGateからターゲット・デプロイメントへの分散パスの作成
- ステップ11.3.5 - ターゲット・デプロイメント・コンソールのレシーバ・サービスでの接続の確認

ステップ11.3.1 - ターゲット・サーバーのルート証明書のダウンロードとソースOracle GoldenGateへのアップロード

ターゲット・デプロイメント・サーバーのルート証明書をダウンロードし、ソース・デプロイメントのサービス・マネージャにCA証明書を追加します。

1. ターゲットGoldenGateで、管理サービスにログインします。
2. NGINXを使用したオンプレミスのOracle GoldenGateからOCI GoldenGateへの接続についてのビデオの"ステップ2 - ターゲット・サーバーのルート証明書のダウンロード"に関する説明に従います。

ステップ11.3.2 - ソースOracle GoldenGateが使用するターゲット・デプロイメントのユーザーの作成

ターゲット・デプロイメントで、次に接続する分散パスのユーザーを作成します。

1. ターゲットGoldenGateで、管理サービスにログインします。
2. 「管理サービス」の「管理者」をクリックします。
3. 「ユーザー」の横のプラス(+)記号をクリックします。
4. 詳細を入力します。

ステップ11.3.3 - ソースOracle GoldenGateの資格証明の作成

前のステップで作成したユーザーでターゲット・デプロイメントを接続するソース・デプロイメントに資格証明を作成します。たとえば、OP2CのドメインとWSSNETの別名です。

1. ソースOracle GoldenGateで、管理サービスにログインします。
2. 「管理サービス」の「構成」をクリックします。
3. 「データベース」ホーム・ページで「資格証明」の横にあるプラス(+)記号をクリックします。「資格証明の追加」ページが表示されます。
4. 詳細を入力します。

ステップ11.3.4 - ソースOracle GoldenGateからターゲット・デプロイメントへの分散パスの作成

分散サーバーから受信サーバーに証跡ファイルを送信するパスが作成されます。パスはDistribution Serviceで作成できます。

ソース・デプロイメントのパスを追加するには:

1. ソースOracle Goldengateで、分散サービスにログインします。
2. 「分散サービス」ホーム・ページで、「パス」の横のプラス(+)記号をクリックします。「パスの追加」ページが表示されます。
3. 次に示すように詳細を入力します。

オプション	説明
パス名	パスの名前を選択します
ソース: トレイル名	ドロップダウン・リストから Extract の名前を選択します(証跡名が自動的に入力されます)。表示されない場合は、Extract の追加時に指定した証跡名を入力します。
生成されたソース URI	サーバーの名前に localhost を指定します。これにより、任意の Oracle RAC のノードで分散パスを起動できます。
ターゲット認証方式	ユーザーID 別名を使用します。
ターゲット	「ターゲット」の転送プロトコルを wss (セキュア Web ソケット)に設定します。「ターゲット・ホスト」を、NGINX が構成された「ポート番号」とともにターゲット・システムへの接続に使用されるターゲット・ホスト名/VIP に設定します(デフォルトは 443)。
ドメイン	「ドメイン」は、前のステップ 11.3.3 で作成した資格証明ドメイン(OP2C など)に設定します。
別名	「別名」は、ステップ 11.3.3 で作成した資格証明別名 wssnet に設定します。
自動再起動オプション	分散サーバーが自動的に起動されたときに分散パスを再起動するように設定します。これは、分散サーバーの Oracle RAC ノードの再配置後に、手動操作を不要にするために必要です。「再試行」の回数は 10 に設定することをお薦めします。「遅延」を 1 に設定します。これは、再起動の試行間に一時停止する分数です。

4. 「パスの作成」をクリックします。
5. 「アクション・メニュー」から、「開始」をクリックします。
6. 分散サービスが稼働していることを確認します。

ステップ11.3.5 - ターゲット・デプロイメント・コンソールのレシーバ・サービスでの接続の確認

1. ターゲット・デプロイメント・コンソールで、管理サービスにログインします。
2. 「レシーバ・サービス」をクリックします。

ステップ11.4 - Replicatの構成

Replicatプロセスにより、証跡データを受信し、それをデータベースに適用します。

次のサブステップを実行して、このステップを完了します。

- ステップ11.4.1 - チェックポイント表の作成
- ステップ11.4.2 - Replicatの追加

ステップ11.4.1 - チェックポイント表の作成

チェックポイント表は、Oracle GoldenGate Replicatプロセスに必須のコンポーネントです。Administration Serviceの資

格証明ページからデータベースに接続した後、チェックポイント表を作成できます。

ターゲット・デプロイメントにチェックポイント表を作成します。

1. ターゲットGoldenGateで、管理サービスにログインします。
2. 「管理サービス」の「構成」をクリックします。
3. 「データベース」をクリックして、ターゲット・データベースまたはPDBに接続します。
4. 「チェックポイント」の横のプラス(+)記号をクリックします。「チェックポイントの追加」ページが表示されます。
5. 詳細を入力します。

チェックポイント表の詳細は、チェックポイント表の概要を参照してください。

ステップ11.4.2 - Replicatの追加

データベース接続を設定して検証した後で、次のステップを実行するとデプロイメントのReplicatを追加できます。

1. ターゲットGoldenGateで、管理サービスにログインします。
2. 「管理サービス」ホーム・ページで「Replicat」の横のプラス(+)記号をクリックします。Replicatの追加ページが表示されます。
3. Replicatのタイプを選択して、「次」をクリックします。
   次に示すように詳細を入力します。

オプション	説明
プロセス名	Replicat プロセスの名前
資格証明ドメイン	ステップ 9.2 で作成した資格証明ドメイン。この例では、`GoldenGate`
資格証明別名	ステップ 9.2 で作成した資格証明別名。この例では、`Target_PDB`
ソース	証跡を使用するソースを選択します。
トレイル名	2 文字の証跡名。
チェックポイント表	既存のチェックポイント表の使用について設定します。

5. 「パスの作成」をクリックします。
6. 「アクション・メニュー」から、「開始」をクリックします。
7. Replicatが稼働していることを確認します。

21 Cloud MAA Platinum: Active Data Guardと統合されたOracle GoldenGate Microservices Architecture

Oracle GoldenGate MicroservicesとOracle Data Guardを組み合せて統合することで、すべての計画停止および計画外停止でゼロまたはほぼゼロの停止時間を達成するMAA Platinumサービス・レベル構成を実現できます。

概要

これらの構成および運用のベスト・プラクティスを使用すると、データ損失ゼロまたはデータ損失ロール・トランジションの後もOracle Data Guardとシームレスに動作するように、Oracle GoldenGateを構成できます。

Oracle GoldenGate Microservicesデプロイメント・ファイルのファイル・システムとしてDatabase File System (DBFS)を使用することにより、Oracle GoldenGate Extract、分散パスおよびReplicatプロセスは、ロール・トランジション後も引き続きデータベースとの同期を維持します。

Oracle Real Application Clusters (Oracle RAC)、Oracle ClusterwareおよびOracle Database File System (DBFS)を使用して、Oracle Exadata Database Service on Dedicated Infrastructure (ExaDB-D)またはOracle Exadata Database Service on Cloud@Customer (ExaDB-C@C)でOracle GoldenGate Microservices Architectureを構成するために、これらのベスト・プラクティスを実装することで、Oracle Data Guardとシームレスに連携できます。

これらのベスト・プラクティスにより、Data Guardスタンバイによって保護されているデータベースを使用したOracle GoldenGate Microservicesレプリケーションは、Data Guard保護モードの構成(最大パフォーマンス、最大可用性または最大保護)に関係なく、Oracle Data Guardロール・トランジションに従って透過的かつシームレスに機能するようになります。

主要なソフトウェア要件を次に示します。

- Oracle Grid Infrastructure 19c以降

  Oracle Grid Infrastructureは、ビジネスクリティカルなアプリケーション高可用性を管理するために必要なコンポーネントを提供します。Oracle Clusterware (Oracle Grid Infrastructureのコンポーネント)ネットワーク、データベースおよびOracle GoldenGateリソースを使用すると、障害発生時の可用性を管理できます。

- Oracle Grid Infrastructure Agentバージョン10.2以降

  Oracle Grid Infrastructure Agentは、Oracle Grid Infrastructureコンポーネントを活用して、Oracle GoldenGateとその依存リソース(データベース、ネットワーク、ファイル・システムなど)との統合を提供します。また、エージェントはOracle GoldenGateとOracle Data Guardを統合します。これにより、Oracle GoldenGateはロール・トランジション後に、新しいプライマリ・データベースで再起動されます。

- Oracle Database 19c以上

  Oracle GoldenGateを使用する場合に推奨されるOracle Databaseパッチの完全なリストは、My Oracle Supportノート2193391.1を参照してください。

- Oracle GoldenGate Microservicesバージョン21c以降

  Oracle GoldenGate 21cでは統合ビルド・サポートが導入されているため、単一のソフトウェア・インストールで、レプリケートされたデータの取得と複数の主要なOracle Databaseバージョン(11gリリース2から21c)への適用がサポートされます。これが可能になるのは、Oracle GoldenGateインストールに必要なOracle Databaseクライアント・ライブラリが含まれており、別途データベースORACLE_HOMEをインストールする必要がないためです。

- 重要なOracle GoldenGateファイルを保護およびレプリケートするOracle DBFS

  Oracle Database File System (DBFS)は、MAAによって検証され、Oracle Data GuardおよびOracle GoldenGate構成に推奨される、唯一のファイル・システムです。これは、チェックポイント・ファイルや証跡ファイルなどの必要なOracle GoldenGateファイルの格納場所を、Oracle Data Guardで保護されている同じデータベース内に置くことができ、Oracle GoldenGateファイルとデータベース間の一貫性をシームレスに確保できるためです。

タスク1 - 始める前に

開始するには、次の前提条件を完了します。

- Oracle GoldenGateのデプロイメント用に、Oracle Exadata Database Service on Dedicated InfrastructureまたはCloud@Customerを調達します。

  ビジネスのニーズに応じて、既存のExaDB-DまたはExaDB-C@Cシステムを使用してOracle GoldenGateをデプロイすることも、新しいシステムを起動することもできます。ExaDB-Dシステムの起動と管理の手順については、Oracle Exadata Database Service on Dedicated Infrastructureを参照してください。また、ExaDB-C@Cの場合は、Oracle Exadata Database Service on Cloud@Customerを参照してください。

- 「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」の説明に従って、Oracle GoldenGateを構成します。

  Data Guardと統合する場合、重要なOracle GoldenGateファイルにはDBFSが必要です。

- 続行する前に、Oracle Data Guardスタンバイ・データベースも構成済で動作している必要があります。

  Oracle Data Guardの詳細は、Oracle Data Guardスタート・ガイドを参照してください。

- セキュアなデプロイメントでは、RESTful APIコールを発行し、証跡データをSSL/TLSを介してDistribution ServerとReceiver Server間で転送します。認証局(CA)から取得した既存のビジネス証明書を使用するか、独自の証明書を作成できます。

  続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

タスク2 - GoldenGateに向けたOracle Databaseの構成

このタスクを完了するには、次のステップを実行します。

- ステップ2.1 - Oracle GoldenGate用のスタンバイ・データベースの構成
- ステップ2.2 - プライマリ・データベース・サービスの変更
- ステップ2.3 - スタンバイ・データベース・サービスの作成

ステップ2.1 - Oracle GoldenGate用のスタンバイ・データベースの構成

スタンバイ・データベースの初期化パラメータは、「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」で指定されているプライマリ・データベースの初期化パラメータと一致する必要があります。

これには次のパラメータが含まれます。

- `ENABLE_GOLDENGATE_REPLICATION=TRUE`
- Oracle GoldenGateソース・データベースの場合は、`FORCE LOGGING`モードを有効にし、最小サプリメンタル・ロギングを有効にします。
- Oracle GoldenGateソース・データベースの場合や統合Replicat (パラレルまたは非パラレル)を実行している場合は、`STREAMS_POOL_SIZE`を構成します。

ステップ2.2 - プライマリ・データベース・サービスの変更

プライマリ・データベース・サーバーで、Oracle Exadata Database Service構成の、元のOracle GoldenGateの一部として作成された既存のデータベース・サービスを検証します。

デフォルトではサービス・ロールが`PRIMARY`として定義されるため、ロール・トランジション後にデータベースがData Guardプライマリ・データベース・ロールになった場合にのみサービスが起動されます。

プライマリ・システムの`oracle` OSユーザーとして、次に示すコマンドを使用して、サービス・ロールを検証します。

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ srvctl config service -db $ORACLE_UNQNAME |
 egrep 'Service name|role|Pluggable database name'
Service name: <CDB_SERVICE_NAME>
Service role: PRIMARY
Pluggable database name:
Service name: <PDB_SERVICE_NAME>
Service role: PRIMARY
Pluggable database name: <PDB_NAME>
```

ロールが`PRIMARY`でない場合は、次に示すコマンドを使用して、サービスを変更します。

```
[oracle@exapri-node1 ~]$ srvctl modify service -db $ORACLE_UNQNAME
 -service <service_name> -role PRIMARY
```

データベースがマルチテナント環境の一部である場合は、必ずマルチテナント・コンテナ・データベース(CDB)とプラガブル・データベース(PDB)の両方のサービスを変更してください。

ステップ2.3 - スタンバイ・データベース・サービスの作成

スタンバイOracle Exadata Database Serviceでは、データベースがプライマリ・ロールで開かれたときにOracle Grid Infrastructure AgentがOracle GoldenGateのデプロイを自動的に開始するために、スタンバイ・データベースにデータ

ベース・サービスが必要です。

ソース・データベースがマルチテナント環境にある場合は、ルート・コンテナ・データベース(CDB)とレプリケートされるスキーマを含むプラガブル・データベース(PDB)に、個別のサービスが必要です。マルチテナント環境のターゲット・データベースの場合は、PDBに単一のサービスが必要です。

プライマリ・データベースで作成したサービスと同じように、スタンバイ・データベースにサービスを作成します。プライマリ・システムで指定したサービス名と同じものを使用することをお薦めします。サービスは-preferredオプションを使用して、シングルトン・サービスとして作成する必要があります。これは、アプリケーション仮想IPアドレス(VIP)、DBFSおよびOracle GoldenGateが、サービスが実行されているシステム・ノードで実行されるためです。

1. oracle OSユーザーとして、完全修飾ドメイン名(FQDN)を取得します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ hostname -f
exadb-node1.<FQDN>
```

2. スタンバイ・システムのoracle OSユーザーとして、次に示すコマンドを使用してサービスを作成します。

```
[opc@exastb-node1 ~]$ sudo su - oracle
[oracle@exastb-node1 ~]$ source <db_name>.env
[oracle@exastb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service <CDB_SERVICE_NAME>.<FQDN> -preferred <SID1> -available <SID2>
 -role PRIMARY
[oracle@exastb-node1 ~]$ srvctl add service -db $ORACLE_UNQNAME
 -service <PDB_SERVICE_NAME>.<FQDN> -preferred <SID1> -available <SID2>
 -pdb <PDB name> -role PRIMARY
```

タスク3 - Oracle Database File Systemの構成

Oracle Data GuardでOracle GoldenGateを構成する場合、お薦めするソリューションはDatabaseファイル・システム(DBFS)のみです。

データベース内のDBFSユーザー、表領域およびファイル・システムは、「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」で説明されているように、事前にプライマリ・データベースに作成されています。

このタスクを完了するには、次のステップを実行します。

- ステップ3.1 - Oracle Exadata Database ServiceのDBFSの構成
- ステップ3.2 - (PDBのみ) TNSNAMESでのエントリの作成
- ステップ3.3 - プライマリ・システムからのmount-dbfsスクリプトのコピーと編集
- ステップ3.4 - DBFSリソースのOracle Clusterwareへの登録

ステップ3.1 - Oracle Exadata Database ServiceのDBFSの構成

1. スタンバイ・システムのopc OSユーザーとして、fuseグループにgridユーザーを追加します。

```
[opc@exastb-node1 ~]$ sudo -u grid
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/bin/olsnodes > ~/dbs_group
[opc@exadb-node1 ~]$ dcli -g ~/dbs_group -l opc sudo usermod -a -G fuse grid
```

2. スタンバイ・システムのopc OSユーザーとして、ファイル/etc/fuse.confが存在していることと、user_allow_otherオプションが含まれていることを確認します。

```
[opc@exastb-node1 ~]$ cat /etc/fuse.conf
# mount_max = 1000
user_allow_other
```

3. スタンバイ・システムのopc OSユーザーとして、このステップをスキップします(オプションuser_allow_otherがすでに/etc/fuse.confファイルにある場合)。それ以外の場合は、次に示すコマンドを実行して、オプションを追加します。

```
[opc@exastb-node1 ~]$ dcli -g ~/dbs_group -l opc “echo user_allow_other |
 sudo tee -a /etc/fuse.conf”
```

4. スタンバイ・システムのopc OSユーザーとして、DBFSファイルシステムのマウント・ポイントとして使用する空のディレクトリを作成します。

> ノート:
>
> マウント・ポイントがプライマリ・システムのマウント・ポイントと同じであることが重要です。これは、Oracle GoldenGate デプロイメントの物理的な場所がデプロイメント構成ファイルに含まれているためです。
>
> ```
> [opc@exastb-node1 ~]$ dcli -g ~/dbs_group -l opc sudo mkdir -p /mnt/dbfs
> ```

5. スタンバイ・システムのopc OSユーザーとして、マウント・ポイント・ディレクトリの所有権を変更して、grid OSユーザーがアクセスできるようにします。

```
[opc@exastb-node1 ~]$ dcli -g ~/dbs_group -l opc
 sudo chown oracle:oinstall /mnt/dbfs
```

ステップ3.2 - (PDBのみ) TNSNAMESでのエントリの作成

1. スタンバイ・システムのoracle OSユーザーとして、$TNS_ADMIN/tnsnames.oraファイルに接続エントリを追加します。ステップ2.3で作成したPDBサービス名を使用します。

```
[oracle@exadb-node1 ~]$ vi $TNS_ADMIN/tnsnames.ora
dbfs =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
 (CONNECT_DATA =
 (SERVICE_NAME = <PDB_SERVICE_NAME> )
 )
 )
```

2. oracle OSユーザーとして、残りのノードに$TNS_ADMIN/tnsnames.oraファイルを配布します。

```
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f $TNS_ADMIN/tnsnames.ora -d $TNS_ADMIN/
```

ステップ3.3 - プライマリ・システムからのmount-dbfsスクリプトのコピーと編集

1. プライマリ・システムのroot OSユーザーとして、mount-dbfs.confファイルとmount-dbfs.shファイルを含むzipファイルを作成します。

```
[opc@exapri-node1 ~]$ sudo su -
[root@exapri-node1 ~]# zip -j /tmp/mount-dbfs.zip
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script/mount-dbfs.sh
 /etc/oracle/mount-dbfs.conf
 adding: mount-dbfs.sh (deflated 67%)
 adding: mount-dbfs.conf (deflated 58%)
```

2. スタンバイ・システムのopc OSユーザーとして、プライマリ・システムからスタンバイ・システムにmount-dbfs.zipファイルをコピーします。

```
[opc@exastb-node1 ~]$ scp exapri-node1.oracle.com:/tmp/mount-dbfs.zip /tmp
```

3. スタンバイ・システムのopc OSユーザーとして、mount-dbfs.zipファイルを解凍し、構成ファイルmount-dbfs.confを編集します。

```
[opc@exastb-node1 ~]$ unzip /tmp/mount-dbfs.zip -d /tmp
Archive: /tmp/mount-dbfs.zip
 inflating: /tmp/mount-dbfs.sh
 inflating: /tmp/mount-dbfs.conf
[opc@exastb-node1 ~]$ vi /tmp/mount-dbfs.conf
```

これらをプライマリ・システムと同じディレクトリに配置することをお薦めします。スタンバイ・データベースと一致するように、mount-dbfs.confファイルの次のパラメータを変更する必要があります。

- DBNAME
- TNS_ADMIN
- PDB_SERVICE

4. スタンバイ・システムのopc OSユーザーとして、mount-dbfs.confをデータベース・ノードの/etc/oracleディレクトリにコピーして、適切な権限を設定します。

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc -d /tmp
 -f /tmp/mount-dbfs.conf
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 cp /tmp/mount-dbfs.conf /etc/oracle
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chown oracle:oinstall /etc/oracle/mount-dbfs.conf
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 chmod 660 /etc/oracle/mount-dbfs.conf
```

5. スタンバイ・システムのopc OSユーザーとして、mount-dbfs.shをデータベース・ノードの$GI_HOME/crs/scriptディレクトリにコピーして、適切な権限を設定します。

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo mkdir
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo chown
 grid:oinstall $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l grid
 -d $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script
 -f /tmp/mount-dbfs.sh
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l grid chmod 770
 $(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)/crs/script/mount-dbfs.sh
```

ステップ3.3 - DBFSリソースのOracle Clusterwareへの登録

このリソースをOracle Clusterwareに登録するときには、必ずcluster_resourceとして作成します。cluster_resourceを使用する理由は、ファイル・システムを一度に1つのノードにのみマウントするようにして、同時ファイル書込みの可能性が生じ、ファイル破損問題の原因になるDBFSの同時ノードのマウントを防止するためです。

Oracle Multitenantを使用する場合は、プライマリ・データベースで作成されたDBFSリポジトリを含む、同じPDBのサービス名を使用してください。

1. スタンバイ・システムのgrid OSユーザーとして、前のDBFSサービス依存関係についてのステップで作成したデータベース・サービスのリソース名を検索します。

```
[opc@exastb-node1 ~]$ sudo su - grid
[grid@exastb-node1 ~]$ crsctl stat res |grep <PDB_NAME>
NAME=ora.<DB_UNIQUE_NAME>.<PDB_SERVICE_NAME>.svc
```

2. スタンバイ・システムのoracle OSユーザーとして、次のスクリプトを実行してClusterwareリソースを登録します。

```
[opc@exadb-node1 ~]$ sudo su - oracle
[oracle@exadb-node1 ~]$ vi add-dbfs-resource.sh
##### start script add-dbfs-resource.sh
#!/bin/bash
ACTION_SCRIPT=$(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/crs/script/mount-dbfs.sh
RESNAME=dbfs_mount
DEPNAME=ora.<DB_UNIQUE_NAME>.<PDB_SERVICE_NAME>.svc
ORACLE_HOME=$(grep ^crs_home /etc/oracle/olr.loc | cut -d= -f2)
PATH=$ORACLE_HOME/bin:$PATH
export PATH ORACLE_HOME
crsctl add resource $RESNAME ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=$ACTION_SCRIPT, ¥
 CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, ¥
 START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',¥
 STOP_DEPENDENCIES='hard($DEPNAME)',¥
 SCRIPT_TIMEOUT=300"
##### end script add-dbfs-resource.sh
[oracle@exadb-node1 ~]$ sh add-dbfs-resource.sh
```

ノート:

$RESNAME リソースの作成後、$RESNAME リソースが ONLINE のときに$DBNAME データベースを停止するために、srvctl の使用時に force フラグを指定する必要があります。

例: srvctl stop database -d $ORACLE_UNQNAME -f

タスク4 - Oracle GoldenGateのインストール

GoldenGate構成の一部となるOracle Exadata Database Service構成のすべてのノードに、Oracle GoldenGateソフトウェアをローカルにインストールします。

ノート:

インストール・ディレクトリがすべてのノードで同一であり、プライマリ・システムのインストール・ディレクトリと一致していることを確認してください。

1. スタンバイ・システムのopc OSユーザーとして、プライマリ・システムからスタンバイ・システムにoggcore.rspレスポンス・ファイルをコピーします。

```
[opc@standby_node_1 ~]$ scp
 primary_node_1:/u02/app_acfs/goldengate/oggcore.rsp
 /u02/app_acfs/goldengate
```

2. スタンバイ・システムで、「タスク4 - Oracle GoldenGateのインストール」で説明されている「ステップ4.2 - Oracle GoldenGateのインストール」に従います。

タスク5 - Oracle GoldenGateデプロイメント・ディレクトリの作成

Oracle GoldenGateサービス・マネージャとデプロイメントは、すでにプライマリ・システムで作成されていますが、特定のディレクトリとシンボリック・リンクをスタンバイ・システム・ノードで構成する必要があります。これらのディレクトリとシンボリック・リンクは、「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」の説明に従って、プライマリ・システムで作成されました。

1. プライマリ・システムのoracle OSユーザーとして、データストア・ディレクトリを確認します。

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ grep RepoDatastorePath /mnt/dbfs/goldengate
 /deployments/<instance_name>/var/log/pmsrvr.log|uniq
"RepoDatastorePath": "",
       "RepoDatastorePath":
"/u02/app/oracle/goldengate/datastores/<instance_name>",
```

2. スタンバイ・システムのoracle OSユーザーとして、すべてのデータベース・ノードにディレクトリを作成します。

```
[oracle@exastb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/datastores/<gg_deployment_name>
```

プライマリ・システムで作成されたシンボリック・リンクと一致するように、Oracle GoldenGateデプロイメントの一時ディレクトリのローカル・ストレージを作成します。

3. プライマリ・システムのoracle OSユーザーとして、データストア・ディレクトリを確認します。

```
[oracle@exapri-node1 ~]$ ls -l
 /mnt/dbfs/goldengate/deployments/<instance_name>/var |grep temp
lrwxrwxrwx  1 oracle oinstall 49 Oct  3 10:20 temp ->
 /u02/app/oracle/goldengate/deployments/<instance_name>/temp
```

4. スタンバイ・システムのoracle OSユーザーとして、スタンバイ・データベース・ノードに同じディレクトリを作成します。

```
[oracle@exastb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/deployments/<instance_name>/temp
```

タスク6 - ネットワーク構成

スタンバイ・システムで、「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」の章の「タスク6 - ネットワークの構成」の手順に従います。

タスク7 - スタンバイNGINXリバース・プロキシの構成

GoldenGateのリバース・プロキシ機能を使用すると、Oracle GoldenGateデプロイメントに関連付けられたすべてのOracle GoldenGate Microservicesに対応する単一のアクセス先を使用できます。

リバース・プロキシを使用しないと、GoldenGateデプロイメント・マイクロサービスは、ホスト名またはIPアドレスと個別のポート番号で構成されるサービスごとに1つずつのURLを使用してアクセスされます。リバース・プロキシは、マイクロサービスに簡単にアクセスし、強化されたセキュリティと管理性を提供するために必須です。

このタスクを完了するには、次のステップを実行します。

- ステップ7.1 - NGINXリバース・プロキシのインストール
- ステップ7.2 - NGINX構成ファイルのプライマリ・システムからのコピー
- ステップ7.3 - NGINXのサーバー証明書のインストール
- ステップ7.4 - 新しいNGINX構成のテストとリロード
- ステップ7.5 - GoldenGate NGINX構成ファイルの配布
- ステップ7.6 - NGINXを管理するClusterwareリソースの作成

ステップ7.1 - NGINXリバース・プロキシのインストール

スタンバイ・システムで、「タスク8 - NGINXリバース・プロキシの構成」の「ステップ8.1 - NGINXリバース・プロキシ・サーバーのインストール」の手順に従って、NGINXをインストールします。

ステップ7.2 - NGINX構成ファイルのプライマリ・システムからのコピー

1. スタンバイ・システムのopcユーザーとして、プライマリからスタンバイ・データベース・システムにNGINX構成ファイルをコピーします。

```
[opc@standby_node_1 ~]$ scp
primary_node_1.oracle.com:/etc/nginx/conf.d/ogg_<deployment_name>.conf
 /tmp
```

2. スタンバイ・システムのrootユーザーとして、NGINX構成ファイルをディレクトリ/tmpからディレクトリ/etc/nginx/conf.dへコピーします。

```
[opc@exastb-node1 ~]$ sudo su -
[root@exastb-node1 ~]# cp /tmp/ogg_<deployment_name>.conf /etc/nginx/conf.d
```

ステップ7.3 - NGINXのサーバー証明書のインストール

スタンバイ・システムはプライマリ・システムとは異なるVIP名/アドレスを使用するため、別のCA署名証明書が必要になります。続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

1. スタンバイ・システムのrootユーザーとして、ファイル権限400 (-r--------)のrootが所有する/etc/nginx/sslディレクトリでサーバーCA証明書とキー・ファイルをコピーします。

```
[opc@exastb-node1 ~]$ sudo su -
[root@exastb-node1 ~]# mkdir /etc/nginx/ssl
[root@exastb-node1 ~]# chmod 400 /etc/nginx/ssl
```

2. スタンバイ・システムのrootユーザーとして、証明書とキー・ファイルの正しいファイル名を、NGINX構成ファイルと同じファイル名と一致するように設定します。

```
[root@exastb-node1 ~]# grep
```

```
ssl_certificate /etc/nginx/conf.d/ogg_<deployment_name>.conf
   ssl_certificate         /etc/nginx/ssl/server.chained.crt;
   ssl_certificate_key     /etc/nginx/ssl/server.key;
```

ノート:

プライマリ・システムからコピーした複数のリバース・プロキシ構成ファイルをコピーした場合は、このプロセスをファイルごとに繰り返す必要があります。

CA署名付き証明書を使用する場合、ssl_certificate NGINXパラメータで指定した証明書には、1) CA署名付き証明書、2)中間証明書および3)ルート証明書を単一ファイルに含める必要があります。この順序は非常に重要です。そうしていないと、NGINXの起動は失敗して、エラー・メッセージが表示されます。

```
(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key
values mismatch).
```

ルート証明書と中間証明書は、CA署名付き証明書プロバイダからダウンロードできます。

単一ファイルは次のコマンド例を使用して生成できます。

```
[root@exastb-node1 ~]# cat CA_signed_cert.crt intermediate.crt root.crt >
 /etc/nginx/ssl/server.chained.crt
```

ssl_certificate_keyファイルは、CA署名付き証明書をリクエストする際に必要になる証明書署名リクエスト(CSR)の作成時に生成されるキー・ファイルです。

3. スタンバイ・システムのrootユーザーとして、server_nameパラメータを、プライマリ・システムからコピーしたリバース・プロキシ構成ファイルの正しいVIP名に変更します。

```
[root@exastb-node1 ~]# vi /etc/nginx/conf.d/ogg_<deployment_name>.conf
# Before:
server_name             exapri-vip.oracle.com;
# After:
server_name             exastb-vip.oracle.com;
```

ステップ7.4 - 新しいNGINX構成のテストとリロード

VIPはプライマリ・データベース・サービスが実行されるまでスタンバイ・システムで実行されないため、/etc/sysctl.confファイルで設定する必要があるパラメータがあります。

1. スタンバイ・システムのopcユーザーとして、次に示すパラメータをファイル/etc/sysctl.confに追加します。

```
[opc@exastb-node1 ~]$ sudo vi /etc/sysctl.conf
# allow processes to bind to the non-local address
net.ipv4.ip_nonlocal_bind = 1
```

2. スタンバイ・システムのopcユーザーとして、/etc/sysctl.confファイルを配布します。

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 -d /tmp -f /etc/sysctl.conf
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo
 cp /tmp/sysctl.conf /etc/sysctl.conf
```

3. スタンバイ・システムのopcユーザーとして、変更した構成をリロードします。

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo sysctl
 -p /etc/sysctl.conf
```

4. スタンバイ・システムのopcユーザーとして、NGINX構成ファイルを検証し、構成のエラーを検出します。ファイルにエラーがある場合は、次のコマンドでエラーが報告されます。

```
[opc@exastb-node1 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

5. スタンバイ・システムのopcユーザーとして、NGINXを新しい構成で再起動します。

```
[opc@exastb-node1 ~]$ sudo systemctl restart nginx
```

ステップ7.5 - GoldenGate NGINX構成ファイルの配布

GoldenGateサービス・マネージャ用のすべてのリバース・プロキシ構成ファイルを作成すると、そのファイルをすべてのデータベース・ノードにコピーする必要があります。

1. スタンバイ・システムのopc OSユーザーとして、すべてのデータベース・ノードにNGINX構成ファイルを配布します。

```
[opc@exastb-node1 ~]$ sudo tar fczP nginx_conf.tar
 /etc/nginx/conf.d/ /etc/nginx/ssl/
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 -d /tmp -f nginx_conf.tar
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 sudo tar fxzP /tmp/nginx_conf.tar
```

2. スタンバイ・システムのopc OSユーザーとして、新しい構成ファイルのコピー先にしたすべてのノードで新しいNGINX構成をテストします。

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc sudo nginx -t

exastb-node1: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
exastb-node1: nginx: configuration file /etc/nginx/nginx.conf test is
successful
exastb-node2: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
exastb-node2: nginx: configuration file /etc/nginx/nginx.conf test is
successful
```

3. スタンバイ・システムのopc OSユーザーとして、新しい構成をロードするために、すべてのノードでNGINXを再起動します。

```
[opc@exastb-node1 ~]$ /usr/local/bin/dcli -g ~/dbs_group -l opc
 sudo systemctl restart nginx
```

ステップ7.6 - NGINXを管理するClusterwareリソースの作成

Oracle Clusterwareは、NGINXリバース・プロキシの起動を制御して、GoldenGateデプロイメントの起動前に自動的にリバース・プロキシを起動できるようにする必要があります。

1. スタンバイ・システムのgrid OSユーザーとして、次のコマンドを使用して、基盤となるネットワークCRSリソースに依存するNGINXリソースの作成に必要なネットワークCRSリソース名を取得します。

```
[opc@exastb-node1 ~]$ sudo su - grid
[grid@exastb-node1 ~]$ crsctl stat res -w "TYPE == ora.network.type"|grep NAME
NAME=ora.net1.network
```

2. スタンバイ・システムのrootユーザーとして、次のサンプル・コマンドを使用して、NGINXを管理するClusterwareリソースを作成します。HOSTING_MEMBERSおよびCARDINALITYは環境に合致するように置き換えます。

```
[opc@exastb-node1 ~]$ sudo su -
[root@exastb-node1 ~]# $(grep ^crs_home /etc/oracle/olr.loc | cut -d=
 -f2)/bin/crsctl add resource nginx -type generic_application -attr
```

```
 "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
x,user:oracle:rwx',
EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl start -f nginx',
STOP_PROGRAM='/bin/systemctl stop -f nginx',
CHECK_PROGRAMS='/bin/systemctl status nginx' ,
START_DEPENDENCIES='hard(ora.net1.network) pullup(ora.net1.network)',
STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)', RESTART_ATTEMPTS=0,
HOSTING_MEMBERS='<exastb-node1, exastb-node2>', CARDINALITY=2"
```

この例で作成したNGINXリソースは、指定のデータベース・ノード(HOSTING_MEMBERSで指定)で同時に実行されます。これは、複数のGoldenGateサービス・マネージャのデプロイメントが構成されていて、それらがデータベース・ノード間で独立して移動できる場合にお薦めします。

タスク8 - Oracle Grid Infrastructure Agentの構成

次の手順では、Oracle Grid Infrastructureスタンドアロン・エージェント(XAG)を使用してOracle GoldenGateを管理するようにOracle Clusterwareを構成する方法を示します。

XAGを使用することで、Oracle RACノード間での再配置時に、共有ファイル・システム(DBFS)のマウントとOracle GoldenGateデプロイメントの停止および起動を自動化します。

このタスクを完了するには、次のステップを実行します。

- ステップ8.1 - プライマリ・クラスタXAG GoldenGateインスタンスの変更
- ステップ8.2 - Oracle Grid Infrastructure Agentのインストール
- ステップ8.3 - Oracle Grid Infrastructure Agentの構成

ステップ8.1 - プライマリ・クラスタXAG GoldenGateインスタンスの変更

プライマリ・システムのOracle Grid Infrastructure Standalone Agent (XAG)のGoldenGateインスタンスは、Oracle Data Guard構成の一部であることを識別するように変更する必要があります。

プライマリ・システムのrootユーザーとして、次のコマンドを使用して、Oracle Data Guardの`autostart`フラグを変更します。

```
[opc@exapri-node1 ~]$ sudo su -
[root@exapri-node1 ~]# /u01/app/grid/xag/bin/agctl modify goldengate <instance_name>
 --dataguard_autostart yes
```

ステップ8.2 - Oracle Grid Infrastructure Agentのインストール

スタンバイ・システムで、「タスク7 - Oracle Grid Infrastructure Agentの構成」の「ステップ7.1 - Oracle Grid Infrastructure Standalone Agentのインストール」の手順に従います。

ステップ8.3 - Oracle Grid Infrastructure Agentの構成

Oracle GoldenGate MicroservicesをXAGに登録するために使用されるパラメータは、プライマリ・システムに登録するときに使用されるパラメータと似ています。

1. プライマリ・システムの`grid`ユーザーとして、次のコマンドを使用してプライマリ・システムの現在のパラメータを確認します。

```
[grid@exapri-node1 ~]$ agctl config goldengate <instance_name>
Instance name: <instance_name>
Application GoldenGate location is: /u02/app/oracle/goldengate/gg21c
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
 /mnt/dbfs/goldengate/deployments/<instance_name>/etc/conf
Goldengate Service Manager var directory:
 /mnt/dbfs/goldengate/deployments/<instance_name>/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: yes
Configured to run on Nodes: exapri-node1 exapri-node2
ORACLE_HOME location is: /u02/app/oracle/goldengate/gg21c/lib/instantclient
Database Services needed: ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.<FQDN>.svc
File System resources needed: dbfs_mount,nginx
Network: 1, IP:<VIP>, User:oracle, Group:oinstall
```

さらに、XAGパラメータ`--filesystem_verify no`を指定して、Oracle GoldenGateインスタンスの登録時にXAGがDBFSデプロイメント・ディレクトリの存在をチェックしないようにする必要があります。このパラメータを設定しないと、DBFSがスタンバイ・システムにマウントされないため、XAGの登録は失敗します。

ノート:

GoldenGate を XAG に登録する場合は、プライマリ・システムで使用されている名前と同じ GoldenGate インスタンス名を使用することをお薦めします。

2. スタンバイ・システムのrootユーザーとして、次のコマンド形式を使用して、XAGにOracle GoldenGate Microservices Architectureを登録します。

```
https://support.oracle.com/rs?type=doc&id=2193391.1
http://www.oracle.com/pls/topic/lookup?ctx=db19&id=SBYDB
[root@exastb-node1 ~]# /u01/app/grid/xag/bin/agctl add goldengate
<instance_name> ¥
--gg_home /u02/app/oracle/goldengate/gg21c ¥
--service_manager ¥
--config_home /mnt/dbfs/goldengate/deployments/<ggsm1>/etc/conf ¥
--var_home /mnt/dbfs/goldengate/deployments/<ggsm1>/var ¥
--port 9100 ¥
--oracle_home /u02/app/goldengate/gg21c/lib/instantclient ¥
--adminuser oggadmin ¥
--user oracle ¥
--group oinstall ¥
--network 1 --ip <virtual_IP_address> ¥
--filesystems dbfs_mount,nginx ¥
--db_services ora.<DB_UNIQUE_NAME>.<SERVICE_NAME>.<FQDN>.svc ¥
--use_local_services ¥
--nodes <exastb-node1>,<exastb-node2> ¥
--filesystem_verify no ¥
--dataguard_autostart yes
```

タスク9 - Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成

IPCプロトコルを使用してプライマリ・データベースのプライマリ・システムで作成された同じTNS別名は、「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」で指定されているIPC通信プロトコルを使用して、スタンバイ・システムの各データベース・ノード上に同じ別名で作成する必要があります。

Oracle GoldenGateデプロイメントで使用されるtnsnames.oraの場所は、プライマリ・システムと同様に、スタンバイ・システム・ノードでも同じである必要があります。

1. プライマリ・システムのoracleユーザーとして、次の問合せREST APIコールを使用して、TNS_ADMINの場所を問い合せます。

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ grep -1 TNS_ADMIN
 /mnt/dbfs/goldengate/deployments/ggsm1/etc/conf/deploymentRegistry.dat
            {
                "name": "TNS_ADMIN",
                "value": "/u02/app/oracle/goldengate/network/admin"
```

tnsnames.oraが、すべてのスタンバイ・データベース・ノードの同じディレクトリにあることを確認します。

2. スタンバイ・システムのoracle OSユーザーとして、ステップに従ってTNS別名の定義を作成し、すべてのデータベース・ノードに分散します。

```
[opc@exastb-node1 ~]$ sudo su - oracle
[oracle@exastb-node1 ~]$ dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/network/admin
[oracle@exastb-node1 ~]$ vi
/u02/app/oracle/goldengate/network/admin/tnsnames.ora
OGGSRV_CDB =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <CDB_SERVICE_NAME>)
       )
    )
OGGSRV_<PDB_NAME> =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
      (CONNECT_DATA =
        (SERVICE_NAME = <PDB_SERVICE_NAME>)
       )
    )
[oracle@exastb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f /u02/app/oracle/goldengate/network/admin/*.ora
 -d /u02/app/oracle/goldengate/network/admin
```

ノート:

Oracle GoldenGate デプロイメントの TNS_ADMIN ディレクトリにある tnsnames.ora または sqlnet.ora を変更した場合は、変更内容を採用するためにデプロイメントを再起動する必要があります。

タスク10 - Oracle GoldenGateプロセスの構成

「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」に記載されているアドバイスに加えて、Extract、分散パスおよびReplicatに関する次の推奨事項に従ってください。

このタスクを完了するには、次のステップを実行します。

- ステップ10.1 - プライマリ・システムのExtract構成の変更
- ステップ10.2 - プライマリおよびスタンバイ・システムの分散パス構成の変更

ステップ10.1 - プライマリ・システムのExtract構成の変更

REDO転送の「最大パフォーマンス」または「最大可用性」モードを使用しているData Guard構成を使用するExtractプロセスについては、プライマリ・システムのExtractプロセス・パラメータ・ファイルに次のパラメータを追加して、トランザクションが失われることで論理データの不整合が発生しないようにする必要があります。

```
TRANLOGOPTIONS HANDLEDLFAILOVER
```

このパラメータにより、まだData Guardスタンバイ・データベースに適用されていないREDOからExtractがトランザクション・データを抽出しないようにします。これは、Oracle GoldenGateがソース・スタンバイ・データベースに存在しないデータをターゲット・データベースにレプリケートしないようにするために重要です。

このパラメータが指定されていないと、ソース・データベースのデータ損失フェイルオーバー後に、ソース・データベースに存在していないデータがターゲット・データベースに保持される可能性があります。それにより、論理データの不整合が発生します。

デフォルトでは、スタンバイ・データベースで適用されたSCN情報を問い合せることができないためにExtractが停止すると、その60秒後に、警告メッセージがExtractレポート・ファイルに書き込まれます。たとえば:

```
WARNING OGG-02721 Extractはスタンバイ・データベースを60秒待機しています。
```

Extractレポート・ファイルに警告メッセージが書き込まれるまでの時間は、ExtractパラメータのTRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNINGを使用すると調整できます。

Extractが30分(デフォルト)後にスタンバイ・データベースによって適用されたSCN情報を問い合せることができない場合、Extractプロセスは異常終了して、次のメッセージがExtractレポート・ファイルに記録されます。

```
ERROR OGG-02722 Extractは、スタンバイ・データベースがアクセス可能になるか、プライマリ・データベースに同期することを1,800秒待機して異常終了しました。
```

デフォルトの30のタイムアウトが満了する前にスタンバイ・データベースが使用可能になると、Extractはソース・データベースからデータのマイニングを続行して、次のメッセージをレポート・ファイルに報告します。

```
INFO OGG-02723 Extractは停止状態から復帰してLCRの処理を開始しました。
```

タイムアウト値の30分は、ExtractパラメータのTRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND <value>を使用すると調整できます。このvalueは、異常終了するまでにスタンバイが使用できない秒数です。

計画メンテナンスの停止時など、スタンバイ・データベースが長時間使用できなくなるときに、Extractでプライマリ・データベースからのデータ抽出を続行する場合は、ExtractパラメータからTRANLOGOPTIONS HANDLEDLFAILOVERパラメータを削除して、Extractを再起動してください。このパラメータは、スタンバイが使用可能になったら必ず設定してください。

ノート:

スタンバイが使用できない間にプライマリ・データベースからの抽出が続行されていると、スタンバイが使用可能になったときにデータ損失フェイルオーバーが発生し、フェイルオーバーの前にすべてのプライマリ REDO が適用されていない可能性もあります。Oracle GoldenGate ターゲット・データベースには、ソース・データベースに存在しないデータが含まれるようになります。

「Cloud: Oracle Exadata Database ServiceでのOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」で説明したように、Extractプロセスに自動再起動プロファイルが割り当てられている場合は、Data Guardのロール・トランジション後にExtractプロセスが自動的に再起動されます。Extractは、デフォルトの5分タイムアウト期間が満了するまで、新しいスタンバイ・データベースの現在の状態を無視して、新しいプライマリ・データベースからのREDOデータのマイニングを続行します。この期間の経過後にスタンバイが使用できない場合、Extractは次のエラーで異常終了します。

`INFO OGG-25053` スタンバイ・データベース復帰のための300秒の待機がタイムアウトです。ただちにHANDLEDLFAILOVERを実施します。

`ERROR OGG-06219` ログマイニング・サーバーOGG$CAP_XXXXXからデータを抽出できません

`ERROR OGG-02078 Extract`は処理スレッドで致命的エラーを検出し、異常終了しています。

Extractは、Oracle GoldenGate Microservicesの自動再起動プロファイルに基づいて、再試行回数に達するか新しいスタンバイ・データベースが使用可能になるまで自動的に再起動し、HANDLEDLFAILOVERタイムアウトに達すると失敗します。

データベース・ロール・トランジション後のタイムアウト期間中、HANDLEDLFAILOVERパラメータは自動的に一時停止されるため、ソース・スタンバイ・データベースが最新の状態に維持されていないことの考慮なしに、データがOracle GoldenGateレプリカ・データベースにレプリケートされます。Extractの異常終了前に起動するスタンバイ・データベースのタイムアウト期間は、ExtractパラメータのTRANLOGOPTIONS DLFAILOVER_TIMEOUTを使用すると調整できます。

DLFAILOVER_TIMEOUTをデフォルトの5分のままにして、古いプライマリがスタンバイに変換できるようにすることをお薦めします。新しいスタンバイ・データベースが長期間使用できなくなるか完全に消失した場合に、Extractを起動して実行状態を維持するには、Extractパラメータ・ファイルからHANDLEDLFAILOVERパラメータを削除する必要があります。このパラメータを削除すると、ExtractはREDOがスタンバイ・データベースに適用されるまで待機することなくデータを抽出します。

スタンバイ・データベースがオンラインに復帰して、プライマリ・データベースからすべてのREDOを適用するまでは、スタンバイ・データベースとOracle GoldenGateレプリカ・データベースの間にデータの相違があります。これは、スタンバイ・データベースが最新状態になると解決されます。その時点で、統合Extractプロセス・パラメータ・ファイルにHANDLEDLFAILOVERパラメータを再追加し、Extractを停止してから再起動します。

プライマリ・データベースが損なわれたときにはブローカがスタンバイ・データベースに自動的にフェイルオーバーできるなど、Oracle Data Guardファスト・スタート・フェイルオーバーが無効になっている場合は、次に示す追加の統合Extractパラメータを指定する必要があります。

`TRANLOGOPTIONS FAILOVERTARGETDESTID n`

このパラメータでは、スタンバイ・データベースにまだ適用されていないREDOデータを抽出しないことに関して、Oracle GoldenGate Extractプロセスを遅らせたままにしておく必要があるスタンバイ・データベースを指定します。

Oracle Data Guardファスト・スタート・フェイルオーバーが無効のときに、追加の統合ExtractパラメータFAILOVERTARGETDESTIDを指定していないと、抽出は次のエラーで異常終了します。

`ERROR OGG-06219` ログマイニング・サーバーOGG$CAP_XXXXXからデータを抽出できません

`ERROR OGG-02078 Extract`は処理スレッドで致命的エラーを検出し、異常終了しています。

FAILOVERTARGETDESTIDの適切な値を判断するには、ソース・スタンバイ・データベースへのREDOの送信に使用されるOracle GoldenGateソース・データベースのLOG_ARCHIVE_DEST_Nパラメータを使用します。たとえば、LOG_ARCHIVE_DEST_2がスタンバイ・データベースを指している場合は2の値を使用します。

プライマリ・システムのoracleユーザーとして、次のコマンドを実行します。

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ sqlplus / as sysdba

SQL> show parameters log_archive_dest
NAME                 TYPE        VALUE
-------------------- ----------- ------------------------------------------------------
log_archive_dest_1   string      location=USE_DB_RECOVERY_FILE_DEST,
                                 valid_for=(ALL_LOGFILES, ALL_ROLES)
log_archive_dest_2   string      service="<db_name>", SYNC AFFIRM delay=0
                                 optional compression=disable max_failure=0
reopen=300
                                 db_unique_name="<db_name>" net_timeout=30,
                                 valid_for=(online_logfile,all_roles)
```

この例では、次のようにExtractパラメータを設定します。

TRANLOGOPTIONS FAILOVERTARGETDESTID 2

Extractパラメータ・ファイルにパラメータを追加するには:

1. ソースOracle GoldenGateのOracle GoldenGate管理サーバーにログインします。
2. 「管理サービス」の「概要」をクリックします。
3. 変更する「Extract」の横にある「アクション」ボタンをクリックします。
4. 「詳細」を選択します。
5. 「パラメータ」タブを選択し、現在のパラメータ・ファイルを編集するための鉛筆アイコンを選択します。
6. TRANLOGOPTIONSパラメータを追加し、変更を保存するために「適用」を選択します。

新しいパラメータを有効にするために、Extractプロセスを停止してから再起動する必要があります。これは管理サーバーを使用して実行できます。

Extract TRANLOGOPTIONSパラメータの詳細は、『Oracle GoldenGateリファレンス』を参照してください。

ステップ10.2 - プライマリおよびスタンバイ・システムの分散パス構成の変更

受信サーバーを実行しているOracle GoldenGate環境のターゲット・データベースがOracle Data Guardで保護されている場合、受信サーバーに証跡ファイルを送信する分散パスには重要な考慮事項があります。Oracle Data Guardロール・トランジション後に受信サーバーが別のシステムに移動される場合は、新しいターゲット・システム・アドレスを反映するように分散パスを変更する必要があります。

分散パスは、受信サーバー・システムのターゲット・データベースでデータベース・ロール・トランジション・トリガーを使用して自動的に変更できます。

プライマリおよびスタンバイ・システムのVIPが異なるルートCA証明書を使用する場合は、「タスク11 - Oracle GoldenGateプロセスの構成」の「ステップ11.3 - 分散パス構成」で説明したように、ソース・デプロイメントのサービス・マネージャにスタンバイ証明書を追加する必要があります。

次の手順に従って、データベース・ロール・トランジション・トリガーを作成します。このトリガーでは、ターゲット・データベースのData Guardロールのトランジション中に受信サーバーがプライマリとスタンバイのシステム間で移動されるときに分散パスのターゲット・アドレスを変更します。

次のサブステップを実行して、このステップを完了します。

- ステップ10.2.1 - 分散パスを変更するシェル・スクリプトの作成
- ステップ10.2.2 - DBMS_SCHEDULERジョブの作成
- ステップ10.2.3 - デプロイメント構成ファイルの作成
- ステップ10.2.4 - データベース・ロール・トランジション・トリガーの作成

ステップ10.2.1 - 分散パスを変更するシェル・スクリプトの作成

「分散パス・ターゲット変更スクリプトの例」には、分散パス・ターゲット・アドレスの変更に使用できるシェル・スクリプトの例が記載されています。適切な変数値の設定については、スクリプト例のコメントを参照してください。

ノート:

スクリプトは、TARGET プライマリおよびスタンバイ・データベース・システムのすべての Oracle RAC ノード上の同じローカル・ディレクトリに配置する必要があります。スクリプト・ファイルの権限は 6751 に設定します。

TARGETプライマリとスタンバイ・システムのoracle OSユーザーとして、ステップに従ってスクリプトchange_path_target.shを作成して配布します。

```
[opc@exadb-node1 ~]$ sudo su – oracle
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group mkdir
 -p /u02/app/oracle/goldengate/scripts
[oracle@exadb-node1 ~]$ vi /u02/app/oracle/goldengate/scripts/change_path_target.sh
# Paste the script from Example Distribution Path Target Change Script
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f /u02/app/oracle/goldengate/scripts/change_path_target.sh
 -d /u02/app/oracle/goldengate/scripts
```

ステップ10.2.2 - DBMS_SCHEDULERジョブの作成

PL/SQL内からオペレーティング・システム・シェル・スクリプトを実行するために、DBMS_SCHEDULERジョブを作成する必要があります。

1. TARGETプライマリ・システムのoracle OSユーザーとして、ルート・コンテナ・データベース(CDB)内でSYSDBAユーザーとしてのスケジューラ・ジョブを作成します。

   ```
   [opc@exapri-node1 ~]$ sudo su - oracle
   [oracle@exapri-node1 ~]$ source <db_name>.env
   [oracle@exapri-node1 ~]$ sqlplus / as sysdba
   SQL> exec dbms_scheduler.create_job(job_name=>'gg_change_path_target',
    job_type=>'EXECUTABLE', number_of_arguments => 6,
    job_action=>'/u02/app/oracle/goldengate/scripts/change_path_target.sh',
    enabled=>FALSE);
   ```

   外部ジョブを実行するには、$ORACLE_HOME/rdbms/admin/externaljob.oraファイルのパラメータrun_userとrun_groupをOracleデータベースのオペレーティング・システムのユーザーとグループに設定する必要があります。

2. TARGETプライマリおよびスタンバイ・システムのroot OSユーザーとして、ファイルexternaljob.oraを作成します。

   ```
   [opc@exadb-node1 ~]$ sudo su –
   [root@exadb-node1 ~]# export DB_NAME=<database_name>
   [root@exadb-node1 ~]# dbaascli database getDetails
    --dbname $DB_NAME |grep homePath |uniq
         "homePath" : "/u02/app/oracle/product/19.0.0.0/dbhome_1",
   [root@exadb-node1 ~]# vi
   ```

```
 /u02/app/oracle/product/19.0.0.0/dbhome_1/rdbms/admin/externaljob.ora
# Before
run_user = nobody
run_group = nobody
# After
run_user = oracle
run_group = oinstall
```

3. このステップは、プライマリおよびスタンバイ・システムのすべてのノードで繰り返します。

ノート:

extrernaljob.ora は、プライマリおよびスタンバイ・データベース・システムのすべての Oracle RAC ノードで構成する必要があります。

ステップ10.2.3 - デプロイメント構成ファイルの作成

このシェル・スクリプト例では、REST APIコールを使用してOracle GoldenGate分散パスにアクセスします。このREST API コールをセキュアにするために、curlで読み取る構成ファイルにはユーザー名とパスワードを含めることをお薦めします。

TARGETプライマリおよびスタンバイ・システムのoracle OSユーザーとして、デプロイメント資格証明が含まれている構成ファイルを作成します。

```
[opc@exadb-node1 ~]$ sudo su – oracle
[oracle@exadb-node1 ~]$
cat > /u02/app/oracle/goldengate/scripts/<INSTANCE_NAME>.cfg << EOF
                user = "oggadmin:<password>"
EOF
[oracle@exadb-node1 ~]$ chmod 600
/u02/app/oracle/goldengate/scripts/<INSTANCE_NAME>.cfg
[oracle@exadb-node1 ~]$ /usr/local/bin/dcli -l oracle -g ~/dbs_group
 -f /u02/app/oracle/goldengate/scripts/<INSTANCE_NAME>.cfg
 -d /u02/app/oracle/goldengate/scripts
```

ステップ10.2.4 - データベース・ロール・トランジション・トリガーの作成

スタンバイ・データベースがプライマリ・データベースになったときに起動するOracle GoldenGateソース・データベースでロール・トランジション・トリガーを作成し、分散パスのターゲット・アドレスを変更します。

TARGETプライマリ・システムのoracle OSユーザーとして、次のSQL文を実行してロール・トランジション・トリガーを作成します。

```
[opc@exapri-node1 ~]$ sudo su - oracle
[oracle@exapri-node1 ~]$ source <db_name>.env
[oracle@exapri-node1 ~]$ sqlplus / as sysdba
CREATE OR REPLACE TRIGGER gg_change_path
AFTER db_role_change ON DATABASE
declare
  role varchar2(30);
  hostname varchar2(64);
begin
  select database_role into role from v$database;
  select host_name into hostname from v$instance;
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',1,'<PRIMARY Source
VIP');
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',2,'<STANDBY Source
VIP');
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',4,'<Distribution path
name');
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',5,'<Instance name>'
  DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',6,'<Config file
containing the deployment credentials>');
  if role = 'PRIMARY' and hostname like '<primary target cluster name>%'
  then
```

```
    DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'<PRIMARY Target
VIP>:443');
  elsif role = 'PRIMARY'
  then
    DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'<STANDBY Target
VIP>:443');
  end if;
  DBMS_SCHEDULER.RUN_JOB(job_name=>'gg_change_path_target');
end;
/
```

ステップ10.3 - プライマリ・システムのReplicat構成

「タスク11 - Oracle GoldenGateプロセスの構成」の「ステップ11.4 - Replicatの構成」で説明したように、すべてのOracle GoldenGate Replicatプロセスにはターゲット・データベースのチェックポイント表が必要です。Oracle Data Guardで構成したReplicatには、その他の構成要件はありません。

分散パス・ターゲット変更スクリプトの例

次のスクリプト例を使用すると、ソースのOracle GoldenGateデプロイメント分散のターゲット・アドレスを変更して、Oracle Data Guardロールの移行後の受信側サーバーの新しい場所を反映させることができます。この例では、ソースのOracle GoldenGateデプロイメントがData Guardを使用したMAAアーキテクチャで構成されており、分散サーバーがプライマリ・システムとスタンバイ・システムの間で再配置できることを前提としています。

```
#!/bin/bash
# change_path_target.sh - changes the target host of a GG Distribution Path
#   when the target moves between primary/standby systems.
# Example usage:
# ./change_path_target.sh <primary source VIP>:443 <standby source VIP>:443
# <path target VIP> <path name> <deployment name> <credentials file>
SOURCE1=$1      # PRIMARY Distribution Server VIP
SOURCE2=$2      # STANDBY Distribution Server VIP
TARGET=$3       # Distribution path target VIP
DPATH=$4        # Distribution path name
DEP=$5          # Deployment name
ACCESS=$6       # Config file containing the deployment credentials.
                # Example contents:
                # user = "oggadmin:<password>"
CONNECT=0
#echo "#${i} - `date`:"
LOGFILE=/tmp/ogg_dpatch_change.txt
result=$(curl -si -K
 $ACCESS https://$SOURCE1/$DEP/distsrvr/services/v2/sources/$DPATH
 -X GET| grep HTTP |  awk '{print $2}')
# Will return NULL of nginx not running, 502 if cannot contact server, 200 if
# contact to server good, and others (404) for other bad reasons:
if [[ -z $result || $result -ne 200 ]]; then  # Managed to access the Distr Server
  echo "`date` - Couldn't contact Distribution Server at $SOURCE1
 Deployment $DEP ****" >> $LOGFILE
else # Try the other source host:
  echo "`date` - Got status of Distribution Server at $SOURCE1 Deployment
 $DEP ***"  >> $LOGFILE
  SOURCE=$SOURCE1
  CONNECT=1
fi
if [ $CONNECT -eq 1 ]; then
# For secure NGINX patch destination (wss)

PAYLOAD='{"target":{"uri":"wss://'${TARGET}'/services/ggnorth/v2/targets?trail=bb"}}'
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X PATCH --data '{"status": "stopped"}'
# Set new target for path:
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X PATCH --data "$PAYLOAD"
  echo "`date` - Set path $DPATH on $SOURCE deployment $DEP:" >> $LOGFILE
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X GET | python -m json.tool | grep uri >> $LOGFILE
  curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH
 -X PATCH --data '{"status": "running"}'
  exit 0
else
  echo "`date` - ERROR: COULDN'T CHANGE DISTRIBUTION PATH ($DPATH) in Deployement
 $DEP at $SOURCE! ***" >> $LOGFILE
fi
# If here, means we couldn't connect to either Distribution Servers
exit 1
```

22 オンプレミス: Oracle Real Application Clustersを使用したOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス

これらのベスト・プラクティスを使用して、Oracle Real Application Clusters (RAC)、Oracle Clusterware、およびOracle Database File System (DBFS)またはOracle Advanced Cluster File System (ACFS)と連携するようにOracle Exadataなどのオンプレミス・システム用のOracle GoldenGate Microservices Architectureを構成します。

Oracle GoldenGate Microservices Architectureをホストするターゲット Oracle RACシステムは、Oracle GoldenGate操作のソース・データベースとして、ターゲット・データベースとして、または場合によってはソース・データベースとターゲット・データベースの両方として機能できます。

次の項を参照してください。

- Oracle RACにOracle GoldenGateをデプロイする場合の推奨事項のサマリー
- タスク1: Oracle GoldenGate用のOracleデータベースの構成
- タスク2: データベース・レプリケーション管理者ユーザーの作成
- タスク3: データベース・サービスの作成
- タスク4: Oracle RACでのファイル・システムの設定
- タスク5: Oracle GoldenGateのインストール
- タスク6: Oracle GoldenGateデプロイメントの作成
- タスク7: Oracle Clusterwareの構成
- タスク8: NGINXリバース・プロキシの構成
- タスク9: Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成
- タスク10: Oracle GoldenGateプロセスの構成
- タスク11: ExtractプロセスとReplicatプロセスの自動起動の構成

Oracle RACにOracle GoldenGateをデプロイする場合の推奨事項のサマリー

Oracle RAC環境でOracle GoldenGateを構成する場合は、次の推奨事項に従います。

- 各Oracle RACノードでローカルに最新バージョンのOracle GoldenGateソフトウェアをインストールしてます。必ず、すべてのOracle RACノードでそのソフトウェアの場所が同じになるようにしてください。
- Oracle GoldenGateファイル(証跡ファイル、チェックポイント・ファイル、一時ファイル、レポート・ファイルおよびパラメータ・ファイル)を格納するファイル・システムに、Oracle Database File System (DBFS)またはOracle Advanced Cluster File System (ACFS)を使用します。
- Oracle GoldenGateが実行される可能性があるOracle RACノードすべてで、同じDBFSまたはACFSマウント・ポイントを使用します。
- GoldenGateデプロイメントの作成時に、デプロイメントの場所にDBFSかACFSを指定します。
- Oracle GoldenGateが実行されるOracle RACノードすべてに、Grid Infrastructure Agent (XAG)バージョン10以降をインストールします。
- GoldenGateプロセスを、そのデプロイメントの起動時に自動的に起動および再起動されるように構成します。

タスク1: Oracle GoldenGate用のOracleデータベースの構成

ソースおよびターゲットのOracle GoldenGateデータベースは、次に示す推奨事項を使用して構成する必要があります。

- データベース初期化パラメータENABLE_GOLDENGATE_REPLICATION=TRUEを設定することで、Oracle GoldenGateレプリケーションを有効にします。
- Oracle GoldenGateソース・データベースをARCHIVELOGモードで実行します。
- Oracle GoldenGateソース・データベースでFORCE LOGGINGモードを有効にします。
- ソース・データベースで最小サプリメンタル・ロギングを有効にします。さらに、すべてのレプリケートされたオブジェクトのスキーマまたは表レベルのログを追加します
- Replicatプロセスを使用する場合は、STREAMS_POOL_SIZE初期化パラメータを使用して、ソース・データベースにあるシステム・グローバル領域(SGA)内のストリーム・プールを構成します。

  なお、統合Replicatを使用する場合は、ターゲット・データベースでのみストリーム・プールが必要です。

  次の式を使用して、STREAMS_POOL_SIZEの値を決定します。

  STREAMS_POOL_SIZE = (#Extracts and #Integrated Replicats * 1GB) * 1.25

  たとえば、2つのExtractと2つの統合Replicatがあるデータベースの場合は、次のようになります。

  STREAMS_POOL_SIZE = 4GB * 1.25 = 5GB

  ExtractプロセスまたはReplicatプロセスを追加する場合は、新しいストリーム・プール・サイズ要件を再計算し構成することが重要です。

Oracle GoldenGate用にデータベースを準備する方法の詳細は、Oracle GoldenGate用のデータベースの準備を参照してください。

タスク2: データベース・レプリケーション管理者ユーザーの作成

ソースおよびターゲットのOracleデータベースには、適切な権限が割り当てられたGoldenGate管理者ユーザーが必要です。

単一テナント(非CDBアーキテクチャ)データベースについては、Oracle GoldenGate資格証明の確立を参照してください

マルチテナント・ソース・データベースの場合は、GoldenGate Extractを、c##アカウントを使用してルート・コンテナ・データベース内のユーザーに接続するように構成する必要があります。マルチテナント・ターゲット・データベースの場合は、Replicatでデータが適用される各PDBに、別個のGoldenGate管理者ユーザーが必要です。

Oracleマルチテナント・データベースでのGoldenGate管理者の作成の詳細は、マルチテナント・コンテナ・データベースでのOracle GoldenGateの構成を参照してください。

タスク3: データベース・サービスの作成

データベースのオープン時にOracle Grid Infrastructure AgentでGoldenGateデプロイメントが自動的に起動されるようにするには、データベース・サービスが必要です。共有ファイル・システムにDBFSを使用している場合は、データベース・サービスを使用してDBFSを正しいOracle RACインスタンスにマウントすることもできます。

ソース・マルチテナント・データベースを使用する場合は、ルート・コンテナ・データベース(CDB)とレプリケートされるスキーマを含むプラガブル・データベース(PDB)に、個別のサービスが必要です。ターゲット・マルチテナント・データベースの場合は、PDBに単一のサービスが必要です。

oracleユーザーとして、次のコマンドを使用してサービスを作成します。

```
$ srvctl add service -db db_name -service service_name
 -preferred instance_1 -available instance_2, instance_3 etc.
 -pdb PDB_name
```

たとえば:

```
$ srvctl add service -db ggdb -service oggserv_pdb -preferred ggdb1
    -available ggdb2 –pdb GGPDB01
```

Oracleマルチテナント・データベースを使用していない場合は、-pdbパラメータを省略します。

タスク4: Oracle RACでのファイル・システムの設定

Oracle GoldenGate Microservices Architectureの設計では、インストールとデプロイメントのディレクトリ構造が簡略化されています。インストール・ディレクトリは、ソフトウェア・パッチ適用中の停止時間を最小限に抑えるために、各Oracle RACノードでローカル・ストレージに配置する必要があります。

デプロイメント・ディレクトリ,は、Oracle GoldenGate Configuration Assistant (oggca.sh)を使用したデプロイメントの作成時に作成されます。このディレクトリは、共有ファイル・システムに配置する必要があります。デプロイメント・ディレクトリには、構成、セキュリティ、ログ、パラメータ、証跡およびチェックポイントのファイルが含まれます。

DBFSまたはACFSにデプロイメントを配置すると、システム障害が発生した場合に最適なリカバリ機能とフェイルオーバー機能が提供されます。クラスタ全体でチェックポイント・ファイルの可用性を確保することは、障害発生後にGoldenGateプロセスで最後に認識された位置から実行を継続できるようにするために不可欠です。

Oracle Data GuardとともにOracle GoldenGateを構成する場合、推奨のファイル・システムはDBFSです。DBFSは、Data Guardで保護されているデータベースに含まれ、XAGと十分に統合できます。Data Guardロール・トランジションが発生した場合、このファイル・システムは新しいプライマリ・サーバーに自動的にマウントされ、その後でOracle GoldenGateが自動起動されます。ACFSはOracle Data Guard構成の一部ではないため、現在これは、ACFSの場合はできません。

DBFSまたはACFSのファイル・システムを構成するには、次の該当する項の手順に従います。

Oracle Database File System(DBFS)

DBFS表領域は、Oracle GoldenGateプロセスで接続しているのと同じデータベース内に作成する必要があります。たとえば、GoldenGate統合ExtractプロセスでGGDBというデータベースから抽出されている場合は、同じGGDBデータベースにDBFS表領域を配置します。

必要なFUSEライブラリがまだインストールされていない場合は、My Oracle Supportのノート869822.1の手順に従ってそれらをインストールします。

My Oracle Supportのノート1054431.1の手順を使用して、データベース、表領域、データベース・ユーザー、tnsnames.oraのOracle Net接続別名、およびDBFSに必要なソースまたはターゲットGoldenGate環境に対する権限を構成します。

ノート:

Oracle マルチテナント・データベースを使用する場合は、プラガブル・データベース(PDB)内に DBFS 表領域を作成する必要があります。GoldenGate Extract プロセスまたは Replicat プロセスで接続しているのと同じ PDB を使用することをお薦めします。これにより、DBFS で、そのデータベース依存関係に、前述のタスク 2 で作成したのと

同じデータベース・サービスを使用できるようになります。

GoldenGateデプロイメント・ファイルを格納するためのファイル・システムを作成するときは、最長12時間の証跡ファイルを格納できる十分な証跡ファイル用ディスク領域を割り当てることをお薦めします。これにより、新しい証跡ファイルを受信できなくなるような問題がターゲット環境で発生した場合の証跡ファイル生成のための十分な領域が確保されます。12時間に必要な領域の量は、実際の本番データで証跡ファイルの生成率をテストすることでのみ決定できます。

DBFS作成の例:

```
$ cd $ORACLE_HOME/rdbms/admin
$ sqlplus dbfs_user/dbfs_password@database_tns_alias
SQL> start dbfs_create_filesystem dbfs_gg_tbs goldengate
```

My Oracle Supportのノート1054431.1の手順に従って、新しく作成されたDBFSファイル・システムを、ノード障害発生後にDBFSインスタンスおよびマウント・ポイント・リソースがCluster Ready Services (CRS)によって自動的に起動されるように構成し、次のようにDBFS構成およびスクリプト・ファイルを変更します。

1. ご使用のデータベース環境を反映するように`mount-dbfs.conf`パラメータを変更します。

   `MOUNT_OPTIONS`パラメータを次のように変更します。

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

   `failover`オプションは、すべてのファイル書込みを`IMMEDIATE WAIT`モードでDBFSデータベースにコミットすることを強制します。これにより、データベースまたはノードの障害の発生時に、データが、`dbfs_client`キャッシュには書き込まれたがデータベースにはまだ書き込まれていなくても、失われることがなくなります。

   `nolock`マウント・オプションは、Oracle Database 18c以降のリリースを使用している場合に必要になります。これは、DBFSファイル・ロック(これにより、ファイルが現在ロックされているときにOracle RACノードの障害が発生するとGoldenGateプロセスに問題が生じる可能性がある)に変更があったためです。

   Oracle Database 12cリリース2 (12.2)の`dbfs_client`を使用している場合は、バグ27056711の修正を含む最新のリリース更新が適用されていることを確認してください。この修正を適用すると、`MOUNT_OPTIONS`に`nolock`オプションも含まれます。

2. CRSリソースの停止時にDBFSを強制的にアンマウントするように、`mount-DBFS.sh`スクリプトを変更します。

   次の2つの出現箇所を変更します。

   ```
   $FUSERMOUNT -u $MOUNT_POINT
   ```

   これを次のように変更します。

   ```
   $FUSERMOUNT -uz $MOUNT_POINT
   ```

3. このリソースをOracle Clusterwareに登録するときは、My Oracle Supportのノートで示されているように、`local_resource`ではなく必ず`cluster_resource`としてそれを作成してください。

   `cluster_resource`を使用するのは、ファイル・システムを一度に1つのノードにのみマウントできるようにして、同時に複数のノードでDBFSがマウントされる(ファイル書込みが同時に発生する可能性が生じて、ファイルが破損する原因になる)ことを防ぐためです。

   DBFSサービスの依存関係には、前のステップで作成したデータベース・サービス名を使用してください。

   たとえば:

   ```
   DBNAME=ggdb
   DEPNAME=ora.$DBNAME.oggserv.svc
   ```

```
crsctl add resource $RESNAME ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=$ACTION_SCRIPT, ¥
 CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, ¥
START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',¥
 STOP_DEPENDENCIES='hard($DEPNAME)',¥
 SCRIPT_TIMEOUT=300"
```

DBFSリソースを作成した後は、ファイル・システムをマウントしテストする必要があります。

```
$ crsctl start res dbfs_mount
$ crsctl stat res dbfs_mount
```

ファイル・システムをマウントした後は、GoldenGateファイルを格納するディレクトリを作成します。

```
$ cd /mnt/dbfs/goldengate
$ mkdir deployments
```

ノート:

共有ファイル・システムは、マウントされたままにします。これは、後のステップで GoldenGate デプロイメントを作成するために必要です。

Oracle Advanced Cluster File System (ACFS)

Oracle ACFSは、Oracle RAC構成での共有GoldenGateファイルのための、DBFSの代替です。

Oracle Exadata Database MachineのACFS構成要件の詳細は、My Oracle Supportのノート1929629.1を参照してください。

Oracleのデプロイメント・ファイルを格納するための単一のACFSファイル・システムを作成します。

最長12時間の証跡ファイルを格納できる十分な証跡ファイル用ディスク領域を割り当てることをお薦めします。これにより、新しい証跡ファイルを受信できなくなるような問題がターゲット環境で発生した場合の証跡ファイル生成のための十分な領域が確保されます。12時間に必要な領域の量は、実際の本番データで証跡ファイルの生成率をテストすることでのみ決定できます。

1. Oracle ASM管理者ユーザーとしてASMCMDを使用してファイル・システムを作成します。

```
ASMCMD [+] > volcreate -G datac1 -s 1200G ACFS_GG
```

ノート:

決定したサイズ要件に応じて、ファイル・システムのサイズを変更します。

```
ASMCMD> volinfo -G datac1 acfs_gg
Diskgroup Name: DATAC1
 Volume Name: ACFS_GG
 Volume Device: /dev/asm/acfs_gg-151
 State: ENABLED
 Size (MB): 1228800
 Resize Unit (MB): 64
 Redundancy: MIRROR
 Stripe Columns: 8
 Stripe Width (K): 1024
 Usage:
 Mountpath:
```

次のmkfsコマンドを使用してファイル・システムを作成します。

```
$ /sbin/mkfs -t acfs /dev/asm/acfs-gg-151
```

2. 新しく作成したACFSファイル・システムのCRSリソースを作成します(まだ作成されていない場合)。

   ファイル・システム・リソースがすでに作成されているかどうかを確認します。

```
$ srvctl status filesystem -volume ACFS_GG -diskgroup DATAC1
ACFS file system /mnt/acfs_gg is mounted on nodes oggadm07, oggadm08
```

   まだ作成されていない場合は、すべてのOracle RACノードでACFSマウント・ポイントを作成します。

```
# mkdir -p /mnt/acfs_gg
```

   rootユーザーとして、ファイル・システム・リソースを作成します。ACFSでの分散ファイル・ロックの実装により、DBFSとは異なり、一度に複数のRACノードにACFSをマウントできるようになっています。

   Oracle Grid InfrastructureのORACLE_HOMEから、srvctlを使用してACFSリソースを作成します。

```
# srvctl add filesystem -device /dev/asm/acfs_gg-151 -volume ACFS_GG
-diskgroup DATAC1 -path /mnt/acfs_gg -user oracle -autostart RESTORE
```

   現在構成されているACFSファイル・システムを確認するには、次のコマンドを使用してファイル・システムの詳細を表示します。

```
$ srvctl config filesystem
Volume device: /dev/asm/acfs_gg-151
Diskgroup name: datac1
Volume name: ACFS_GG
Canonical volume device: /dev/asm/acfs_gg-151
Accelerator volume devices:
Mountpoint path: /mnt/acfs_gg
Mount point owner: oracle
Check the status of the ACFS resource and mount it.
$ srvctl status filesystem -volume ACFS_GG -diskgroup DATAC1
ACFS file system /mnt/acfs is not mounted
$ srvctl start filesystem -volume ACFS_GG -diskgroup DATAC1 –node dc1north01
```

   作成されたCRSリソースには、ora.diskgroup_name.volume_name.acfsという形式で名前が付けられます。前述のファイル・システムの例を使用すると、CRSリソースの名前は`ora.datac1.acfs_gg.acfs`になります。

   現在存在しているすべてのACFSファイル・システムのCRSリソースを表示するには、次のコマンドを使用します。

```
$ crsctl stat res -w "((TYPE = ora.acfs.type) OR (TYPE =
ora.acfs_cluster.type))"
NAME=ora.datac1.acfs_gg.acfs
TYPE=ora.acfs.type
TARGET=ONLINE              , OFFLINE
STATE=ONLINE on dc1north01, OFFLINE
```

3. ACFSにGoldenGateデプロイメント・ディレクトリを作成します。

   ファイル・システムをマウントした後は、GoldenGateデプロイメントを格納するディレクトリを作成します。

```
$ cd /mnt/acfs_gg
$ mkdir deployments
```

ノート:

共有ファイル・システムは、マウントされたままにします。これは、後のタスクで GoldenGate デプロイメントを作成

するために必要です。

タスク5: Oracle GoldenGateのインストール

Oracle GoldenGate 21c Microservicesソフトウェアまたはそれ以降のリリースをダウンロードしインストールします。

https://www.oracle.com/middleware/technologies/goldengate-downloads.htmlでこのソフトウェアをダウンロードします。

GoldenGate構成の一部となるOracle RAC構成内のすべてのノードでローカルにOracle GoldenGateソフトウェアをインストールします。すべてのノードでインストール・ディレクトリが同一であることを確認します。

Oracle GoldenGate Microservicesのドキュメントに記載されている一般的なインストール手順に従います。

タスク6: Oracle GoldenGateデプロイメントの作成

Oracle GoldenGateソフトウェアをインストールした後のステップでは、Oracle GoldenGate Configuration Assistant (oggca)を使用してデプロイメントを作成します。

現時点では、Oracle GoldenGateとXAGには2つの制限があります。

1. XAGに登録されているサービス・マネージャごとに、単一のデプロイメントのみを管理できます。複数のデプロイメントが必要な場合は、それぞれのデプロイメントに専用のサービス・マネージャを使用する必要があります。Oracle GoldenGateリリース21cでは、様々なバージョンのOracle Databaseに接続するExtractプロセスおよびReplicatプロセスをサポートする単一のデプロイメントが使用されるため、この要件が簡素化されています。
2. XAGに登録されている各サービス・マネージャは、別々のOGG_HOMEソフトウェア・インストール・ディレクトリに属している必要があります。Oracle GoldenGateを複数回インストールするのではなく、Oracle GoldenGateを1回インストールし、その後はサービス・マネージャのOGG_HOMEごとにシンボリック・リンクを作成することをお薦めします。

たとえば:

```
$ echo $OGG_HOME
/u01/oracle/goldengate/gg21c_MS
$ ln –s /u01/oracle/goldengate/gg21c_MS /u01/oracle/goldengate/gg21c_MS_ggnorth
$ export OGG_HOME=/u01/oracle/goldengate/gg21c_MS_ggnorth
$ $OGG_HOME/bin/oggca.sh
```

このシンボリック・リンクおよびOGG_HOME環境変数は、すべてのOracle RACノードで、Oracle GoldenGate Configuration Assistantを実行する前に構成する必要があります。

Oracle GoldenGate Configuration AssistantでのGoldenGateデプロイメントの作成に関する推奨事項は次のとおりです。

1. 「サービス・マネージャ・オプション」で、新しいサービス・マネージャの作成のために次の内容を指定します。
    a. 「サービス・マネージャの詳細」ペインで、「新規サービス・マネージャの作成」を選択します。
    b. 共有DBFSまたはACFSファイル・システム上のサービス・マネージャ・デプロイメント・ホームの場所を入力します。
    c. 「XAGと統合」を選択します。
    d. 「サービス・マネージャ接続の詳細」ペインで、「リスニング・ホスト名/アドレス」フィールドにlocalhostと指定します。

localhostを使用すると、仮想IPアドレス(VIP)を必要とせずに、すべてのOracle RACノードでこのデプロイメントを起動できます。

e. 「リスニング・ポート」にポート番号を入力します。

2. 「デプロイメント・ディレクトリ」で、共有DBFSまたはACFSファイル・システム上のデプロイメント・ホーム・ディレクトリを指定します。

3. 「環境変数」で、正しいTNS_ADMINディレクトリを指定します。

Oracle GoldenGateリリース21.3以降では、必要なデータベース・ライブラリがOracle GoldenGateインストールの一部としてインストールされるため、データベースのORACLE_HOMEは不要になりました。既存のORACLE_HOMEディレクトリの外部にあるTNS_ADMINディレクトリを使用することをお薦めします。

4. 「セキュリティ・オプション」で「SSL/TLSセキュリティ」を選択しないでください。

Oracle GoldenGate Microservicesサーバーへの外部アクセスは、NGINXリバース・プロキシでのSSL終了を使用することで実現されます。GoldenGateデプロイメントへのセキュアなアクセスおよび通信は、SSLポート443を介して排他的に実行されます。同じローカル・ホスト内のNGINXとGoldenGateとの内部接続には、SSLは必要ありません。

5. Management Pack for Oracle GoldenGateがライセンス付与されている場合は、「ポート設定」で、「モニタリングの有効化」を選択して、Berkeley Database (BDB)またはLightening Memory Database (LMDB)を使用してパフォーマンス・メトリック・サーバーを使用します。

BDBとLMDBの両方のメトリック・サービス・データストア・タイプについて、「メトリック・サービス・データストア・ホーム」のディレクトリを、すべてのOracle RACノードに存在するローカル・ディレクトリに設定します。例: /u01/oracle/goldengate/datastores/デプロイメント名

6. デプロイメントが作成されるまで、Oracle GoldenGate Configuration Assistantの使用を続行します。

7. 共有ファイル・システムにDBFSを使用しておりデータベース・バージョンがOracle Databaseリリース21c (21.3)より前のリリースである場合は、デプロイメントの作成後に、次のコマンドを実行してOracle GoldenGateデプロイメントの一時ディレクトリをDBFSからローカル・ストレージに移動します。

最初のノードでは、次のようにします。

```
$ cd <DBFS GoldenGate deployment home directory/var
$ mkdir -p local_storage_directory/deployment_name
$ mv temp local_storage_directory/deployment_name
$ ln -s local_storage_directory/deployment_name/temp temp
```

その他すべてのノードでは、次のようにします。

```
$ mkdir local_storage_directory/deployment_name/temp
```

最初のノードの例:

```
$ cd /mnt/dbfs/goldengate/deployments/ggnorth/var
$ mkdir –p /u01/oracle/goldengate/deployments/ggnorth
$ mv temp /u01/oracle/goldengate/deployments/ggnorth
$ ln -s /u01/oracle/goldengate/deployments/ggnorth/temp temp
```

その他すべてのノードでは、次のようにします。

```
$ mkdir /u01/oracle/goldengate/deployments/ggnorth/temp
```

タスク7: Oracle Clusterwareの構成

次の手順では、Oracle Grid Infrastructureスタンドアロン・エージェント(XAG)を使用してOracle GoldenGateを管理するようにOracle Clusterwareを構成する方法を示します。

XAGを使用することで、Oracle RACノード間の再配置時に、共有ファイル・システム(DBFSまたはACFS)のマウントとGoldenGateデプロイメントの停止および起動を自動化します。

1. Oracle Grid Infrastructure Standalone Agentをインストールします。

   XAGソフトウェアはスタンドアロン・エージェントとしてGrid InfrastructureのORACLE_HOMEの外部にインストールすることをお薦めします。これにより、入手可能な最新のXAGリリースを使用できるようになり、Grid Infrastructureに影響を与えることなくソフトウェアを更新できます。

   Oracle GoldenGate Microservices Architectureを使用する場合は、XAGバージョン10.2以降を使用する必要があります。

   最新のエージェント・ソフトウェアは、次の場所からダウンロードできます。

   http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/xag-agents-downloads-3636484.html

   Oracle Grid Infrastructureホーム・ディレクトリの外部に、XAGスタンドアロン・エージェントをインストールします。XAGは、Oracle GoldenGateがインストールされているクラスタ内のすべてのRACノード上の同じディレクトリにインストールする必要があります。

   たとえば、Oracle Grid Infrastructureユーザーとして、デフォルトのoracleを使用した場合:

   ```
   $ ./xagsetup.sh --install --directory /u01/oracle/xag --all_nodes
   ```

   oracleユーザーがマシンにログオンしたときにagctlの場所がわかるように、新しくインストールしたXAGソフトウェアの場所をPATH変数に追加します。

   ```
   $ cat .bashrc
   export PATH=/u01/oracle/xag/bin:$PATH
   ```

   > ノート:
   >
   > 正しい agctl バイナリが見つかるように、Grid Infrastructure の bin ディレクトリの前に XAG の bin ディレクトリが指定されていることを必ず確認してください。ログオン時に有効になるように、oracle ユーザー環境でこの場所を設定します(たとえば、Bash シェル使用時には.bashrc ファイル内)。

2. アプリケーション仮想IPアドレス(VIP)の作成を準備します。

   専用のアプリケーションVIPは、どのOracle RACノードでサービスがホストされていても同じホスト名を使用してGoldenGate Microservicesにアクセスできるようにするために必要です。また、アプリケーションVIPにより、GoldenGate Distribution Serverが、現在のOracle RACノードを実行しているDistribution Receiverと通信できるようになります。

   VIPは、Oracle Clusterwareが管理するクラスタ・リソースです。VIPはクラスタ・ノードに割り当てられ、ノード障害が発生すると自動的に別のノードに移行されます。

   アプリケーションVIPを作成する前に、次の2つの情報が必要です。

- ネットワーク番号。次のコマンドを使用して特定できます。

```
$ crsctl status resource -p -attr ADDRESS_TYPE,NAME,USR_ORA_SUBNET -w
"TYPE = ora.network.type" |sort | uniq
ADDRESS_TYPE=IPV4
NAME=ora.net1.network
USR_ORA_SUBNET=10.133.16.0
```

  NAME=ora.net1.networkでのnet1は、これがネットワーク1であり、タイプがIPV4であることを示します。

- システム管理者から提供された、新しいアプリケーションVIPのIPアドレス。このIPアドレスは、前述のクラスタ環境と同じサブネット内にある必要があります。

  このVIPは、Oracle Grid Infrastructure Agentを構成するときに、次のステップで作成されます。

3. Oracle Grid Infrastructure Agent (XAG)を構成します。

Oracle GoldenGateは、データベースの起動時およびファイル・システムのマウント時に自動的にデプロイメントが起動および停止されるように、XAGに登録しておく必要があります。

Oracle GoldenGate Microservices ArchitectureをXAGに登録するには、次のコマンド形式を使用します。

```
agctl add goldengate instance_name
 --gg_home GoldenGate_Home
 --service_manager
 --config_home GoldenGate_SvcMgr_Config
 --var_home GoldenGate_SvcMgr_Var_Dir
 --port port_number
 --oracle_home $OGG_HOME/lib/instantclient
 --adminuser OGG_admin_user
 --user GG_instance_user
 --group GG_instance_group
 --network network_number
 --ip ip_address
 --vip_name vip_name
 --filesystems CRS_resource_name
 --db_services service_name
 --use_local_services
 --nodes node1, node2, ... ,nodeN
```

説明:

--gg_homeでは、GoldenGateソフトウェアの場所を指定します。複数のサービス・マネージャを登録する場合は、OGG_HOMEにOGG_HOMEシンボリック・リンクを指定します(「タスク6: Oracle GoldenGateデプロイメントの作成」を参照)。

--service_managerでは、これがGoldenGate Microservicesインスタンスであることを示します。

--config_homeでは、GoldenGateサービス・マネージャのデプロイメント構成のホーム・ディレクトリを指定します。

--var_homeでは、GoldenGateサービス・マネージャのデプロイメント変数のホーム・ディレクトリを指定します。

--portでは、デプロイメント・サービス・マネージャのポート番号を指定します。

--oracle_homeでは、Oracle GoldenGate 21c以降のリリースの一部として含まれるOracleデータベース・ライブラリの場所を指定します。例: $OGG_HOME/lib/instantclient

--adminuserでは、Oracle GoldenGate Microservices管理者アカウント名を指定します。

--userでは、GoldenGateデプロイメント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します。

--groupでは、GoldenGateデプロイメントを所有するオペレーティング・システム・グループの名前を指定します。

--networkでは、前述の、VIPのネットワーク・サブネットを指定します。

--IPでは、前述の、VIPのIPアドレスを指定します。VIPをすでに作成してある場合は、--networkと--ipのかわりに--vip_nameパラメータを使用してそれを指定します。

--vip_nameでは、以前に作成したアプリケーションVIPのCRSリソース名を指定します。このパラメータは、--networkと--ip (オプション)のかわりに指定できます。

--filesystemsでは、デプロイメントの開始前にマウントする必要があるDBFSまたはACFS CRSファイル・システム・リソースを指定します。

--db_servicesでは、前のステップで作成したora.database.service_name.svcサービス名を指定します。Oracleマルチテナント・データベースを使用している場合、ReplicatにはPDBデータベース・サービス、ExtractにはCDBデータベース・サービスを指定します。ReplicatとExtractを両方使用する場合は、両方のサービス名をカンマで区切って指定します。

--use_local_servicesでは、db_servicesサービスが実行されているのと同じOracle RACノードにGoldenGateインスタンスも配置されている必要があることを指定します。

--nodesでは、このGoldenGateインスタンスを実行できるOracle RACノードを指定します。クラスタ内のいずれかのOracle RACノードで実行されるようにGoldenGateが構成されている場合は、引き続きこのパラメータを使用して、Oracle GoldenGateを実行するノードの優先順序を決める必要があります。

ノート:

- XAGへのGoldenGateインスタンス登録は、rootユーザーとして実行する必要があります。
- XAGへのGoldenGate登録はrootユーザーとして実行されるため、userパラメータとgroupパラメータは必須です。

次に、Oracle GoldenGateをXAGに登録する例を示します。

例1: DBFSを使用するOracle RACクラスタ(すでに作成されているアプリケーションVIPを使用する)

```
# agctl add goldengate GGNORTH ¥
--gg_home /u01/oracle/goldengate/gg21c_MS ¥
--service_manager ¥
--config_home /mnt/dbfs/goldengate/deployments/ggsm01/etc/conf ¥
--var_home /mnt/dbfs/goldengate/deployments/ggsm01/var ¥
--port 9100 ¥
--oracle_home /u01/oracle/goldengate/gg21c_MS/lib/instantclient
--adminuser oggadmin
--user oracle ¥
--group oinstall ¥
--vip_name gg_vip_prmy ¥
--filesystems dbfs_mount ¥
--db_services ora.ds19c.oggserv.svc ¥
--use_local_services ¥
--nodes dc1north01,dc1north02
```

説明:

- GoldenGateインスタンスはGGNORTHです
- GoldenGateホーム・ディレクトリは/u01/oracle/goldengate/gg21c_MSです
- これは、Oracle GoldenGate Microservices Architectureインスタンス(--service_manager)です。
- GoldenGateデプロイメント構成のホーム・ディレクトリは/mnt/dbfs/goldengate/deployments/ggsm01/etc/confです
- GoldenGateデプロイメント変数のホーム・ディレクトリは/mnt/dbfs/goldengate/deployments/ggsm01/varです

- デプロイメント・サービス・マネージャのポート番号は9100です
- Oracle GoldenGate Microservices管理者アカウント名はoggadminです
- GoldenGateユーザーは、oinstallグループ内のoracleです
- CRSで管理されるアプリケーションVIPの名前はgg_vip_prmyです
- このデプロイメントが依存するファイル・システムのCRSリソース名はdbfs_mountです
- GoldenGateインスタンスは、ora.ds19c.oraserv.svcというCRSサービスがこのGoldenGateインスタンスと同じノードに配置されるため、同じOracle RACノードで起動されます。

例2: ACFSを使用するOracle RACクラスタ(アプリケーションVIPはクラスタ内のノードのサブセットで実行される)

```
# agctl add goldengate GGNORTH ¥
--gg_home /u01/oracle/goldengate/gg21c_MS ¥
--service_manager ¥
--config_home /mnt/acfs/goldengate/deployments/ggsm01/etc/conf ¥
--var_home /mnt/acfs/goldengate/deployments/ggsm01/var ¥
--port 9100 ¥
--oracle_home /u01/oracle/goldengate/gg21c_MS/lib/instantclient
--adminuser admin ¥
--user oracle ¥
--group oinstall ¥
--network 1 --ip 10.13.11.203 ¥
--filesystems ora.datac1.acfs_gg.acfs ¥
--db_services ora.ds19c.oraserv.svc ¥
--use_local_services ¥
--nodes dc1north01,dc1north02
```

説明:

- GoldenGateインスタンスはGGNORTHです
- GoldenGateホーム・ディレクトリは/u01/oracle/goldengate/gg21c_MSです
- これは、Oracle GoldenGate Microservices Architectureインスタンス(--service_manager)です。
- GoldenGateデプロイメント構成のホーム・ディレクトリは/mnt/acfs/goldengate/deployments/ggsm02/etc/confです
- GoldenGateデプロイメント変数のホーム・ディレクトリは/mnt/acfs/goldengate/deployments/ggsm02/varです
- デプロイメント・サービス・マネージャのポート番号は9100です
- Oracle GoldenGate Microservices管理者アカウント名はadminです
- GoldenGateユーザーは、oinstallグループ内のoracleです
- ネットワークはデフォルトのora.net1.networkであり、VIPは10.13.11.203です。
- このデプロイメントが依存するファイル・システムのCRSリソース名はora.datac1.acfs_gg.acfsです
- このGoldenGateインスタンスは、ora.ds19c.oraserv.svcというCRSサービスがこのGoldenGateインスタンスと同じノードに配置されるため、同じOracle RACノードで起動されます
- Oracle GoldenGateは、Oracle RACノードdc1north01およびdc1north02でのみ実行されます(優先度順で示されている)。

AGCTLコマンドの例

次に、XAGでGoldenGateデプロイメントを管理するために使用する、いくつかのagctlコマンドの例を示します。

Oracle GoldenGateのステータスを確認するには、次のようにします。

```
% agctl status goldengate
Goldengate instance 'GGNORTH' is running on dc1north01
```

GoldenGateデプロイメント、および自動起動するように構成されているすべてのExtractプロセスとReplicatプロセスを起動するには(後のステップで示す手順)、次のようにします。

```
% agctl start goldengate GGNORTH --node dc1north02
```

GoldenGateデプロイメントを停止するには、次のようにします。

```
% agctl stop goldengate GGNORTH
```

GoldenGateデプロイメントを別のノードに手動で再配置するには、次のようにします。

```
% agctl relocate goldengate GGNORTH --node dc1north02
```

GoldenGateリソースの構成パラメータを表示するには、次のようにします。

```
% agctl config goldengate GGNORTH
Instance name: GGNORTH
Application GoldenGate location is: /u01/oracle/goldengate/gg21c_MS
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
/mnt/dbfs/goldengate/deployments/ggsm01/etc/conf
Goldengate Service Manager var directory: /mnt/dbfs/goldengate/deployments/ggsm01/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: no
Configured to run on Nodes: dc1north01 dc1north02
ORACLE_HOME location is: /u01/oracle/goldengate/gg21c_MS/lib/instantclient
Database Services needed: ora.cdb1.oggcdb.svc [use_local_services]
File System resources needed: ora.datac1.acfs_gg.acfs
Network: 1, IP: 10.13.11.203, User:oracle, Group:oinstall
```

GoldenGate XAGリソースを削除するには、次のようにします。

```
$ agctl stop goldengate GGNORTH
# agctl remove goldengate GGNORTH
```

Oracle Grid Infrastructure Bundled Agentの詳細は、Oracle Grid Infrastructure Standalone Agents for Oracle Clusterware 11gリリース2、12c、18cおよび19cを参照してください。

タスク8: NGINXリバース・プロキシの構成

My Oracle Supportのノート2826001.1で示されている手順に従って、SSL接続を使用するNGINXリバース・プロキシをインストールし構成し、すべての外部通信が十分にセキュリティ保護されていることを確認します。

> ノート:
>
> NGINX で CA 署名付き証明書を使用する場合は、CA 署名付き証明書、中間証明書、ルート証明書という正しい順序で証明書が含まれている証明書ファイルが NGINX の `ssl_certificate` パラメータで指し示されていることを確認します。

Oracle Clusterwareは、NGINXリバース・プロキシの起動を制御して、GoldenGateデプロイメントの起動前に自動的にリバース・プロキシを起動できるようにする必要があります。

NGINXリソースは、基礎となるネットワークCRSリソースに依存して作成され、その名前は、次のコマンドを使用して決定できます。

```
$ $GRID_HOME/bin/crsctl stat res -w "TYPE == ora.network.type"|grep NAME
NAME=ora.net1.network
```

rootユーザーとして、次のコマンド例を使用して、NGINXを管理するClusterwareリソースを作成します。

```
# $GRID_HOME/bin/crsctl add resource nginx -type generic_application -attr
 "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-x,user:oracle:rwx',
EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl
 start -f nginx',STOP_PROGRAM='/bin/systemctl
 stop -f nginx',CHECK_PROGRAMS='/bin/systemctl
 status nginx' ,START_DEPENDENCIES='hard(ora.net1.network)
 pullup(ora.net1.network)', STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='dc1north01,dc1north02', CARDINALITY=2"
```

この例で作成したNGINXリソースは、指定した複数のクラスタ・ノード(HOSTING_MEMBERSで指定)で同時に実行されます。これは、GoldenGateサービス・マネージャのデプロイメントが複数構成されており、それらをクラスタ・ノード間で単独で移動できる場合にお薦めします。

NGINX Clusterwareリソースの作成後には、GoldenGateデプロイメントの起動前にNGINXが起動されるように、GoldenGate XAGリソースを変更する必要があります。

oracleユーザーとして、次のコマンド例を使用してXAGリソースを変更します。

現在の--filesystemsパラメータを確認します。

```
$ agctl config goldengate SOURCE|grep "File System"
File System resources needed: ora.datac1.acfs_gg.acfs
```

その--filesystemsパラメータを変更します。

```
$ agctl modify goldengate SOURCE --filesystems ora.datac1.acfs_gg.acfs,nginx
```

NGINXに依存しているXAG GoldenGate登録ごとに、前述のコマンドを繰り返します。

タスク9: Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成

Oracle RACノード間の切替え時にGoldenGateプロセスにローカル・データベース接続を提供するために、Oracle GoldenGateが起動される可能性があるOracle RACノードすべてでTNS別名を作成します。TNS別名は、デプロイメントの作成時に指定したTNS_ADMINディレクトリ内のtnsnames.oraファイルで作成します。

ソース・データベースがマルチテナント・データベースの場合は、2つのTNS別名エントリが必要になります。その1つはコンテナ・データベース(CDB)用、もう1つは、レプリケートされるプラガブル・データベース(PDB)用です。ターゲットのマルチテナント・データベースでは、TNS別名によって、レプリケートしたデータが適用されているPDBを接続しますプラガブル・データベースSERVICE_NAMEは、前のステップ(「タスク3: データベース・サービスの作成」を参照)で作成したデータベース・サービスに設定する必要があります。

次に、IPCプロトコルを使用する、ソース・データベースのTNS別名の定義の例を示します。これは、すべてのRACノードでローカルに定義する必要があります。

```
OGGSOURCE_CDB =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
 (CONNECT_DATA =
 (SERVICE_NAME = oggserv_cdb)
 )
 )
```

```
OGGSOURCE_PDB =
 (DESCRIPTION =
 (ADDRESS = (PROTOCOL=IPC)(KEY=LISTENER))
 (CONNECT_DATA =
 (SERVICE_NAME = oggserv_pdb)
 )
 )
```

ノート:

GoldenGate デプロイメントの TNS_ADMIN ディレクトリにある tnsnames.ora または sqlnet.ora を変更した場合は、変更内容を採用するためにデプロイメントを再起動する必要があります。

GoldenGateデプロイメントを作成したら、管理サーバーのホーム・ページで、前述のTNS別名を使用してデータベース資格証明を作成します。「ユーザーID」フィールド内のデータベース・ユーザー名に追加されているTNS別名を使用したデータベース資格証明の作成の例は、次の図6を参照してください。

ソース・データベースがマルチテナント・データベースの場合は、CDBとPDBのデータベース資格証明を作成します。ターゲット・データベースがマルチテナント・データベースの場合は、PDB用の単一の資格証明を作成します。

タスク10: Oracle GoldenGateプロセスの構成

Oracle GoldenGate Microservices Architectureを使用してExtract、分散パスおよびReplicatプロセスを作成する場合、Oracle RACノード間で共有する必要があるファイルはすべて、共有ファイル・システム(DBFSまたはACFS)に格納されているデプロイメント・ファイルですでに共有されています。

次に示す重要な構成詳細は、Extract、分散パスおよびReplicatプロセスのために Oracle RACでOracle GoldenGate Microservicesを実行する場合にお薦めします。

Extract構成

1. Oracle GoldenGate Administration Server GUIインタフェースを使用してExtractを作成するときには、「トレイルのサブディレクトリ」パラメータを空白のままにして、共有ファイル・システムに格納されているデプロイメント・ディレクトリに自動的に証跡ファイルが作成されるようにします。

   証跡ファイルのデフォルトの場所は、/<deployment directory>/var/lib/dataです

2. 共有ストレージにDBFSを使用しているときに、デプロイメントのvar/tempディレクトリが「タスク6: Oracle GoldenGateデプロイメントの作成」の説明に従ってローカル・ストレージに移動された場合は、ExtractのCACHEMGRパラメータを使用して一時キャッシュ・ファイルを共有ストレージに配置することをお薦めします。

DBFSデプロイメントのマウント・ポイントの下に新しいディレクトリを作成します。たとえば:

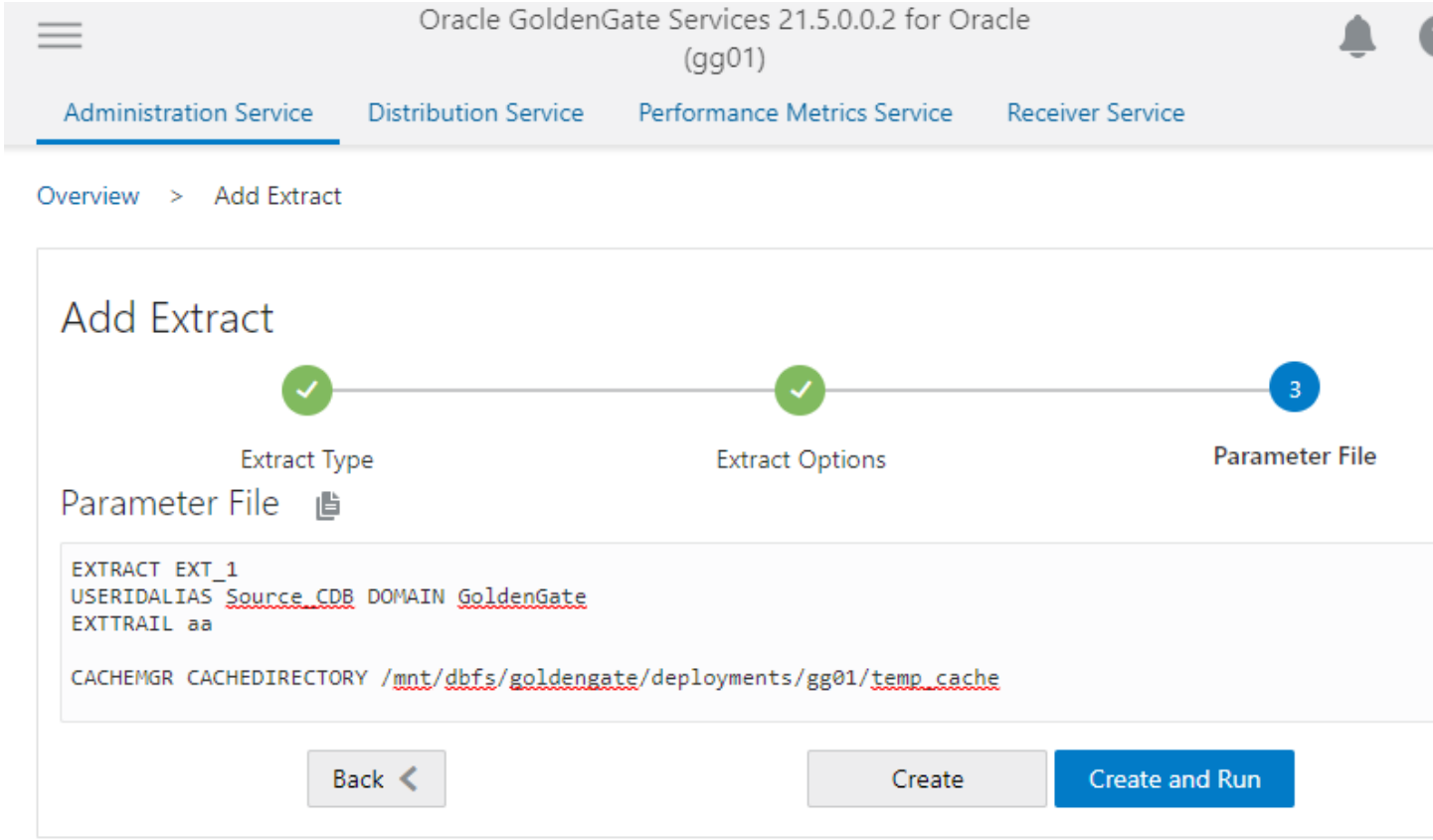
```
$ mkdir –p /mnt/dbfs/goldengate/deployments/ggnorth/temp_cache
```

Extractパラメータを新しいディレクトリに設定します。

```
CACHEMGR CACHEDIRECTORY /mnt/dbfs/goldengate/deployments/ggnorth/temp_cache
```

次に、Oracle GoldenGate Administration Server GUIで統合Extract用に指定されたパラメータがUIでどのように表示されるかの例を示します。

図22-1 一時キャッシュ・ファイルを定義するためのExtractパラメータ



分散パス構成

NGINXリバース・プロキシでOracle GoldenGate分散パスを使用する場合、パスのサーバー証明書が構成されるように、追加でステップを実行する必要があります。

次のビデオで示されている手順に従って、証明書を正しく構成します:
https://apexapps.oracle.com/pls/apex/f?p=44785:112:0::::P112_CONTENT_ID:31380

このビデオで示されている、構成の主要部分:

1. ソース・デプロイメントのクライアント証明書を作成し、そのクライアント証明書をソース・デプロイメントのサービス・マネージャに追加します。(これは、Oracle GoldenGate 21c以降のリリースを使用する場合は必要ありません。)
2. ターゲット・デプロイメント・サーバーのルート証明書をダウンロードし、ソース・デプロイメントのサービス・マネージャにCA証明書を追加します。
3. ターゲット・デプロイメントで、接続する分散パスのユーザーを作成します。
4. 前のステップで作成したユーザーを使用して、ターゲット・デプロイメントに接続するソース・デプロイメント内に資格証明を作成します。

   たとえば、ドメインが`GGNORTH_to_GGSOUTH`、別名が`PathReceiver`です。

クライアント証明書とサーバー証明書を構成した後は、次の構成オプションを設定する必要があります。これらのオプションをUIのどこで設定するかを確認するには、次の図を参照してください。

1. 「生成されたソースURI」を変更してサーバー名に`localhost`を指定します。

   これにより、任意のOracle RACノードで分散パスを起動できるようになります。

2. 「ターゲット認証方式」を`UserID Alias`に、「ターゲット」転送プロトコルを`wss` (セキュアWebソケット)に設定しま

す。

ターゲット・ホストを、ターゲット・システムへの接続に使用するターゲット・ホスト名/VIPに設定し、NGINXの構成に使用したポート番号を指定します(デフォルトは443)。

ターゲット・ホスト名/VIPは、NGINXで使用されるCA署名付き証明書での共通名と一致している必要があります。

3. 「ドメイン」を、前述のステップ4で作成した、ビデオで示されている資格証明ドメインに設定します(GGNORTH_to_GGSOUTHなど)。

   「別名」は、ビデオ内のステップ4でも作成した、資格証明別名に設定されます。

4. 分散サーバーの起動時に分散パスが自動的に再起動されるように設定します。

   これは、分散サーバーのOracle RACノード再配置の後に手動操作を不要にするために必要です。「再試行」の回数を10に設定することをお薦めします。「遅延」を1に設定します。これは、再起動を試行してから次に試行するまでの一時停止時間(分数)です。

図22-2 分散パス作成のステップ1から3

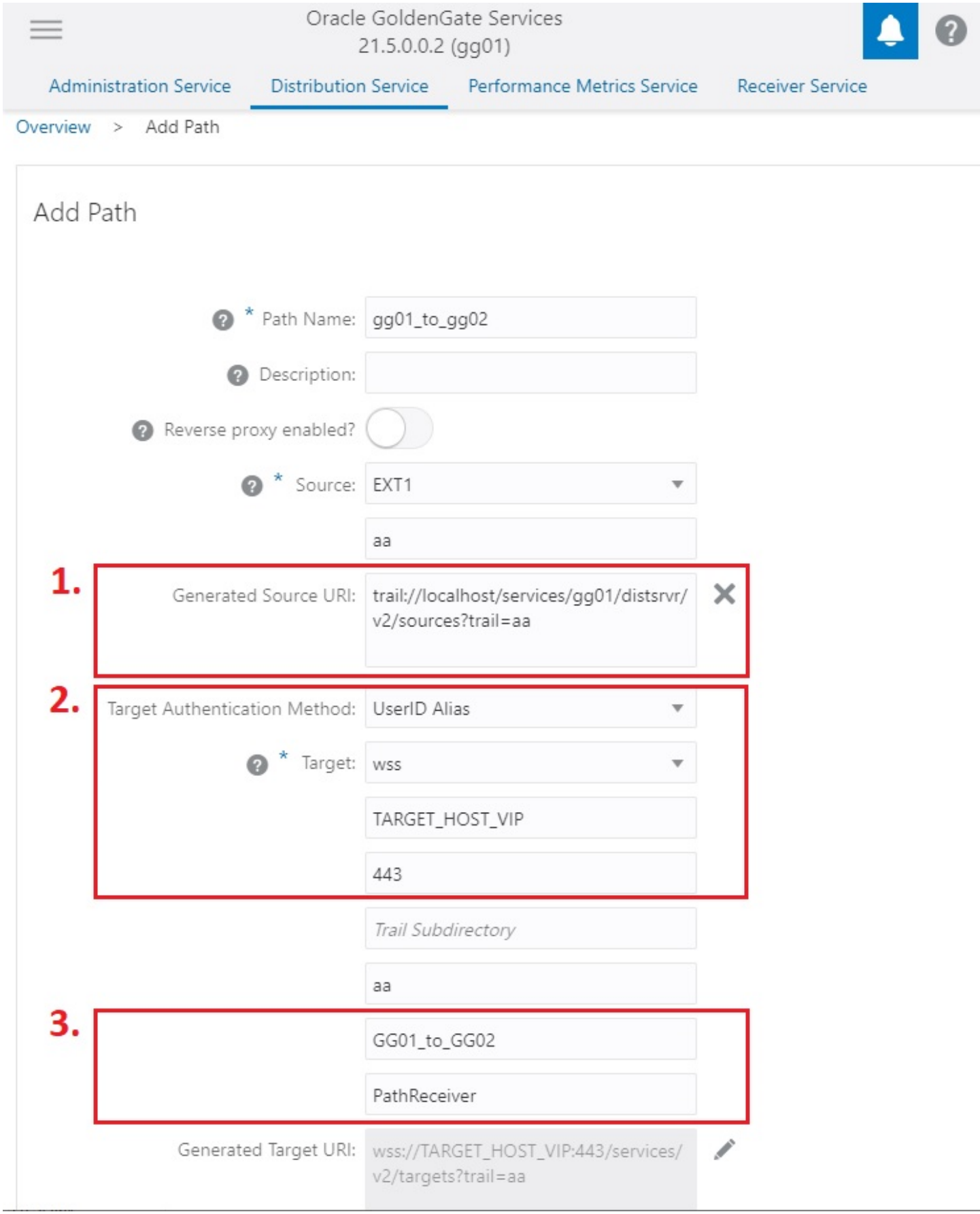
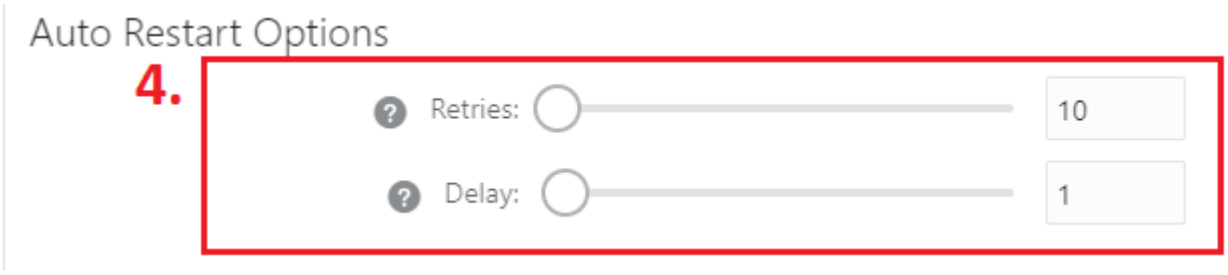


図22-3 分散パス作成のステップ4

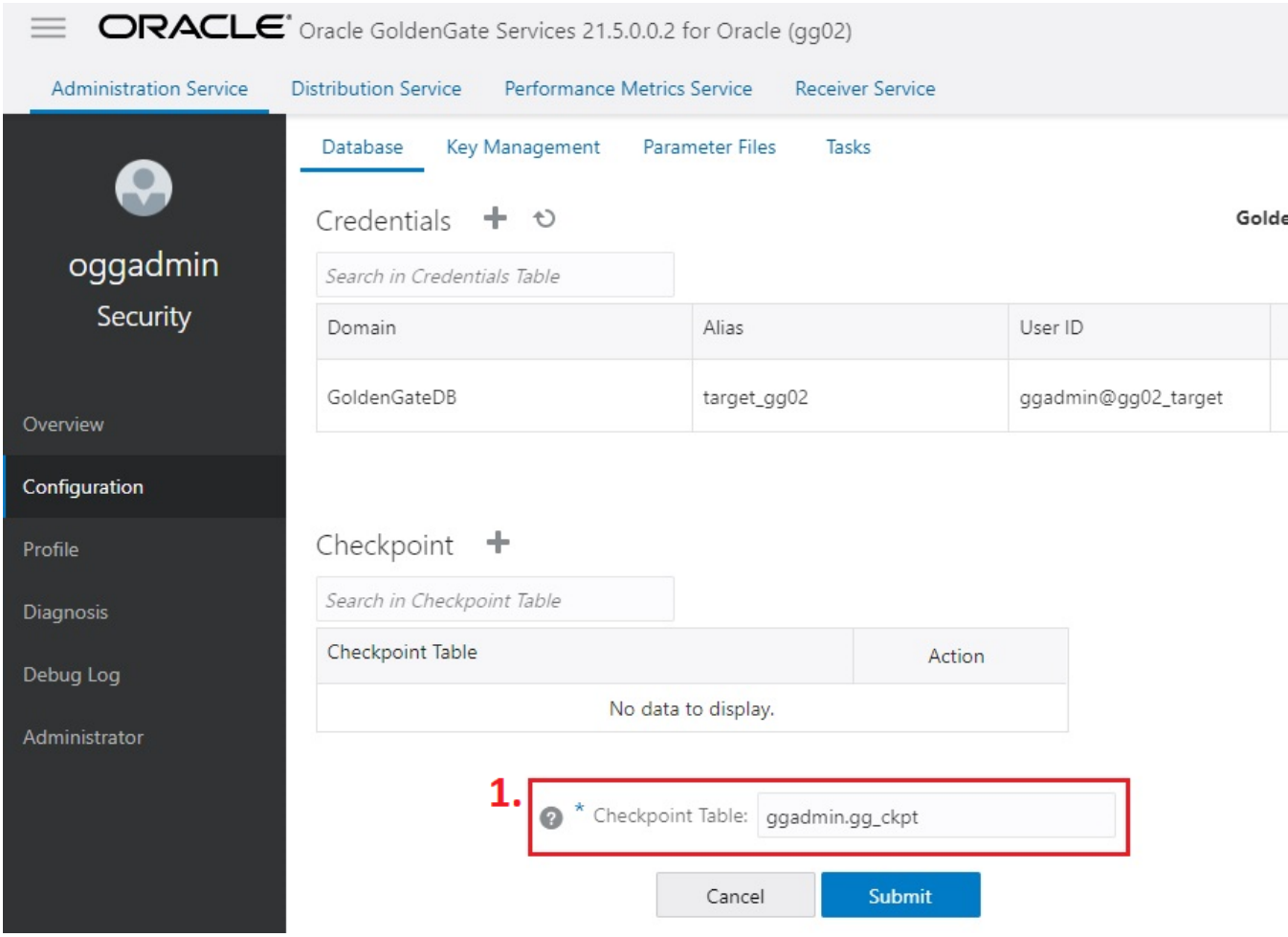


Replicat構成

1. チェックポイント表は、GoldenGate Replicatプロセスに必要なコンポーネントです。必ず、データベースGoldenGate管理者(GGADMIN)スキーマにチェックポイント表が作成されていることを確認してください。

   チェックポイント表を作成するには、Oracle GoldenGate Administration Server GUIを使用し、「+」ボタンをクリックし、schema.tablenameという形式でチェックポイント表名を入力します。これを次の図で示します。

   図22-4 Replicatプロセス用のチェックポイント表の作成

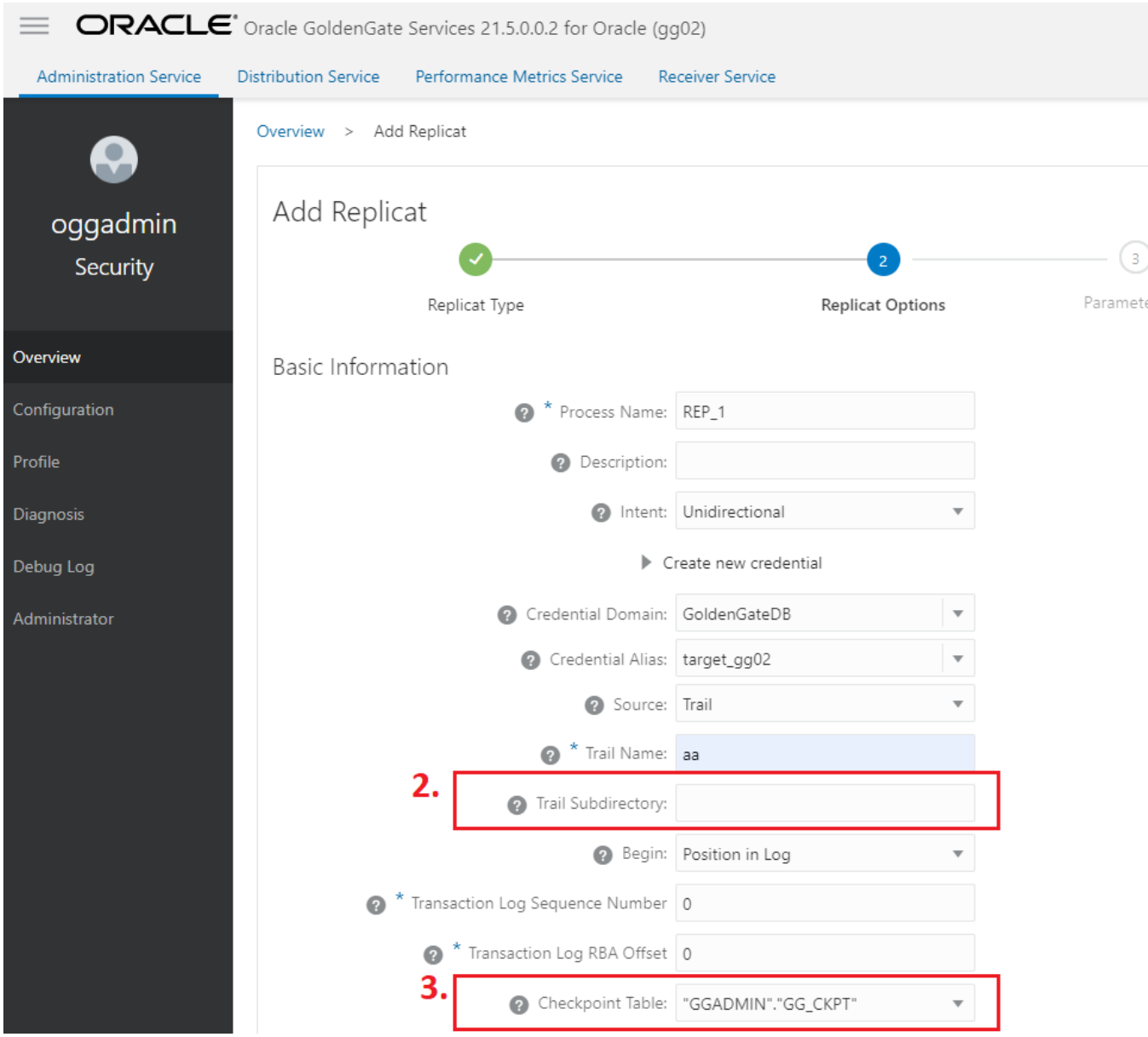


   チェックポイント表の作成の詳細は、チェックポイント表の概要を参照してください。

2. Oracle GoldenGate Administration Server GUIインタフェースを使用してReplicatを作成するときは、「トレイルのサブディレクトリ」パラメータを、分散パスまたはローカルExtractで証跡ファイルが作成されている場所に設定します。

3. チェックポイント表が以前に作成されている場合は、「チェックポイント表」プルダウン・リストからその表名を選択します。

図22-5「トレイルのサブディレクトリ」と「チェックポイント表」を使用したReplicat作成



タスク11: ExtractプロセスとReplicatプロセスの自動起動の構成

ExtractプロセスとReplicatプロセスを、Oracle GoldenGate Administration Serverの起動時に自動的に起動されて、ExtractプロセスかReplicatプロセスが異常終了した場合に再起動されるように構成します。GoldenGate Microservicesでは、自動起動と再起動はプロファイルによって管理されます。

Oracle GoldenGate Administration Server GUIを使用して、各Oracle GoldenGateプロセスに割り当てることができる新しいプロファイルを作成します。

プロファイルの構成オプション	推奨設定
デフォルト・プロファイル	有効
自動開始	有効
開始の遅延	1 分

プロファイルの構成オプション	推奨設定
自動再起動	有効
最大再試行回数	5
再試行の遅延	30 秒
再試行期間	30 分
失敗時にのみ再起動	有効
試行回数に達したらタスクを無効化	有効

プロファイルが作成され、デフォルト・プロファイルとして設定された後は、作成された新しいGoldenGateプロセスすべてにこのプロファイルが割り当てられます。既存のすべてのプロセスに対して、このプロファイルをプロセスごとに割り当てる必要があります。

「概要」ペインの「プロセス情報」タブで、「管理対象オプション」の下の「プロファイル名」を選択します。

ノート:

XAG とともに Oracle GoldenGate Microservices を使用する場合は、Extract プロセスや Replicat プロセスに対して「デプロイメントのヘルスに不可欠」フラグを有効にしないでおくことをお薦めします。これを行うと、単一の Extract 障害または Replicat 障害によって GoldenGate デプロイメント全体が停止する可能性があり、XAG で GoldenGate を再起動できなくなる可能性もあります。Replicat を不可欠と設定したことによる停止のトラブルシューティングの例は、「Oracle RAC での Oracle GoldenGate のトラブルシューティング」を参照してください。

23 オンプレミスMAA Platinum: Active Data Guardと統合されたOracle GoldenGate Microservices Architecture

Oracle GoldenGate MicroservicesとOracle Data Guardを組み合せて統合することで、すべての計画停止および計画外停止でゼロまたはほぼゼロの停止時間を達成するMAA Platinumサービス・レベル構成を実現できます。

これらの構成ベスト・プラクティスに従って、Data Guardスタンバイで保護されているデータベースを使用したOracle GoldenGate Microservicesレプリケーションが、Data Guard保護モードの構成(最大パフォーマンス、最大可用性または最大保護)に関係なく、Oracle Data Guardロール・トランジションに従って透過的かつシームレスに動作するようにします。

次の内容について説明します。

前提条件

オンプレミスMAA Platinumアーキテクチャ構成のタスクを実行する前に、必ず次の前提条件を満たしてください。

- オンプレミスのMAA Platinumの前提条件であるため、「オンプレミス: Oracle Real Application Clustersを使用したOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」の説明に従ってOracle GoldenGateを構成します。
- データベース・ファイル・システム(DBFS)は、Data Guardとの統合時に、重要なOracle GoldenGateファイルのために必要です。
- 続行する前に、Oracle Data Guardスタンバイ・データベースも構成済で動作している必要があります。

次に、MAA Platinum構成のベースとなるソフトウェア要件を示します。

- Oracle Grid Infrastructure 19c以降

  Oracle Grid Infrastructureは、ビジネスクリティカルなアプリケーション高可用性を管理するために必要なコンポーネントを提供します。Oracle Clusterware (Oracle Grid Infrastructureのコンポーネント)ネットワーク、データベースおよびOracle GoldenGateリソースを使用すると、障害発生時の可用性を管理できます。

- Oracle Grid Infrastructure Agentバージョン10.2以降

Oracle Grid Infrastructure Agentは、Oracle Grid Infrastructureコンポーネントを活用して、Oracle GoldenGateとその依存リソース(データベース、ネットワーク、ファイル・システムなど)との統合を提供します。また、エージェントはOracle GoldenGateとOracle Data Guardを統合します。これにより、Oracle GoldenGateはロール・トランジション後に、新しいプライマリ・データベースで再起動されます。

- Oracle Database 19c以上

  Oracle GoldenGateを使用する場合の推奨されるOracle Databaseパッチをすべて示すリストは、My Oracle Supportのドキュメント2193391.1を参照してください。

- Oracle GoldenGate Microservicesバージョン21c以降

  Oracle GoldenGate 21cでは統合ビルド・サポートが導入されているため、単一のソフトウェア・インストールで、レプリケートされたデータの取得と複数の主要なOracle Databaseバージョン(11gリリース2から21c)への適用がサポートされます。これが可能になるのは、Oracle GoldenGateインストールに必要なOracle Databaseクライアント・ライブラリが含まれており、別途データベースORACLE_HOMEをインストールする必要がないためです。

- 重要なOracle GoldenGateファイルを保護およびレプリケートするOracle DBFS

  Oracle Database File System (DBFS)は、MAAによって検証され、Oracle Data GuardおよびOracle GoldenGate構成に推奨される、唯一のファイル・システムです。これは、チェックポイント・ファイルや証跡ファイルなどの必要なOracle GoldenGateファイルの格納場所を、Oracle Data Guardで保護されている同じデータベース内に置くことができ、Oracle GoldenGateファイルとデータベース間の一貫性をシームレスに確保できるためです。

前提条件を満たしている場合は、以降のタスクで構成のベスト・プラクティスに従います。これらのタスクは、Oracle GoldenGate MicroservicesとOracle Data Guardをシームレスに統合するために実行する必要があります。これにより、GoldenGateが、Data Guardロール・トランジションの後に引き続き実行されるようになります。

タスク1: Oracle GoldenGate用のスタンバイ・データベースの構成

スタンバイ・データベースの初期化パラメータは、プライマリ・データベースの初期化パラメータと一致している必要があります。

詳細は、「タスク1: Oracle GoldenGate用のOracleデータベースの構成」を参照してください。これには次のパラメータが含まれます。

- ENABLE_GOLDENGATE_REPLICATION=TRUE
- Oracle GoldenGateソース・データベースの場合は、FORCE LOGGINGモードを有効にし、最小サプリメンタル・ロギングを有効にします。
- GoldenGateソース・データベースの場合や統合Replicat (パラレルまたは非パラレル)を実行している場合は、STREAMS_POOL_SIZEを構成します。

タスク2: プライマリ・データベース・サービスの変更

プライマリ・データベース・サーバーで、Oracle RAC構成での元のOracle GoldenGateの一部として作成された既存のデータベース・サービスを変更します。

サービス・ロールをPRIMARYに設定して、ロール・トランジション後にデータベースがData Guardプライマリ・データベース・ロールになったときのみこのサービスが起動されるようにします。

oracleユーザーとして、次のコマンドを使用してこのサービスを変更します。

```
$ srvctl modify service -db dbName -service service_name
```

```
 -role PRIMARY
```

データベースがマルチテナント環境の一部である場合は、必ずマルチテナント・コンテナ・データベース(CDB)とプラガブル・データベース(PDB)の両方のサービスを変更してください。

タスク3: スタンバイ・データベース・サービスの作成

スタンバイ・クラスタでは、スタンバイ・データベースにデータベース・サービスが必要です。これにより、そのデータベースがプライマリ・ロールでオープンされたときにOracle Grid Infrastructure AgentによってOracle GoldenGateデプロイメントが自動的に起動されるようになります。

ソース・データベースがマルチテナント環境にある場合は、ルート・コンテナ・データベース(CDB)とレプリケートされるスキーマを含むプラガブル・データベース(PDB)に、個別のサービスが必要です。マルチテナント環境のターゲット・データベースの場合は、PDBに単一のサービスが必要です。

プライマリ・クラスタでサービスを作成したのと同じ方法で、`oracle`ユーザーとして次のコマンドを使用してサービスを作成します。

```
$ srvctl add service -db dbName -service service_name
 -preferred instance_1 -available instance_2, instance_3 etc.
 -pdb pdbName -role PRIMARY
```

プライマリ・クラスタで指定したのと同じサービス名を使用することをお薦めします。このサービスは、`-preferred`オプションを使用してシングルトン・サービスとして作成する必要があります。これは、このサービスが実行されているクラスタ・ノードでアプリケーション仮想IPアドレス(VIP)、DBFSおよびOracle GoldenGateが実行されるためです。

データベースがマルチテナント環境にない場合、またはデータベースがOracle GoldenGateのターゲット・データベースである場合は、`-pdb`パラメータを省略します。

タスク4: スタンバイ・クラスタ・ノードでのDBFSの構成

Oracle Data GuardでOracle GoldenGateを構成する場合、お薦めするソリューションはDatabaseファイル・システム(DBFS)のみです。

データベース内のDBFSユーザー、表領域およびファイル・システムは、「タスク4: Oracle RACでのファイル・システムの設定」の説明に従って、以前にプライマリ・データベース内に作成してあります。

残りの構成ステップは、Oracle GoldenGateが実行される可能性があるスタンバイ・クラスタのすべてのノードで必要です。

1. My Oracle Supportのドキュメント869822.1の手順に従って、必要なFUSEライブラリをインストールします(まだインストールされていない場合)。
2. プライマリ・クラスタで作成したのと同じように、IPCプロトコルを使用して`tnsnames.ora`のOracle Net接続別名を作成します。

   ```
   dbfs =
    (DESCRIPTION =
    (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
    (CONNECT_DATA =
    (SERVICE_NAME = NAME)
    )
    )
   ```

3. プライマリ・クラスタで使用されているのと同一の、DBFSのマウントポイントを作成します。

   Oracle GoldenGateデプロイメントの物理的な場所はデプロイメント構成ファイルに含まれているため、このマウント・

ポイントが同じであることが重要です。

たとえば:

```
# mkdir /mnt/dbfs
```

4. mount-dbfs.confファイルとmount-dbfs.shファイルをプライマリ・クラスタからスタンバイ・クラスタ・ノードにコピーします。

   これらをプライマリ・クラスタと同じディレクトリに配置することをお薦めします。

5. 次のコマンド例を使用して、DBFSリソースをOracle Clusterwareに登録します。

   Oracle Multitenantを使用している場合は、必ず、プライマリ・データベースで作成したのと同一の、DBFSリポジトリを含むPDBに、このサービス名を使用してください。

```
DBNAME=dbName
DEPNAME=ora.$DBNAME.oggserv_pdb.svc
crsctl add resource $RESNAME ¥
 -type cluster_resource ¥
 -attr "ACTION_SCRIPT=$ACTION_SCRIPT, ¥
 CHECK_INTERVAL=30,RESTART_ATTEMPTS=10, ¥
 START_DEPENDENCIES='hard($DEPNAME)pullup($DEPNAME)',¥
 STOP_DEPENDENCIES='hard($DEPNAME)',¥
 SCRIPT_TIMEOUT=300"
```

タスク5: Oracle GoldenGateソフトウェアのインストール

Oracle GoldenGateソフトウェアを、Oracle GoldenGate構成の一部となるスタンバイ・クラスタ内のすべてのノードでローカルにインストールします。

インストール・ディレクトリがすべてのノードで同一であり、プライマリ・クラスタのインストール・ディレクトリと一致していることを確認してください。

次の場所で、Oracle GoldenGate 21cソフトウェアまたはそれ以降のバージョンをダウンロードします。

http://www.oracle.com/technetwork/middleware/goldengate/downloads/index.html

タスク6: Oracle GoldenGateデプロイメント・ディレクトリの作成

Oracle GoldenGateサービス・マネージャおよびデプロイメントは、前提条件で必要となるため、プライマリ・クラスタにすでに作成されていますが、特定のディレクトリおよびシンボリック・リンクはスタンバイ・クラスタ・ノードで構成する必要があります。

これらのディレクトリおよびシンボリック・リンクは、「オンプレミス: Oracle Real Application Clustersを使用したOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」の一部として実行したタスクで、プライマリ・クラスタ上に作成してあります。

今度は、次のように、スタンバイ・クラスタにあるすべてのOracle RACノードに次のディレクトリおよびシンボリック・リンクを作成します。

1. プライマリ・クラスタ上に複数のGoldenGateサービス・マネージャが構成されており、それぞれ固有のデプロイメントがあり、XAGに個別に登録されている場合、それらは別々のOGG_HOMEソフトウェア・インストール・ディレクトリに属している必要があります。

   プライマリ・クラスタ上で構成されているOGG_HOMEディレクトリのディレクトリおよびシンボリック・リンクは、スタンバイ・クラスタ上と同じである必要があります。

2. パフォーマンス・メトリック・サーバーを有効にしてGoldenGateデプロイメントを作成した場合は、スタンバイOracle RACノードにメトリック・データストアのホーム・ディレクトリを作成する必要があります。

   たとえば、次のように、プライマリ・クラスタ・ノードにあるデータストア・ディレクトリを特定します。

   ```
   $ grep RepoDatastorePath <deployment directory>/var/log/pmsrvr.log|uniq
   "RepoDatastorePath": "",
   "RepoDatastorePath": "/u01/oracle/goldengate/datastores/ggnorth",
   ```

   その後、次のように、すべてのスタンバイ・クラスタ・ノードにそのディレクトリを作成します。

   ```
   $ mkdir -p /u01/oracle/goldengate/datastores/ggnorth
   ```

3. データベース・リリースがOracle Database 21c (21.3)より前の場合は、プライマリ・クラスタ上に作成されたシンボリック・リンクと一致するように、Oracle GoldenGateデプロイメントの一時ディレクトリのローカル・ストレージを作成します。

   たとえば、プライマリ・クラスタ上に次がある場合は、

   ```
   $ ls –lrt DBFS_GoldenGate_deployment_home_directory/var/temp
   lrwxrwxrwx 1 oracle oinstall 32 Aug 31 12:27 temp
    -> /u01/oracle/goldengate/deployments/ggnorth/temp
   ```

   次のように、スタンバイ・クラスタ・ノードにそれと同じディレクトリを作成します。

   ```
   $ mkdir –p /u01/oracle/goldengate/deployments/ggnorth/temp
   ```

タスク7: スタンバイNGINXリバース・プロキシの構成

スタンバイNGINXリバース・プロキシを構成するには、次のステップを実行します。

1. NGINXリバース・プロキシをインストールします。

   NGINXリバース・プロキシがまだインストールされていない場合は、
   https://nginx.org/en/linux_packages.htmlのインストール手順に従います。

   `root`ユーザーとして、Oracle GoldenGateデプロイメントの`NGINX`構成ファイルをプライマリ・クラスタ・ノードから単一のスタンバイ・ノード・ディレクトリ`/etc/NGINX/conf.d`にコピーします。

   たとえば:

   ```
   [root@dc2north01]# scp dc1north01:/etc/nginx/conf.d/ogg_north.conf
    /etc/nginx/conf.d
   ```

   スタンバイ・クラスタでは、プライマリ・クラスタとは異なるVIP名/アドレスが使用されるため、別のCA署名付き証明書が必要になります。続行する前に、システム管理者に連絡して、会社の標準に従ってサーバー証明書を作成または取得してください。VIPとサービス・マネージャのペアごとに、個別の証明書が必要です。

2. NGINXのサーバー証明書をインストールします。

   ファイル権限400 (-r--------)がある`root`が所有する`/etc/nginx/ssl`ディレクトリにサーバーCA証明書およびキー・ファイルをインストールします。

   ```
   # mkdir /etc/nginx/ssl
   # chmod 400 /etc/nginx/ssl
   ```

   プライマリ・クラスタからコピーしたリバース・プロキシ構成ファイルごとに、次の例を使用して証明書およびキー・ファイルの正しいファイル名を設定します。

```
ssl_certificate /etc/nginx/ssl/gg-stby-vip1.pem;
ssl_certificate_key /etc/nginx/ssl/gg-stby-vip1.key;
```

CA署名付き証明書を使用する場合、ssl_certificate NGINXパラメータで指定した証明書は、1つのファイルにルート、中間およびCA署名付き証明書が含まれたものである必要があります。この順序は非常に重要です。順序が異なると、NGINXは起動に失敗し、エラー・メッセージが表示されます

```
(SSL: error:0B080074:x509 certificate routines: X509_check_private_key:key values mismatch)
```

ルート証明書と中間証明書は、CA署名付き証明書プロバイダからダウンロードできます。

単一ファイルは次のコマンド例を使用して生成できます。

```
# cat CA_signed_cert.crt intermediate.crt root.crt
 > gg-stby-vip1.pem
```

ssl_certificate_keyファイルは、CA署名付き証明書をリクエストする際に必要になる証明書署名リクエスト(CSR)の作成時に生成されるキー・ファイルです。

プライマリ・クラスタからコピーしたリバース・プロキシ構成ファイル内のserver_nameパラメータを変更して、正しいVIP名に設定します。たとえば:

変更前:

server_name dc1north-vip1.example.com;

変更後:

server_name dc2north-vip1.example.com;

3. NGINXを検証し再起動します。

VIPはプライマリ・データベース・サービスが実行されるまでスタンバイ・システムで実行されないため、/etc/sysctl.confファイル内で設定する必要があるパラメータがあります。

a. rootユーザーとして、/etc/sysctl.confに次の変更を加えます。

```
# vi /etc/sysctl.conf
```

b. 次のパラメータを追加します。

```
# allow processes to bind to the non-local address
net.ipv4.ip_nonlocal_bind = 1
```

c. 変更した構成を再ロードします。

```
# sysctl -p /etc/sysctl.conf
```

d. NGINX構成ファイルを検証して構成内のエラーを検出します。ファイル内にエラーがある場合は、次のコマンドでそれらが報告されます。

```
# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginxconf test is successful
```

e. 新しい構成でNGINXを再起動します。

```
# systemctl restart nginx
```

NGINX構成が完了したら、その構成ファイルおよび証明書を、他のスタンバイ・クラスタ・ノード上の一致するディレクト

リにコピーします。

4. NGINX Clusterwareリソースを作成します。

   Oracle Clusterwareは、NGINXリバース・プロキシの起動を制御して、GoldenGateデプロイメントの起動前に自動的にリバース・プロキシを起動できるようにする必要があります。

   NGINXリソースは、基礎となるネットワークCRSリソースに依存して作成され、その名前は、次のコマンドを使用して決定できます。

```
$ $GRID_HOME/bin/crsctl stat res -w "TYPE == ora.network.type"|grep NAME
NAME=ora.net1.network
```

   a. rootユーザーとして、次のコマンド例を使用して、NGINXを管理するClusterwareリソースを作成します。

```
# $GRID_HOME/bin/crsctl add resource nginx -type generic_application
 -attr "ACL='owner:root:rwx,pgrp:root:rwx,other::r--,group:oinstall:r-
x,user:oracle:rwx',EXECUTABLE_NAMES=nginx,START_PROGRAM='/bin/systemctl
 start -f nginx',STOP_PROGRAM='/bin/systemctl
 stop -f nginx',CHECK_PROGRAMS='/bin/systemctl
 status nginx' ,START_DEPENDENCIES='hard(ora.net1.network)
pullup(ora.net1.network)',
 STOP_DEPENDENCIES='hard(intermediate:ora.net1.network)',
 RESTART_ATTEMPTS=0, HOSTING_MEMBERS='dc1north01,dc1north02',
CARDINALITY=2"
```

      この例で作成したNGINXリソースは、指定した複数のクラスタ・ノード(HOSTING_MEMBERSで指定)で同時に実行されます。これは、GoldenGateサービス・マネージャのデプロイメントが複数構成されており、それらをクラスタ・ノード間で単独で移動できる場合にお薦めします。

   b. NGINX Clusterwareリソースの作成時には、GoldenGateデプロイメントの起動前にNGINXが起動されるように、GoldenGate XAGリソースを変更する必要があります。

      rootユーザーとして、次のコマンド例を使用してXAGリソースを変更します。

      現在の--filesystemsパラメータを確認します。

```
# agctl config goldengate GGNORTH | grep "File System"
File System resources needed: dbfsgg
```

タスク8: Oracle Clusterwareの構成

1. プライマリ・クラスタのXAG GoldenGateインスタンスを変更します。

   rootユーザーとして次のコマンド例を使用して、プライマリ・クラスタ上のOracle Grid Infrastructure Standalone Agent (XAG) GoldenGateインスタンスに変更を加えそれがOracle Data Guard構成の一部であることを特定する必要があります。

```
# agctl modify goldengate instance_name --dataguard_autostart yes
```

2. スタンバイ・クラスタで、「タスク7: Oracle Clusterwareの構成」の手順に従って、次のステップ3から5を実行します。
3. 各スタンバイ・クラスタ・ノードにXAGソフトウェアをインストールします。

   XAGソフトウェアをプライマリ・クラスタと同じディレクトリにインストールすることをお薦めします。

4. XAGのアプリケーションVIPの作成を準備します。

   想定ではこのVIPおよびVIP名はプライマリ・クラスタ上とは異なるため、VIPアドレスをスタンバイ・クラスタのシステム管

理者が割り当てる必要があります。

5. Oracle GoldenGate MicroservicesをXAGに登録します。

   Oracle GoldenGate MicroservicesをXAGに登録するために使用するパラメータは、プライマリ・クラスタに登録するときに使用するパラメータと似ています。

   a. 次のコマンドを使用して、プライマリ・クラスタでの現在のパラメータを確認します。

```
$ agctl config goldengate GoldenGate_instance_name
Instance name: GoldenGate_instance_name
Application GoldenGate location is: /u01/oracle/goldengate/gg21c_MS
Goldengate MicroServices Architecture environment: yes
Goldengate Service Manager configuration directory:
 /mnt/dbfs/goldengate/deployments/ggnorth_sm/etc/conf
Goldengate Service Manager var directory:
 /mnt/dbfs/goldengate//deployments/ggnorth_sm/var
Service Manager Port: 9100
Goldengate Administration User: oggadmin
Autostart on DataGuard role transition to PRIMARY: yes
Configured to run on Nodes: dc1north01,dc1north02
ORACLE_HOME location is:
/u01/oracle/goldengate/gg21c_MS/lib/instantclient
Database Services needed:
ora.ggdg.oggserv_cdb.svc,ora.ggdg.oggserv_pdb.svc
File System resources needed: dbfsgg,nginx
VIP name: gg_vip_prmy
```

   また、XAGパラメータ--filesystem_verify noを指定して、GoldenGateインスタンスの登録時にXAGでDBFSデプロイメント・ディレクトリの存在が確認されないようにする必要があります。このパラメータを設定しないと、スタンバイ・クラスタでDBFSがマウントされないため、XAGの登録に失敗します。

   ノート:

   GoldenGate を XAG に登録するときは、プライマリ・クラスタで使用したのと同じ GoldenGate インスタンス名を使用することをお薦めします。

   b. rootユーザーとして、スタンバイ・クラスタでGoldenGateをXAGに登録します。

```
# agctl add goldengate GoldenGate_instance_name ¥
--gg_home /u01/oracle/goldengate/gg21c_MS ¥
--service_manager ¥
--config_home /mnt/dbfs/goldengate/deployments/ggnorth_sm/etc/conf ¥
--var_home /mnt/dbfs/goldengate/deployments/ggnorth_sm/var ¥
--port 9100 ¥
--oracle_home /u01/goldengate/gg21c_MS/lib/instantclient ¥
--adminuser oggadmin ¥
--user oracle ¥
--group oinstall ¥
--vip_name gg_vip_stby ¥
--filesystems dbfsgg,nginx ¥
--db_services ora.ggdgs.oggserv_cdb.svc,ora.ggdgs.oggserv_pdb.svc ¥
--use_local_services ¥
--nodes dc2north01,dc2north02 ¥
--filesystem_verify no ¥
--dataguard_autostart yes
```

Oracle Grid Infrastructure Bundled Agentの詳細は、
http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/xag-agents-downloads-3636484.htmlを参照してください

タスク9: Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成

「タスク9: Oracle GoldenGateデータベース接続用のOracle Net TNS別名の作成」で示されているIPC通信プロトコルを使用して、プライマリ・クラスタ上でIPCプロトコルを使用してプライマリ・データベース用に作成したのと同じTNS別名を、スタンバイ・クラスタの各ノードで、同じ別名で作成する必要があります。

Oracle GoldenGateデプロイメントで使用されるtnsnames.oraの場所は、スタンバイ・システム・ノード上でもプライマリ・システム上のその場所と同じである必要があります。

次の問合せREST APIコールを使用して、プライマリ・クラスタ上のTNS_ADMINの場所を問い合せます。

```
$ curl -s -u OGG_admin_username
 https://vip_name/services/v2/deployments/deployment_name
 -XGET|python -m json.tool|grep TNS_ADMIN -A1
```

Oracle GoldenGateサービス・マネージャ管理者ユーザーのパスワードの入力を求められます。

たとえば:

```
$ curl -s -u oggadmin https://dc1north01-vip1/services/v2/deployments/ggnorth
 -XGET|python -m json.tool|grep TNS_ADMIN -A1
 "name": "TNS_ADMIN",
 "value": "/u01/goldengate/network/admin"
```

必ず、tnsnames.oraがスタンバイ・クラスタ・ノードすべてでこの同じディレクトリにあることを確認してください。

GoldengateデータベースのTNS別名の例:

```
ggnorth_pdb =
 (DESCRIPTION =
 (SDU = 2097152)
 (ADDRESS = (PROTOCOL = IPC)(KEY=LISTENER))
 (CONNECT_DATA =
 (SERVICE_NAME = oggserv_pdb.us.oracle.com)
 )
 )
```

タスク10: Oracle GoldenGateプロセスの構成

「タスク10: Oracle GoldenGateプロセスの構成」で示されているガイダンスに加え、Extract、分散パスおよびReplicatに関する次の推奨事項に従います。

プライマリ・クラスタでのExtract構成

REDO転送の最大パフォーマンス・モードまたは最大可用性モードを使用しているData Guard構成を使用するGoldenGate Extractプロセスでは、トランザクションが失われて論理データの不整合が発生しないように、プライマリ・クラスタ上のExtractプロセス・パラメータ・ファイルに次のパラメータを追加する必要があります。

```
TRANLOGOPTIONS HANDLEDLFAILOVER
```

このパラメータにより、まだData Guardスタンバイ・データベースに適用されていないREDOからExtractがトランザクション・データを抽出しないようにします。これは、Oracle GoldenGateがソース・スタンバイ・データベースに存在しないデータをターゲット・データベースにレプリケートしないようにするために重要です。

このパラメータが指定されていないと、ソース・データベースのデータ損失フェイルオーバー後に、ソース・データベースに存在してい

ないデータがターゲット・データベースに保持される可能性があります。それにより、論理データの不整合が発生します。

デフォルトでは、スタンバイ・データベースで適用されたSCN情報を問い合せることができないためにExtractが停止すると、その60秒後に、警告メッセージがExtractレポート・ファイルに書き込まれます。たとえば:

WARNING OGG-02721 Extractはスタンバイ・データベースを60秒待機しています。

Extractレポート・ファイルに警告メッセージが書き込まれるまでの時間は、ExtractパラメータのTRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_WARNINGを使用すると調整できます。

Extractが30分(デフォルト)後にスタンバイ・データベースによって適用されたSCN情報を問い合せることができない場合、Extractプロセスは異常終了して、次のメッセージがExtractレポート・ファイルに記録されます。

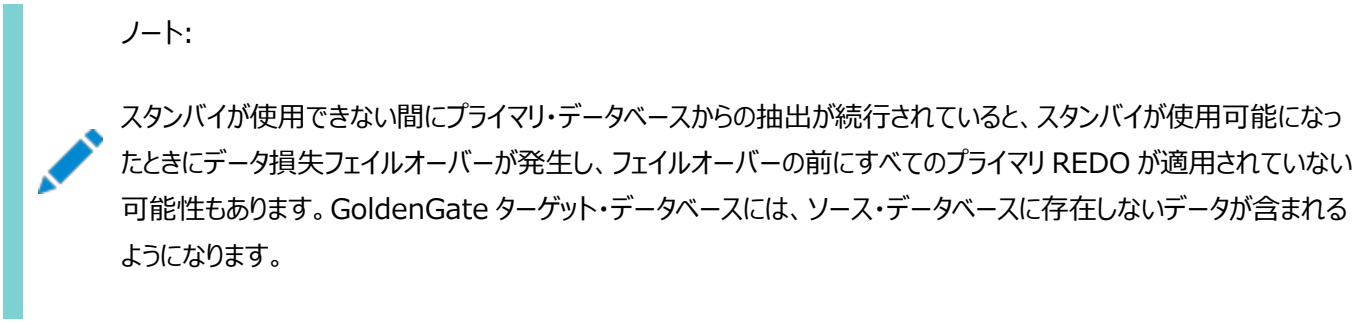
ERROR OGG-02722 Extractは、スタンバイ・データベースがアクセス可能になるか、プライマリ・データベースに同期することを1,800秒待機して異常終了しました。

デフォルトのタイムアウトである30分が経過するまでにスタンバイ・データベースが使用可能になると、Extractでソース・データベースからのデータのマイニングが続行され、次のメッセージがレポート・ファイルに報告されます。

INFO OGG-02723 Extractは停止状態から復帰してLCRの処理を開始しました。

タイムアウト値の30分は、ExtractパラメータTRANLOGOPTIONS HANDLEDLFAILOVER STANDBY_ABEND valueを使用して調整できます。ここでのvalueは、スタンバイが使用不可になってから異常終了するまでの秒数です。

計画メンテナンスの停止時など、スタンバイ・データベースが長時間使用できなくなるときに、Extractでプライマリ・データベースからのデータ抽出を続行する場合は、Extractパラメータ・ファイルからTRANLOGOPTIONS HANDLEDLFAILOVERパラメータを削除して、Extractを再起動してください。このパラメータは、スタンバイが使用可能になったら必ず設定してください。



ノート:

スタンバイが使用できない間にプライマリ・データベースからの抽出が続行されていると、スタンバイが使用可能になったときにデータ損失フェイルオーバーが発生し、フェイルオーバーの前にすべてのプライマリ REDO が適用されていない可能性もあります。GoldenGate ターゲット・データベースには、ソース・データベースに存在しないデータが含まれるようになります。

TRANLOGOPTIONS HANDLEDLFAILOVERパラメータの詳細は、Oracle GoldenGateリファレンス・ガイド (https://docs.oracle.com/en/middleware/goldengate/core/21.3/reference/reference-oracle-goldengate.pdf)を参照してください。

「タスク11: ExtractプロセスとReplicatプロセスの自動起動の構成」の説明に従ってExtractプロセスに自動再起動プロファイルが割り当てられている場合は、Data Guardロール・トランジションの後に、Extractプロセスが自動的に再起動されます。Extractでは、デフォルトのタイムアウト期間である5分が経過するまで、新しいスタンバイ・データベースの現在の状態が無視されて、新しいプライマリ・データベースからのREDOデータのマイニングが続行されます。この期間の経過後にスタンバイが使用できない場合、Extractは次のエラーで異常終了します。

INFO OGG-25053 スタンバイ・データベース復帰のための300秒の待機がタイムアウトです。ただちにHANDLEDLFAILOVERを実施します。

エラーOGG-06219 ロギング・サーバーOGG$CAP_EXT1からデータを抽出できません

ERROR OGG-02078 Extractは処理スレッドで致命的エラーを検出し、異常終了しています。

Extractは、Oracle GoldenGate Microservicesの自動再起動プロファイルに基づいて、再試行回数に達するか新しいスタンバイ・データベースが使用可能になるまで自動的に再起動し、HANDLEDLFAILOVERタイムアウトに達すると失敗します。

データベース・ロール・トランジション後のタイムアウト期間中、HANDLEDLFAILOVERパラメータは自動的に一時停止されるため、ソース・スタンバイ・データベースが最新の状態に維持されていないことの考慮なしに、データがOracle GoldenGateレプリカ・データベースにレプリケートされます。Extractの異常終了前に起動するスタンバイ・データベースのタイムアウト期間は、ExtractパラメータのTRANLOGOPTIONS DLFAILOVER_TIMEOUTを使用すると調整できます。

DLFAILOVER_TIMEOUTをデフォルトの5分のままにして、古いプライマリがスタンバイに変換できるようにすることをお薦めします。新しいスタンバイ・データベースが長期間使用できなくなるか完全に消失した場合に、Extractを起動して実行状態を維持するには、Extractパラメータ・ファイルからHANDLEDLFAILOVERパラメータを削除する必要があります。このパラメータを削除すると、ExtractはREDOがスタンバイ・データベースに適用されるまで待機することなくデータを抽出します。

スタンバイ・データベースがオンラインに戻りプライマリ・データベースからのすべてのREDOが適用されるまでの期間は、

それとOracle GoldenGateレプリカ・データベースとでデータの相違があります。これは、スタンバイ・データベースが最新状態になると解決されます。その時点で、統合Extractプロセス・パラメータ・ファイルにHANDLEDLFAILOVERパラメータを再追加し、Extractを停止してから再起動します。

プライマリ・データベースが失われた場合にブローカでスタンバイ・データベースに自動的にフェイルオーバーできるなど、ファスト・スタート・フェイルオーバーを有効にしてOracle Data Guardが構成されている場合は、次に示す統合Extractパラメータをさらに指定する必要があります。

```
TRANLOGOPTIONS FAILOVERTARGETDESTID n
```

このパラメータでは、スタンバイ・データベースにまだ適用されていないREDOデータを抽出しないことに関して、Oracle GoldenGate Extractプロセスを遅らせたままにしておく必要があるスタンバイ・データベースを指定します。

FAILOVERTARGETDESTIDの正しい値を判断するには、ソース・スタンバイ・データベースへのREDOの送信に使用されるGoldenGateソース・データベースのLOG_ARCHIVE_DEST_Nパラメータを使用します。たとえば、LOG_ARCHIVE_DEST_2がスタンバイ・データベースを指している場合は2の値を使用します。

たとえば:

```
SQL> show parameters log_archive_dest
NAME TYPEVALUE
-------------------- -------- ------------------------------------------------------
log_archive_dest_1 string location=USE_DB_RECOVERY_FILE_DEST,
 valid_for=(ALL_LOGFILES, ALL_ROLES)
log_archive_dest_2 string service="ggnorths", SYNC AFFIRM delay=0
 optional compression=disable max_failure=0 reopen=300
db_unique_name="GGNORTHS" net_timeout=30,
 valid_for=(online_logfile,all_roles)
```

この例では、次のようにExtractパラメータを設定します。

```
TRANLOGOPTIONS FAILOVERTARGETDESTID 2
```

Extractパラメータ・ファイルにこのパラメータを追加するには、Oracle GoldenGate Administration Serverを使用してExtract詳細の表示を選択します

1. 「管理サービス」タブで、Extractの「アクション」メニューを選択し、「詳細」を選択します。
2. Extract詳細のビューで、「パラメータ」タブを選択してから、鉛筆アイコンを選択して現在のパラメータ・ファイルを編集します
3. TRANLOGOPTIONSパラメータを追加し、変更を保存するために「適用」を選択します。

新しいパラメータを有効にするために、Extractプロセスを停止してから再起動する必要があります。これは管理サーバーを使用して実行できます。

前述のExtract TRANLOGOPTIONSパラメータの詳細は、Oracle GoldenGateのリファレンス(https://docs.oracle.com/en/middleware/goldengate/core/21.3/reference/tranlogoptions.html#GUID-B6ADFEC9-10E6-456D-9477-088513E113AF)を参照してください。

プライマリおよびスタンバイ・クラスタでの分散パス構成

受信サーバーを実行しているOracle GoldenGate環境のターゲット・データベースがOracle Data Guardで保護されている場合、受信サーバーに証跡ファイルを送信する分散パスには重要な考慮事項があります。Oracle Data Guardロール・トランジションの後に受信サーバーを別のクラスタに移動する場合は、新しいターゲット・クラスタ・アドレスを反映するように分散パスを変更する必要があります。

分散パスは、受信サーバー・クラスタにあるターゲット・データベースでデータベース・ロール・トランジション・トリガーを使用して自動的に変更できます。

プライマリ・クラスタとスタンバイ・クラスタのVIPで別々のルートCA証明書が使用されている場合は、「オンプレミス: Oracle Real Application Clustersを使用したOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」の説明に従って、スタンバイ証明書をソース・デプロイメントのサービス・マネージャに追加する必要があります。

次の手順に従って、データベース・ロール・トランジション・トリガーを作成します。このトリガーでは、ターゲット・データベースのData Guardロール・トランジションの間の、受信サーバーがプライマリ・システムとスタンバイ・システムの間で移動されるときに、分散パスのターゲット・アドレスが変更されます。

1. 分散パスを変更するシェル・スクリプトを作成します。

   「分散パス・ターゲット変更スクリプトの例」には、分散パス・ターゲット・アドレスの変更に使用できるシェル・スクリプトの例が記載されています。適切な変数値の設定については、スクリプト例のコメントを参照してください。

   このスクリプトは、プライマリ・データベース・クラスタおよびスタンバイ・データベース・クラスタのすべてのOracle RACノードで同じローカル・ディレクトリに配置する必要があります。スクリプト・ファイルの権限は6751に設定します。

   たとえば:

   ```
   $ chmod 6751 /u01/oracle/goldengate/scripts/change_path_target.sh
   ```

   このシェル・スクリプト例では、REST APIコールを使用してOracle GoldenGate分散パスにアクセスします。このREST APIコールをセキュアにするために、次に示すように、GoldenGateデプロイメント管理者のユーザー名とパスワードを構成ファイル(access.cfg)に含めることをお薦めします。

   ```
   $ cat /u01/oracle/goldengate/scripts/access.cfg
   user = "oggadmin:<password>"
   ```

   access.cfgファイルは、次のデータベース・ロール・トランジション・トリガーでも参照されます。

2. DBMS_SCHEDULERジョブを作成します。

   PL/SQL内からオペレーティング・システム・シェル・スクリプトを実行するために、DBMS_SCHEDULERジョブを作成する必要があります。ルート・コンテナ・データベース(CDB)内にSYSDBAユーザーとしてスケジューラ・ジョブを作成します。

   たとえば:

   ```
   SQL> exec dbms_scheduler.drop_job('gg_change_path_target');
   SQL> exec dbms_scheduler.create_job(job_name=>'gg_change_path_target',
    job_type=>'EXECUTABLE', number_of_arguments => 6,
    job_action=>'/u01/oracle/goldengate/scripts/change_path_target.sh',
    enabled=>FALSE);
   ```

   外部ジョブを実行するには、$ORACLE_HOME/rdbms/admin/externaljob.oraファイルのパラメータ

run_userとrun_groupをOracleデータベースのオペレーティング・システムのユーザーとグループに設定する必要があります。

たとえば:

```
run_user = oracle
run_group = oinstall
```

extrernaljob.oraは、プライマリ・データベース・クラスタおよびスタンバイ・データベース・クラスタのすべてのOracle RACノードで構成する必要があります。

3. データベース・ロール・トランジション・トリガーを作成します。

次の例を使用して、GoldenGateターゲット・データベースに、スタンバイ・データベースがプライマリ・データベースになったときに起動され分散パスのターゲット・アドレスを変更するロール・トランジション・トリガーを作成します。

```
CREATE OR REPLACE TRIGGER gg_change_path
AFTER db_role_change ON DATABASE
declare
 role varchar2(30);
 hostname varchar2(64);
begin
 select database_role into role from v$database;
 select host_name into hostname from v$instance;

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',1,'source_primary
_cluster_VIP');
DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',2,'source_standby
_cluster_VIP');

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',4,'dist_path_name
');

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',5,'deployment_nam
e');
 DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',6,
'<dir/access.cfg>');
 if role = 'PRIMARY' and hostname like 'primary_target_cluster_name%'
 then

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'primary_target
_cluster_VIP:443');
 elsif role = 'PRIMARY'
 then

DBMS_SCHEDULER.SET_JOB_ARGUMENT_VALUE('gg_change_path_target',3,'standby_target
_cluster_VIP:443');
 end if;
 DBMS_SCHEDULER.RUN_JOB(job_name=>'gg_change_path_target');
end;
/
```

データベース・トリガーを作成した後、次のコマンドを使用して、プライマリ・データベースにあるログ・ファイルを切り替えてコードがスタンバイ・データベースに伝播されるようにします。

```
SQL> alter system switch all logfile;
```

プライマリ・クラスタでのReplicat構成

「オンプレミス: Oracle Real Application Clustersを使用したOracle GoldenGate Microservices Architectureの構成のベスト・プラクティス」で説明されているように、ターゲット・データベース内のチェックポイント表は、すべてのOracle GoldenGate Replicatプロセスに必要です。Oracle Data Guardで構成したReplicatには、その他の構成要件はありま

せん。

分散パス・ターゲット変更スクリプトの例

次のスクリプト例を使用すると、ソースOracle GoldenGateデプロイメントの分散パスのターゲット・アドレスを変更して、Oracle Data Guardロール・トランジションの後の受信サーバーの新しい場所を反映できます。この例では、Data Guardを使用したMAAアーキテクチャでソースGoldenGateデプロイメントが構成されており、分散サーバーをプライマリ・システムとスタンバイ・システムの間で再配置できることを前提としています。

```
#!/bin/bash
# change_path_target.sh - changes the target host of a GG Distribution Path when the
target
# moves between primary/standby clusters.
# Example usage:
# ./change_path_target.sh <primary source VIP>:443 <standby source VIP>:443 <path
target VIP> <path name> <deployment name> <credentials file>
SOURCE1=$1 # PRIMARY Distribution Server VIP
SOURCE2=$2 # STANDBY Distribution Server VIP
TARGET=$3 # Distribution path target VIP
DPATH=$4 # Distribution path name
DEP=$5 # Deployment name
ACCESS=$6 # access.cfg file containing the deployment credentials. Example contents:
 # user = "oggadmin:<password>"
CONNECT=0
#echo "#${i} - `date`:"
LOGFILE=/tmp/ogg_dpatch_change.txt
result=$(curl -si -K $ACCESS
https://$SOURCE1/$DEP/distsrvr/services/v2/sources/$DPATH -X GET| grep HTTP | awk
'{print $2}')
# Will return NULL of nginx not running, 502 if cannot contact server, 200 if contact
to server good, and others (404) for other bad reasons:
if [[ -z $result || $result -ne 200 ]]; then # Managed to access the Distr Server
 echo "`date` - Couldn't contact Distribution Server at $SOURCE1 Deployment $DEP
****" >> $LOGFILE
else # Try the other source host:
 echo "`date` - Got status of Distribution Server at $SOURCE1 Deployment $DEP ***" >>
$LOGFILE
 SOURCE=$SOURCE1
 CONNECT=1
fi
if [ $CONNECT -eq 1 ]; then
# For secure NGINX patch destination (wss)

PAYLOAD='{"target":{"uri":"wss://'${TARGET}'/services/ggnorth/v2/targets?trail=bb"}}'
 curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X PATCH
--data '{"status": "stopped"}'
# Set new target for path:
 curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X PATCH
--data "$PAYLOAD"
 echo "`date` - Set path $DPATH on $SOURCE deployment $DEP:" >> $LOGFILE
 curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X GET |
python -m json.tool | grep uri >> $LOGFILE
 curl -s -K $ACCESS https://$SOURCE/$DEP/distsrvr/services/v2/sources/$DPATH -X PATCH
--data '{"status": "running"}'
exit 0
else
 echo "`date` - ERROR: COULDN'T CHANGE DISTRIBUTION PATH ($DPATH) in Deployement $DEP
at $SOURCE! ***" >> $LOGFILE
fi
# If here, means we couldn't connect to either Distribution Servers
exit 1
```

24 Oracle GoldenGateのトラブルシューティング

次の内容について説明します。

- MAA GoldenGate Hubのトラブルシューティング
- Oracle RACでのOracle GoldenGateのトラブルシューティング

MAA GoldenGate Hubのトラブルシューティング

ACFSレプリケーションのトラブルシューティング

ACFSレプリケーションのヘルスは、acfsutil repl util verifyprimary/verifystandbyコマンドによって判断されます。これらのコマンドは、サンプルのCRSアクション・スクリプトacfs_primary.scrおよびacfs_standby.scrによってコールされますが、ファイル・システムのロール・トランジション中にも暗黙的にコールされます。

問題が検出されていない場合は、どちらのコマンドも'0'の値を返します。ゼロ以外の値が返された場合は、verboseフラグを指定して同じコマンドを実行し、検証テストの包括的な出力を確認します。

スタンバイGGHubシステムのgridユーザーとして、プライマリGGHubを使用してACFSレプリケーションを確認します。

```
[grid@gghub_stby1]$ acfsutil repl util verifyprimary /mnt/acfs_gg -v
- Attempting to ping clust1-vip1
- ping successful
- Attempting to ssh
 '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@clust1-vip1 true 2>&1'
- ssh output: Host key verification failed.
- ssh output: Host key verification failed.
- ssh attempt failed, ret=255
verifyprimary return code: 255
```

検証コマンドHost key verification failedによって報告されたエラーは、失敗した理由を明確に示しています。この例では、スタンバイおよびプライマリ・ファイル・システムGGHubの間のssh構成に問題があります。問題が解決したら、検証コマンドを再実行して、それ以上の問題がないことを確認します。

フェイルオーバーが完了したら、acfsutilトレース・ファイルでフェイルオーバーの背後にある理由を確認することをお薦めします。acfsutilトレース・ファイルはCRSトレース・ファイル・ディレクトリ(デフォルトは/u01/app/grid/diag/crs/`hostname`/crs/trace/crsd_scriptagent_grid.trc)にあります。

次に、誤ったACFSレプリケーション構成で発生する可能性のある、一般的な障害の一部を示します。

SSHデーモンが停止しているか、VIPで実行するように構成されていない

ACFSプライマリとスタンバイGGHubsでアプリケーションVIPを使用する場合は、sshデーモンを、VIPアドレスで受信接続をリスニングするように構成する必要があります。この構成が行われない場合や、現在のプライマリ/スタンバイ・ホストでsshデーモンが実行されていない場合は、verifyprimaryまたはverifystandbyコマンドが次のエラーで失敗します。

プライマリGGHubシステムのgridユーザーとして、スタンバイGGHubを使用してACFSレプリケーションを確認します。

```
[grid@gghub_prim1]$ acfsutil repl util verifystandby /mnt/acfs_gg -v
- Attempting to ping gghubstby.goldengate.com
- ping successful
- Attempting to ssh
  '/usr/bin/ssh -o BatchMode=true -o Ciphers=aes128-ctr -o ConnectTimeout=3 -x
oracle@gghub_stby-avip true 2>&1'
- ssh output: ssh: connect to host gghub_stby1 port 22: Connection refused
- ssh output: ssh: connect to host gghub_stby2 port 22: Connection refused
- ssh attempt failed, ret=255
verifystandby return code: 255
```

スタンバイGGHubシステムのgridユーザーとして、リソース・アプリケーションVIPとsshd_restartが実行中であることを確認し、実行していない場合は再起動します。

```
[grid@gghub_stby1 ~]$ crsctl stat res -w "TYPE co app.appviptypex2"
NAME=gghubstby
TYPE=app.appviptypex2.type
TARGET=OFFLINE
STATE=OFFLINE
[grid@gghub_stby1 ~]$ crsctl start res gghubstby
CRS-2672: Attempting to start 'gghubstby' on 'gghub_stby1'
CRS-2676: Start of 'gghubstby' on 'gghub_stby1' succeeded
CRS-2672: Attempting to start 'sshd_restart' on 'gghub_stby1'
CRS-2676: Start of 'sshd_restart' on 'gghub_stby1' succeeded
```

`acfsutil repl verifystandby/verifyprimary`がプライマリおよびスタンバイ・ホストの両方から、'0'の結果を返すことを確認します。

プライマリACFSバックグラウンド・リソースが実行されていない

1. プライマリまたはスタンバイのACFSサーバーにアクセスできません
2. ACFSレプリケーションsshユーザーの問題
3. SSHホスト・キーの検証に失敗しました

Oracle GoldenGateのトラブルシューティング

Oracle RACノードでGoldenGateプロセスが正常に開始されない場合があります。問題の原因を特定するために確認する必要がある、GoldenGate、XAGおよびCRSによって生成されたファイルが多数あります。

重要なログ・ファイルとトレース・ファイルのリスト、場所の例および出力の例を次に示します。

XAGログ・ファイル

場所: <XAG installation directory>/log/<hostname>

場所の例: /u01/app/grid/xag/log/'hostname'

ファイル名: agctl_goldengate_grid.trc

agctlで実行されるすべてのコマンドと、CRSが実行するコマンドを含むコマンドの出力が含まれます。

```
2022-04-18 11:52:21: stop resource success
2022-04-18 11:52:38: agctl start goldengate <instance_name>
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start resource
xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:45: Command output:
> CRS-2672: Attempting to start 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-node1'
> CRS-2676: Start of 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-node1' succeeded
>End Command output
2022-04-18 11:52:45: start resource success
```

XAG GoldenGateインスタンス・トレース・ファイル

場所: <XAG installation directory>/log/<hostname>

場所の例: /u01/app/grid/xag/log/'hostname'

ファイル名: <GoldenGate_instance_name>_agent_goldengate.trc

これには、agctlによって実行されたコマンドの出力、使用された環境変数、基礎となるコマンドに対して有効になっているデバッ

グ出力が含まれます。

```
2022-04-18 12:14:46: Exported ORACLE_SID ggdg1
2022-04-18 12:14:46: Exported GGS_HOME /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: Exported OGG_CONF_HOME
/mnt/dbfs/goldengate/deployments/ggsm01/etc/conf
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH
/u01/oracle/goldengate/gg21c_MS:/u01/app/19.0.0.0/grid/lib:/etc/ORCLcluster/lib
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH_64 /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: Exported LIBPATH /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: ogg input =
{"oggHome":"/u01/oracle/goldengate/gg21c_MS","serviceManager":{"oggConfHome":"/mnt/db
fs/goldengate/deployments/ggsm01/etc/conf","portNumber":9100},"username":"<username>"
,"credential":"*****"}
2022-04-18 12:14:46: About to exec /u01/oracle/goldengate/gg21c_MS/bin/XAGTask
HealthCheck
2022-04-18 12:14:47: XAGTask retcode = 0
```

CRSトレース・ファイル

場所: /u01/app/grid/diag/crs/<hostname>/crs/trace

例の場所: /u01/app/grid/diag/crs/'hostname'/crs/trace

ファイル名: crsd_scriptagent_oracle.trc

XAGまたはdbfs_mountなどのCRSリソース・アクション・スクリプトによって作成された出力が含まれます。このトレース・ファイルは、DBFSまたはGoldenGateがRACノードで起動しなかった理由を判断するために重要です。

```
2022-04-18 11:52:38.634 : AGFW:549631744: {1:30281:59063} Agent received the message:
RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID 4098:4125749
2022-04-18 11:52:38.634 : AGFW:549631744: {1:30281:59063} Preparing START command
for: xag.<INSTANCE_NAME>.goldengate 1 1
2022-04-18 11:52:38.634 : AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: OFFLINE to: STARTING
2022-04-18 11:52:38.634 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] Executing action script:
/u01/oracle/XAG_MA/bin/aggoldengatescaas[start]
2022-04-18 11:52:38.786 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] GG agent running command
'start' on xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] ServiceManager fork pid =
265747
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] Waiting for
/mnt/dbfs/goldengate/deployments/ggsm01/var/run/ServiceManager.pid
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] Waiting for SM to start
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] ServiceManager PID = 265749
2022-04-18 11:52:43.643 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] XAGTask retcode = 0
2022-04-18 11:52:43.643 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] XAG HealthCheck after start
returned 0
2022-04-18 11:52:43.643 : AGFW:558036736: {1:30281:59063} Command: start for
resource: xag.<INSTANCE_NAME>.goldengate 1 1 completed with status: SUCCESS
2022-04-18 11:52:43.643 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [check] Executing action script:
/u01/oracle/XAG_MA/bin/aggoldengatescaas[check]
2022-04-18 11:52:43.644 : AGFW:549631744: {1:30281:59063} Agent sending reply for:
RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID 4098:4125749
2022-04-18 11:52:43.795 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [check] GG agent running command
'check' on xag.<INSTANCE_NAME>.goldengate
```

```
2022-04-18 11:52:45.548 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [check] XAGTask retcode = 0
2022-04-18 11:52:45.548 : AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: STARTING to: ONLINE
```

GoldenGateデプロイメント・ログ・ファイル

場所: <Goldengate_deployment_directory>/<instance_name>/var/log

例の場所: /mnt/dbfs/goldengate/deployments/<instance_name>/var/log

ファイル名: adminsrvr.log、recvsrvr.log、pmsrvr.log、distsrvr.log

Oracle GoldenGateデプロイメント・プロセス(管理サーバー、分散サーバー、受信サーバー、パフォーマンス・メトリック・サーバー)の開始、停止およびステータス・チェックの出力が含まれます。

```
2022-04-18T11:52:42.645-0400 INFO | Setting deploymentName to '<instance_name>'.
(main)
2022-04-18T11:52:42.665-0400 INFO | Read SharedContext from store for length 19 of
file '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/conf/adminsrvr-
resources.dat'. (main)
2022-04-18T11:52:42.723-0400 INFO | XAG Integration enabled (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring security. (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring user authorization secure store path
as '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/credential/secureStore/'.
(main)
2022-04-18T11:52:42.731-0400 INFO | Configuring user authorization as ENABLED. (main)
2022-04-18T11:52:42.749-0400 INFO | Set network configuration. (main)
2022-04-18T11:52:42.749-0400 INFO | Asynchronous operations are enabled with default
synchronous wait time of 30 seconds (main)
2022-04-18T11:52:42.749-0400 INFO | HttpServer configuration complete. (main)
2022-04-18T11:52:42.805-0400 INFO | SIGHUP handler installed. (main)
2022-04-18T11:52:42.813-0400 INFO | SIGINT handler installed. (main)
2022-04-18T11:52:42.815-0400 INFO | SIGTERM handler installed. (main)
2022-04-18T11:52:42.817-0400 WARN | Security is configured as 'disabled'. (main)
2022-04-18T11:52:42.818-0400 INFO | Starting service listener... (main)
2022-04-18T11:52:42.819-0400 INFO | Mapped 'ALL' interface to address 'ANY:9101' with
default IPV4/IPV6 options identified by 'exadb-node1.domain'. (main)
2022-04-18T11:52:42.821-0400 INFO | Captured 1 interface host names: 'exadb-
node1.domain' (main)
2022-04-18T11:52:42.824-0400 INFO | The Network ipACL specification is empty.
Accepting ANY address on ALL interfaces. (main)
2022-04-18T11:52:42.826-0400 INFO | Server started at 2022-04-18T11:52:42.827-05:00
(2022-04-18T15:52:42.827Z GMT) (main)
```

GoldenGateレポート・ファイル

場所: <Goldengate_deployment_directory>/<instance_name>/var/lib/report

例の場所: /mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/report

GoldenGateレポート・ファイルには、Managerプロセスを含む、すべてのGoldenGateプロセスの重要な情報、警告メッセージおよびエラーが含まれています。実行中にGoldenGateプロセスのいずれかが開始に失敗したか異常終了した場合、プロセス・レポート・ファイルには、失敗の原因を特定するために使用できる重要な情報が含まれます。

```
2022-04-23 13:01:50 ERROR OGG-00446 Unable to lock file "
/mnt/acfs_gg/deployments/<instance_name>/var/lib/checkpt/EXT_1A.cpe" (error 95,
Operation not supported).
2022-04-23 13:01:50 ERROR OGG-01668 PROCESS ABENDING.
```

Oracle RACでのOracle GoldenGateのトラブルシューティング

Oracle RACノードでOracle GoldenGateプロセスが正常に開始されない場合があります。問題の原因を特定するには、Oracle GoldenGate、XAGおよびCRSによって生成された複数のファイルを確認する必要があります。

重要なログ・ファイルとトレース・ファイルのリスト、その場所の例および出力例を次に示します。

XAGログ・ファイル

場所: <XAG installation directory>/log/<hostname>

場所の例: /u01/app/grid/xag/log/'hostname'

ファイル名: agctl_goldengate_grid.trc

agctlで実行されるすべてのコマンドと、CRSが実行するコマンドを含むコマンドの出力が含まれます。

例:

```
2022-04-18 11:52:21: stop resource success
2022-04-18 11:52:38: agctl start goldengate <instance_name>
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl status res
xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:38: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start resource
xag.<INSTANCE_NAME>.goldengate -f
2022-04-18 11:52:45: Command output:
> CRS-2672: Attempting to start 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-node1'
> CRS-2676: Start of 'xag.<INSTANCE_NAME>.goldengate' on 'exadb-node1' succeeded
>End Command output
2022-04-18 11:52:45: start resource success
```

XAG GoldenGateインスタンス・トレース・ファイル

場所: <XAG installation directory>/log/<hostname>

場所の例: /u01/app/grid/xag/log/'hostname'

ファイル名: <GoldenGate_instance_name>_agent_goldengate.trc

これには、agctlによって実行されたコマンドの出力、使用された環境変数、基礎となるコマンドに対して有効になっているデバッグ出力が含まれます。

例:

```
2022-04-18 12:14:46: Exported ORACLE_SID ggdg1
2022-04-18 12:14:46: Exported GGS_HOME /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: Exported OGG_CONF_HOME
/mnt/dbfs/goldengate/deployments/ggsm01/etc/conf
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH
/u01/oracle/goldengate/gg21c_MS:/u01/app/19.0.0.0/grid/lib:/etc/ORCLcluster/lib
2022-04-18 12:14:46: Exported LD_LIBRARY_PATH_64 /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: Exported LIBPATH /u01/oracle/goldengate/gg21c_MS
2022-04-18 12:14:46: ogg input =
{"oggHome":"/u01/oracle/goldengate/gg21c_MS","serviceManager":{"oggConfHome":"/mnt/db
fs/goldengate/deployments/ggsm01/etc/conf","portNumber":9100},"username":"admin","cre
dential":"xyz"}
```

```
2022-04-18 12:14:46: About to exec /u01/oracle/goldengate/gg21c_MS/bin/XAGTask
HealthCheck
2022-04-18 12:14:47: XAGTask retcode = 0
```

CRSトレース・ファイル

場所: /u01/app/grid/diag/crs/<hostname>/crs/trace

例の場所: /u01/app/grid/diag/crs/'hostname'/crs/trace

ファイル名: crsd_scriptagent_oracle.trc

XAGまたはdbfs_mountなどのCRSリソース・アクション・スクリプトによって作成された出力が含まれます。このトレース・ファイルは、DBFSまたはGoldenGateがRACノードで起動しなかった理由を判断するために重要です。

例:

```
2022-04-18 11:52:38.634 : AGFW:549631744: {1:30281:59063} Agent received the message:
RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID 4098:4125749
2022-04-18 11:52:38.634 : AGFW:549631744: {1:30281:59063} Preparing START command
for: xag.<INSTANCE_NAME>.goldengate 1 1
2022-04-18 11:52:38.634 : AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: OFFLINE to: STARTING
2022-04-18 11:52:38.634 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] Executing action script:
/u01/oracle/XAG_MA/bin/aggoldengatescaas[start]
2022-04-18 11:52:38.786 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] GG agent running command
'start' on xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] ServiceManager fork pid =
265747
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] Waiting for
/mnt/dbfs/goldengate/deployments/ggsm01/var/run/ServiceManager.pid
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] Waiting for SM to start
2022-04-18 11:52:42.140 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] ServiceManager PID = 265749
2022-04-18 11:52:43.643 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] XAGTask retcode = 0
2022-04-18 11:52:43.643 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [start] XAG HealthCheck after start
returned 0
2022-04-18 11:52:43.643 : AGFW:558036736: {1:30281:59063} Command: start for
resource: xag.<INSTANCE_NAME>.goldengate 1 1 completed with status: SUCCESS
2022-04-18 11:52:43.643 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [check] Executing action script:
/u01/oracle/XAG_MA/bin/aggoldengatescaas[check]
2022-04-18 11:52:43.644 : AGFW:549631744: {1:30281:59063} Agent sending reply for:
RESOURCE_START[xag.<INSTANCE_NAME>.goldengate 1 1] ID 4098:4125749
2022-04-18 11:52:43.795 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [check] GG agent running command
'check' on xag.<INSTANCE_NAME>.goldengate
2022-04-18 11:52:45.548 :CLSDYNAM:558036736:
[xag.<INSTANCE_NAME>.goldengate]{1:30281:59063} [check] XAGTask retcode = 0
2022-04-18 11:52:45.548 : AGFW:549631744: {1:30281:59063}
xag.<INSTANCE_NAME>.goldengate 1 1 state changed from: STARTING to: ONLINE
```

GoldenGateデプロイメント・ログ・ファイル

場所: <Goldengate_deployment_directory>/<instance_name>/var/log

例の場所: /mnt/dbfs/goldengate/deployments/<instance_name>/var/log

ファイル名: adminsrvr.log、recvsrvr.log、pmsrvr.log、distsrvr.log

Oracle GoldenGateデプロイメント・プロセス(管理サーバー、分散サーバー、受信サーバー、パフォーマンス・メトリック・サーバー)の開始、停止およびステータス・チェックの出力が含まれます。

例:

```
2022-04-18T11:52:42.645-0400 INFO | Setting deploymentName to '<instance_name>'.
(main)
2022-04-18T11:52:42.665-0400 INFO | Read SharedContext from store for length 19 of
file '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/conf/adminsrvr-
resources.dat'. (main)
2022-04-18T11:52:42.723-0400 INFO | XAG Integration enabled (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring security. (main)
2022-04-18T11:52:42.723-0400 INFO | Configuring user authorization secure store path
as '/mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/credential/secureStore/'.
(main)
2022-04-18T11:52:42.731-0400 INFO | Configuring user authorization as ENABLED. (main)
2022-04-18T11:52:42.749-0400 INFO | Set network configuration. (main)
2022-04-18T11:52:42.749-0400 INFO | Asynchronous operations are enabled with default
synchronous wait time of 30 seconds (main)
2022-04-18T11:52:42.749-0400 INFO | HttpServer configuration complete. (main)
2022-04-18T11:52:42.805-0400 INFO | SIGHUP handler installed. (main)
2022-04-18T11:52:42.813-0400 INFO | SIGINT handler installed. (main)
2022-04-18T11:52:42.815-0400 INFO | SIGTERM handler installed. (main)
2022-04-18T11:52:42.817-0400 WARN | Security is configured as 'disabled'. (main)
2022-04-18T11:52:42.818-0400 INFO | Starting service listener... (main)
2022-04-18T11:52:42.819-0400 INFO | Mapped 'ALL' interface to address 'ANY:9101' with
default IPV4/IPV6 options identified by 'exadb-node1.domain'. (main)
2022-04-18T11:52:42.821-0400 INFO | Captured 1 interface host names: 'exadb-
node1.domain' (main)
2022-04-18T11:52:42.824-0400 INFO | The Network ipACL specification is empty.
Accepting ANY address on ALL interfaces. (main)
2022-04-18T11:52:42.826-0400 INFO | Server started at 2022-04-18T11:52:42.827-05:00
(2022-04-18T15:52:42.827Z GMT) (main)
```

GoldenGateレポート・ファイル

場所: <Goldengate_deployment_directory>/<instance_name>/var/lib/report

例の場所: /mnt/dbfs/goldengate/deployments/<instance_name>/var/lib/report

GoldenGateレポート・ファイルには、Managerプロセスを含む、すべてのGoldenGateプロセスの重要な情報、警告メッセージおよびエラーが含まれています。実行中にGoldenGateプロセスのいずれかが開始に失敗したか異常終了した場合、プロセス・レポート・ファイルには、失敗の原因を特定するために使用できる重要な情報が含まれます。

Extractレポート・ファイルのエラー例:

```
2022-04-23 13:01:50 ERROR OGG-00446 Unable to lock file "
/mnt/acfs_gg/deployments/<instance_name>/var/lib/checkpt/EXT_1A.cpe" (error 95,
Operation not supported).
2022-04-23 13:01:50 ERROR OGG-01668 PROCESS ABENDING.
```

構成の問題例

次に、RAC環境のGoldenGateで発生する可能性がある構成の問題と、それらの診断および解決方法を示します。

mount-dbfs.confファイルの不適切なパラメータ設定

XAGがDBFSのマウントに失敗すると、コマンドライン(手動のagctlコマンドを実行している場合)かXAGログ・ファイルのいずれかで、失敗が報告されます。

```
$ agctl start goldengate <instance_name> --node exadb-node1
CRS-2672: Attempting to start 'dbfs_mount' on 'exadb-node1'
CRS-2674: Start of 'dbfs_mount' on 'exadb-node1' failed
CRS-2679: Attempting to clean 'dbfs_mount' on 'exadb-node1'
CRS-2681: Clean of 'dbfs_mount' on 'exadb-node1' succeeded
CRS-4000: Command Start failed, or completed with errors.
```

XAGログ・ファイル(agctl_goldengate_grid.trc)には、他のログ・ファイルやトレース・ファイルの参照時に使用できるタイムスタンプが表示されるという利点があります。

```
2022-04-19 15:32:16: executing cmd: /u01/app/19.0.0.0/grid/bin/crsctl start resource
xag.<INSTANCE_NAME>.goldengate -f -n exadb-node1
2022-04-19 15:32:19: Command output:
> CRS-2672: Attempting to start 'dbfs_mount' on 'exadb-node1'
> CRS-2674: Start of 'dbfs_mount' on 'exadb-node1' failed
> CRS-2679: Attempting to clean 'dbfs_mount' on 'exadb-node1'
> CRS-2681: Clean of 'dbfs_mount' on 'exadb-node1' succeeded
> CRS-4000: Command Start failed, or completed with errors.
>End Command output
2022-04-19 15:32:19: start resource failed rc=1
```

次に、CRSトレース・ファイル(crsd_scriptagent_oracle.trc)を確認します。これにはDBFSがマウントに失敗した理由が示されています。次に、mount-dbfs.confファイルの誤ったパラメータ設定によって発生するエラーの例を示します。

- 不適切なDBNAME

```
2022-04-19 15:32:16.679 : AGFW:1190405888: {1:30281:17383} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 15:32:16.680 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 15:32:16.732 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdg
2022-04-19 15:32:17.883 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 ORACLE_SID is
2022-04-19 15:32:17.883 :CLSDYNAM:1192507136: [dbfs_mount]{1:30281:17383}
[start]
 No running ORACLE_SID available on this host, exiting
2022-04-19 15:32:17.883 : AGFW:1192507136: {1:30281:17383} Command: start for
 resource: dbfs_mount 1 1 completed with invalid status: 2
```

- 不適切なMOUNT_POINT

```
2022-04-19 16:45:14.534 : AGFW:1734321920: {1:30281:17604} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 16:45:14.535 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 16:45:14.586 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdgs
```

```
2022-04-19 16:45:15.638 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 ORACLE_SID is ggdg1
2022-04-19 16:45:15.738 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 spawning dbfs_client command using SID ggdg1
2022-04-19 16:45:20.745 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 fuse: bad mount point `/mnt/dbfs': No such file or directory
2022-04-19 16:45:21.747 :CLSDYNAM:1736423168: [dbfs_mount]{1:30281:17604}
[start]
 Start - OFFLINE
2022-04-19 16:45:21.747 : AGFW:1736423168: {1:30281:17604} Command: start for
 resource: dbfs_mount 1 1 completed with status: FAIL
```

- 不適切なDBFS_USERまたはDBFS_PASSWD

```
2022-04-19 16:47:47.855 : AGFW:1384478464: {1:30281:17671} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 16:47:47.856 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 16:47:47.908 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdgs
2022-04-19 16:47:48.959 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 ORACLE_SID is ggdg1
2022-04-19 16:47:49.010 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 spawning dbfs_client command using SID ggdg1
2022-04-19 16:47:55.118 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 Fail to connect to database server. Error: ORA-01017: invalid
username/password;
 logon denied
2022-04-19 16:47:55.118 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
2022-04-19 16:47:56.219 :CLSDYNAM:1386579712: [dbfs_mount]{1:30281:17671}
[start]
 Start - OFFLINE
2022-04-19 16:47:56.220 : AGFW:1386579712: {1:30281:17671} Command: start for
 resource: dbfs_mount 1 1 completed with status: FAIL
```

- 不適切なORACLE_HOME

```
2022-04-19 16:50:38.952 : AGFW:567502592: {1:30281:17739} dbfs_mount
 1 1 state changed from: UNKNOWN to: STARTING
2022-04-19 16:50:38.953 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 Executing action script: /u01/oracle/scripts/mount-dbfs.sh[start]
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 mount-dbfs.sh mounting DBFS at /mnt/dbfs from database ggdgs
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 /u01/oracle/scripts/mount-dbfs.sh: line 136:
 /u01/app/oracle/product/19.0.0.0/rdbms/bin/srvctl: No such file or directory
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 /u01/oracle/scripts/mount-dbfs.sh: line 139:
 /u01/app/oracle/product/19.0.0.0/rdbms/bin/srvctl: No such file or directory
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 ORACLE_SID is
```

```
2022-04-19 16:50:39.004 :CLSDYNAM:569603840: [dbfs_mount]{1:30281:17739}
[start]
 No running ORACLE_SID available on this host, exiting
2022-04-19 16:50:39.004 : AGFW:569603840: {1:30281:17739} Command: start for
 resource: dbfs_mount 1 1 completed with invalid status: 2
```

これらの構成の問題を解決するには、mount-dbfs.confに適切なパラメータ値を設定します。

DBFSのファイル・ロックの問題

Oracle Database 12cリリース2 (12.2)を使用している場合にnolock DBFSマウント・オプションを使用しないと、GoldenGateプロセスがチェックポイントまたは証跡ファイルをロックしようとすると問題が発生する可能性があります。バグ22646150のパッチを適用したOracle Database 11gリリース2 (11.2.0.4)または12cリリース1 (12.1)を使用している場合も、同じ問題が発生します。このパッチは、Oracle Database 12cリリース2 (12.2)に合わせて、DBFSがファイルのロックを処理する方法を変更します。nolock DBFSマウント・オプションを追加するには、データベースにバグ27056711のパッチを適用する必要があります。データベースにバグ22646150のパッチが適用されていない場合は、バグ27056711のパッチとnolockマウント・オプションは必要ありません。

次に、GoldenGate Microservices Architectureのロック問題を診断する例を示します。

XAGを使用してデプロイメントを開始すると、1つ以上のファイルでロック競合が検出されたために、1つ以上のプロセスが開始されないことがあります。これは、デプロイメントが正常に停止する機会を得られなかったRACノードのフェイルオーバー後に頻発します。

デプロイメント・サーバーのプロセスの1つの起動に失敗した場合(管理サーバー、パフォーマンス・メトリック・サーバー、分散サーバー、レシーバ・サーバーまたはサービス・マネージャ)は、そのデプロイメントのvar/logディレクトリにある特定のサーバーのログ・ファイルを確認します。

たとえば、ログ・ファイル/mnt/dbfs/goldengate/deployments/<INSTANCE_NAME>/var/log/pmsrvr.logでは、起動時に次のエラーが表示されます。

```
2022-04-11T12:41:57.619-0700 ERROR| SecureStore failed on open after
 retrying due to extended file lock. (main)
2022-04-11T12:41:57.619-0700 ERROR| SecureStore failed to close (28771). (main)
2022-04-11T12:41:57.619-0700 INFO | Set network configuration. (main)
2022-04-11T12:41:57.619-0700 INFO | Asynchronous operations are enabled with default
 synchronous wait time of 30 seconds (main)
2022-04-11T12:41:57.619-0700 INFO | HttpServer configuration complete. (main)
2022-04-11T12:42:07.674-0700 ERROR| Unable to lock process file, Error is [1454]
 - OGG-01454 (main)
2022-04-11T12:42:07.675-0700 ERROR| Another Instance of PM Server is Already Running
 (main)
```

Extractプロセスは、デプロイメント・ログ・ファイル・ディレクトリにあるER-events.logログ・ファイルで、起動の失敗を報告します。

たとえば、/mnt/dbfs/goldengate/deployments/<instance_name>/var/log/ER-events.logは次のエラーを示します。

```
2022-04-11T00:14:56.845-0700 ERROR OGG-01454 Oracle GoldenGate Capture for
 Oracle, EXT1.prm: Unable to lock file
 "/mnt/dbfs/goldengate/deployments/<instance_name>/var/run/EXT1.pce" (error 11,
Resource
 temporarily unavailable). Lock currently held by process id (PID) 237495.
2022-04-11T00:14:56.861-0700 ERROR OGG-01668 Oracle GoldenGate Capture for Oracle,
 EXT1.prm: PROCESS ABENDING.
```

次に、起動に失敗するプロセスが、どのRACノードでも実行されていないことを確認します。

例:

```
$ ps -ef|grep EXT1|grep -v grep
```

プロセスが実行されていないと判断したら、デプロイメントを正常に停止し、ファイル・システムをアンマウントして、正しいDBFSパッチを適用する必要があります。

例:

```
$ agctl stop goldengate <INSTANCE_NAME>
$ crsctl stop resource dbfs_mount
```

DBFSマウント・オプションを確認します。

```
$ ps -ef|grep dbfs_client
oracle 204017 1 0 14:37 ?
 00:00:00 /u01/app/oracle/product/19.1.0.0/dbhome_1/bin/dbfs_client dbfs@dbfs.local
 -o allow_other,failover,direct_io /mnt/dbfs
```

明らかにnolockマウント・オプションが使用されていないため、これがロック・エラーの原因になります。

前述のガイドラインを使用して、DBFSパッチが必要かどうかを判断します。その後、デプロイメントの一部である、すべてのOracle RACノードのmount-dbfs.confファイルに、nolockマウント・オプションを追加します。

例:

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

最後に、デプロイメントを再起動します。

```
$ agctl start goldengate <INSTANCE_NAME>
```

第VI部 Oracle Database Cloudのベスト・プラクティス

25 Oracle Maximum Availability ArchitectureとOracle Autonomous Database

Oracle Maximum Availability Architecture (MAA)は、Oracleの高可用性、データ保護および障害時リカバリ・テクノロジを統合して使用するために、Oracleのエンジニアが何年もかけて開発した一連のベスト・プラクティスです。

Oracle MAAの主な目的は、Oracle Cloud MAAアーキテクチャおよびソリューションを使用して、Oracleのシステムやデータベース・プラットフォームで実行されているOracleデータベースとアプリケーションのリカバリ時間目標(RTO)およびリカバリ・ポイント目標(RPO)を満たすことです。

MAAリファレンス・アーキテクチャおよびそれに関連する利点と潜在的なRTOおよびRPOターゲットの概要は、Oracle MAAリファレンス・アーキテクチャを参照してください。また、Autonomous Database CloudはExadata Cloudプラットフォームで実行されるため、差別化された固有のOracle Exadata Cloud HAおよびデータ保護の利点については、「Oracle Exadata Cloud SystemsにおけるOracle Maximum Availability Architecture」を参照してください。

最大可用性アーキテクチャでは、テストおよび開発ライフ・サイクルを通してカオス・エンジニアリングを活用することで、Oracle Cloudにおける障害やメンテナンス・イベントの際にアプリケーションとデータベースのエンドツーエンドの可用性を確保したり、最適なレベルに維持することを確実化します。カオス・エンジニアリングは、システムが本番環境で乱流状態に耐えることができるよう信頼性を高めるためにシステムで実験を行う分野です。具体的には、MAAでは様々な障害や計画メンテナンス・イベントを積極的に注入して、開発、ストレスおよびテスト・サイクルを通してアプリケーションとデータベースに対する影響を評価します。この実験により、ベスト・プラクティス、不具合および知見が導出され、その知識を応用してクラウドのMAAソリューションを進化および向上させます。

デフォルトの高可用性オプションを使用したOracle Autonomous Database (MAA Silver)

高可用性は、稼働時間の要件が厳しく、データ損失の許容範囲がゼロであるか小さいすべての開発、テストおよび本番データベースに適しています。デフォルトでは、Autonomous Databaseの可用性は高く、局所的なソフトウェアおよびハードウェア障害から保護するマルチノード構成が組み込まれています。

それぞれのAutonomous Databaseアプリケーション・サービスは、少なくとも1つのOracle Real Application Clusters (Oracle RAC)インスタンスに配置され、計画外停止または計画メンテナンス・アクティビティの際に別の使用可能なOracle RACインスタンスにフェイルオーバーするオプションを備えており、ゼロまたはゼロに近い停止時間を実現します。

Autonomous Databaseの自動バックアップはOracle Cloud Infrastructure Object Storageに格納され、別の可用性ドメイン(使用可能な場合)にレプリケートされます。障害発生時には、これらのバックアップを使用してデータベースをリストアできます。Exadata Cloud at Customerを使用するAutonomous Databaseの場合、NFSまたはZero Data Loss Recovery Appliance (ZDLRA)にバックアップするオプションがありますが、これらのバックアップのレプリケーションはお客様の責任です。

データベースのメジャー・アップグレードは自動化されています。Autonomous Database Serverlessの場合、停止時間は最小限です。

1か月当たりの稼働時間に関するサービス・レベル合意(SLA)は99.95% (1か月当たり最大22分の停止時間)です。ほとんどの月で停止時間がゼロになるアプリケーションの稼働時間SLAを達成するには、次の「アプリケーション稼働時間の確保」を参照してください。

次の表は、様々な停止のリカバリ時間目標およびリカバリ・ポイント目標(データ損失の許容範囲)を示しています。

表25-1 デフォルトの高可用性ポリシーのリカバリ時間(RTO)およびリカバリ・ポイント(RPO)に関するサービス・レベル目標

障害およびメンテナンス・イベント	データベースの停止時間	サービス・レベルの停止時間(RTO)	潜在的なサービス・レベルのデータ損失(RPO)
次のものを含む、局所的なイベント: ● Exadata クラスタ・ネットワーク・トポロジの障害 ● ストレージ(ディスクおよびフラッシュ)の障害 ● データベース・インスタンスの障害 ● データベース・サーバーの障害 ● ソフトウェアおよびハードウェアの定期的なメンテナンス更新	ゼロ	ほぼゼロ	ゼロ
スタンバイ・データベースが存在しないため、バックアップからのリストアを必要とするイベント: ● データ破損 ● データベース全体の障害 ● 全面的なストレージ障害 ● マルチ AD リージョンの可用性ドメイン(AD)	数分から数時間 (Autonomous Data Guard なし)	数分から数時間 (Autonomous Data Guard なし)	Oracle Autonomous Database on Dedicated Exadata Infrastructure の場合、15 分 Autonomous Database Serverless の場合、1 分 (Autonomous Data Guard なし)
非ローリング・ソフトウェア更新またはデータベース・アップグレードを必要とするイベント	Autonomous Database Serverless の場合、10 分未満 Autonomous Database on Dedicated Infrastructure の場合、数分から数時間 (Autonomous Data Guard なし)	Autonomous Database Serverless の場合、10 分未満 Autonomous Database on Dedicated Infrastructure の場合、数分から数時間 (Autonomous Data Guard なし)	ゼロ

上の表において、バックアップからのリストアを必要とするイベントの停止時間は、障害の性質によって異なります。最も楽観的なケースでは、物理的なブロック破損が検出され、ブロック・メディア・リカバリを使用してブロックが数分で修復されます。この場合、影響を受けるのはデータベースのごく一部のみであり、データ損失はゼロです。より悲観的なケースでは、データベースまたはクラスタ全体で障害が発生し、すべてのアーカイブを含む最新のデータベース・バックアップを使用して、データベースがリストアおよびリカバリされます。

データ損失は、最後に成功したアーカイブ・ログ・バックアップによって制限されます。この頻度は、Autonomous Database on Dedicated Infrastructureの場合に15分ごと、Autonomous Database Serverlessの場合に1分ごとです。アーカイブまたはREDOは、今後のリカバリ目的でOracle Cloud Infrastructure Object Storageまたはファイル・ストレージ・サービスにバックアップされます。データ損失は数秒間、または最悪の場合、最後に成功したアーカイブ・ログと、外部ストレージにアーカイブされなかったオンラインREDOログの残りのREDOの前後における数分間のデータ損失となります。

Autonomous Data Guardオプションを使用したOracle Autonomous Database (MAA Gold)

Autonomous Data Guardは、データの破損による障害やデータベースまたはサイトの障害に対する稼働時間の要件がより厳しく、同時にAutonomous Databaseの高可用性オプションの利点も求められるミッションクリティカルな本番データベースについて有効化します。

また、読取り専用のスタンバイ・データベースは、レポート作成、問合せおよび一部の更新をオフロードするための拡張アプリケーション・サービスを提供します。読取り専用のスタンバイ・データベースは、Autonomous Data Guard on Dedicated Infrastructureでのみ使用可能です。

Autonomous Data Guardを有効にすると、同じ可用性ドメイン、別の可用性ドメインまたは別のリージョンに配置されたExadataラックに、対称スタンバイ・データベースが1つ追加されます。プライマリ・データベース・システムとスタンバイ・データベース・システムは、Data Guardのロール・トランジション後にパフォーマンス・サービス・レベルが維持されるように、対称的に構成されます。Autonomous Database Serverlessでは2つのスタンバイ・データベースの構成がサポートされており、Autonomous Database on Dedicated Infrastructureは、現時点では単一のデータベースに制限されています。Autonomous Database Serverlessの場合、複数スタンバイ構成は、同じリージョン内のローカル・スタンバイ・データベースとリージョン間スタンバイ・データベースで構成されます。

Oracle Autonomous Data Guardでは、アプリケーション・パフォーマンスへの影響をゼロにするために、デフォルトで非同期REDO転送(最大パフォーマンス・モード)が提供されます。スタンバイ・データベースは、同じ可用性ドメイン内に配置することも、複数の可用性ドメインやリージョンにわたって配置することもできます。MAAでは、最適な障害分離のために、スタンバイを別々の可用性ドメインまたは別のリージョンに配置することをお薦めします。Data Guardのデータ損失ゼロの保護は、同期REDO転送(最大可用性モード)を構成することで実現できます。ただし、同期REDO転送による最大可用性データベース保護モードは、Autonomous Database on Dedicated Infrastructureでのみ使用でき、スタンバイ・データベースは、通常、同じリージョン内の異なる可用性ドメインに配置されます。あるいは、リージョン間のラウンド・トリップ待機時間が最小(5ミリ秒未満)の場合は、複数リージョン間に配置され、障害分離を提供しながらアプリケーションのレスポンス時間とスループットへの影響が無視できる程度になります。さらに、ローカルおよびリモートの仮想クラウド・ネットワーク・ピアリングによって、プライマリ・サーバーとスタンバイ・サーバーの間のあらゆるトラフィックについて、複数の可用性ドメインおよびリージョンにわたってセキュアで高帯域幅のネットワークが提供されます。

バックアップはプライマリ・データベースとスタンバイ・データベースの両方で自動的にスケジュールされ、Oracle Cloud Infrastructure Object Storageに格納されます。Exadata Cloud at Customerを使用するAutonomous Databaseの場合、NFSまたはZero Data Loss Recovery Applianceにバックアップするオプションがありますが、これらのバックアップのレプリケーションはお客様の責任です。プライマリ・データベースとスタンバイ・データベースの両方が失われる二重障

害が発生したときには、これらのバックアップを使用してデータベースをリストアできます。

1か月当たりの稼働時間に関するサービス・レベル合意(SLA)は99.995% (1か月当たり最大132秒の停止時間)で、下の表に記載されているように、リカバリ時間目標(停止時間)とリカバリ・ポイント目標(データ損失)は低くなっています。ほとんどの月で停止時間がゼロになるアプリケーションの稼働時間SLAを達成するには、「アプリケーション稼働時間の確保」(XREF)を参照してください。

Autonomous Database Serverlessでの自動Data Guardフェイルオーバーでは、データ損失しきい値サービス・レベルがサポートされており、データ損失がそのしきい値を下回った場合にスタンバイ・データベースへの自動フェイルオーバーが開始されます。Autonomous Database Serverlessではデータ損失ゼロのフェイルオーバーは保証されませんが、プライマリ・データベースに障害が発生し、プライマリ・システム・コンテナおよびインフラストラクチャが使用可能な状態で、残りのREDOをスタンバイ・データベースに送信して適用できる場合には可能です。Autonomous Database on Dedicated Infrastructureによる自動Data Guardフェイルオーバーでは、データ損失ゼロまたはデータ損失しきい値の低いサービス・レベルがサポートされます。いずれの場合も、プライマリ・データベース、クラスタまたはデータ・センターの障害に対して、データ損失のサービス・レベルを保証できる場合、自動Autonomous Data Guardフェイルオーバーが発生します。ターゲット・スタンバイが新しいプライマリ・データベースになり、すべてのアプリケーション・サービスが自動的に有効になります。OCIコンソールには、手動のデータ・フェイルオーバー・オプションが用意されています。手動Data Guardフェイルオーバー・オプションの場合、稼働時間SLAの計算停止時間は、Data Guardフェイルオーバー操作を実行した時点で開始され、新しいプライマリ・サービスが有効化されると終了します。

アプリケーションやビジネスの要件とデータ・センターの可用性に応じて、データベース・フェイルオーバー・サイトを、同じ可用性ドメイン、同じリージョン内の別の可用性ドメイン、または別のリージョンに配置するかを選択できます。

表25-2 Autonomous Data Guardのリカバリ時間(RTO)およびリカバリ・ポイント(RPO)に関するサービス・レベル目標

障害およびメンテナンス・イベント	**サービス・レベルの停止時間(RTO)[1]**	**潜在的なサービス・レベルのデータ損失(RPO)**
次のものを含む、局所的なイベント: ● Exadata クラスタ・ネットワーク・ファブリックの障害 ● ストレージ(ディスクおよびフラッシュ)の障害 ● データベース・インスタンスの障害 ● データベース・サーバーの障害 ● ソフトウェアおよびハードウェアの定期的なメンテナンス更新	ゼロまたはほぼゼロ	ゼロ
次のものを含む、Autonomous Data Guard を使用してスタンバイ・データベースにフェイルオーバーする必要があるイベント: ● データ破損(Data Guard には物理的な破損用の自動ブロック修復があるため [2]、フェイルオーバー操作	数秒から 2 分 [4]	最大可用性保護モード(同期 REDO 転送を使用)ではゼロ。リージョン内スタンバイ・データベースで最もよく使用されます。これは Autonomous Data Guard on Dedicated Infrastructure で使用できます。 最大パフォーマンス保護モード(非同期 REDO 転送を使用)ではほぼゼロ。リージョン間スタンバイ・データベースで最もよく使用されます。リージョン内スタンバイ・データベースにも、アプリケー

障害およびメンテナンス・イベント	サービス・レベルの停止時間(RTO)[1]	潜在的なサービス・レベルのデータ損失(RPO)
は、論理的な破損または広範囲に及ぶデータ破損についてのみ必要です) ● データベース全体の障害 ● 全面的なストレージ障害 ● 可用性ドメインまたはリージョンの障害 3		ションへの影響をゼロにするために使用されます。これは、Autonomous Data Guard on Dedicated Infrastructure と Autonomous Database Serverless の両方に適用されます。RPO は 10 秒未満です。 RPO は、プライマリ・クラスタとスタンバイ・クラスタの間のネットワーク帯域幅およびスループットによって影響を受ける可能性があります。

[1] サービス・レベルの停止時間(RTO)では、自動フェイルオーバーの開始前にソースにアクセスできないようにするために、複数のハートビートを含む検出時間が除外されます。

[2] 物理的な破損用のActive Data Guard自動ブロック修復機能は、Autonomous Data Guard on Dedicated Infrastructureでのみ使用可能です。

[3]リージョン障害保護を使用できるのは、スタンバイが複数のリージョンにまたがって配置されている場合のみです。

[4] バックエンドのAutonomous Data Guardロール・トランジションのタイミングは、クラウド・コンソールのリフレッシュ率で示されるタイミングよりもはるかに高速です。

Autonomous Database on Dedicated InfrastructureとAutonomous Database Serverlessの両方がMAA Goldの検証と認定を受けています。Autonomous Database on Dedicated Infrastructureは、同じリージョン内のスタンバイ・データベース、および別のリージョンのスタンバイ・データベースで検証され、スタンバイ・ターゲットがプライマリと対称であったときに前述のSLAが満たされました。RTOおよびRPO SLAは、最大1000 MB/秒のREDO率で満たされました。Autonomous Database Serverlessは、同じリージョン内のスタンバイ・データベースでのみ検証および認定され、スタンバイ・ターゲットに対称リソースがあるときに前述のSLAを満たしました。RTOおよびRPO SLAは、ターゲットのAutonomous Data Guardプラガブル・データベースが存在するコンテナ・データベース(CDB)全体に対して最大300 MB/秒のREDO率で満たされました。

アプリケーション稼働時間の確保

テナンシのAutonomous Databaseリソースにアクセスできるように、Oracle Cloud Infrastructureへのネットワーク接続の信頼性が高いことを確認します。

ガイドラインに従ってAutonomous Databaseに接続します(Autonomous Database Serverlessまたは Autonomous Database on Dedicated Exadata Infrastructureを参照)。アプリケーションは、事前定義されたサービス名に接続し、適切な`tnsnsames.ora`ファイルおよび`sqlnet.ora`ファイルを含むクライアント資格証明をダウンロードする必要があります。要件に合うように、特定のアプリケーション・サービスの`drain_timeout`属性を変更することもできます。

計画停止および計画外停止を通して継続的なアプリケーション・サービスを実現する方法の詳細は、「アプリケーションの継続的な可用性の構成」を参照してください。Validating Application Failover Readiness (Doc ID 2758734.1)に従って、アプリケーションの準備状況をテストすることをお薦めします。

データベース・インスタンスの再起動が必要なOracle Exadata Cloud Infrastructureの計画メンテナンス・イベントの場合、

Oracle RACインスタンスをすべて停止する前に、Oracleにより自動的に、サービスおよび排出セッションが別の使用可能なOracle RACインスタンスに再配置されます。MAAチェックリストに準拠するOLTPアプリケーションについては、サービスの排出および再配置によってアプリケーションの停止時間がなくなります。

実行時間が長いバッチ・ジョブやレポートなど、一部のアプリケーションでは、排出タイムアウトを長くしても、排出および再配置を正常に行うことができない場合があります。そうしたアプリケーションについては、これらのタイプのアクティビティを除外してソフトウェアの計画メンテナンス・ウィンドウをスケジュールするか、計画メンテナンス・ウィンドウの前にこれらのアクティビティを停止することをお薦めします。たとえば、バッチ・ウィンドウ外になるように計画メンテナンス・ウィンドウを再スケジュールしたり、計画メンテナンス・ウィンドウの前にバッチ・ジョブを停止できます。

26 Oracle Exadata Cloud SystemsにおけるOracle Maximum Availability Architecture

Oracle Exadata Cloud Infrastructure (ExaDB-D)とOracle Exadata Cloud@Customer (ExaDB-C@C)のOracle Maximum Availability Architectureでは、クラウドの自動化とライフサイクル操作の両方と統合された固有の高可用性、データ保護および障害時リカバリ保護が提供され、Oracle Exadata Cloudシステムをエンタープライズ・データベースおよびアプリケーションに最適なクラウド・ソリューションにすることができます。

Oracle CloudのMAAアーキテクチャおよび機能の詳細なウォークスルーは、Oracle Cloud: Maximum Availability Architectureを参照してください。

Oracle Maximum Availability Architectureの利点

- デプロイメント: Oracle Exadata Cloudシステム(ExaDB-DおよびExaDB-C@C)は、ストレージ、ネットワーク、オペレーティング・システム、Oracle Grid InfrastructureおよびOracle Databaseの構成のベスト・プラクティスを含む、Oracle Maximum Availability Architectureのベスト・プラクティスを使用してデプロイされます。ExaDB-Dは、きわめて高いスケーラビリティ、可用性および拡張度でエンタープライズOracleデータベースを実行するように最適化されています。
- Oracle Maximum Availability Architectureデータベース・テンプレート: Oracle Cloudの自動化で作成されたすべてのOracle Cloudデータベースでは、ExaDB-D用に最適化されたOracle Maximum Availability Architectureのデフォルト設定を使用します。

  カスタム・スクリプトを使用してクラウド・データベースを作成することはお薦めしません。メモリーおよびシステム・リソースの設定を調整する以外に、以前のデータベース・パラメータ設定(特に文書化されていないパラメータ)を移行しないようにしてください。潜在的なオーバーヘッドが原因で、1つの有益なデータベース・データ保護パラメータDB_BLOCK_CHECKINGはデフォルトで有効になっていません。MAAでは、アプリケーションのパフォーマンスへの影響を評価し、パフォーマンスへの影響が妥当である場合はこの設定を有効にすることを推奨しています。
- バックアップとリストアの自動化: Oracle Cloud Infrastructure Object Storageへの自動バックアップを構成すると、リージョンに複数の可用性ドメインが存在する場合にバックアップ・コピーによって追加の保護が提供され、RMANによってクラウド・データベースのバックアップに物理的な破損がないかどうかが検証されます。

  データベースのバックアップは毎日行われ、完全バックアップが週に1回、増分バックアップが他のすべての日に行われます。アーカイブ・ログのバックアップは、障害発生時のデータ損失の可能性を低減するために頻繁に行われます。アーカイブ・ログ頻度は通常30分です。
- Oracle Exadata Database Machine固有の利点: Oracle Exadata Database Machineは、Oracleが提供する最適なOracle Maximum Availability Architectureプラットフォームです。Exadataは、最もミッション・クリティカルなエンタープライズ・アプリケーションをサポートするハードウェア、ソフトウェア、データベースおよび可用性のイノベーションを使用して設計されています。

  具体的に言うと、Exadataには、Oracleを他のプラットフォーム・ベンダーやクラウド・ベンダーと差別化する独自の高可用性、データ保護およびサービス品質機能が用意されています。最高の可用性、安定性およびパフォーマンスを確保するには、アプリケーションおよびデータベース・システムのリソース・ニーズ(十分なCPU、メモリーおよびI/Oリソースなど)を満たすようにExadataクラウド・システムのサイズを設定することが非常に重要です。多数のデータベースを同じクラスタに統合する場合は、適切なサイズ設定が特に重要です。

Oracle Exadata Database MachineシステムにおけるOracle Maximum Availability Architectureの利点の包括的なリストは、Exadata Database Machine: Maximum Availability Architectureのベスト・プラクティスを参照してください。

これらの利点の例は次のとおりです。

- 高い可用性と短時間のブラウンアウト: 完全な冗長性を備えたフォルト・トレラントなハードウェアがストレージ、ネットワークおよびデータベース・サーバーに存在します。Oracle Real Application Clusters (Oracle RAC)、Oracle Clusterware、Oracle Database、Oracle Automatic Storage Management、Oracle Linux、Oracle Exadata Storage Serverなど、可用性に優れたリジリエントなソフトウェアを使用すると、アプリケーションは、計画外停止や計画メンテナンス・イベントを通してアプリケーション・サービス・レベルを維持できます。

  たとえば、Exadataには、データベース・ノード、ストレージ・サーバーおよびネットワークの障害を2秒未満で検出して修復し、アプリケーションおよびデータベース・サービスの稼働時間とパフォーマンスを回復できる即時障害検出が用意されています。他のプラットフォームでは、同じタイプの障害について、30秒あるいは数分に及ぶブラックアウトと長期のアプリケーション・ブラウンアウトが発生することがあります。アプリケーションとデータベースのエンドツーエンドのブラウンアウトおよびブラックアウトを評価するための広範囲にわたる計画外停止および計画メンテナンス・テストを提供しているのは、Exadataプラットフォームのみです。

- データ保護: Exadataは、物理的および論理的なブロック破損の防止と検出に加えて、場合によっては自動修正をOracle Databaseに提供します。

  Exadata Hardware Assisted Resilient Data (HARD)チェックには、サーバー・パラメータ・ファイル、制御ファイル、ログ・ファイル、Oracleデータ・ファイルおよびOracle Data Guard Brokerファイルのサポートが含まれます(これらのファイルがExadataストレージに格納されている場合)。このインテリジェントなExadataストレージ検証では、HARDチェックが失敗すると、破損したデータがディスクに書き込まれなくなるため、データベース業界で以前は防止できなかった大規模なクラスの障害が排除されます。

  Exadata HARDチェックの例は次のとおりです。

  - REDOおよびブロック・チェックサム
  - 正しいログ順序
  - ブロック・タイプの検証
  - ブロック番号の検証
  - ブロック・マジック番号、ブロック・サイズ、順序番号、ブロック・ヘッダーおよびテール・データ構造などのOracleデータ構造

  Exadata HARDチェックは、Exadataストレージ・ソフトウェア(セル・サービス)から開始され、データベースの`DB_BLOCK_CHECKSUM`パラメータを有効にした後(クラウドではデフォルトで有効になっている)、透過的に機能します。Exadataは、HARDイニシアチブを現在サポートしている唯一のプラットフォームです。

  さらに、Oracle Exadata Storage Serverには、非侵入型の自動ハード・ディスク・スクラブおよび修復が用意されています。この機能では、アイドル時間中に定期的にハード・ディスクが検査され、修復されます。ハード・ディスクで不良セクターが検出された場合、Oracle Exadata Storage Serverは、別のミラー・コピーからデータを読み取ることによって不良セクターを修復するようOracle Automatic Storage Management (ASM)に自動的にリクエストを送信します。

  最後に、ExadataおよびOracle ASMでは、データ・ブロックがバッファ・キャッシュに読み込まれるときに破損を検出し、後続のデータベース書込みでデータ・ブロックの良好なコピーを使用してデータ破損を自動的に修復できます。この固有のインテリジェントなデータ保護により、Exadata Database MachineとExaDB-Dは、Oracleデータベースに最適

なデータ保護ストレージ・プラットフォームになります。

包括的なデータ保護のための最大可用性アーキテクチャのベスト・プラクティスは、別個のExadataインスタンス上のスタンバイ・データベースを使用して、Exadataのみで対処できない破損の検出、防止および自動修復を行うことです。また、スタンバイ・データベースによって、サイト、クラスタおよびデータベースの障害に起因する障害時の停止時間とデータ損失も最小限に抑えられます。

- レスポンス時間に関するサービス品質: レスポンス時間を確実に短く最適に維持するようにエンドツーエンドのサービス品質機能を備えているのはExadataのみです。データベース・サーバーのI/Oレイテンシ制限とExadataストレージのI/Oレイテンシ制限により、レスポンス時間が特定のしきい値を超えた場合に、読取りまたは書込みI/Oをパートナ・セルにリダイレクトできます。

  パフォーマンスが低下したり、予測できないためにストレージの信頼性がなくなった(ただし障害は発生していない)場合、ディスクまたはフラッシュ・キャッシュをオフラインに限定し、後でI/Oパフォーマンスが許容可能なレベルに戻ったことをヒューリスティックが示した場合にオンラインに戻すことができます。リソース管理は、最適化されたレベルでアプリケーションおよびデータベースが動作するように、重要なデータベース・ネットワークまたはI/O機能を優先するために役立ちます。

  たとえば、データベース・ログの書込みは、Exadataネットワークおよびストレージでのバックアップ・リクエストよりも優先されます。さらに、パートナ・フラッシュ・キャッシュがウォームされてフラッシュ・ミスが最小限に抑えられるようにすることで、ストレージ・ソフトウェアの更新中に迅速なレスポンス時間が維持されます。

- エンドツーエンドのテストと全体的なヘルス・チェック: OracleはOracle Exadata Cloud Infrastructure全体を所有しているため、エンドツーエンドのテストおよび最適化によって、オンプレミスとクラウドのどちらでホストされているかに関係なく、世界中のすべてのExadata顧客に利点がもたらされます。ミッション・クリティカルなシステムを実行するために必要な検証済の最適化および修正が、厳密なテストの後に一律に適用されます。ヘルス・チェックは、スタック全体を評価するように設計されています。

  ExadataのヘルスチェックユーティリティEXACHKは、Exadataクラウド対応であり、顧客の変更によって発生した可能性がある構成およびソフトウェアのアラートを強調表示します。現在、他のクラウド・プラットフォームでは、このようなエンドツーエンドのヘルス・チェックは提供されていません。Oracle Autonomous Databaseについては、EXACHKが自動的に実行されて、最大可用性アーキテクチャへの準拠が評価されます。Autonomous Database以外については、少なくとも月に1回と、ソフトウェア更新の前後にEXACHKを実行して、新しいベスト・プラクティスおよびアラートを評価することをお薦めします。

- 稼働時間の向上: 1か月当たりの稼働時間に関するサービス・レベル合意は99.95% (1か月当たり最大22分の停止時間)ですが、継続的なサービスのためのMAAベスト・プラクティスを使用すると、ほとんどの月で停止時間がゼロになります。

Exadataの機能および利点の完全なリスト: Oracle Exadata Database Machineの新機能

Oracle Maximum Availability Architectureのベスト・プラクティス・ペーパー: Oracle Maximum Availability Architecture (MAA)エンジニアリングはOracle Cloudチームと協力して、Oracle Cloud InfrastructureおよびセキュリティについてMAAプラクティスを統合します。継続的な可用性、Oracle Data Guard、Hybrid Data Guard、Oracle GoldenGate、および最大可用性アーキテクチャに関連する他のトピックについての追加情報は、Oracle CloudのMAAベスト・プラクティスを参照してください。

計画外停止に伴う予想される影響

次の表に、様々な計画外停止とそれに関連する潜在的なデータベースの停止時間、アプリケーション・レベルのリカバリ時間目標(RTO)およびデータ損失の可能性またはリカバリ・ポイント目標(RPO)を示します。Oracle Data Guardアーキテクチャの

場合、データベースの停止時間またはサービス・レベルの停止時間には、検出時間や顧客がクラウド・コンソールのData Guardフェイルオーバー操作を開始するまでに要した時間は含まれません。

表26-1 Exadata Cloudのソフトウェア更新による可用性およびパフォーマンスへの影響

障害およびメンテナンス・イベント	**データベースの停止時間**	**サービス・レベルの停止時間(RTO)**	**潜在的なサービス・レベルのデータ損失(RPO)**
次のものを含む、局所的なイベント: Exadata クラスタ・ネットワーク・トポロジの障害 ストレージ(ディスクおよびフラッシュ)の障害 データベース・インスタンスの障害 データベース・サーバーの障害	ゼロ	ほぼゼロ	ゼロ
スタンバイ・データベースが存在しないため、バックアップからのリストアを必要とするイベント: データ破損 データベース全体の障害 全面的なストレージ障害 可用性ドメイン	数分から数時間 (Data Guard なし)	数分から数時間 (Data Guard なし)	30 分 (Data Guard なし)
Data Guard を使用してフェイルオーバーするイベント: データ破損 データベース全体の障害 全面的なストレージ障害 可用性ドメインまたはリージョンの障害	数秒から数分 [1] 自動ブロック修復機能により、物理的な破損については停止時間ゼロ	数秒から数分 [1] 自動ブロック修復を完了する間、物理的な破損を検出するフォアグラウンド・プロセスは一時停止します	最大可用性(SYNC)についてはゼロ 最大パフォーマンス(ASYNC)についてはほぼゼロ

[1] リージョン障害から保護するには、プライマリ・データベースとは異なるリージョンにスタンバイ・データベースが必要です。

計画メンテナンスに伴う予想される影響

次の表に、様々なソフトウェア更新とそれに関連するデータベースおよびアプリケーションへの影響を示します。これは、Oracle Exadata Cloud@Customer (ExaDB-C@C)、Oracle Exadata Cloud Infrastructure (ExaDB-D) Gen2およびOracle Autonomous Database (ADB)を含む、Oracle Exadata Cloudインフラストラクチャすべてに適用可能です。

表26-2 Oracle Exadata Cloudのソフトウェア更新による可用性およびパフォーマンスへの影響

ソフトウェア更新	データベースへの影響	アプリケーションへの影響	スケジュール作成者	実行者
Exadata ネットワーク・ファブリック・スイッチ	データベースの再起動なしで停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト	Oracle が顧客の好みに基づいてスケジュール。顧客は再スケジュールできます	ADB と非 ADB の両方に対応した Oracle Cloud
Exadata ストレージ・サーバー	データベースの再起動なしで停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト Exadata ストレージ・サーバーは、冗長性を維持しながらローリング方式で更新されます Oracle Exadata System Software は、フラッシュ・キャッシュに最も頻繁にアクセスされる OLTP データのセカンダリ・ミラーをプリフェッチして、ストレージ・サーバーの再起動中にアプリケーション・パフォーマンスを維持します データベース・バッファ用の Exadata スマート・フラッシュは、ストレージ・サーバーの再起動後も維持されます Exadata 21.2 ソフトウェアでは、永続ストレージ索引および永続列キャッシュ	Oracle が顧客の好みに基づいてスケジュール。顧客は再スケジュールできます	ADB と非 ADB の両方に対応した Oracle Cloud

ソフトウェア更新	データベースへの影響	アプリケーションへの影響	スケジュール作成者	実行者
		機能により、ストレージ・サーバーのソフトウェア更新後に一貫した問合せパフォーマンスを実現できます		
Exadata Database ホスト - 月次インフラストラクチャ・セキュリティ・メンテナンス	ホストまたはデータベースの再起動なしで停止時間ゼロ	停止時間ゼロ	Oracle がスケジュール。顧客は再スケジュールが可能	ADB と非 ADB の両方に対応した Oracle Cloud
Exadata Database ホスト - 四半期インフラストラクチャ・メンテナンス	Oracle RAC ローリング更新により停止時間ゼロ	停止時間ゼロ 計画メンテナンスが完了するまで、Exadata Database のコンピュート・リソースが削減されます	Oracle が顧客の好みに基づいてスケジュール。顧客は再スケジュールできます	ADB と非 ADB の両方に対応した Oracle Cloud
Exadata データベース・ゲスト	Oracle RAC ローリング更新により停止時間ゼロ	停止時間ゼロ 計画メンテナンスが完了するまで、Exadata Database のコンピュート・リソースが削減されます	ADB の顧客	ADB に対応した Oracle Cloud ADB 以外については顧客が Oracle Cloud コンソール/API を使用
Oracle Database の四半期更新またはカスタム・イメージ更新	Oracle RAC ローリング更新により停止時間ゼロ	停止時間ゼロ 計画メンテナンスが完了するまで、Exadata Database のコンピュート・リソースが削減されます データベース OJVM を使用するアプリケーションのデータベースを四半期ごとにローリング更新する際には、特別な考慮事項が必要になります。詳細は、MOS ノート 2217053.1 を参照してください。	ADB の顧客	ADB に対応した Oracle CloudADB-D の場合は、Standby-First Patch のプラクティスが自動的に適用されます。 ADB 以外については顧客が Oracle Cloud コンソール/API または dbaascli ユーティリティを使用データベース・ホーム・パッチによるインプレースと、データベース移動によるアウトオブプレースでは、ソフトウェア更新が存在しま

ソフトウェア更新	データベースへの影響	アプリケーションへの影響	スケジュール作成者	実行者
				す。Data Guard およびスタンバイ・データベースで動作します(MOS 2701789.1 を参照)。
Oracle Grid Infrastructure の四半期更新またはアップグレード	Oracle RAC ローリング更新により停止時間ゼロ	停止時間ゼロ 計画メンテナンスが完了するまで、Exadata Database のコンピュート・リソースが削減されます	ADB の顧客	ADB に対応した Oracle Cloud ADB 以外については顧客が Oracle Cloud コンソール/API または dbaascli ユーティリティを使用
停止時間を伴う Oracle Database のアップグレード	数分から数時間の停止時間	数分から数時間の停止時間	ADB の顧客	ADB に対応した Oracle Cloud ADB 以外については顧客が Oracle Cloud コンソール/API または dbaascli ユーティリティを使用 Data Guard およびスタンバイ・データベースで動作します(MOS 2628228.1 を参照)。
停止時間がゼロに近い Oracle Database のアップグレード	DBMS_ROLLING、Oracle GoldenGate レプリケーション、またはプラガブル・データベースの再配置により最小限の停止時間	DBMS_ROLLING、Oracle GoldenGate レプリケーション、またはプラガブル・データベースの再配置により最小限の停止時間	非 ADB の顧客	共有 Exadata インフラストラクチャ上の ADB (ADB-S)に対応した Oracle Cloud は、アップグレードのユースケースでプラガブル・データベースの再配置を実行できます DBMS_ROLLING を利用する Autonomous 以外については、顧客が dbaascli を使用。 Exadata Cloud

ソフトウェア更新	データベースへの影響	アプリケーションへの影響	スケジュール作成者	実行者
				Database 19c Rolling Upgrade With DBMS_ROLLING (Doc ID 2832235.1)を参照してください ADB 以外については顧客が最大可用性アーキテクチャの一般的なベスト・プラクティスを使用

Exadataクラウド・システムには、データベースおよびアプリケーションのパフォーマンス・ニーズを調整するために使用できるエラスティック機能が多数用意されています。必要に応じてリソースを再配置することで、ターゲット・データベースおよびアプリケーションのシステム・リソースを最大化できるとともに、コストを最小化できます。次の表に、Oracle Exadata Cloud InfrastructureおよびVMクラスタのエラスティック更新と、それらの更新に関連するデータベースおよびアプリケーションへの影響を示します。特に指定されていないかぎり、これらの操作はすべて、Oracle CloudコンソールまたはAPIを使用して実行できます。

表26-3 Exadataのエラスティック操作による可用性およびパフォーマンスへの影響

VMクラスタの変更	データベースへの影響	アプリケーションへの影響
VM クラスタ・メモリーのスケール・アップまたはスケール・ダウン	Oracle RAC ローリング更新により停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト
VM クラスタ CPU のスケール・アップまたはスケール・ダウン	データベースの再起動なしで停止時間ゼロ	停止時間ゼロ アプリケーションのパフォーマンスとスループットは、使用可能な CPU リソースの影響を受けることがあります
データベース使用状況に応じた ASM ストレージのスケール・アップまたはスケール・ダウン(サイズ変更)	データベースの再起動なしで停止時間ゼロ	停止時間ゼロ アプリケーションのパフォーマンスへの影響は最小限に抑えられる可能性があります。
VM のローカル/u02 ファイル・システム・サイズのスケール・アップ (Exadata X8M 以降のシステム)	データベースの再起動なしで停止時間ゼロ	停止時間ゼロ
VM のローカル/u02 ファイル・システム・サイズのスケール・アップ (Exadata X8 以前のシステム)	Oracle RAC ローリング更新により停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト

VMクラスタの変更	データベースへの影響	アプリケーションへの影響
VM のローカル/u02 ファイル・システム・サイズのスケール・ダウン	スケール・ダウンについては Oracle RAC ローリング更新により停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト
Exadata ストレージ・セルの追加	データベースの再起動なしで停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト アプリケーションのパフォーマンスへの影響は最小限に抑えられる可能性があります
Exadata データベース・サーバーの追加	データベースの再起動なしで停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト Oracle RAC インスタンスと CPU リソースを追加すると、アプリケーションのパフォーマンスとスループットが向上する可能性があります
仮想マシン(VM)クラスタでのデータベース・ノードの追加/削除	データベースの再起動なしで停止時間ゼロ	ゼロから数秒(1 桁台)のブラウンアウト Oracle RAC インスタンスと CPU リソースを追加または削除すると、アプリケーションのパフォーマンスとスループットが向上または低下する可能性があります

これらのエラスティック変更の中にはかなりの時間を要するものがあり、アプリケーションの使用可能なリソースに影響を及ぼす可能性があるため、計画が必要です。

スケール・ダウンおよび削除による変更では、使用可能なリソースが減少します。リソースが減少してデータベースおよびアプリケーションの安定性に必要な量を下回ることがないように、また、アプリケーション・パフォーマンス目標を満たすように注意を払う必要があります。推定時間および計画の推奨事項は、次の表を参照してください。

表26-4 Exadataのエラスティック操作に関する顧客計画の推奨事項

VMクラスタの変更	推定タイミング	顧客計画の推奨事項
VM クラスタ・メモリーのスケール・アップまたはスケール・ダウン	サービスを排出する時間および Oracle RAC ローリング再起動 通常はノード当たり 15-30 分ですが、アプリケーションの排出によって異なる場合があります	アプリケーションの排出についての理解。 アプリケーションの継続的な可用性の実現を参照してください メモリーをスケール・ダウンする前に、データベースの SGA を引き続き hugepages に格納できることと、アプリケーション・パフォーマンスが引き続き許容可能であることを確認します。 予測可能なアプリケーション・パフォーマンスおよび安定性を確保するには: ● 監視して、重要な高ワークロード・パターンでメモ

VMクラスタの変更	推定タイミング	顧客計画の推奨事項
		リー・リソースが必要になる前にスケール・アップします ● データベースの SGA および PGA メモリーすべてが新しいメモリー・サイズに収まり、すべての SGA がシステムの hugepages に収容されないかぎり、メモリーのスケール・ダウンを回避します。
VM クラスタ CPU のスケール・アップまたはスケール・ダウン	オンライン操作は、通常、VM クラスタ当たり 5 分未満です。非常に小さい値から非常に大きい値へのスケール・アップ(10 以上の oCPU の増加)には、10 分かかる場合があります。	予測可能なアプリケーション・パフォーマンスおよび安定性を確保するには: ● 監視して、重要な高ワークロード・パターンで CPU リソースが必要になる前や、常に許容時間内に OCPU しきい値に到達する場合にスケール・アップします。 ● 負荷平均が少なくとも 30 分間しきい値を下回る場合にのみスケール・ダウンするか、固定ワークロード・スケジュールに基づいてスケール・ダウンします(たとえば、営業時間は 60 OCPU、営業時間外は 10 OCPU、バッチは 100 oCPU) ● 2 時間以内に複数のスケール・ダウン・リクエストを行わないでください
データベース使用状況に応じた ASM ストレージのスケール・アップまたはスケール・ダウン(サイズ変更)	時間は、使用されているデータベース・ストレージ容量およびデータベース・アクティビティによって異なります。使用されているデータベース・ストレージの割合が高くなるほど、サイズ変更操作(ASM リバランスを含む)にかかる時間が長くなります。 通常は数分から数時間。	Oracle ASM リバランスは自動的に開始されます。ストレージの冗長性は保持されます。非侵入型の ASM 指数制限を使用する固有のベスト・プラクティスにより、アプリケーション・ワークロードへの影響は最小限です。 サイズ変更およびリバランス操作を最適化できるように、非ピーク・ウィンドウを選択します。 時間は大きく異なる可能性があるため、操作が数時間かけて完了するものと想定して計画します。VM クラスタ当たりの既存のサイズ変更またはリバランス操作の時間を推定するには、GV$ASM_OPERATION を問い合せます。たとえば、顧客は 30 分ごとに次の問合せを実行して、どれくらいの作業が必要になる可能性があるか(EST_WORK)と、あとどれくらいの時間が必要になる可能性があるか(EST_MINUTES)を評価できます。 `select operation, pass, state, sofar, est_work, est_minutes from gv$asm_operation where operation='REBAL';`

VMクラスタの変更	推定タイミング	顧客計画の推奨事項
		リバランスが進行するにつれて、推定された統計はより正確になる傾向がありますが、同時ワークロードによって異なる場合があることに注意してください。
VM のローカル/u02 ファイル・システム・サイズのスケール・アップ (Exadata X8M 以降)	オンライン操作は、通常、VM クラスタ当たり 5 分未満です	VM ローカル・ファイル・システムの領域はローカル・データベース・ホスト・ディスクに割り当てられ、そのデータベース・ホストにプロビジョニングされたすべての VM クラスタの、すべての VM ゲストによって共有されます。ローカル/u02 ファイル・システムのスケール・ダウンは RAC ローリング方式で実行する必要があり、アプリケーションの中断を引き起こす可能性があるため、同じ Exadata インフラストラクチャ上の他の VM クラスタでスケール・アップするための領域が残らないように、1 つの VM クラスタ上でローカル/u02 ファイル・システムの領域を不必要にスケール・アップしないでください。
VM のローカル/u02 ファイル・システム・サイズのスケール・アップ (Exadata X8 以前)	サービスを排出する時間および Oracle RAC ローリング再起動通常はノード当たり 15-30 分ですが、アプリケーションの排出設定によって異なる場合があります。	アプリケーションの排出についての理解。 アプリケーションの継続的な可用性の実現を参照してください
VM のローカル/u02 ファイル・システム・サイズのスケール・ダウン	サービスを排出する時間および Oracle RAC ローリング再起動通常はノード当たり 15-30 分ですが、アプリケーションの排出設定によって異なる場合があります。	アプリケーションの排出についての理解 アプリケーションの継続的な可用性の実現を参照してください
Exadata ストレージ・セルの追加	管理者が分散方法を選択するために、より多くの使用可能領域を作成するオンライン操作。 VM クラスタの数、データベース・ストレージの使用状況およびストレージ・アクティビティに応じて、通常は操作当たり 3-72 時間。非常にアクティブなデータベースと負荷が大きいストレージ・アクティビティでは、最大 72 時間かかることがあります。 ストレージ・セルの追加操作の一部として、この操作には 2 つの部分があり	操作が数日かけて完了することがあるため、ストレージ容量の使用率が 1 か月以内に 80%に達したときにストレージを追加することを計画します。 Oracle ASM リバランスは自動的に開始されます。ストレージの冗長性は保持されます。非侵入型の ASM 指数制限を使用する固有のベスト・プラクティスにより、アプリケーション・ワークロードへの影響は最小限です。 時間は大きく異なる可能性があるため、ストレージが使用可能になる前に操作が数日かけて完了するものと想定して計画します。VM クラスタ当たりの既存のサイズ変更またはリバランス操作の時間を推定するには、GV$ASM_OPERATION を問い合せます。たとえば、顧客は 30 分ごとに次の問合せ

VMクラスタの変更	推定タイミング	顧客計画の推奨事項
	ます。1) ストレージの追加の一部としてシステムにストレージが追加されます。2) 管理者は別個の操作として、ASM ディスク・グループを拡張する VM クラスタを決定する必要があります。	を実行して、どれくらいの作業が必要になる可能性があるか(EST_WORK)と、あとどれくらいの時間が必要になる可能性があるか(EST_MINUTES)を評価できます。 `select operation, pass, state, sofar, est_work, est_minutes from gv$asm_operation where operation='REBAL';` リバランスが進行するにつれて、推定された統計はより正確になる傾向がありますが、同時ワークロードによって異なる場合があることに注意してください。
Exadata データベース・サーバーの追加	VM クラスタを拡張するオンライン操作。データベース・コンピュートを ExaDB-D に追加して、VM クラスタを拡張する 1 ステップ・プロセス。 Exadata Database Server 当たり、およそ 1 から 6 時間	データベース・リソースの使用率が 1 か月以内に 80%に達したときに、データベース・コンピュートを追加することを計画します。この操作には数時間から 1 日かかることに注意して計画してください。 データベース・コンピュートの追加操作をより短時間で完了できるように、非ピーク・ウィンドウを選択します Oracle Clusterware によって登録され、Oracle Cloud コンソールに表示される各 Oracle RAC データベースが拡張されます。Oracle Cloud コンソール外で、または dbaascli を使用せずにデータベースが構成された場合、そのデータベースは拡張されません。
仮想マシン(VM)クラスタでのデータベース・ノードの追加/削除	VM クラスタでデータベース・ノードを追加する際、VM クラスタ内のデータベースの数に応じて通常は 3-6 時間かかり、データベースの停止時間はゼロです VM クラスタでデータベース・ノードを削除する際、VM クラスタ内のデータベースの数に応じて通常は 1-2 時間かかり、データベースの停止時間はゼロです	追加/削除操作は即時ではなく、操作が完了するまでに数時間かかる場合があることを理解してください 削除操作によって、データベース・コンピュート、OCPU およびメモリー・リソースが削減されるため、アプリケーション・パフォーマンスに影響が生じる可能性があります

アプリケーションの継続的な可用性の実現

Oracle Exadata Database Service (ExaDB-DおよびExaDB-C@C)の一部として、すべてのソフトウェア更新(非ローリング・データベース・アップグレードまたは非ローリング・パッチを除く)をオンラインで、あるいはOracle RACローリング更新を使用

して行うことで、継続的なデータベース稼働時間を実現できます。さらに、ストレージ、ExadataネットワークまたはExadataデータベース・サーバーのローカルな障害が自動的に管理され、データベース稼働時間が維持されます。

Oracle RACのスイッチオーバーまたはフェイルオーバー・イベント中に継続的なアプリケーション稼働時間を実現するには、次に示すアプリケーション構成のベスト・プラクティスに従ってください。

- Oracle Clusterware管理データベース・サービスを使用してアプリケーションを接続します。Oracle Data Guard環境では、ロール・ベースのサービスを使用します。
- タイムアウト、再試行および遅延が組み込まれた推奨の接続文字列を使用して、停止中に着信接続でエラーが表示されないようにします。
- 高速アプリケーション通知を使用して接続を構成します。
- サービスを排出および再配置します。次の表を参照し、作業のバッチを流用または開始する際の接続のテストや、使用間におけるプールへの接続の返却など、排出をサポートする推奨ベスト・プラクティスを使用してください。
- 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を利用して、障害後に、実行中のコミットされていないトランザクションを透過的にリプレイします。

前述のチェックリストの詳細は、「アプリケーションの継続的な可用性の構成」を参照してください。アプリケーション・フェイルオーバーの準備状況の検証(ドキュメントID 2758734.1)に従って、アプリケーションの準備状況をテストすることをお薦めします。

Oracle Exadata Database Serviceの計画メンテナンス・イベントに応じて、Oracleでは、Oracle RACインスタンスを停止する前にデータベース・サービスを自動的に排出および再配置しようと試みます。OLTPアプリケーションについては、サービスの排出および再配置が通常は非常に適切に機能し、アプリケーションの停止時間がなくなります。

実行時間が長いバッチ・ジョブやレポートなど、一部のアプリケーションでは、排出および再配置を最大排出時間内に正常に行うことができない場合があります。そうしたアプリケーションについては、これらのタイプのアクティビティを避けてソフトウェアの計画メンテナンス・ウィンドウをスケジュールするか、計画メンテナンス・ウィンドウの前にこれらのアクティビティを停止することをお薦めします。たとえば、バッチ・ウィンドウ外で実行するように計画メンテナンス・ウィンドウを再スケジュールしたり、計画メンテナンス・ウィンドウの前にバッチ・ジョブを停止できます。

データベースOJVMを使用するアプリケーションのデータベースを四半期ごとにローリング更新する際には、特別な考慮事項が必要になります。詳細は、MOSノート2217053.1を参照してください。

次の表に、Oracle RACインスタンスのローリング再起動を実行する計画メンテナンス・イベントと、アプリケーションに影響を与える可能性がある、関連するサービスの排出タイムアウト変数を示します。

表26-5 Exadata Cloudのソフトウェア更新とエラスティック操作のためのアプリケーション・ドレイン属性

Oracle Exadata Database Serviceのソフトウェア更新またはエラスティック操作	**排出タイムアウト変数**
Oracle DBHOME パッチ適用およびデータベース移動	Oracle Cloud ソフトウェアの自動化により、データベース・サービス構成(srvctl など)によって定義された drain_timeout 設定を保持したまま、データベース・サービスが停止/再配置されます。[1] サービスで定義された drain_timeout は、オプション`-drainTimeoutInSeconds` をコマンドライン操作 `dbaascli dbHome patch` または `dbaascli database move`

Oracle Exadata Database Serviceのソフトウェア更新またはエラスティック操作	排出タイムアウト変数
	とともに使用してオーバーライドできます。 サポートされている Oracle Cloud 内部の最大排出時間は 2 時間です。
Oracle Grid Infrastructure (GI)パッチ適用およびアップグレード	Oracle Cloud ソフトウェアの自動化により、データベース・サービス構成(srvctl など)によって定義された drain_timeout 設定を保持したまま、データベース・サービスが停止/再配置されます。1 コマンドライン操作 `dbaascli grid patch` または `dbaascli grid upgrade` でオプション`-drainTimeoutInSeconds` を使用すると、サービスで定義された `drain_timeout` をオーバーライドできます。 サポートされている Oracle クラウド内部の最大排出時間は 2 時間です。
仮想マシン・オペレーティング・システムのソフトウェア更新(Exadata データベース・ゲスト)	Exadata の patchmgr/dbnodeupdate ソフトウェア・プログラムによって、排出オーケストレーション(rhphelper)がコールされます。 排出オーケストレーションには、次の排出タイムアウト設定があります(詳細は、Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)を参照してください)。 ● DRAIN_TIMEOUT - サービスに drain_timeout が定義されていない場合、この値が使用されます。デフォルト値は 180 seconds です。 ● MAX_DRAIN_TIMEOUT - データベース・サービス構成によって定義された、より大きい drain_timeout 値をオーバーライドします。デフォルト値は 300 seconds です。最大値はありません。 データベース・サービス構成によって定義された DRAIN_TIMEOUT 設定は、サービスの停止/再配置中に適用されます。
Exadata X8 以前のシステム ● VM のローカル/u02 ファイル・システム・サイズのスケール・アップとスケール・ダウン ● VM クラスタ・メモリーのスケール・アップまた	Exadata X8 以前のシステムのローカル・ファイル・システムのサイズ変更操作によって、排出オーケストレーション(rhphelper)がコールされます。 排出オーケストレーションには、次の排出タイムアウト設定があります(詳細は、Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)を参照してください)。 ● DRAIN_TIMEOUT - サービスに drain_timeout が定義されていない場合、この値が使用されます。デフォルト値は 180 seconds です。

Oracle Exadata Database Serviceのソフトウェア更新またはエラスティック操作	排出タイムアウト変数
はスケール・ダウン	● MAX_DRAIN_TIMEOUT - データベース・サービス構成によって定義された、より大きい drain_timeout 値をオーバーライドします。デフォルト値は 300 seconds です。 データベース・サービス構成によって定義された DRAIN_TIMEOUT 設定は、サービスの停止/再配置中に適用されます。 この操作でサポートされている Oracle Cloud 内部の最大排出時間は 300 秒です。
Exadata X8M 以降のシステム ● VM のローカル・ファイル・システム・サイズのスケール・ダウン	Exadata X8M 以降のシステムによって、排出オーケストレーション(rhphelper)がコールされます。 排出オーケストレーションには、次の排出タイムアウト設定があります(詳細は、Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)を参照してください)。 ● DRAIN_TIMEOUT - サービスに drain_timeout が定義されていない場合、この値が使用されます。デフォルト値は 180 seconds です。 ● MAX_DRAIN_TIMEOUT - データベース・サービス構成によって定義された、より大きい drain_timeout 値をオーバーライドします。デフォルト値は 300 seconds です。 データベース・サービス構成によって定義された DRAIN_TIMEOUT 設定は、サービスの停止/再配置中に適用されます。 この操作でサポートされている Oracle Cloud 内部の最大排出時間は 300 秒です。
Exadata X8M 以降のシステム ● VM クラスタ・メモリーのスケール・アップまたはスケール・ダウン	Exadata X8M 以降のシステムによって、排出オーケストレーション(rhphelper)がコールされます。 排出オーケストレーションには、次の排出タイムアウト設定があります(詳細は、Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)を参照してください)。 ● DRAIN_TIMEOUT - サービスに drain_timeout が定義されていない場合、この値が使用されます。デフォルト値は 180 seconds です。 ● MAX_DRAIN_TIMEOUT - 特定のサービスに定義されている、より大きい drain_timeout 値をオーバーライドします。デフォルトは 300 です。 データベース・サービス構成によって定義された DRAIN_TIMEOUT 設定は、サービスの停

Oracle Exadata Database Serviceのソフトウェア更新またはエラスティック操作	排出タイムアウト変数
	止/再配置中に適用されます。 この操作でサポートされている Oracle Cloud 内部の最大排出時間は 300 秒です。
Oracle Exadata Cloud Infrastructure (ExaDB-D)のソフトウェア更新	ExaDB-D データベース・ホストによって、ドレイン・オーケストレーション(rhphelper)がコールされます。 排出オーケストレーションには、次の排出タイムアウト設定があります(詳細は、Using RHPhelper to Minimize Downtime During Planned Maintenance on Exadata (Doc ID 2385790.1)を参照してください)。 ● DRAIN_TIMEOUT - サービスに drain_timeout が定義されていない場合、この値が使用されます。デフォルト値は 180 seconds です。 ● MAX_DRAIN_TIMEOUT - データベース・サービス構成によって定義された、より大きい drain_timeout 値をオーバーライドします。デフォルト値は 300 seconds です。 データベース・サービス構成によって定義された DRAIN_TIMEOUT 設定は、サービスの停止/再配置中に適用されます。 この操作でサポートされている Oracle Cloud 内部の最大排出時間は、次のとおりです ● Exadata X8 以前のシステムについては、タイムアウトは 300 秒です。 ● Exadata X8M 以降のシステムについては、タイムアウトは 500 秒です。 拡張インフラストラクチャ・メンテナンス制御機能: Oracle Cloud の内部最大値より長い排出時間を実現するには、拡張インフラストラクチャ・メンテナンス制御機能のカスタム・アクション機能を利用します。これにより、次のデータベース・サーバー更新の開始前にインフラストラクチャ・メンテナンスを一時停止し、データベース・サーバーで実行中のデータベース・サービスを直接停止/再配置してから、次のデータベース・サーバーに進むためにインフラストラクチャ・メンテナンスを再開できます。この機能は現在、Oracle Exadata Cloud@Customer (ExaDB-C@C)でも使用できます。詳細は、Oracle Cloud Infrastructure ドキュメントの Oracle 管理のインフラストラクチャ・メンテナンスの構成に関する項を参照してください。

[1] このサービス排出機能を入手するための最小ソフトウェア要件は、1) Oracle Database 12.2以降、2) 最新のOracle Cloud DBaaSツール・ソフトウェアです

Oracle Exadata CloudにおけるOracle Maximum Availability Architectureリファレンス・アーキテクチャ

Oracle Exadata Cloud (ExaDB-DおよびExaDB-C@C)は、固有の高可用性、データ保護および障害時リカバリのサービス・レベル合意に関係なく、すべてのOracle Maximum Availability Architectureリファレンス・アーキテクチャをサポートし、すべてのOracle Databaseについてサポートを提供します。Oracle Exadata CloudにおけるOracle Maximum Availability Architectureの詳細は、Oracle CloudのMAAベスト・プラクティスを参照してください。

27 Oracle Data Guardハイブリッド・クラウド構成

ハイブリッドOracle Data Guard構成は、プライマリ・データベースと、一部はオンプレミスにあり一部はクラウドにある1つ以上のスタンバイ・データベースで構成されます。ここで説明するプロセスでは、Oracleゼロ・ダウンタイム移行ツールを使用して、既存のオンプレミス・プライマリ・データベースからクラウド・スタンバイ・データベースを作成します。

ゼロ・ダウンタイム移行は、MAAのベスト・プラクティスを組み込んで、クラウドにスタンバイ・データベースを作成するプロセスを合理化および簡素化します

ここで説明するようにクラウド・スタンバイ・データベースを設定した後、プライマリ・データベースがオンプレミスではなくクラウドで動作するようにロール・トランジションを実行できます。

Oracle CloudでのハイブリッドData Guardの利点

Oracle CloudでハイブリッドData Guard構成を使用する主なメリットは次のとおりです。

- クラウド・データ・センターとインフラストラクチャはOracleが管理します。
- スケジュールされたメンテナンスまたは計画外の停止中に、本番をクラウド内のスタンバイ・データベースにスイッチオーバー(計画イベント)またはフェイルオーバー(計画外イベント)できます。障害が発生したオンプレミス・データベースが修復されると、クラウドの現在の本番データベースと同期できます。その後、本番環境をオンプレミス・データベースに切り替えることができます。
- オンプレミス・デプロイメントと同じOracle MAAベスト・プラクティスを使用します。ハイブリッドData Guardデプロイメントに固有のその他のOracle MAAベスト・プラクティスは、次のトピックで説明しています。MAAプラクティスで構成すると、ハイブリッドData Guard構成によって次のものが提供されます。
    - Data Guardファスト・スタート・フェイルオーバーで構成されている場合の自動フェイルオーバーを使用したリカバリ時間目標(RTO)の秒数
    - ASYNCト転送を使用するData Guardの1秒未満のリカバリ・ポイント目標(RPO)
    - SYNCまたはFAR SYNC構成でのData GuardのゼロのRPO

ノート:

スイッチオーバー、フェイルオーバー、復元などの Data Guard ライフ・サイクル管理操作は、ハイブリッド Data Guard 構成の手動プロセスです。

障害時リカバリにExadata Cloudを使用するためのMAAの推奨事項

障害時リカバリのためにExadata Cloudをデプロイする場合、Oracle MAAでは次のことをお薦めします。

- オンプレミスのプライマリ・データベースと対称または類似のクラウド・データベース・システム・ターゲットを作成して、ロール・トランジション後にパフォーマンスSLAを満たすことができるようにします。たとえば、Oracle RACソースにはOracle RACターゲット、ExadataにはExadataなどを作成します。
- 既存のネットワーク・トラフィックに加えて、ネットワーク帯域幅でピークREDO率を処理できることを確認します。

  My Oracle SupportドキュメントData GuardおよびRMANのネットワーク・パフォーマンスの評価およびチューニング(ドキュメントID 2064368.1)では、Data GuardおよびRMANのネットワーク・パフォーマンスを評価およびチューニン

グするための追加のネットワーク帯域幅トラブルシューティング・ガイダンスが提供されます。

- オンプレミス環境とクラウド環境間のネットワークの信頼性とセキュリティを確保します。
- 追加の自動ブロック修復、データ保護およびオフロードのために、Oracle Active Data Guardを使用します。
- プライマリ・データベースとスタンバイ・データベースの両方にOracle Transparent Data Encryption (TDE)を使用します。

  My Oracle SupportドキュメントのOracle CloudでのOracle Database表領域の暗号化動作(ドキュメントID 2359020.1)に、クラウド構成のTDE動作に関する追加の詳細が記載されています。
- 自動クラウド・バックアップは、クラウド・インスタンスをプライマリ・データベースにするオプションのData Guardロール・トランジション後に構成する必要があります。

サービス・レベルの要件

Oracle Data Guardハイブリッド・デプロイメントは、ユーザー管理環境です。特定の構成およびアプリケーションで実用的な可用性、データ保護およびパフォーマンスに対するサービス・レベルの期待値は、ユーザーの要件によって決定する必要があります。

Data Guard構成に適用可能な障害時リカバリに関連する次の各ディメンションについて、サービス・レベルを設定する必要があります。

- リカバリ時間目標(RTO)は、停止が発生した場合の最大許容停止時間を示します。これには、サービスを再開するために、停止を検出し、データベースおよびアプリケーションの接続をフェイルオーバーするために必要な時間が含まれています。
- リカバリ・ポイント目標(RPO)は、許容可能なデータ損失の最大量を示します。目的のRPOの達成は、次によって異なります。
  - ネットワーク・ボリュームに関連する使用可能な帯域幅
  - 信頼性が高く、中断のない転送を実現するネットワークの機能
  - 使用されるData Guard転送方法: ほぼゼロのデータ損失保護の場合は非同期、データ損失防止の場合は同期
- データ保護 - Oracle Active Data GuardおよびMAAを使用して、最も包括的なブロック破損検出、防止および自動修復を構成できます。
- パフォーマンス - コンピュート、メモリー、I/Oなどの容量がスタンバイ・システムにプロビジョニングされていない場合、データベースのレスポンス時間はフェイルオーバー後に、オンプレミスの本番システムと異なる場合があります。

  これは、管理者が意図的にスタンバイ・リソースを低く構成してコストを削減し、DRモードでサービス・レベルが低下した場合に発生します。MAAのベスト・プラクティスでは、フェイルオーバー後のレスポンス時間に変更がないように、プライマリ・データベース・ホストとスタンバイ・データベース・ホストの両方で対称的に容量を構成することをお薦めしています。

  クラウドで迅速にプロビジョニングできるため、安定した状態でデプロイされる容量が少なくなるという妥協点を容易に実現できますが、フェイルオーバーが必要な場合は、新しいプライマリ・データベース・システムを迅速にスケールアップできます。

ノート:

高速プロビジョニング・アプローチでの安定した状態のリソースの削減により、スタンバイ・データベースをプライマリ・データベースに遅れていないようにするためのリカバリ機能が影響を受け、適用ラグおよび影響のある RTO が作成される可能性があります。このアプローチは、徹底的なテスト後にのみ考慮する必要があります。

RTOおよびRPOの要件およびその他の考慮事項の決定の詳細は、「高可用性およびデータ保護 - 要件からのアーキテクチャの取得」を参照してください。

「データ破損の検出および監視」を参照してください。

セキュリティ要件および考慮事項

Oracle MAAのベスト・プラクティスでは、Oracle Transparent Data Encryption (TDE)を使用してプライマリ・データベースとスタンバイ・データベースを暗号化し、保存データを暗号化することをお薦めします。

TDEを使用したデータの保護は、システムのセキュリティの向上に不可欠な部分ですが、次のような暗号化ソリューションを使用する場合は、特定の考慮事項に注意する必要があります。

- 追加のCPUオーバーヘッド - 暗号化では、暗号化および復号化された値を計算するために追加のCPUサイクルが必要です。ただし、TDEは、データベース・キャッシング機能を利用し、Exadata内のハードウェア・アクセラレーションを活用して、オーバーヘッドを最小化するように最適化されています。ほとんどのTDEユーザーは、TDEを有効にした後、本番システムでのパフォーマンスへの影響がほとんどありません。
- より低いデータ圧縮 - 暗号化されたデータは元のプレーン・テキスト・データに関する情報を表示する必要がないため、圧縮率は低くなります。そのため、TDEで暗号化されたデータに適用される圧縮は圧縮率が低くなります。

  TDE暗号化を使用する場合、REDO転送の圧縮はお薦めしません。ただし、TDEをAdvanced CompressionやHybrid Columnar圧縮などのOracle Database圧縮テクノロジとともに使用する場合は、暗号化が実行される前に圧縮が実行され、圧縮と暗号化の利点の両方が得られます。
- キー管理 - 暗号化は使用される暗号化キーと同じくらい強力で、暗号化キーが失われると、そのキーで保護されているすべてのデータが失われます。

  いくつかのデータベースで暗号化が有効になっている場合、キーとそのライフ・サイクルを追跡することは比較的簡単です。暗号化されたデータベースの数が増えるにつれて、キーの管理はますます難しくなります。多数の暗号化データベースを管理している場合は、Oracle Key Vaultをオンプレミスで使用してTDEマスター・キーを格納および管理することをお薦めします。

データは移行プロセス中に変換できますが、最も安全なOracle Data Guard環境を提供するために、移行を開始する前にTDEを有効にすることをお薦めします。データ・ファイルやREDOヘッダーなど、TDEで暗号化されていない他のデータベース・ペイロードの処理中の暗号化には、VPN接続またはOracle Net暗号化も必要です。詳細は、My Oracle SupportドキュメントOracle CloudでのOracle Database表領域の暗号化動作(ドキュメントID 2359020.1)を参照してください。

オンプレミス・データベースがTDEでまだ有効になっていない場合は、My Oracle Supportドキュメントの透過的データ暗号化(TDE)のプライマリ・ノート(ドキュメントID 1228046.1)の手順に従い、TDEを有効にしてウォレット・ファイルを作成します。

TDEをオンプレミス・データベースで有効にできない場合は、クラウド・データベースがTDEで暗号化されていてオンプレミス・データベースが暗号化されていないハイブリッド・クラウド・ディザスタ・リカバリ構成でのREDO操作の復号化の詳細について、*Oracle Database Advanced Security*ガイドのOracle Data Guard環境での表領域の暗号化を参照してください。

プラットフォーム、データベースおよびネットワークの前提条件

クラウド・スタンバイ・データベースへの移行を確実に成功させるためには、次の要件を満たす必要があります。

要件タイプ	オンプレミス要件	Oracle Cloud要件
オペレーティング・システム	Linux、Windows または Solaris X86 (Data Guard クロス・プラットフォーム互換性のための My Oracle Support ノート 413484.1)	Oracle Enterprise Linux (64 ビット)
Oracle Database のバージョン*	ゼロ・ダウンタイム移行でサポートされるすべての Oracle リリース* オンプレミス・ソース・データベースの Oracle リリースおよびエディションのサポートの詳細は、移行でサポートされているデータベース・バージョンを参照してください。	Extreme performance / BYOL* Oracle Cloud のデータベース・サービス・オプションの詳細は、サポートされているデータベース・エディションおよびバージョンを参照してください。
Oracle Database アーキテクチャ	Oracle RAC または単一インスタンス	Oracle RAC または単一インスタンス
Oracle Multitenant	Oracle 12.1 以降では、プライマリ・データベースはマルチテナント・コンテナ・データベース(CDB)である必要があります	マルチテナント・コンテナ・データベース(CDB)または非 CDB
物理ホストまたは仮想ホスト	物理または仮想	Exadata 仮想
データベース・サイズ	任意のサイズ	任意のサイズ。 形状の制限については、Exadata Cloud のドキュメントを参照してください
TDE 暗号化	推奨	クラウド・データベースに必須

* プライマリ・データベースとスタンバイ・データベースのOracle Databaseリリースは、初期インスタンス化時に一致している必要があります。スタンバイ・ファーストで互換性のあるデータベース・ソフトウェア更新の場合、プライマリ・データベースとスタンバイ・データベースのOracleホーム・ソフトウェアは異なる場合があります。Oracle Patchの保証 - Data Guard Standby-First Patch適用(ドキュメントID 1265700.1)を参照してください。

クラウド・ネットワークの前提条件

オンプレミスからOracle Cloud Infrastructure (OCI)へのデータ転送では、パブリック・ネットワーク、VPNまたはOracle FastConnectが提供する高帯域幅オプション(あるいはそのすべて)が使用されます。

Oracle Data Guard構成では、プライマリ・データベースとスタンバイ・データベースが双方向で通信できる必要があります。これには、システム間のポートへのアクセスを許可するための追加のネットワーク構成が必要です。

ノート:

Oracle Exadata Database Service on Cloud@Customer はオンプレミス・ネットワークにデプロイされているため、ネットワーク接続構成は必要ありません。ExaDB-C@C を使用する場合は、「オンプレミスの前提条件」にスキップします。

安全な接続

Oracle Exadata Database Service (ExaDB-C@Cには不要)には、仮想クラウド・ネットワークをオンプレミス・ネットワークにプライベート接続するための2つのオプション(FastConnectとIPSec VPN)があります。どちらの方法でも、プライベート仮想クラウド・ネットワーク(VCN)に接続するには、動的ルーティング・ゲートウェイ(DRG)が必要です。

DRGの作成の詳細は、オンプレミス・ネットワークへのアクセスを参照してください。

- OCI FastConnect - データ・センターとOCI間に専用のプライベート接続を簡単に作成できます。FastConnectは、インターネットベースの接続と比較して、より高い帯域幅オプションおよびより信頼性の高い一貫性のあるネットワーキング・エクスペリエンスを提供します。詳細は、FastConnectの概要を参照してください(リンクhttps://docs.oracle.com/en-us/iaas/Content/Network/Concepts/fastconnectoverview.htm)。
- IPSec VPN - Internet Protocol SecurityまたはIP Security (IPSec)は、パケットがソースから宛先に転送される前にIPトラフィック全体を暗号化するプロトコル・スイートです。OCIのIPSecの概要は、サイト間VPNの概要を参照してください。

パブリック・インターネット接続

OCIとオンプレミス間の接続も、パブリック・インターネットを使用して実現できます。

この方法はデフォルトでは安全ではありません。転送を保護するには、追加の手順を実行する必要があります。ハイブリッドData Guard構成のステップでは、パブリック・インターネット接続を想定しています。

デフォルトでは、ポート1521のクラウド・セキュリティは無効になっています。また、仮想マシン(VM)またはベア・メタル(BM)用のクラウド内のこのデフォルトの事前構成済ポートは、パブリック・インターネットからオープン・アクセスが可能です。

1. スタンバイ・データベースの仮想クラウド・ネットワーク(VCN)にインターネット・ゲートウェイがない場合は、追加する必要があります。

   インターネット・ゲートウェイを作成するには、インターネット・ゲートウェイを参照してください。

2. オンプレミス・データベースとの間で接続するには、受信および送信ルールをVCNセキュリティ・リストで構成する必要があります。

   追加情報は、セキュリティ・リストを参照してください。

オンプレミスの前提条件

スタンバイ・データベースをインスタンス化する前に、次の前提条件を満たす必要があります。

oratcptestを使用したネットワークの評価

Oracle Data Guard構成では、プライマリ・データベースとスタンバイ・データベースが両方向で情報を送信します。これには、プライマリ・データベースとスタンバイ・データベースの両方で、基本的な構成、ネットワーク・チューニングおよびポートのオープンが

必要です。

プライマリ・データベースのREDO生成率をサポートするために、帯域幅が存在することが重要です。

Data GuardおよびRMANのネットワーク・パフォーマンスの評価およびチューニング(ドキュメントID 2064368.1)の手順に従って、オンプレミス環境とクラウド環境の間のネットワーク・リンクを評価およびチューニングします。

構成

- 名前解決
  - ExaDB-C@Cの場合、クラスタがオンプレミス・ネットワークに存在するため、オンプレミスDNSは各クラスタを解決する必要があり、それ以上の構成は必要ありません。
  - Oracle Exadata Database Serviceの場合、クラスタ間の名前解決を構成する必要があります。

    これを実行するには、/etc/hostsなどの静的ファイルを使用するか、OCIインスタンスのパブリックIPアドレスを適切に解決するようにオンプレミスDNSを構成します。また、オンプレミス・ファイアウォールでアクセス制御リストを構成して、SSHおよびOracle Netでオンプレミス・システムからOCIへのアクセスを許可する必要があります。
- DR構成のOracle Data Guardでは、クラウド・インスタンスからオンプレミス・データベースへのアクセスが必要です。プライマリ・データベース・リスナー・ポートは、iptablesなどの機能を使用してクラウドIPアドレスからのアクセスが制限されている状態でオープンする必要があります。

  各企業には異なるネットワーク・セキュリティ・ポリシーがあるため、ネットワーク管理者は、「クラウド・ネットワークの前提条件」に示すクラウド側のネットワーク構成などの操作を実行する必要があります。
- Oracle Cloudからオンプレミス・マシンへのプロンプトレスssh。これは、プロビジョニング・プロセス中にオンプレミスからクラウドに対して、およびクラウドからオンプレミスに対して構成されます。
- クラウドからオンプレミス・マシンへのインバウンドSSH接続を許可するオンプレミス・ファイアウォールの構成。
- 「oratcptestを使用したネットワークの評価」で前述したネットワーク評価を完了することをお薦めします。適切なTCPソケット・バッファ・サイズの設定は、ASYNC REDO転送に特に重要です。
- RDBMSソフトウェアは、インスタンス化のためにプライマリ・データベースとスタンバイ・データベースで同じである必要があります。現在のオンプレミスのOracle Databaseリリースが Oracle Exadata Database Serviceで使用できない場合、プライマリ・データベースにパッチを適用するか、使用可能なクラウド・バンドル・パッチにアップグレードする必要があります。

プライマリ・データベースでのMAAベスト・プラクティスのパラメータ設定の実装

Data GuardのほとんどのMAAベスト・プラクティスはここで説明するプロセスの一部です。ただし、このプロセスを開始する前に、プライマリ・データベースでスタンバイREDOログを作成する必要があります。

詳細は、「Oracle Data Guardの構成ベスト・プラクティス」を参照してください。

オンプレミス・ホストとExadataクラウド・ホスト間の接続の検証

ネットワーク・ステップが正常に実装されたら、次のコマンドを実行して、すべてのソースとすべてのターゲット間の接続が両方向で成功したことを確認します。

オンプレミス・ホストで次を実行します。

```
[root@onpremise1 ~]# telnet TARGET-HOST-IP-ADDRESS PORT
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
```

```
^C^]q
telnet> q
Connection closed.
```

クラウド・ホストで次を実行します。

```
[root@oci2 ~]# telnet TARGET-HOST-IP-ADDRESS PORT
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
^]q
telnet> q
Connection closed.
```

telnetが成功した場合は、次の手順に進みます。

ノート:

netcat (nc -zv)は、telnet のかわりに使用できます。

ゼロ・ダウンタイム移行を使用したスタンバイのインスタンス化

ゼロ・ダウンタイム移行環境を準備し、物理移行方法を使用してスタンバイ・データベースをインスタンス化します。

各タスクは、『*Zero Downtime Migration*を使用した*Oracle Cloud*への移行』の最新のゼロ・ダウンタイム移行ドキュメントの手順を参照し、これにはハイブリッドData Guard構成に関する追加情報が含まれています。

Oracle Data Guardハイブリッド・ユースケースでは、ゼロ・ダウンタイム移行をスタンバイ・データベースのインスタンス化と呼ぶこともできます。

スタンバイ・データベースがインスタンス化された後、完全移行ワークフローを完了する前に、移行ジョブが停止し、スタンバイがクラウドに配置されたままになります。ハイブリッドData Guard構成を完了するには、追加の修正が必要です。

タスク1: ゼロ・ダウンタイム移行のインストールおよび構成

ゼロ・ダウンタイム移行アーキテクチャには、プライマリ・データベース・ホストとスタンバイ・データベース・ホストとは別のゼロ・ダウンタイム移行サービス・ホストが含まれています。ゼロ・ダウンタイム移行ソフトウェアは、ゼロ・ダウンタイム移行サービス・ホストにインストールおよび構成されます。

DBCSコンピュート・リソースなどの任意のLinuxサーバーは、要件を満たし、ターゲットおよびソース・データベース・システムによって双方向にアクセス可能な場合に、サービス・ホストとして使用できます。

ホストの構成およびインストール手順については、ゼロ・ダウンタイム移行ソフトウェアの設定を参照してください。

タスク2: 物理データベースのインスタンス化の準備

ハイブリッドData Guard構成プロセスでは、ターゲット・データベースのインスタンス化後に移行ジョブを一時停止するオプションとともに、ゼロ・ダウンタイム移行物理データベースのオンライン移行ワークフローが使用されます。

スタンバイ・データベースがインスタンス化および検証されると、スタンバイ・データベースをそのままにして、移行ジョブを停止できます。

物理的な移行を準備するには、『Zero Downtime Migrationを使用したOracle Cloudへの移行』の物理データベース移行の準備の手順に従います。

次に、ハイブリッドData Guard構成に固有の追加情報を示します。

ソース・データベースでの透過的データ暗号化の構成

ハイブリッドData Guard構成の一部であるスタンバイ・データベースを含め、Oracle Cloudデータベースでは透過的データ暗号化(TDE)が必要です。

オンプレミス・データベースも暗号化することをお薦めしますが、ハイブリッドData Guard構成の一部としてプライマリ・データベースを暗号化しないまま構成することも可能で、Oracle Database 19c (19.16)以降のリリースでは新しいパラメータによってより適切にサポートされます。

Oracle Data Guardを使用するすべてのTDE構成では、暗号化ウォレットをプライマリ・データベースに作成し、マスター・キーを設定する必要があります。

TDE構成に必要なパラメータは、Oracle Databaseリリースによって異なります。値は、Data Guard構成のデータベースごとに異なる場合があります。

- Oracle Databaseリリース19c (19.16)以降では、TDEを正しく構成するために、パラメータ`TABLESPACE_ENCRYPTION`、`WALLET_ROOT`および`TDE_CONFIGURATION`が必要です。
- 19.16より前のOracle Database 19cリリースでは、パラメータ`WALLET_ROOT`、`TDE_CONFIGURATION`および`ENCRYPT_NEW_TABLESPACES`を設定します。
- Oracle Database19cより前のリリースでは、パラメータ`ENCRYPTION_WALLET_LOCATION`および`ENCRYPT_NEW_TABLESPACES`を設定します。

ノート:

`TABLESPACE_ENCRYPTION=DECRYPT_ONLY` パラメータで特に指定されていないかぎり、スタンバイ・データベースでの新しい表領域の暗号化はプライマリのものと同じになります。

次の表では、リンクを使用して、プライマリおよびスタンバイ・データベース・パラメータを設定するためのリファレンスを検索します。

パラメータ	定義	19cより前のすべてのOracle Databaseリリース	Oracle Databaseリリース19.15以前	Oracle Databaseリリース19.16以降
`ENCRYPTION_WALLET_LOCATION`	ウォレットの場所を定義します sqlnet.ora ファイル内のキーストアの場所についてを参照	RECOMMENDED	DEPRECATED	DEPRECATED
`WALLET_ROOT` and `TDE_CONFIGURATION`	`WALLET_ROOT` は、CDB内の各 PDB のウォレット・ストレージのディレクトリのルートの場所を設定します。 WALLET_ROOT を参照 `TDE_CONFIGURATION`	N/A	RECOMMENDED	RECOMMENDED

パラメータ	定義	19cより前のすべてのOracle Database リリース	Oracle Database リリース19.15以前	Oracle Database リリース19.16以降
	は、キーストアのタイプを定義します。たとえば、ウォレット・キーストアの場合は FILE です。キーストア・タイプは、プライマリ・データベースとスタンバイ・データベースで同じ値に設定する必要があります。TDE_CONFIGURATION を参照			
ENCRYPT_NEW_TABLESPACES	プライマリ・データベースの新規表領域を暗号化するかどうかを示します ENCRYPT_NEW_TABLESPACES パラメータは、次のように設定できます。 ● CLOUD_ONLY - デフォルト設定。作成された新しい表領域は、CREATE TABLESPACE 文の ENCRYPTION 句に別のアルゴリズムが指定されていないかぎり、AES128 アルゴリズムで透過的に暗号化されます。オンプレミス・データベースの場合、表領域は CREATE TABLESPACE... ENCRYPTION 句が指定されている場合にのみ暗号化されます。 ● ALWAYS - プライマリ・データベース(オン	RECOMMENDED	RECOMMENDED	NOT RECOMMENDED TABLESPACE_ENCRYPTION の推奨設定でオーバーライドします

パラメータ	定義	19cより前のすべてのOracle Databaseリリース	Oracle Databaseリリース19.15以前	Oracle Databaseリリース19.16以降
	プレミスまたはクラウド)で作成された新しい表領域は、CREATE TABLESPACE ENCRYPTION 句で別の暗号化アルゴリズムが指定されていないかぎり、AES128 アルゴリズムで透過的に暗号化されます。 ● DDL - CREATE TABLESPACE 文に続く暗号化の有無に関係なく表領域を作成でき、暗号化アルゴリズムを変更することもできます。ノート: この値は、Oracle Database 19c (19.16)以降のリリースがあるクラウドのプライマリ・データベースでは表領域の暗号化が強制されるため、適用されません。 ENCRYPT_NEW_TABLESPACES を参照			
TABLESPACE_ENCRYPTION (前述のノートを参照)	Oracle Database 19c (19.16)以降のリリース - 新しい表領域を暗号化する必要があるかどうかを示します。使用可能なオプションは、AUTO_ENABLE、MANUAL_ENABLE および	N/A	N/A	RECOMMENDED

パラメータ	定義	19cより前のすべてのOracle Databaseリリース	Oracle Databaseリリース19.15以前	Oracle Databaseリリース19.16以降
	DECRYPT_ONLY です。 Oracle Database 19c (19.16)以降、Oracle Cloud はクラウド・データベース内のすべての表領域に対して暗号化を強制します。これはオーバーライドできません。 オンプレミス・データベース(プライマリまたはスタンバイ)で暗号化表領域を防止するには、TABLESPACE_ENCRYPTION パラメータを DECRYPT_ONLY に設定します。 DECRYPT_ONLY は、オンプレミス・データベースでのみ有効です。 TABLESPACE_ENCRYPTION を参照			

TDEを構成するには、Zero Downtime Migrationを使用したOracle Cloudへの移行の透過的データ暗号化ウォレットの設定のステップに従います。

インスタンス化前のTDEマスター・キーの確認

プライマリ・データベースが暗号化されないままの場合でも、TDEをプライマリ・データベースで構成する必要があります。この構成には、暗号化ウォレットの作成およびマスター・キーの設定が含まれています。

プロセス中に、ウォレットがスタンバイ・データベースにコピーされます。ウォレットに格納されたマスター・キーは、スタンバイ・データベースによって暗号化に使用されます。

クラウド・スタンバイ・データベースがプライマリ・データベースになるスイッチオーバーの場合、暗号化されたREDOをクラウド・データベースから復号化するために、暗号化されていないオンプレミス・データベースによってキーが使用されます。

マスター・キーの設定に失敗すると、Data Guard管理のリカバリが失敗します。

マスター・キーが正しく設定されていることを確認するには:

- V$DATABASE_KEY_INFOのMASTERKEYID列が、ソース・データベースのV$ENCRYPTION_KEYSに存在する

キーと一致することを確認します。

マルチテナント・コンテナ・データベース(CDB)環境で、CDB$ROOT、およびPDB$SEEDを除くすべてのPDBを確認します。

オンラインREDOログの構成

REDOログ・スイッチは、REDO転送および適用のパフォーマンスに大きな影響を与える可能性があります。インスタンス化の前に、プライマリ・データベースのオンラインREDOログのサイズ設定に関する次のベスト・プラクティスに従います。

- すべてのオンラインREDOログ・グループは、同じサイズのログ(バイト)を持つ必要があります。
- オンラインREDOログは、高パフォーマンスのディスク(DATAディスク・グループ)に存在する必要があります。
- Oracle RACインスタンスでREDOのスレッドごとに3つ以上のオンラインREDOログ・グループを作成します。
- Oracle RAC環境の共有ディスクにオンラインREDOログ・グループを作成します。
- 高冗長性ディスク・グループに配置されていないかぎり、オンラインREDOログを多重化します(ログ・グループごとに複数のメンバー)。
- ログ・スイッチが1時間に12回(5分ごと)以下になるようにオンラインREDOログのサイズを設定します。ほとんどの場合、ピーク時のワークロードであっても、15分から20分ごとのログ・スイッチが最適です。

REDOログ・サイズの設定

プライマリ・データベースのピークREDO生成率に基づいてREDOログ・サイズを設定します。

ピーク生成率を確認するには、ピーク時のワークロードを含む期間に次の問合せを実行します。最大速度は、月末、四半期末または年次で確認できます。REDO適用をこれらのワークロード中に一貫して実行するためには、最大速度を処理するようにREDOログ・サイズを設定します。

```
SQL> SELECT thread#,sequence#,blocks*block_size/1024/1024 MB,
(next_time-first_time)*86400 sec,
 blocks*block_size/1024/1024)/((next_time-first_time)*86400) "MB/s"
 FROM v$archived_log
 WHERE ((next_time-first_time)*86400<>0)
 and first_time between to_date('2015/01/15 08:00:00','YYYY/MM/DD HH24:MI:SS')
 and to_date('2015/01/15 11:00:00','YYYY/MM/DD HH24:MI:SS')
 and dest_id=1 order by first_time;
   THREAD#  SEQUENCE#         MB        SEC       MB/s
---------- ---------- ---------- ---------- ----------
         2       2291 29366.1963        831  35.338383
         1       2565 29365.6553        781 37.6000708
         2       2292 29359.3403        537  54.672887
         1       2566 29407.8296        813 36.1719921
         2       2293 29389.7012        678 43.3476418
         2       2294 29325.2217       1236 23.7259075
         1       2567 11407.3379       2658 4.29169973
         2       2295 29452.4648        477 61.7452093
         2       2296 29359.4458        954 30.7751004
         2       2297 29311.3638        586 50.0193921
         1       2568 3867.44092       5510 .701894903
```

次のチャートを使用して、ピーク生成率に基づいてREDOログ・サイズを選択します。

ピークREDO率	推奨REDOログ・サイズ
<= 1 MB/s	1 GB
<= 5 MB/s	4 GB
<= 25 MB/s	16 GB

ピークREDO率	推奨REDOログ・サイズ
<= 50 MB/s	32 GB
> 50 MB/s	64 GB

ターゲット・データベースの作成

スタンバイ・データベースになるターゲット・データベースは、最初にOracle Cloudの自動化によって作成されます。このアプローチにより、Oracle Cloudユーザー・インタフェースでデータベースが表示され、パッチ適用などのクラウド自動化のサブセットで使用できるようになります。

ノート:

スイッチオーバー、フェイルオーバー、復元などの Oracle Data Guard 操作は、Data Guard Broker で実行される手動操作です。Data Guard ライフ・サイクル管理は、ハイブリッド Data Guard 構成のユーザー・インタフェースではサポートされていません。

データベースが作成されると、ゼロ・ダウンタイム移行ワークフローによって既存のファイルが削除され、スタンバイ・データベースがかわりにインスタンス化されます。

ターゲット・データベースのハイブリッドData Guard構成(ゼロ・ダウンタイム移行と比較)の例外を次に示します。

- ターゲット・データベースでは、ソース・データベースと同じdb_nameを使用する必要があります。
- ターゲット・データベースでは、異なるdb_unique_nameを使用する必要があります。

インスタンス化方法の選択

ゼロ・ダウンタイム移行を使用するハイブリッドData Guardスタンバイ・インスタンス化に推奨される2つのオプションは、直接データ転送とオブジェクト・ストレージ・サービスです。

- 直接データ転送 - DATA_TRANSFER_MEDIUM=DIRECT - RMANを使用して、データ・ファイルをプライマリ・データベースから直接コピーします。
- オブジェクト・ストレージ・サービス - DATA_TRANSFER_MEDIUM=OSS - OSSバケットへのプライマリ・データベースのバックアップを実行し、バックアップからスタンバイ・データベースをインスタンス化します。

既存のバックアップまたは既存のスタンバイからインスタンス化するための追加オプションがありますが、この手順では対応していません。詳細は、『Zero Downtime Migrationを使用したOracle Cloudへの移行』のデータ・ソースとしての既存のRMANバックアップの使用および既存のスタンバイを使用したターゲット・データベースのインスタンス化を参照してください。

ゼロ・ダウンタイム移行パラメータの設定

次に示すゼロ・ダウンタイム移行の物理移行レスポンス・ファイル・パラメータは、ほとんどの場合に設定される主なパラメータです。

- TGT_DB_UNIQUE_NAME - クラスタウェアに登録されているターゲット・クラウド・データベースのdb_unique_name (srvctl)
- MIGRATION_METHOD=ONLINE_PHYSICAL - ハイブリッドData Guardの設定はすべてONLINE_PHYSICALメソッドを使用します
- DATA_TRANSFER_MEDIUM=DIRECT | OSS - DIRECTは、Oracle 12.1より前のバージョンのソース・データベースではサポートされていません
- PLATFORM_TYPE=EXACS | EXACC | VMDB - 適切な構成を確保するために、適切なターゲットのOracle

Cloudプラットフォームを選択します

- `HOST=cloud-storage-REST-endpoint-URL` - OSSデータ転送メディアを使用する場合は必須です
- `OPC_CONTAINER=object-storage-bucket` - OSSデータ転送メディアを使用する場合は必須です
- `ZDM_RMAN_COMPRESSION_ALGORITHM=BASIC`
- `ZDM_USE_DG_BROKER=TRUE` - Data Guard BrokerはMAA構成のベスト・プラクティスです

要塞ホストまたはその他の複雑さが関係する場合は、『Zero Downtime Migrationを使用したOracle Cloudへの移行』の物理移行パラメータの設定を参照してください。

タスク3: スタンバイ・データベースのインスタンス化

準備が完了したら、ゼロ・ダウンタイム移行オンライン物理移行ジョブを実行して、クラウド・スタンバイ・データベースをインスタンス化できます。

実際には、ZDMCLIのゼロ・ダウンタイム移行コマンドを使用して、評価ジョブと実際の移行ジョブの2つのジョブを実行します。

1. 評価ジョブを実行します。

   評価ジョブでは、トポロジ構成および移行ジョブ設定が分析され、本番データベースに対して実行したときにプロセスが成功することを確認します。

   次に示すように、`ZDMCLI migrate database`コマンドの`-eval`オプションを使用して、評価ジョブを実行します。

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb source_db_unique_name_value
 -sourcenode source_database_server_name
 -srcauth zdmauth
 -srcarg1 user:source_database_server_login_user_name
 -srcarg2 identity_file:ZDM_installed_user_private_key_file_location
 -srcarg3 sudo_location:/usr/bin/sudo
 -targetnode target_database_server_name
 -backupuser Object_store_login_user_name
 -rsp response_file_location
 -tgtauth zdmauth
 -tgtarg1 user:target_database_server_login_user_name
 -tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
 -tgtarg3 sudo_location:/usr/bin/sudo
 -eval
```

   『Zero Downtime Migrationを使用したOracle Cloudへの移行』の移行ジョブの評価には、評価ジョブ・オプションの例が他にもあります。

   ノート:

   ハイブリッド Data Guard クラウド・スタンバイ・インスタンス化プロセスは物理的な移行であるため、クラウド移行前アドバイザ・ツール(CPAT)はサポートされていません。

2. 移行ジョブを実行します。

   デフォルトでは、ゼロ・ダウンタイム移行で、ターゲット・データベースのインスタンス化直後にスイッチオーバー操作が実行されるため、`ZDMCLI migrate database`コマンドで`-stopafter`オプションを使用して、スタンバイ・データベースの作成後に移行ジョブを停止します。

   `-stopafter`オプションを使用して、次に示すように`ZDM_CONFIGURE_DG_SRC`に設定します。

```
zdmuser> $ZDM_HOME/bin/zdmcli migrate database
 -sourcedb source_db_unique_name_value
 -sourcenode source_database_server_name
 -srcauth zdmauth
 -srcarg1 user:source_database_server_login_user_name
 -srcarg2 identity_file:ZDM_installed_user_private_key_file_location
 -srcarg3 sudo_location:/usr/bin/sudo
 -targetnode target_database_server_name
 -backupuser Object_store_login_user_name
 -rsp response_file_location
 -tgtauth zdmauth
 -tgtarg1 user:target_database_server_login_user_name
 -tgtarg2 identity_file:ZDM_installed_user_private_key_file_location
 -tgtarg3 sudo_location:/usr/bin/sudo
 -stopafter ZDM_CONFIGURE_DG_SRC
```

データベース移行ジョブが発行されると、コマンド出力にジョブIDが表示されます。後で診断が必要な場合は、この情報を保存します。

『Zero Downtime Migrationを使用したOracle Cloudへの移行』のデータベースの移行には、`ZDMCLI migrate database`コマンドの使用例が他にもあります。

タスク4: スタンバイ・データベースの検証

スタンバイ・データベースのインスタンス化後にゼロ・ダウンタイム移行ジョブが停止した場合、スタンバイ・データベースを検証します。

Oracle Data Guard Broker構成の確認

ゼロ・ダウンタイム移行レスポンス・ファイルでパラメータ`ZDM_USE_DG_BROKER=TRUE`を使用すると、Data Guard Broker構成が作成されます。Oracle Cloudユーザー・インタフェースではオンプレミス・データベースが認識されないため、Data Guard Brokerが、ハイブリッドData Guard構成のライフ・サイクル操作を管理する主要なユーティリティになります。

DGMGRLを使用して、Data Guard Broker構成を検証します。リストされているData Guard Brokerコマンドは、プライマリ・データベースまたはスタンバイ・データベースから実行できます。

```
DGMGRL> show configuration
Configuration - ZDM_primary db_unique_name
  Protection Mode: MaxPerformance
  Members:
  primary db_unique_name - Primary database
    standby db_unique_name - Physical standby database
Fast-Start Failover:  Disabled
Configuration Status:
SUCCESS   (status updated 58 seconds ago)
```

構成ステータスはSUCCESSである必要があります。その他のステータスが表示された場合は、2分間待った後でコマンドを再実行して、ブローカが更新する時間を指定します。問題が解決しない場合は、Oracle Data Guard Brokerのドキュメントを参照して、問題を診断および修正してください。

スタンバイ・データベースの検証

DGMGRLを使用して、スタンバイ・データベースを検証します。

```
DGMGRL> validate database standby db_unique_name
  Database Role:     Physical standby database
  Primary Database:  primary db_unique_name
  Ready for Switchover:  Yes
  Ready for Failover:    Yes (Primary Running)
  Flashback Database Status:
```

```
    primary db_unique_name:  On
    standby db_unique_name:  Off  <- see note below
  Managed by Clusterware:
    primary db_unique_name:  YES
    standby db_unique_name:  YES
```

ノート:

スタンバイでフラッシュバック・データベースを有効にするステップは、次のステップで対処します。

タスク5: お薦めするMAAのベスト・プラクティスの実装

スタンバイ・インスタンス化の後、より優れたデータ保護と可用性を実現するために、次のOracle MAAのベスト・プラクティスの実装を評価します。

主なベスト・プラクティスを次に示します。Oracle Data GuardのOracle MAAでお薦めするベスト・プラクティスの詳細は、Oracle Data Guard構成のベスト・プラクティスも参照してください。

フラッシュバック・データベースの有効化

フラッシュバック・データベースでは、フェイルオーバー後に古いプライマリ・データベースをスタンバイ・データベースとして復元できます。フラッシュバック・データベースを有効にしない場合、フェイルオーバー後に古いプライマリ・データベースをスタンバイとして再作成する必要があります。フラッシュバック・データベースがまだ有効になっていない場合は、ここで有効にします。

フラッシュバック・データベースを有効にするには、高速リカバリ領域またはRECOディスク・グループに十分な領域およびI/Oスループットがあることを確認し、パフォーマンスへの影響を評価します。

1. プライマリ・データベースで、次のコマンドを実行して、フラッシュバック・データベースがまだ有効になっていない場合はプライマリでこれを有効にします。

```
SQL> alter database flashback on;
Database altered.
```

2. スタンバイ・データベースで、フラッシュバック・データベースを有効にするには、まずREDO適用を無効にし、フラッシュバック・データベースを有効にしてから、REDO適用を再度有効にします。

```
DGMGRL> edit database standby-database set state=apply-off;
Succeeded.
SQL> alter database flashback on;
Database altered.
DGMGRL> edit database standby-database set state=apply-on;
Succeeded.
```

CONTROL_FILESパラメータの設定およびスタンバイのデフォルト・オープン・モードの変更

Oracle MAAベスト・プラクティスでは、高い冗長性のディスク・グループに配置されるときに制御ファイルを1つのみ持つことをお薦めします。すべてのOracle Cloud製品で高い冗長性が使用されるため、必要な制御ファイルは1つのみです。

1. スタンバイ・データベースで、CONTROL_FILESパラメータを編集します。

```
SQL> show parameter control_files
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
control_files                        string      controlfile-1
                                                 , controlfile-2
SQL> ALTER SYSTEM SET control_files='controlfile-1' scope=spfile sid='*';
System altered.
```

2. oracleユーザーとしてデータベースを停止し、gridユーザーとして余分な制御ファイルを削除します(opcユーザーからgridユーザーへのsu)。

```
$ srvctl stop database -db standby-unique-name
[grid@standby-host1 ~]$ asmcmd rm controlfile-2
```

3. データベースが停止している間に、スタンバイ・データベースのデフォルトが読取り専用でオープンされるように起動オプションを変更し、データベースを起動します。

```
$ srvctl modify database -db standby-unique-name -startoption 'read only'
$ srvctl start database -db standby-unique-name
```

ノート:

Oracle MAA ベスト・プラクティスは、自動ブロック・メディア・リカバリを有効化するためにスタンバイを読取り専用でオープンすることですが、Oracle Cloud はマウントされたスタンバイをサポートします。マウントされたスタンバイがユーザーの優先構成の場合は、構成できます。

代替ローカル・アーカイブ・ログの場所の設定

リカバリ領域で領域が消費されると、プライマリ・データベースはアーカイブを停止し、オンラインREDOログをアーカイブするための領域が使用可能になるまですべての操作が停止します。

このシナリオを回避するには、DATAディスク・グループに代替のローカル・アーカイブの場所を作成します。

1. LOG_ARCHIVE_DEST_10を設定してDATAディスク・グループを使用し、状態をALTERNATEに設定します。

```
SQL> ALTER SYSTEM SET log_archive_dest_10='LOCATION=+DATAC1
 VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1
 REOPEN=5 DB_UNIQUE_NAME=standby-unique-name
 ALTERNATE=LOG_ARCHIVE_DEST_1' scope=both sid='*';
SQL> ALTER SYSTEM SET log_archive_dest_state_10=ALTERNATE scope=both sid='*';
```

2. LOG_ARCHIVE_DEST_10を代替として使用するようにLOG_ARCHIVE_DEST_1を設定します。

```
SQL> ALTER SYSTEM SET log_archive_dest_1='LOCATION=USE_DB_RECOVERY_FILE_DEST
 VALID_FOR=(ALL_LOGFILES,ALL_ROLES) MAX_FAILURE=1
 REOPEN=5 DB_UNIQUE_NAME=standby-unique-name
 ALTERNATE=LOG_ARCHIVE_DEST_10' scope=both sid='*';
```

ノート:

バックアップが構成されていない場合、デフォルトでは 24 時間より古いアーカイブ・ログは 30 分ごとにスイープされます。

データ保護パラメータの設定

MAAのベスト・プラクティスの推奨事項には、プライマリ・データベースとスタンバイ・データベースに関する次の設定が含まれています。

db_block_checksum=TYPICAL

db_lost_write_protect=TYPICAL

db_block_checking=MEDIUM

```
SQL> show parameter db_block_checksum
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
```

```
db_block_checksum                    string      TYPICAL
SQL> alter system set db_block_checksum=TYPICAL scope=both sid='*';
SQL> show parameter db_lost_write_protect
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_lost_write_protect                string      typical
SQL> alter system set db_lost_write_protect=TYPICAL scope=both sid='*';
SQL> show parameter db_block_checking
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_block_checking                    string      OFF
SQL> alter system set db_block_checking=MEDIUM scope=both sid='*';
```

db_block_checking設定は、プライマリ・データベースのパフォーマンスに影響するため、本番と同様のより低い環境で本番ワークロードを使用して十分にテストする必要があります。

パフォーマンスへの影響がプライマリ・データベースで許容できないと判断された場合、両方のデータベースに対してスタンバイ・データベースではdb_block_checking=MEDIUMを設定し、cloudautomation Data Guard Brokerプロパティを'1'に設定して、ロール・トランジション後に値が適切に変更されるようにする必要があります。

```
DGMGRL> edit database primary-unique-name set property cloudautomation=1;
Property "cloudautomation" updated
DGMGRL> edit database standby-unique-name set property cloudautomation=1;
Property "cloudautomation" updated
```

両方のデータベースでcloudautomationプロパティが正しく動作するように設定する必要があることに注意してください。

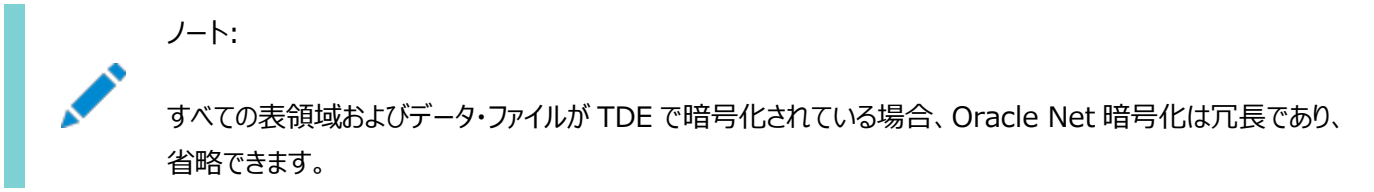
REDO転送の構成 - Oracle Net暗号化

プレーン・テキストまたは暗号化されていない表領域のREDOがWANに表示されないようにするには、すべてのオンプレミス・データベースおよびクラウド・データベースのsqlnet.oraファイルに次のエントリを配置します。

クラウド・デプロイメントでは、TNS_ADMIN変数を使用して、共有データベース・ホームでtnsnames.oraとsqlnet.oraを分けます。したがって、特定のデータベースのクラウドsqlnet.oraおよび拡張tnsnames.oraは、$ORACLE_HOME/network/admin/db_nameにあります。

これらの値は、クラウド構成のデプロイメント・ツールによってすでに設定されている必要があります。

```
SQLNET.ORA ON ON-PREMISES HOST(S)
SQLNET.ENCRYPTION_SERVER=REQUIRED
SQLNET.CRYPTO_CHECKSUM_SERVER=REQUIRED
SQLNET.ENCRYPTION_TYPES_SERVER=(AES256,AES192,AES128)
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER=(SHA1)
SQLNET.ENCRYPTION_CLIENT=REQUIRED
SQLNET.CRYPTO_CHECKSUM_CLIENT=REQUIRED
SQLNET.ENCRYPTION_TYPES_CLIENT=(AES256,AES192,AES128)
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT=(SHA1)
```



ノート:

すべての表領域およびデータ・ファイルが TDE で暗号化されている場合、Oracle Net 暗号化は冗長であり、省略できます。

REDO転送の構成 - 完全接続記述子を使用したREDO転送の再構成

わかりやすくするために、ゼロ・ダウンタイム移行ではEZconnect識別子を使用してOracle Data Guard REDO転送を設定します。

完全なゼロ・ダウンタイム移行ワークフローを使用する構成など、短期間の構成の場合、このソリューションは許容可能です。た

だし、ハイブリッドData Guard構成の場合、MAAのベスト・プラクティスの推奨事項は、tnsnames.oraで構成された完全接続記述子を使用することです。

次の例を使用して、属性値を構成に関連する値に置き換えます。

データベースのTNS記述子は、SCANリスナーが他のシステムから解決可能かどうかによって異なります。

次の説明は、SCAN名が解決可能であり、TNS記述子で使用できることを前提としています。SCAN名を解決できない場合は、ADDRESS_LISTを使用できます。詳細は、tnsnames.oraの複数のアドレス・リストを参照してください。

適切な置換を行った後に、プライマリおよびスタンバイ・データベース・システムの共有tnsnames.oraファイルに次の記述子を追加します。

```
standby-db_unique_name =
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= standby-cluster-scan-name )
        (PORT=standby-database-listener-port))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= standby-database-service-name)))
primary-db_unique_name=
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL=TCP)
        (HOST=primary-cluster-scan-name)
        (PORT=primary-database-listener-port))
      (CONNECT_DATA=
        (SERVER=DEDICATED)
        (SERVICE_NAME=primary-database-service-name)
    ))
```

ノート:

プライマリ db_unique_name の名前を持つ記述子は、クラウド自動化またはゼロ・ダウンタイム移行によって作成されている可能性があります。このエントリは間違ったデータベースを指しているため、置き換えます。

REDO転送の構成 - REDO転送のData Guard Broker設定の変更

ゼロ・ダウンタイム移行ワークフロー中に設定されたEZconnect識別子を変更して、各データベースのtnsnames.oraファイルに追加された接続記述子を使用します。

```
DGMGRL> show database primary-db_unique_name DGConnectIdentifier
DGConnectIdentifier = 'ZDM-created-EZconnect-string>'
DGMGRL> edit database primary-db_unique_name
         set property DGConnectIdentifier='primary-db_unique_name';
DGMGRL> show database standby-db_unique_name DGConnectIdentifier
DGConnectIdentifier = 'ZDM-created-EZconnect-string'
DGMGRL> edit database standby-db_unique_name
         set property DGConnectIdentifier='standby-db_unique_name';
```

スタンバイ自動ワークロード・リポジトリの構成

スタンバイ自動ワークロード・リポジトリ(AWR)を使用すると、スタンバイに対してAWRレポートを生成できます。これらのレポートは、スタンバイ・データベースでのREDO適用およびその他のパフォーマンスの問題を診断する際に非常に重要です。

すべてのOracle Data Guard構成に対してスタンバイAWRを構成することをお薦めします。

詳細は、My Oracle SupportノートActive Data Guardスタンバイ・データベースでAWRを生成する方法(ドキュメントID

2409808.1)を参照してください。

ヘルス・チェックおよび監視

スタンバイ・データベースをインスタンス化した後、ヘルス・チェックを実行して、Oracle Data Guardデータベース(プライマリおよびスタンバイ)がOracle MAAのベスト・プラクティスに準拠していることを確認する必要があります。

また、ヘルス・チェックは毎月、データベース・メンテナンスの前後に実行することをお薦めします。Data Guard構成のヘルスをチェックするには、Oracle Autonomous Health FrameworkおよびOraChkまたはExaChkを使用したOracle MAAスコアカードを含む自動ツールをお薦めします。

Oracle Data Guard構成の定期的な監視は、ハイブリッドData Guard構成では提供されないため、手動で実行する必要があります。詳細は、「Oracle Data Guard構成の監視」を参照してください。

第VII部 アプリケーションの継続的な可用性

28 アプリケーションの継続的な可用性の構成

計画メンテナンスおよび計画外停止の際に、ワークロードを使用可能なOracle RACインスタンスまたはスタンバイ・データベースに迅速かつ自動的に移動するようにアプリケーションが構成されていることを確認します。

停止時に次の推奨事項に従うことで、アプリケーションの稼働時間が最大化されます。

このドキュメントの主な対象読者は、アプリケーション開発者とアプリケーション所有者です。データベース管理者とPDB管理者向けの操作例が含まれています。

次の内容について説明します。

アプリケーション高可用性のレベルについて

アプリケーション高可用性の要件に応じて、必要な高可用性(HA)保護のレベルを実装できます。

HA保護レベルは次の表で定義され、前のレベルに基づいて段階的にレベルが高くなります。

HAレベル	構成	エクスペリエンス	メリット
レベル 1: 基本的なアプリケーション高可用性 「レベル 1: 基本的なアプリケーション高可用性の構成」を参照	データベースまたはセキュリティの管理者: ● ロールベースのデータベース・サービスを構成します ● 推奨のデータベース接続文字列を利用して、LDAP およびウォレットはオプションで構成します ● 停止時の即時割込みのために高速アプリケーション通知(FAN)を有効にします アプリケーション開発者: ● MAA 推奨の接続文字列を使用します ● 基本的な例外処理を使用します 実装作業: 最小 - 管理者の場合は 1 時間、開発者の場合は 1 時間未満	レベル 1 保護を実装すると、アプリケーションの影響を軽減するアプリケーション + Oracle 統合およびインテリジェンスによって、サード・パーティ・アプリケーションのフェイルオーバー・ソリューションと比較して大きなメリットが得られます。 ● アプリケーションの停止時間が短縮されます ● アプリケーションは、計画メンテナンス時および計画外停止時にエラーを認識し、ターゲット・サービスのある別の Oracle RAC インスタンスまたはデータベースに自動的に再接続します。 ● 計画外停止および計画メンテナンスに適用できます。場合によっては、長期実行トランザクションは、計画メンテナンス中に延期または一時停止する必要があります。	アプリケーションが自動的にフェイルオーバーして再接続する高可用性 ● 迅速なタイムアウトとデータベース接続文字列を使用した自動的な接続の再試行 ● サービスによる場所の透過性: スタンバイ・データベースと読取り専用データベースに応じたロールベースのサービスにより、アプリケーションが自動的に正しいロールの適切なインスタンスにルーティングされるようにします。 ● データベース接続文字列で自動構成される FAN ● FAN を使用すると停止時に即座に割込みが発生します(タイムアウトを調整して待機する必要はありません)。 ● Clusterware は RAC と VIP のヘルスを認識するため、FAN の働きによりダウンしたエンド・ポイントで待機することはありません

HAレベル	構成	エクスペリエンス	メリット
レベル 2: 計画メンテナンスに向けたアプリケーションの準備 「レベル 2: 計画メンテナンスに向けたアプリケーションの準備の構成」を参照	レベル 1 の構成 ＋ アプリケーション開発者: ● Oracle 接続プールまたは接続テストを使用し、各使用の間で接続をプールに返却します 開発者の追加の実装作業: Oracle 接続プールにかかわる最小の作業量 - 開発者がアプリケーションで使用される接続テストを特定し、場合によってはデータベース内に新しい接続テストを作成するために、アプリケーション・サーバーを使用する場合は最大で数時間(アプリケーション・サーバーを使用しない場合はアプリケーションの複雑さに応じて数日間)	● 計画メンテナンス中のエラーを回避します(計画外停止のエラーの可能性は残ります)。 ● アプリケーションの割込みなしにワークロードを適切にドレインおよび移動する機能性 ● 計画外停止と計画メンテナンスのイベントに適用できます。場合によっては、長期実行トランザクションは、計画メンテナンス中に延期または一時停止する必要があります。	ワークロードは、計画メンテナンス中にわずかな遅延とエラーなしでインスタンス間で正常に移動されます。
レベル 3: 計画外および計画フェイルオーバーのアプリケーションからのマスク 「レベル 3: 計画外および計画フェイルオーバーのアプリケーションからのマスクの構成」を参照	レベル 1 および 2 の構成 ＋ "アプリケーション・コンティニュイティ "ソリューション データベースまたはセキュリティの管理者: ● 追加のセキュリティと特権が必要です アプリケーション開発者: ● データベース外の外部アクション(副次的作用など)を考慮する必要があります 追加の実装作業: 保護範囲を確認するために、開発者とデータベース管理者の間の共同作業に数日から数週間(アプリケー	● 処理中のトランザクションは、アプリケーション・コードの変更なしに自動的にコミットを承認するかリプレイします ● データベース管理者とアプリケーション開発者は、AWR 統計を使用して保護範囲を評価し、ACCHK (アプリケーション・コンティニュイティ・ヘルス・チェック)を使用してトランザクションを再実行できる場合とできない場合の範囲または例外を特定することで、準備状況の確認のために調整を実施します	アプリケーションから計画外および計画フェイルオーバーをマスクします ● アプリケーションは、計画メンテナンスおよび停止中のエラーの認識を回避します ● 処理中のコミットされていないトランザクションがリプレイされ、コミットされたトランザクションは承認されてリプレイされません

HAレベル	構成	エクスペリエンス	メリット
	ションの複雑さに応じて異なります)		

前の表で説明したすべてのHAレベルは、次の理由から、単一の接続VIPエンドポイントとしてロード・バランサを使用する接続管理のアプローチよりも優れています。

- スマート・サービス・ヘルスおよびインテリジェント再接続: Oracle ClusterwareとOracle Data Guard Brokerは、クラスタおよびデータベースのヘルスと状態を詳細にモニターして、プライマリでオープンされているデータベース・サービスに接続がルーティングされるようにします。
- 透過的および自動的フェイルオーバー: データベースのヘルスを問い合せて、VIPの移動に適したデータベースを決定する必要はありません。表で説明した高可用性アプローチでは、すべてが透過的になります。
- 高速通知および自動的な接続の再試行: すでに接続されているセッションの切断は即時に実施され、Oracle ClusterwareとData Guard Brokerがプライマリおよびスタンバイ・データベースで停止またはロールの変更を検出したときにはインテリジェントに実施されます。

用語

このドキュメント全体を通じて次の用語が使用されています。

- ドレイン: あるインスタンスから別の使用可能なインスタンスに接続を移動します。

  あるインスタンスから別のインスタンスに正常にセッションを移動するためのドレインは、計画メンテナンス中および負荷のリバランス中に使用されます。接続は、アプリケーションがプールに接続を返却した後に新しい接続を取得するときや、別のルールが満たされたときに移動されます。

- フェイルオーバー: サービスを提供する新しいインスタンスで同等のセッションを再確立します。

  フェイルオーバーは、計画外停止中および計画メンテナンス中に、割り当てられた期間内にセッションがドレインされない場合に発生します。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は、アプリケーションがエラーを受信することはありません。

ソフトウェアの推奨事項

HAレベルの構成に推奨のソフトウェアは次のとおりです。

- Oracle Real Application Clusters (Oracle RAC)およびOracle Clusterware (効率的な停止の管理のためのサービスおよびインフラストラクチャを提供します)。Oracle Grid Infrastructure (GI)リリース19c以降の使用をお薦めします
- Oracle Active Data Guardは、データベース、クラスタ、ストレージまたはサイトの障害から保護するためにお薦めです。
- Oracle Database 19cのクライアントとデータベース、またはそれ以降の長期サポート・バージョン(最新のパッチ・レベルが適用されたもの)

レベル1: 基本的なアプリケーション高可用性の構成

アプリケーションがインスタンス、ノードまたはデータベースの障害に即座に対応して、障害が発生していないデータベース・インスタンスへの新しい接続を迅速に確立できる、高可用性のレベルを実装します。

アプリケーションHAレベル1では、計画外停止および計画停止の停止時間が最小限に抑えられます。こうしたメリットは、次の推奨事項がアプリケーション構成で確実に実装されるようにすることで得られます。設計時のコード変更は不要です。

レベル1を実装するステップの概要は次のとおりです。

- ステップ1: 高可用性データベース・サービスの構成
- ステップ2: 高可用性のための接続文字列の構成
- ステップ3: FANが使用されていることの確認
- ステップ4: アプリケーションへの再接続ロジックの実装の確認

ステップ1: 高可用性データベース・サービスの構成

高可用性機能を使用するために、デフォルト以外のロールベースのデータベース・サービスを作成します。

データベース・サービスとは、ワークロードを管理するための論理的な抽象化であり、同様のSLAやワークロードのタイプ(OLTPとバッチなど)を共有するアプリケーションのグループです。データベース・サービスは、場所の透過性を提供し、基礎となるシステムの複雑な側面をクライアントから隠します。

高可用性機能を使用するために、アプリケーションはデフォルト以外のデータベース・サービスに接続する必要があります。デフォルトのデータベース・サービスまたはデフォルトのPDBサービス(データベースまたはPDBと同じ名前のサービス)を使用するかわりに、単一のサービス(または各種アプリケーション・ワークロードに応じて必要になる複数のサービス)を明示的に作成する必要があります。

Oracle Autonomous Databaseでは、推奨属性を使用することで適切なサービスが作成されます。

サービスのサーバー側構成について

こうしたサービスは、Oracle Clusterwareを通じてサービスを設定するためにデータベース管理者が構成します。

Oracle Data Guardとスタンバイ・データベースを使用する場合は、読取り/書込み操作に対してアプリケーションがプライマリ・データベースに接続するようにするためのプライマリ・ロールを使用するサービスと、読取り専用の操作と少数低頻度の書込みを必要に応じてスタンバイ・データベースにオフロードするためのスタンバイ・ロールを使用するサービスを作成します。

サービスは、ロールに基づいてData Guardのロール・トランジション(スイッチオーバーやフェイルオーバーなど)の後に自動的に起動および停止します。

次のいずれかのセクションのアーキテクチャに応じてサービスを構成します。

- 高可用性サービスの構成
- Oracle Active Data Guardまたはスタンバイ・ロールの高可用性サービスの構成

ノート:

サービスは、作成後に使用できるように起動する必要があります。次のようなコマンドを使用します。

```
$ srvctl start service -db mydb -service my_service
```

関連項目:

『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』のOracleサービスの使用

高可用性サービスの構成

高可用性機能を使用するために、デフォルト以外のロールベースのデータベース・サービスを作成します。

サービスは、単一の優先インスタンスに接続するように構成することも、優先インスタンスが停止している場合には使用可能なインスタンスに接続するように構成することもできます。単一のインスタンスでのみ使用可能なサービスは、シングルトン・サービスと呼ばれます。これにより、クラスタ内のインスタンス間でワークロードの分離が可能になります。

また、クラスタの複数のインスタンスに接続して、すべてのインスタンスに作業を分散するようにサービスを構成することもできます。さらに、1つのインスタンスが停止している場合には、正常に稼働しているインスタンスに接続できます。

それ以外にも、インスタンスのサブセットを「優先」として構成し、インスタンスの別のサブセットを「使用可能」として構成できる組合せもあります。こうしたサブセットを使用すると、一部のインスタンス間で負荷を分散しながら、別のインスタンスからは作業を分離できます(また、障害発生時に使用可能なインスタンスを確保できます)。

例1: シングルトン・サービス

この例では、プライマリ・ロールにMyServiceというシングルトン・サービスを作成します。このサービスは、インスタンスinst1に接続します(そのインスタンスが使用できない場合を除きます)。インスタンスが使用できない場合は、inst2に接続します。また、セッションのドレインを待機するデフォルトの300秒のドレイン・タイムアウトも構成します。その時間の終了時点で、残りのセッションは`IMMEDIATE`オプションによってすべて終了します。

`commit_outcome`と`failovertype`の設定により、今後、透過的アプリケーション・コンティニュイティ(TAC)の実装を決定した場合に、その機能を使用できるようにします(これは高度な機能です。詳細は、Oracle MAAのOracle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)。TACを有効にしても悪影響はありません。HAレベル3への移行を決定したときに、前提条件が満たされていれば自動的にメリットが得られます。

```
$ srvctl add service -db mydb -service my_service -pdb mypdb
 –preferred inst1 -available inst2 -commit_outcome TRUE
 -failovertype AUTO -notification TRUE -drain_timeout 300
 -stopoption IMMEDIATE -role PRIMARY
```

アプリケーションが別のOracle RACインスタンスにアプリケーションのブラックアウトなしで正常に切り替えられるようにするために、`drain_timeout`の間隔は、アプリケーションがトランザクション間の接続をクローズして、正常に停止または別のインスタンスに移動できる十分なタイムアウトに設定します。`drain_timeout`の間隔は、短いOLTPアプリケーションに対して最適です。大規模なバッチ操作の場合は、計画メンテナンス・ウィンドウの前に、そのような操作を遅延させるか一時停止することをお薦めします。

例2: 複数のインスタンスを持つサービス

この例では、前述のシングルトンと同様のサービスを作成しますが、このクラスタ内の複数のインスタンスに接続が分散されます。

```
$ srvctl add service -db mydb -service my_service -pdb mypdb
 –preferred inst1,inst2 -commit_outcome TRUE -failovertype AUTO
 -notification TRUE -drain_timeout 300 -stopoption IMMEDIATE
 -role PRIMARY
```

Oracle Active Data Guardまたはスタンバイ・ロールの高可用性サービスの構成

スタンバイ・データベース(読取り専用フィジカル・スタンバイ)への接続に使用するサービスを作成します。

次の例に示すように、サービスを作成します。

```
$ srvctl add service -db mydb -service my_standby_service
 -pdb mypdb –preferred inst1 -available inst2 -notification TRUE
 -drain_timeout 300 -stopoption IMMEDIATE -role PHYSICAL_STANDBY
```

ステップ2: 高可用性のための接続文字列の構成

データベース・スイッチオーバーや別のサイトへのフェイルオーバーなどの様々なシナリオで適切な接続ができるように、ここに示す接続文字列の構成を使用することをお薦めします。

例1: Oracle RACプライマリ・データベースとの接続文字列、スタンバイなし

```
Alias = (DESCRIPTION =
(CONNECT_TIMEOUT=
90)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=1000ms)
  (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=clu_site1-scan)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME = my_service)))
```

例2: Oracle RACプライマリ・データベースおよびスタンバイ・データベースとの接続文字列

この例では、Oracle RACプライマリ・データベースまたはスタンバイ・データベースのどちらかの使用可能なデータベースに接続します。

```
Alias = (DESCRIPTION =
(CONNECT_TIMEOUT=
90)(RETRY_COUNT=100)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=1000ms)
  (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=clu_site1-scan)(PORT=1521)))
  (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=clu_site2-scan)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME = my_service)))
```

ノート:

clu_site1-scan および clu_site2-scan は、それぞれ site1 および site2 のクラスタ内の SCAN リスナーを表します。

最新ドライバの使用をお薦めしますが、前述の接続文字列の例はリリース12.2以降のすべてのOracleドライバに使用する必要があります。特定の値で調整できますが、この例では出発点として妥当な値を示しているため、ほとんどすべてのケースに使用できます。

接続文字列またはURLは、LDAPや`tnsnames.ora`などの中央の場所に維持するようにしてください。接続文字列またはURLはプロパティ・ファイルやプライベートの場所に分散しないでください。そのようにすると、メンテナンスが非常に困難になります。集中管理された場所を使用すると、標準のフォーマット、チューニングおよびサービスの設定を維持できます。このためのOracleのソリューションは、Oracle Unified Directory製品とともにLDAPを使用することです。

関連項目:

- Data Guardスイッチオーバーまたはフェイルオーバー時の接続時間の推定
- Oracle Net TNS文字列パラメータ
- 『*Oracle Unified Directory*の管理』のOracle Unified Directory
- 『*Oracle Database Net Services*リファレンス』のローカル・ネーミング・パラメータの概要

ステップ3: FANが使用されていることの確認

FANは、停止の発生時にインテリジェントで即時的な割込みを提供して、アプリケーションへの影響やブラウンアウトを大幅に縮小します。

サービスが定期的な保守や計画外の障害(ノードやネットワークの停止など)のためにドレインする必要がある場合、アプリケーションは別のインスタンスやサイトに接続を迅速に移動できるようにリアルタイムで通知されることが必要です。これは、Oracleの高速アプリケーション通知(FAN)機能を使用することで実現できます。FANを有効にすると、ノード、ネットワーク、サイトの障害などの物理的な障害が発生したときに、アプリケーションがハングしないようになります。

FANは、Oracle ClusterwareのOracle Notification Service (ONS)を使用して、クラスタからイベントを受信します。ONSにはクライアントとサーバーの間で使用可能なポートが必要になり、場合によってはファイアウォール・ポートのオープンが必要になります。

前述のステップ1および2で推奨されているサービスと接続文字列を使用していると、FANイベントを受信するための登録が自動的に有効になります。

ONSポート(デフォルトでは6200)は、すべてのデータベース・サーバー、ファイアウォールおよびOracle Active Data Guardノードでオープンしておく必要があります。例に示すように、Oracle Autonomous Database on Dedicated Exadata Infrastructure (ADB-D)、Exadata Database Service on Dedicated Infrastructure (ExaDB-D)、Oracle Exadata Database Service on Cloud@Customer (ExaDB-C@C)などのクラウド環境では、このステップは非常に重要です。

クライアントのFANの有効化

FANを使用するためのアプリケーション・コードの変更は不要です。FANに必要なものは、Oracleドライバおよび推奨のデータベース接続文字列のみです。

FANは自動構成され、設定なしで有効化されます。Oracleデータベースへの接続時に、データベースはURLまたはTNS接続文字列を使用してクライアントでFANを自動構成します。

FANの自動構成には、ステップ2に示したTNS形式を使用することが重要です。別の形式の構文を使用すると、FANが自動構成されないことがあります。FANを使用するには、データベース・サービス(ステップ1で構成したサービス)に接続して、Oracle Notification Service (ONS)からイベントを受信できるようにする必要があります。そのためには、前述したようにポートのオープンが必要になることがあります。

また、接続プールの設定(後述)を使用して、必要に応じて手動でFANを構成することもできます。

各種プール・タイプの構成要件については、次を参照してください。

JDBC FANの要件

UCPを使用するクライアント・ドライバの場合:

- ONSの自動構成には、推奨の接続URL/文字列(前述)を使用します。
- JDBC JARファイル`ojdbc8.jar` (またはそれ以降)、`ons.jar`および`simplefan.jar`をCLASSPATHに含め

ます(また、オプションのウォレットjar: `osdt_cert.jar`、`osdt_core.jar`および`oraclepki.jar`を必要に応じて加えます)。

- プールまたはドライバのプロパティを設定して、高速接続フェイルオーバーを有効にします(たとえば、UCPでは`setFastConnectionFailoverEnabled(true)`を使用して`PoolDataSource`に対して設定します)。
- 自動コミット接続プロパティを無効にします(たとえば、UCPでは、`setConnectionProperty(OracleConnection.CONNECTION_PROPERTY_AUTOCOMMIT, "false");`を使用して`PoolDataSource`に対して無効にします)。
- サード・パーティのJDBCプールの場合は、データ・ソースとしてユニバーサル接続プール(UCP)の使用をお薦めします。
- データベース・サーバーからのONS通信用のポート6200をオープンします(6200はデフォルト・ポートです。別のポートが選択されている可能性があります)。

推奨の接続URL/文字列を使用できない場合は、次の設定によってクライアントを手動で構成します。

```
oracle.ons.nodes=Node01:6200, Node02:6200, Node03:6200
```

手動で構成する場合は、追加の設定が必要になることがあります。たとえば、`walletfile`と`walletpassword`です。

UCP以外の接続プールにも同様の要件があります。

OCI FANの要件

- Oracle Call Interface (OCI)クライアントの場合:

  OCIクライアントは、FANをドライバ・レベルで埋め込むことで、プール・ソリューションに関係なくすべてのクライアントが使用できるようにします。

  データベース・サービスには、属性"-notification TRUE"を設定しておく必要があります。

  oraaccess.xmlを使用している場合は、eventsタグがTRUEであることを確認します。

  ```
  <oraaccess> xmlns="http://xmlns.oracle.com/oci/oraaccess"
  xmlns:oci="http://xmlns.oracle.com/oci/oraaccess"
  schemaLocation="http://xmlns.oracle.com/oci/oraaccess
  http://xmlns.oracle.com/oci/oraaccess.xsd">
  <default_parameters>
  <events>true</events>
  </default_parameters>
  </oraaccess>
  ```

- ODP.Netクライアントの場合

  接続文字列でHA eventsを指定します

  ```
  "user id=oracle; password=oracle; data source=HA; pooling=true; HA
  events=true;"
  ```

関連項目:

『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』のOracle統合クライアントおよびFANの概要

ステップ4: アプリケーションへの再接続ロジックの実装の確認

アプリケーションは、データベース・コール中の接続障害の例外とエラーを捕捉するように記述して、新しい接続を取得して新しい作業を続行できるようにする必要があります。

JDBCベースのアプリケーションの場合は、`SQLRecoverableException`を捕捉することで、接続エラーと通常のアプリケーション・エラーまたはSQLエラーの区別ができます。接続エラーが捕捉された場合は、新しい接続を取得する必要があります。こ

れは、SQLExceptionクラスの個別のOracleエラー(Oracle Databaseリリースによる調整が可能)を確認するよりも簡単で堅牢になります。

関連項目:

接続再試行ロジックの例

レベル2: 計画メンテナンスに向けたアプリケーションの準備の構成

アプリケーションHAレベル1: 基本的なアプリケーション高可用性に基づいたレベル2には、計画メンテナンス時のアプリケーションの影響を最小限に抑えるためのセッション・ドレインの構成が追加されます。

レベル1の実装後は、次のいずれかの選択肢から、対象のアプリケーションに適した計画メンテナンス・ソリューションを実装できます。計画操作は、サービスを再配置または停止する場合やスイッチオーバーする場合に、ユーザーの作業が正常に完了できるようにするために使用できます。

アプリケーションへの影響を回避するためにお薦めする方法は、Oracle RACローリング方式で作業をドレインすることです。通常、ドレインの実行には一定の期間が割り当てられています。ドレインの開始には、FANと統合されたOracle接続プールの使用をお薦めします。

ドレインできない場合は、メンテナンスを開始する前に作業をドレインする方法もあります。

Oracle接続プールを使用できない場合は、その他の選択肢を使用できます。

次のプラクティスを採用して、アプリケーション高可用性をレベル2に引き上げます。

- 推奨オプション: Oracle接続プールの使用 - Oracle接続プールを使用して、リクエスト間には接続をプールに戻します。

  または、サード・パーティの接続プールやリクエスト境界付きのプールを使用してUCPを構成します。
- 代替オプション: 接続テストの使用 - Oracle接続プールを使用できない場合は、接続テストを使用できます。
- 計画メンテナンスに対するサーバー側操作の利用
- メンテナンス期間中にワークロードに影響を与えることなく、あるインスタンスから別の使用可能なインスタンスに負荷を分散できるように、十分なノード容量が使用できることを確認してください。

推奨オプション: Oracle接続プールの使用

計画メンテナンスの管理にお薦めのソリューションは、FAN対応のOracle接続プールを使用することです。

Oracleプールは、ノードとサイトの間のドレイン、再接続、リバランスなどの完全なライフサイクル管理を提供します。メンテナンスが進行して完了すると(インスタンスまたはノードごと)、セッションはインスタンス間で移動およびリバランスされます。アプリケーションがOracleプールとFANを使用していて、リクエスト間に接続をプールに返却する場合は、ユーザーに影響はありません。

サポートされているOracleプールには、次のものがあります。

- ユニバーサル接続プール(UCP)
- WebLogic Active GridLink
- Tuxedo
- OCIセッション・プール
- ODP.NET管理対象および管理対象外プロバイダ
- Python用のOracleセッション・プール

これらのプールを使用する場合、リクエスト間に接続がプールに返却されるようにすること以外に、アプリケーションの変更は不要です。

ベスト・プラクティスは、アプリケーションが必要な間のみ接続を取得して、データベース・コールの完了後にすぐに接続をプールに返却することです。接続をプールに戻さずに保持すると、プールは使用可能なインスタンスにセッションを正常に移動できなくなり、リソースの使用が非効率的になるため、それ以外の場合に使用されるよりも多くの接続が必要になります。そのため、アプリケーションは接続を取得したら、その接続を作業の完了直後に返却する必要があります。接続は、その後で別のスレッドで使用することも、再度必要になったときには同じスレッドで使用することもできます。接続プールに接続を返却することは、ドレインの実装方法に関係なく、一般的な推奨事項です。

ノート:

接続の取得と返却のための構文は、プールの実装によって異なります。

たとえば、UCP では、`PoolDataSource` オブジェクトの `getConnection()`メソッドを使用して接続を取得し、データベースでの作業の完了後に `close()`メソッドを使用して接続を返却します。

Oracle接続プールは、接続が借用されるたびに接続を検証することで、その接続をエラーなしで使用できるようにします。

『Universal Connection Pool開発者ガイド』を参照してください

代替オプション: 接続テストの使用

Oracleプールを使用できない場合は、Oracleクライアント・ドライバ19cまたはOracle Database 19cによってセッションがドレインされます。

サービスが再配置または停止された場合や、Oracle Data Guardによるスタンバイ・サイトへのスイッチオーバーがある場合、Oracle DatabaseとOracleクライアント・ドライバは、次の項目に応じて接続を解放するための安全な場所を検索するように通知されます。

- 接続の有効性に対する標準接続テスト(JDBCの`isValid()`など)
- 接続の有効性に対するカスタムSQLテスト

カスタム・バッチ・アプリケーションの場合は、バッチ間の接続をテストします。接続テストが失敗したときには、別の接続を作成するか借用します。

サードパーティ接続プールの場合は、ベンダーが提供する接続テストを有効にします。接続テストが失敗すると、サード・パーティのプールは接続をクローズして、別の接続を借用できるようになります。

ノート:

接続テストの失敗時にプールをフラッシュおよび破棄するための接続プールのオプションはすべて無効にしてください。

JDBC Thinドライバでの標準接続テストの使用

Oracle以外のプールでは、JDBC Thinドライバで接続テストを使用するために、次のステップを実行します。

1. プールで接続テストを有効にして(実装はサードパーティ・プールによって異なります)、テスト`java.sql.Connection.isValid(int timeout)`を使用します
2. Javaシステムのプロパティを設定します
    - `-Doracle.jdbc.fanEnabled=true`

- -Doracle.jdbc.defaultConnectionValidation=SOCKET (Oracle Database 19cでは、isValid()コールはクライアントに対してローカルであり、データベースへのトリップは不要です)

OCIドライバでドレインするためのOCI接続テストの使用

Oracle Call Interface (OCI)セッション・プールを使用しているときには、この接続チェックが自動的に実施されます。OCIドライバを直接使用しているときには、OCI_ATTR_SERVER_STATUSを使用します。これは、コード変更がある唯一のメソッドです。

対象のコードでは、接続の借入時または返却時にサーバー・ハンドルをチェックし、セッションが切断されているかどうかを確認します。サービスが停止または再配置されているときに、値OCI_ATTR_SERVER_STATUSはOCI_SERVER_NOT_CONNECTEDに設定されます。

次のコード例は、OCI_ATTR_SERVER_STATUSの使用方法を示しています。

```
ub4 serverStatus = 0
OCIAttrGet((dvoid *)srvhp, OCI_HTYPE_SERVER,
 (dvoid *)&serverStatus, (ub4 *)0, OCI_ATTR_SERVER_STATUS, errhp);
if (serverStatus == OCI_SERVER_NORMAL)
printf("Connection is up.¥n");
else if (serverStatus == OCI_SERVER_NOT_CONNECTED)
printf("Connection is down.¥n");
/* Close connection and get a new one */
```

Oracle Databaseでドレインするための接続テストの使用

Oracle Database 19cでは、セッションのドレインが可能です。メンテナンス中に接続テストを実行すると、データベースがセッションを閉じて、アプリケーション・サーバーが接続を閉じます。

ビューDBA_CONNECTION_TESTSを使用して、有効になっている接続テストとルールを確認します。SQLベースの接続テストを使用している場合は、接続プールまたはアプリケーション・サーバーのデータベースで有効にされているものと同じSQL (同じ文)を使用します。

追加の接続テストが必要な場合は、サービス、プラガブル・データベース、または非コンテナ・データベースの接続テストを追加、削除、有効化、または無効化できます。

たとえば:

```
SQL> EXECUTE
 dbms_app_cont_admin.add_sql_connection_test('SELECT COUNT(1) FROM DUAL');
SQL> EXECUTE
dbms_app_cont_admin.enable_connection_test(dbms_app_cont_admin.sql_test,
'SELECT COUNT(1) FROM DUAL');
SQL> SELECT * FROM DBA_CONNECTION_TESTS
```

PL/SQLベースのワークロードをドレインするためのUSERENV関数の使用

関数USERENVは、セッションがドレイン・モードであるかどうかを確認するために使用します。たとえば、この関数を使用して、レコードを処理する長時間実行のPL/SQLループの場合に接続を停止するタイミングと新しい接続を取得するタイミングを判断します。

```
SQL> select SYS_CONTEXT('USERENV', 'DRAIN_STATUS') from dual ;
SYS_CONTEXT('USERENV','DRAIN_STATUS')
--------------------------------------------------------------------------------
DRAINING
SQL> select SYS_CONTEXT('USERENV', 'DRAIN_STATUS') from dual ;
SYS_CONTEXT('USERENV','DRAIN_STATUS')
--------------------------------------------------------------------------------
NONE
```

計画メンテナンスに対するサーバー側操作の利用

計画メンテナンスの接続を管理するために、サーバー側操作が必要になります。

Oracle Databaseに接続されているサービスは、接続テストとドレインに許可する時間を指定するドレイン・タイムアウト、およびドレイン・タイムアウトの経過後に適用される`stopoption` (通常は`IMMEDIATE`)で構成されます。SRVCTLによって管理される`stop`、`relocate`および`switchover`コマンドには、サービスに設定された値を必要に応じてオーバーライドするための`drain_timeout`および`stopoption`スイッチが含まれます。

サービスの構成時には、そのサービスがメンテナンス操作中に自動的に使用されるように、適用可能な必須のドレイン・タイムアウトを指定することをお薦めします。

メンテナンス・コマンドは、「サーバー側の計画メンテナンスのコマンド例」の例で説明したコマンドに類似しています。こうしたコマンドは、ドレインの開始に使用できます。追加のオプションは、My Oracle Support (MOS)のノート: Doc ID 1593712.1の説明に従って必要に応じて含めます。Oracleツールのフリート・パッチ適用およびプロビジョニング(FPP)なども、これらのコマンドを使用します。

Oracle Clusterwareは、サービスの実行に必要な現在実行されていないインスタンスを起動できます。再配置できないサービスや再配置が不要なサービスは停止されます。シングルトン・サービスが、その他に使用可能なインスタンスなしで定義されていると、完全な停止時間が発生する可能性があり、これは予期される動作です。優先インスタンスと少なくとも1つの使用可能なインスタンスを常に定義することをお薦めします。

メンテナンスが完了してインスタンスが再起動されると、Oracle Clusterwareサービス属性によってサービスが終了する場所が自動的に決定されるため、追加のSRVCTLアクションは必要ありません。

関連項目:

『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』の計画メンテナンス前のサーバーのドレイン

レベル3: 計画外および計画フェイルオーバーのアプリケーションからのマスクの構成

レベル1およびレベル2に基づいたアプリケーション・コンティニュイティは、アプリケーションからデータベースの割込みをマスクし、タイムアウトと停止を処理するためにお薦めします。

アプリケーションからのデータベースの割込みには、ドレインしないアプリケーション・ワークロードが含まれる場合があります(計画フェイルオーバー)。アプリケーション・コンティニュイティは、ACとTACの2つのモードで動作するように構成できるデータベース・サービスで有効になります。

アプリケーション・コンティニュイティ(AC)

アプリケーション・コンティニュイティは、JDBC Thinアプリケーション用のOracle Database 12.1以降、およびOracle Database 19以降のNode.jsやPythonなどのオープンソース・ドライバをサポートするOCIおよびODP.NETアプリケーション用のOracle Database 12.2.0.1から停止を非可視化します。

アプリケーション・コンティニュイティは、セッション状態およびトランザクション状態が含まれている既知のポイントからセッションを再構築およびリカバリし、割り込まれたすべての処理中の作業をリプレイします(すでにコミットされている作業はリプレイされません)。リプレイが完了すると、割込みの発生がなかったかのようにアプリケーションに結果が返されます。

アプリケーション・コンティニュイティは、Oracle接続プールを使用するOLTPアプリケーションにお薦めします。これは、アプリケーションがデータベースに接続するデータベース・サービスで有効にされます。

透過アプリケーション・コンティニュイティ(TAC)

Oracle Database 19c以降、透過的アプリケーション・コンティニュイティ(TAC)は、セッションとトランザクションの状態を自動的に追跡して記録します。これにより、リカバリ可能な停止の後にデータベース・セッションのリカバリと再構築が可能になります。これは、アプリケーションの知識やアプリケーションコードの変更に依存することなく、対象のアプリケーションでTACを有効にすることができます。

アプリケーションの透過性とフェイルオーバーは、アプリケーションがデータベースへのコールを発行するときのセッション状態の使用状況を取得して分類する状態追跡情報を利用することで実現されます。対象のサービスで、FAILOVERTYPEをAUTOに設定します。

Oracle接続プールを使用していない場合(SQL*PLUSなど)、またはアプリケーションに関する知識がない場合は、データベース・サービスでTACを有効にします。

ACおよびTACによる計画フェイルオーバー

計画フェイルオーバーは、セッションがリプレイ可能であり、ドレインが想定されていないとデータベースが判断した時点でOracle Databaseによって起動されるフェイルオーバーです。

計画フェイルオーバーは、ACまたはTACの使用時にデフォルトで有効になります。これは、FANや接続テストが構成されていないなど、その他のドレイン方法がアクティブでない状況を改善します。

計画フェイルオーバーでは、リプレイが有効な場合に、早期フェイルオーバーすることでメンテナンスを短縮できます。

たとえば、TACによる計画フェイルオーバーは、SQL*Plusで使用されるメンテナンス・ソリューションです。

関連項目:

- 『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』のアプリケーション・コンティニュイティの確保

およびアプリケーション・コンティニュイティに関する制限および他の考慮事項

- https://database-heartbeat.com/category/application-continuity/のブログ

接続プールへの接続の返却

リクエスト境界は、アプリケーション・コンティニュイティ(AC)の必須事項であり、透過的アプリケーション・コンティニュイティ(TAC)の推奨事項です。

ユニバーサル接続プール(UCP)やOCIセッション・プールなどのOracle接続プールを使用すると、アプリケーションの変更なしに適切なポイントで、リクエスト境界がセッションに自動的に埋め込まれます。アプリケーションは、リクエスト境界の終了を挿入するために作業単位(データベース・リクエスト)の完了時にOracle接続プールに接続を返却する必要があります。これは、ODP.Net管理対象外プロバイダ、WebLogic Active GridLinkおよびRedHatの使用時にも適用されます。

また、透過的アプリケーション・コンティニュイティ(TAC)は、リクエスト境界を検出します。Oracle Database 19cでは、このような境界を次の条件で検出します。

- 処理中のトランザクションが存在しない
- カーソルが文キャッシュに戻されているか取り消されている(カーソルはトランザクション間でオープンしたままにしません)
- リストア不可能なセッション状態が存在していない(PL/SQLグローバル、OJVM、移入された一時表)

サービスのFAILOVER_RESTOREの設定

フェイルオーバー時にセッション状態をリストアするために、データベース・サービスで属性FAILOVER_RESTOREを設定します。

アプリケーションは、データベース・セッション状態を変更するように記述できます(通常はALTER SESSIONコマンドを使用します)。フェイルオーバー後に作業をリプレイする場合は、そうした状態が存在している必要があります。

サービス構成で、アプリケーション・コンティニュイティにFAILOVER_RESTORE LEVEL1、TACにFAILOVER_RESTORE AUTOを使用します。アプリケーションのHAレベル1のステップに従うと、自動的にFAILOVER_RESTORE AUTOを設定するFAILVOERTYPE AUTOを使用するサービスが作成されます。

ウォレットの使用をお薦めします。ACおよびTACは、ウォレットを利用することで、すべての変更可能なデータベース・パラメータがFAILOVER_RESTOREによって自動的にリストアされるようにします。ウォレットは、ADB-DおよびADB-Sに対して有効化されていて、データベース・リンクに使用されるものと同じです。

関連項目:

データベースのウォレットの詳細な設定方法は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』の*FAILOVER_RESTORE*のキーストアの構成を参照してください。

リプレイ時の元の関数値のリストア

Oracle Database 19cでは、リプレイ時にSQLのSYSDATE、SYSTIMESTAMP、SYS_GUIDと、sequence.NEXTVAL, CURRENT_TIMESTAMPおよびLOCALTIMESTAMPの値が保持されます。

PL/SQLを使用している場合は、アプリケーション・ユーザーにはGRANT KEEPを使用し、順序所有者にはKEEP句を使用します。KEEP権限が付与されていると、再実行時に元の関数結果が適用されます。

```
SQL> GRANT KEEP DATE TIME to scott;
SQL> GRANT KEEP SYSGUID to scott;
SQL> GRANT KEEP SEQUENCE mySequence on mysequence.myobject to scott;
```

副次的作用

データベース・リクエストにデータベースからの外部コール(メールの送信やファイルの転送など)が含まれている場合、これは副次的作用と呼ばれます。

リプレイの発生時には、副次的作用をリプレイするかどうかを選択できます。多くのアプリケーションでは、ジャーナルのエントリ、メールの送信、ファイル書込みなどの副次的作用の繰り返しが要求されます。アプリケーション・コンティニュイティの場合、副次的作用はリプレイされますが、プログラムで回避することもできます。それとは逆に、透過的アプリケーション・コンティニュイティでは副次的作用がリプレイされません。

JDBC構成

スタンドアロン方式で`oracle.jdbc.replay.OracleDataSourceImpl`を使用するか、Java接続プール(UCPなど)またはWebLogic AGL Server接続プールの接続ファクトリ・クラスとして構成します。

UCPのAC/TACの有効化の詳細は、『Oracle Universal Connection Pool開発者ガイド』のアプリケーション・コンティニュイティのためのデータ・ソースの構成を参照してください。JDBCドライバのデータ・ソース・クラス`oracle.JDBC.replay.OracleDataSourceImpl`は、UCPデータ・ソースPoolDataSourceImplの接続ファクトリ・クラスとして構成します。

正確なデータ・ソースおよび接続プールの構成は、常にサードパーティ接続プール、フレームワーク、アプリケーション・サーバー、コンテナなどの特定のベンダー製品に固有であることに注意してください。

モニタリング

アプリケーション・コンティニュイティでは、保護レベルをモニターするために統計を収集します。

そうした統計は自動ワークロード・リポジトリ(AWR)に保存され、自動ワークロード・リポジトリのレポートに使用できます。統計情報を確認して、保護されたコールの範囲、または保護されたコール数の減少や保護された時間の短縮について判断します。原因に関する詳細は、ACCHKユーティリティを使用します。

関連項目:

『*Oracle Real Application Clusters*管理およびデプロイメント・ガイド』のアプリケーション・コンティニュイティ保護チェック

リファレンス

アプリケーションの継続的な可用性の構成に関する参照トピック。

次の内容について説明します。

- Data Guardスイッチオーバーまたはフェイルオーバー時の接続時間の推定
- Oracle Net TNS文字列パラメータ
- 接続再試行ロジックの例
- サーバー側の計画メンテナンスのコマンド例

Data Guardスイッチオーバーまたはフェイルオーバー時の接続時間の推定

接続文字列の設定により、スイッチオーバー時またはフェイルオーバー時に次の最大接続時間を許容します。

- Data Guardスイッチオーバー:

  RETRY_COUNT X RETRY_DELAY = 100 x 3秒= 300秒。

- Data Guardフェイルオーバー:

  (3 SCAN x TRANSPORT_CONNECT_TIMEOUT) + (RETRY_COUNT x (RETRY_DELAY + (3 SCAN x TRANSPORT_CONNECT_TIMEOUT))) = (3 x 1) + (100 x (3 + (3 x 1)) = 3 + 600 = 603秒

clu-site2へのData Guardスイッチオーバー/Data Guardフェイルオーバー後、clu-site1が停止すると、clu-site2への初期接続に3秒かかります(この遅延は接続プールを使用すると軽減できます)。clu-site1が再びアクセス可能になると(スタンバイになると)、スタンバイ上のリスナーがサービスが存在しないと即座に応答し、クライアントに他のADDRESS_LISTへの接続を促すため、ほぼ瞬時に接続が行われます。

- スイッチオーバーまたはフェイルオーバーが最大時間よりもかなり早く完了した場合、アプリケーションの影響は少なくなります。
- システムでスイッチオーバーまたはフェイルオーバーを完了するまでに300秒以上かかる可能性がある場合は、RETRY_COUNTを増やします。Data Guardスイッチオーバーの完了にさらに時間が必要な場合は、RETRY_COUNTを100より大きい値に変更します。
- Oracle Clusterwareを使用していない場合、HOSTアドレスはSCAN VIPではなく単一のVIPを参照します。つまり、ネットワーク待機時間を考慮するには、TRANSPORT_CONNECT_TIMEOUTの値を大きくまたは小さく設定する必要があります。

Oracle Net TNS文字列パラメータ

ここでは、接続文字列で使用されるパラメータについて説明します。

CONNECT_TIMEOUT

リスナー・アドレスへの接続が試行されたときに適用します。

この設定は、特定のADDRESSエンドポイントを使用した接続が完了する必要がある最大時間を表します。これにはトランスポート接続時間と、発生する必要があるその他のアクション(SCAN VIPからリスナーVIPへ、最終的にフォアグラウンドで生成されたプロセスへのリダイレクション)が含まれます。

CONNECT_TIMEMOUTはTRANSPORT_CONNECT_TIMEOUTより大きい必要があります。そうでない場合、TRANSPORT_CONNECT_TIMEOUTはCONNECT_TIMEOUTによって事実上制限されます。TRANSPORT_CONNECT_TIMEOUTが指定されていない場合、CONNECT_TIMEOUTはADDRESSエンドポイントへの接続試行全体(データベース・フォアグラウンドへのトランスポート接続と最終接続の両方)のタイムアウトとして機能します。

CONNECT_TIMEOUTの値は、ビジー状態のリスナーやホストへの接続時に発生する可能性のある遅延に加えて、TRANSPORT_CONNECT_TIMEOUTの値も考慮に入れた十分な大きさにすることをお薦めします。接続文字列の例にある90秒という値は非常に長いため、場合によっては短縮する必要があります。ただし、短すぎると設定が非生産的になる可能性があります。これは、失敗する可能性もある追加の試行の原因となり、途中で放棄される可能性のある接続リクエストを処理するために、より非生産的なワークロードをサーバーに導入する可能性があるためです。

RETRY_COUNT

すべてのADDRESS_LISTSで接続試行が失敗した場合は、最初のADDRESS_LISTから始まる接続の試行がRETRY_COUNT回行われます。

これは、スタンバイへのスイッチオーバーまたはフェイルオーバーが進行中で、接続を操作が完了するまで試行し続ける必要がある場合に便利です。

RETRY_DELAY

再試行間の秒数です。

新しいプライマリ・データベースを開くことが許可される短い時間を指定します。このパラメータにRETRY_COUNTを指定して使用すると、新しく開かれたデータベースへの接続を適切な時間待機できます。

プライマリ・データベースが開いた時間に近い時間で接続が完了できるように、再試行回数を多くして再試行の遅延を短くすることをお薦めします。

TRANSPORT_CONNECT_TIMEOUT=1000ms

ADDRESSのTCPホストを使用してリスナーに接続するには、最大1000ミリ秒かかります。接続が確立されない場合は、次のADDRESSを試します。

Oracle RAC SCANホスト名を使用すると、SCANアドレスの各IPは内部で個別のADDRESS文字列に展開されます。接続の試行が失敗すると、各ADDRESSが試されます。

このパラメータを環境に合わせて調整すると、リスナー・エンドポイント接続が成功するまでの待機時間を最小限に抑えることができます。この時間が短すぎると、接続の試行をユーザーがすぐに諦めてしまい、遅延が増えて接続に失敗する可能性があります。時間が長すぎると、リスナー・エンドポイントに到達できない場合にユーザーが諦めて別のエンドポイントを試すまでの待機時間が長くなりすぎる可能性があります。

ホスト名はSCAN VIPを指定します。クラスタを使用する場合、これらは常に使用可能です。これは、ノードまたはネットワークが停止した場合にVIPがインスタント・リプライを送信することを意味します。これにより、接続時にVIPアドレスでサービスが提供されない場合は、次のアドレスがすぐに使用されます。

LOAD_BALANCE=ON within ADDRESS_LIST

ADDRESS内のHOSTがOracle RAC SCANの複数のアドレスに解決されると、すべてのアドレスがランダムに試行されます。

LOAD_BALANCE=OFFに設定すると、順序が毎回同じになり、SCANリスナーの1つに過度な負荷がかかる可能性があるため、ONに設定することをお薦めします。

接続再試行ロジックの例

再接続ロジックの参照コード例。

詳細は、「ステップ4：アプリケーションへの再接続ロジックの実装の確認」を参照してください。

簡易再試行(SANITY CHECK)

```
Connection jdbcConnection = getConnection();
int iterationCount = 0;
int maxIterations = 10;
for (int i = 0; i < maxIterations, i++)) {
  try {
    // apply the raise (DML + commit):
    giveRaiseToAllEmployees(jdbcConnection, i * 5);
    // no exception, the procedure completed:
    iterationCount++;
    Thread.sleep(1000);
  } catch (SQLRecoverableException recoverableException) {
    // Get a new connection only if the error was recoverable.
    System.out.println("SQLRecoverableException on iteration " + iterationCount )
     System.out.println("DB Connection lost - will attempt to get a new connection to
continue with the other iterations")
      // IF its OK to lose this work and move onto the next
        // iteration you could now try to get a new connection
        // This depends on what the code is doing; in many use
        // cases you must stop working, in others you can proceed
        // after logging a message to a log file
        // In our example, we assume we can proceed with the rest
        // of the loop if possible.
        // Using Transaction Guard, we can know if the work
        // committed and move on safely (covered in another example).
    try {
      jdbcConnection.close(); // close old connection:
      System.out.println("Connection closed - getting a new one")
      jdbcConnection = getConnection(); // reconnect to continue with other
iterations
    } catch (Exception ex) {
     System.out.println("Unable to close or get a new connection - giving up")
     throw ex;
    }
   } catch (SQLException nonRecoverableException) {
       // This is not a recoverable exception, so give up
    System.out.println("SQL UN-recoverable exception...give up the rest of the
iterations")
       throw nonRecoverableException;
  }
}
```

トランザクション・ガードを使用した接続再試行ロジック

```
Connection jdbcConnection = getConnection();
boolean isJobDone = false;
while (!isJobDone) {
  try {
    // apply the raise (DML + commit):
    giveRaiseToAllEmployees(jdbcConnection, 5);
    // no exception, the procedure completed:
    isJobDone = true;
  } catch (SQLRecoverableException recoverableException) {
    // Retry only if the error was recoverable.
    try {
      jdbcConnection.close(); // close old connection:
     } catch (Exception ex) {} // pass through other exceptions
```

```
    Connection newJDBCConnection = getConnection(); // reconnect to allow retry
    // Use Transacton Guard to force last request: committed or uncommitted
    LogicalTransactionId ltxid
      = ((OracleConnection) jdbcConnection).getLogicalTransactionId();
    isJobDone = getTransactionOutcome(newJDBCConnection, ltxid);
    jdbcConnection = newJDBCConnection;
  }
}
void giveRaiseToAllEmployees(Connection conn, int percentage) throws SQLException {
  Statement stmt = null;
  try {
    stmt = conn.createStatement();
    stmt.executeUpdate("UPDATE emp SET sal=sal+(sal*" + percentage + "/100)");
  } catch (SQLException sqle) {
    throw sqle;
  } finally {
    if (stmt != null)
      stmt.close();
  }
  // At the end of the request we commit our changes:
  conn.commit(); // commit can succeed but the commit outcome is lost
}
/**
* GET_LTXID_OUTCOME_WRAPPER wraps DBMS_APP_CONT.GET_LTXID_OUTCOME
*/
private static final String GET_LTXID_OUTCOME_WRAPPER =
  "DECLARE PROCEDURE GET_LTXID_OUTCOME_WRAPPER(" +
  " ltxid IN RAW," +
  " is_committed OUT NUMBER ) " +
  "IS " +
  " call_completed BOOLEAN; " +
  " committed BOOLEAN; " +
  "BEGIN " +
  " DBMS_APP_CONT.GET_LTXID_OUTCOME(ltxid, committed, call_completed); " +
  " if committed then is_committed := 1; else is_committed := 0; end if; " +
  "END; " +
  "BEGIN GET_LTXID_OUTCOME_WRAPPER(?,?); END;";
/**
* getTransactionOutcome returns true if the LTXID committed or false otherwise.
* note that this particular version is not considering user call completion
*/
boolean getTransactionOutcome(Connection conn, LogicalTransactionId ltxid)
throws SQLException {
  boolean committed = false;
  CallableStatement cstmt = null;
  try {
    cstmt = conn.prepareCall(GET_LTXID_OUTCOME_WRAPPER);
    cstmt.setObject(1, ltxid); // use this starting in 12.1.0.2
    cstmt.registerOutParameter(2, OracleTypes.BIT);
    cstmt.execute();
    committed = cstmt.getBoolean(2);
  } catch (SQLException sqlexc) {
    throw sqlexc;
  } finally {
    if (cstmt != null)
      cstmt.close();
  }
  return committed;
}
```

サーバー側の計画メンテナンスのコマンド例

ノート:

- これらのコマンドをスクリプトで使用している場合は、`wait = yes` を含めると便利です。
- パラメータ`-force -failover` により、各サービスで構成された他の使用可能なインスタンスでサービスが開始されます。
- 詳細は、『Oracle Real Application Clusters 管理およびデプロイメント・ガイド』のメンテナンスのためのサービスのグループの管理を参照してください。

関連付けられたすべてのサービスの構成済`-drain_timeout`パラメータと`-stopoption`パラメータを使用して、ノード(node1)上のすべてのインスタンスを停止します。

```
srvctl stop instance -db myDB -node node1 -force -failover
 -role primary
```

関連付けられたすべてのサービスの構成済`-drain_timeout`パラメータと`-stopoption`パラメータを使用して、1つのインスタンス(inst1)を停止します

```
srvctl stop instance -db myDB -instance inst1 -force -failover
 -role primary
```

関連付けられたサービスに対して構成されたパラメータをオーバーライドする、明示的なドレイン・パラメータを持つすべてのインスタンスを停止します。

```
srvctl stop instance -db db_name -node node_name
 -stopoption IMMEDIATE -drain_timeout <#> -force -failover
```

明示的なドレイン・パラメータを使用して、サービスを停止します。

```
srvctl stop service -db db_name -service service_name
 -instance instance_name -drain_timeout <#> -stopoption IMMEDIATE
 -force -failover
```

5分間のドレイン・タイムアウトおよびIMMEDIATE停止オプションを使用して、inst1 (特定のインスタンス)という名前のインスタンスで、GOLDという名前のサービスを停止します。

```
srvctl stop service -db myDB -service GOLD -instance inst1
 -drain_timeout 300 -stopoption IMMEDIATE -force -failover
```

明示的なドレイン・パラメータを使用して、Data Guardインスタンスを停止します。

```
srvctl stop instance -db db_name -node node_name
 -stopoption IMMEDIATE -drain_timeout <#> -force -failover
 -role primary
```

データベース、ノードまたはPDB別に、すべてのサービスを再配置します。

```
srvctl relocate service -database db_unique_name
  -pdb pluggable_database
 {-oldinst old_inst_name [-newinst new_inst_name] |
                          -currentnode current_node
                         [-targetnode target_node]}
 -drain_timeout timeout -stopoption stop_option -force
srvctl relocate service -database db_unique_name
```

```
 -oldinst old_inst_name [-newinst new_inst_name]
 -drain_timeout timeout -stopoption stop_option
 -force
srvctl relocate service -database db_unique_name
 -currentnode current_node [-targetnode target_node]
 -drain_timeout timeout -stopoption stop_option
 -force
```

Data Guard Brokerを使用して、待機タイムアウトが60秒のData Guardセカンダリ・サイトにスイッチオーバーします。

```
SWITCHOVER TO dg_south WAIT 60
```

Data Guard Brokerを使用して、サービスからの待機タイムアウトでData Guardセカンダリ・サイトにスイッチオーバーします。

```
SWITCHOVER TO dg_south WAIT
```

第VIII部 Oracle Multitenantのベスト・プラクティス

29 Oracle Multitenantのベスト・プラクティスの概要

Oracle Multitenantは、データベース統合のためのOracleの戦略的製品です。

Oracle Multitenantアーキテクチャには、次のような利点があります。

- 同じコンテナ・データベース(CDB)に格納されている個々のプラガブル・データベース(PDB)間のアクセス分離
- 多くのPDBを含む1つのCDBのみを単純に管理することで、多数のデータベースを管理する機能。CDBのバックアップ、CDBソフトウェアの更新、または障害時リカバリのためのスタンバイCDBの設定によって、多くの独立したデータベースに同じ管理ステップを適用するかわりに1つのCDBを管理することで、本質的に複雑さとステップが削減されます。管理タスク、ステップおよびエラーを削減します。
- リソース制限の柔軟な設定(メモリー、I/O、PDBレベルごとなど)による、CAPEXを削減するためのシステム・リソースの共有
- 単一のPDBを別のコンテナに再配置し、そのPDBのみをアップグレードするなど、個々のPDBで操作できる柔軟性
- 迅速なクローニングとプロビジョニング
- Oracle RACとの緊密な統合

次の表に、様々なOracle Multitenant構成および運用のベスト・プラクティスを示します。

表29-1 Oracle Multitenantの構成および運用のベスト・プラクティス

ユースケース	ベスト・プラクティス
プラガブル・データベース(PDB)構成	すべての Oracle RDBMS リリース 12c リリース 2 (12.2)から 21c では、ローカル UNDO モードで CDB を構成します Undo Modes in 12.2 Multitenant Databases - Local and Shared Modes (Doc ID 2169828.1)を参照してください
PDB サービス管理	Oracle Clusterware を使用する Oracle データベース(Oracle RAC や、Oracle Clusterware がインストールされている単一インスタンス・データベースなど)の必須の MAA ベスト・プラクティス 1. PDB のデフォルト・サービス、SAVED STATE (再配置操作時を除く)またはデータベース・トリガーを使用してロールベースのサービスを管理しないでください。 2. アプリケーション・サービスには、PDB ごとにクラスタウェア管理の個別サービスを使用し、そのアプリケーション・サービスを利用してデータベースに接続します。 3. クラスタウェア管理アプリケーション・サービスを定義する場合は、起動する PDB とサービス、およびどの RAC インスタンスとデータベース・ロール内かを定義します。 4. Data Guard の場合、ロールを各クラスタウェア管理サービスに割り当てることで、常にロールベースのサービスを使用します。 前述のプラクティスが適用されると、PDB のオープンおよび Data Guard ロールの遷移中に予測可能なサービス管理が可能になります。これにより、アプリケーション・サービスの可用性が向上し、アプリケーション・エラーが回避されます。

ユースケース	ベスト・プラクティス
	MAA 推奨の Oracle クラスタウェア設定のない単一インスタンス・データベースの場合、次のプラクティスに従います。 1. PDB のデフォルト・サービスは使用しないでください。 2. アプリケーション・サービスには、PDB ごとに個別サービスを使用し、そのアプリケーション・サービスを利用してデータベースに接続します。 3. Data Guard 以外の場合は、SAVED 状態のみを使用して PDB をオープンし、明示的なアプリケーション・サービスを起動します。または、Data Guard の場合は、AFTER STARTUP データベース・トリガーのみを使用して、プライマリ、READ ONLY またはスナップショット・スタンバイ・データベース・ロールに応じて起動する必要があるアプリケーション・サービスをプログラム的に管理します。 Best Practices for Pluggable Database End User and Application Connection and Open on Database Startup (Doc ID 2833029.1)を参照してください
Oracle Multitenant での Data Guard の使用	My Oracle Support の次のノートでは、Oracle Data Guard 構成で Oracle Multitenant を使用する場合の、運用のベスト・プラクティスに関する推奨事項について説明しています ● Data Guard Impact on Oracle Multitenant Environments (Doc ID 2049127.1) ● ユースケース: PDB 作成、PDB 移行および PDB クローニング時の Oracle Multitenant での遅延 PDB リカバリおよび STANDBYS=NONE 機能の使用(ドキュメント ID 1916648.1) ● PDB 移行の Data Guard 構成のプライマリ・データベースに PDB を接続する場合のソース・スタンバイ・データベース・ファイルの再利用(ドキュメント ID 2273829.1) ● PDB 移行の Data Guard 構成のプライマリ・データベースに非 CDB を PDB として接続する場合のソース・スタンバイ・データベース・ファイルの再利用(ドキュメント ID 2273304.1) ● PDB リモート・クローンまたは PDB プラグインの PDB リモート・クローンまたは PDB プラグインを実行する場合の standby_pdb_source_file_dblink および standby_pdb_source_file_directory を使用したスタンバイ・データベースのメンテナンス(ドキュメント ID 2274735.1) ● サブセット・スタンバイをサポートするための、Data Guard Subset Standby を使用したパラメータ enabled_pdbs_on_standby および STANDBYS オプション(ドキュメント ID 2417018.1)

ユースケース	ベスト・プラクティス
Data Guard: PDB スイッチオーバーおよびフェイルオーバーのユースケース	Data Guard Broker を使用した新しい Data Guard 構成へのプラガブル・データベースの移行ドキュメント 2887844.1 Data Guard 環境での PDB フェイルオーバー: Data Guard Broker を使用した、障害が発生した単一の PDB のスタンバイ・データベースからの切断および新しいコンテナへの接続または単一 PDB の新規コンテナへの移行ドキュメント 2088201.1
PDB 移行	My Oracle Support の次のノートでは、最小限の停止時間で様々なタイプの PDB を移行するための運用のベスト・プラクティスについて説明しています: ● ターゲットが Exadata プラットフォームまたはクラウドの場合は、ゼロ・ダウンタイム移行を使用します。 https://www.oracle.com/database/technologies/rac/zdm.html を参照してください ● Step by Step Process of Migrating non-CDBs and PDBs Using ASM for File Storage (Doc ID 1576755.1) ● Cloning a Pluggable Database from an RMAN Container Database Backup (Doc ID 2042607.1) ● Data Guard Broker を使用した新しい Data Guard 構成へのプラガブル・データベースの移行ドキュメント 2887844.1
PDB 再配置	● Using PDB Relocation to Upgrade an Individual PDB (Doc ID 2771716.1) ● Using PDB Relocation to Move a Single PDB to Another CDB Without Upgrade (Doc ID 2771737.1)
PDB リソース管理	My Oracle Support の次のノートでは、Oracle Multitenant リソース管理の運用ユースケースについて説明しています。 How to Control and Monitor the Memory Usage (Both SGA and PGA) Among the PDBs in Mutitenant Database- 12.2 New Feature (Doc ID 2170772.1)

Oracle Multitenant MAAソリューションを使用すると、様々なMAAソリューションの利点を得ながら、管理とシステム・リソースを節約できます。次の表に、様々な計画外停止および計画メンテナンス・アクティビティについて、ゼロおよびゼロに近い停止時間とデータ損失を示します。

表29-2 計画外停止

計画外停止	ソリューションの主な機能	RTO	RPO
リカバリ可能なノードまたはインスタンスの障害	Real Application Cluster (RAC) アプリケーション・コンティニュイティ(AC)	秒	ゼロ
データベース、クラスタおよびサイトの障害	Active Data Guard ファスト・スタート・フェイルオーバー	<2 分	ゼロまたは数秒
データ破損	物理破損の自動ブロック修復を含む Active Data Guard	ゼロ	ゼロ
PDB のリカバリ不能な障害または障害の発生した PDB	Data Guard の移行コマンドを使用した PDB フェイルオーバー 同じクラスタ上の別のターゲット CDB が必要です PDB Failover in a Data Guard environment: Using Data Guard Broker to Unplug a Single Failed PDB from a Standby Database and Plugging into a New Container or Migrate a Single PDB into a New Container (Doc ID 2088201.1)を参照	<2 分	ゼロまたは数秒
アクティブ・レプリカへの PDB フェイルオーバー	オプション 1: プライマリ CDB とスタンバイ CDB の Data Guard アーキテクチャを使用した CDB 全体のフェイルオーバー オプション 2: Oracle GoldenGate を使用した PDB レプリカの作成。異なる CDB の PDB レプリカを使用して PDB アクティブ・フェイルオーバーを実行します。 アプリケーション・フェイルオーバーについては、グローバル・データ・サービスおよび MAA ソリューションの継続的サービスのためのアプリケーション・チェックリストのプラクティスを使用します	ゼロの可能性	ゼロまたは数秒

表29-3 計画メンテナンス

計画停止時間	ソリューション	RTO
ソフトウェアおよびハードウェアの更新	Real Application Cluster (RAC) MAA ソリューションの継続的サービスのためのアプリケーション・チェックリスト	ゼロ
CDB 全体に対するデータベースのメジャー・アップグレード	Active Data Guard DBMS_ROLLING	秒

計画停止時間	ソリューション	RTO
CDB 内の単一の PDB に対するデータベースのメジャー・アップグレード	PDB 再配置およびアップグレード Using PDB Relocation to Upgrade an Individual PDB (Doc ID 2771716.1)を参照	分
リモート CDB への移行	PDB の再配置 Using PDB Relocation to Move a Single PDB to Another CDB Without Upgrade (Doc ID 2771737.1)を参照	分
リモート CDB への移行(論理移行)	Data Pump および Oracle GoldenGate またはゼロ・ダウンタイム移行	ゼロの可能性

30 マルチテナント構成でのPDBスイッチオーバーおよびフェイルオーバー

ここで説明するユースケースでは、多数のPDBがあるコンテナ・データベース(CDB)を含むOracle Data Guard構成に対して、単一のプラガブル・データベース(PDB)フェイルオーバーおよびスイッチオーバーを設定する方法を示します。

Oracle Multitenantと、複数のプラガブル・データベース(PDB)をコンテナ・データベース(CDB)に統合する機能により、同様のSLAと計画メンテナンス要件を持つ多数のデータベースを、より少ないシステム・リソースで、運用投資を抑えながら管理できます。Oracle MultitenantとそのCDB/PDBテクノロジをOracleのリソース管理で活用することで、全体的なハードウェア・コストと運用コストを効果的に削減できます。

計画とサイズ設定は、同じCDBに統合するデータベースを決定するための重要な前提条件です。HAおよびDRの保護を必要とするミッション・クリティカルなデータベースの場合、および計画メンテナンスの停止時間を最小限に抑える場合は、次のことが重要です

- PDBごとに十分なリソースを確保してレスポンスおよびスループットの期待値内で実行するためのサイズ設定およびリソース管理の活用
- 同じ計画メンテナンス要件とスケジュールを持つターゲットPDBデータベース
- CDB、クラスタまたはサイトの障害など、計画外停止が発生した場合にすべて同じCDBスタンバイにフェイルオーバーできるターゲットPDBデータベース

PDBおよびそれらに関連付けられたアプリケーション・サービスを追加すると、Data Guardのフェイルオーバーおよびスイッチオーバー時間が長くなる可能性があります。Data Guardのスイッチオーバーおよびフェイルオーバーの時間を削減する場合は、Data Guardを使用したミッション・クリティカルなゴールドCDBに対して、CDB当たり25未満のPDBを使用することをお薦めします。

ミッション・クリティカルなデータベースと開発/テスト・データベースを異なるCDBに分割することが重要です。たとえば、スタンバイがあるミッション・クリティカルなゴールドCDBには、同じHA/DR要件を持つPDBが5つのみ存在し、十分なシステム・リソース・ヘッドルームを持つようにサイズ設定できますが、スタンバイを持つ重要なCDBには開発、UATおよびアプリケーションのテストの目的で100個のPDBを含めることができ、コストを削減するために一部のレベルのオーバー・サブスクリプションを設定できます。マルチテナントMAAおよびマルチテナントのベスト・プラクティスの詳細は、「Oracle Multitenantのベスト・プラクティスの概要」を参照してください。

このユースケースでは、CDB Data Guardの完全なスイッチオーバーおよびフェイルオーバー操作が不可能な例外の概要およびステップ・バイ・ステップの手順を示します。PDBのフェイルオーバーおよびスイッチオーバー・ステップでは、1つのPDBへのData Guardロール・トランジションを分離して、リカバリ時間目標(RTO)を5分未満、リカバリ・ポイント目標(RPOまたはデータ損失)をゼロまたはほぼゼロにすることができます。

Oracle RDBMS 19c (19.15)以降では、Data Guard Brokerコマンドライン・インタフェース(DGMGRL)を使用して、PDBをData Guard構成間で移行できます。ブローカを使用すると、同じCDB内の他のPDBに影響を与えることなく、PDB障害時リカバリ(DR)およびスイッチオーバー操作を分離して開始できます。

Data Guard Brokerの移行では、次の主要なユースケースについて説明します。

- PDBスイッチオーバーのユースケース - Data Guard CDBの既存のPDBに影響を与えずにPDBスイッチオーバー操作を起動する計画メンテナンスDR検証
- PDBフェイルオーバーのユースケース - Data Guard CDB内の既存のPDBに影響を与えずにPDBフェイルオーバーを起動する計画外停止DR

ノート:

CDB 内の他の PDB に影響を与えずに、アップグレードが不要な場合に単一の PDB を再配置するには、PDB 再配置を使用したアップグレードなしの別の CDB への単一 PDB の移動(ドキュメント ID 2771737.1)を参照してください。CDB 内の他の PDB に影響を与えずに、アップグレードが必要な単一の PDB を再配置するには、次を参照してください。

PDBスイッチオーバーのユースケース

このPDBスイッチオーバーまたはDRテストのユースケースでは、PDBはOracle Data Guardで保護されたCDBから別のData Guardで保護されたCDBに移行されます。

このユースケースの一部として、ソースCDBのプライマリ・データベースとスタンバイ・データベースの両方のPDBのファイルは、宛先CDBのそれぞれのプライマリ・データベースとスタンバイ・データベースで直接使用されます。

ソースCDBには複数のPDBが含まれていますが、他のPDBは影響を受け入れられないため、1つのPDBに対してのみロール・トランジション・テストを実行します。移行を開始する前に、2番目のCDBを作成し、ソースCDBと同じデータベース・オプションを持つ必要があります。宛先CDBもData Guard構成にありますが、最初はPDBが含まれていません。対応する2つのプライマリ・データベースとスタンバイ・データベースは同じストレージを共有し、データ・ファイルの移動は実行されません。

前提条件

使用している環境が、ユースケースの前提条件を満たしていることを確認します。

Oracle Data Guard Broker CLI (DGMGRL)では、単一のフィジカル・スタンバイ・データベースを使用した構成のメンテナンスがサポートされます。

ここで説明する方法を使用して、移行するPDB (ソース)に対して、プライマリ・データベースとスタンバイ・データベースの両方のデータ・ファイルがソースにある既存のディレクトリ構造に物理的に残り、宛先CDBとそのスタンバイ・データベースによって消費されます。

- 必要なOracleパッチ/バージョン
  - Oracle RDBMS 19c (19.15)以降
  - スイッチオーバー・プロセスを管理するブローカ機能を提供するソースおよび宛先のCDB RDBMS Oracleホームにインストールされているパッチ33358233。Oracle RDBMS 19c (19.18)以降にパッチを適用する必要はありません。これにはパッチが含まれています。
  - ソースおよび宛先CDB RDBMS Oracleホームにインストールされているパッチ34904997では、PDBフェイルオーバー・ユースケースの実行後にPDBを元の構成に移行する機能が提供されます。
- 構成
  - `DB_CREATE_FILE_DEST = ASM_Disk_Group`
  - `DB_FILE_NAME_CONVERT=""`
  - `STANDBY_FILE_MANAGEMENT=AUTO`
  - ソースと宛先のスタンバイCDBが同じクラスタで実行されている必要があります
  - ソースと宛先のプライマリCDBは同じOracleホームから実行する必要があり、ソースと宛先のスタンバイCDBは同じOracleホームから実行する必要があります

- ソースと宛先のプライマリCDBが同じホストで実行されている必要があります
- ソースと宛先のプライマリ・データベースは同じASMディスク・グループを使用し、ソースと宛先のスタンバイ・データベースは同じASMディスク・グループを使用する必要があります

- 次へのアクセス権が必要です
  - 宛先CDB sysdbaユーザーのパスワード
  - スタンバイ・サイトのASM sysasmユーザーのパスワード(別名の管理用)
  - TDEが有効な場合の宛先CDBの透過的データ暗号化(TDE)キーストアのパスワード

ノート:

PDB スナップショット・クローンおよび PDB スナップショット・クローンの親は、移行またはフェイルオーバーではサポートされていません。

複数のフィジカル・スタンバイ・データベースがある宛先プライマリ・データベースの場合、Data Guard 構成のプライマリ・データベースに PDB を接続する場合のソース・スタンバイ・データベース・ファイルの再使用(ドキュメント ID 2273829.1)の手動ステップを使用するか、スタンバイ・データベースの ENABLED_PDBS_ON_STANDBY 初期化パラメータを使用して、このプロセスによって管理されるスタンバイを制限する必要があります。
ENABLED_PDBS_ON_STANDBY の使用の詳細は、『*Oracle Data Guard* 概要および管理』の CDB のフィジカル・スタンバイの作成を参照してください。

移行されたソース PDB の既存の ASM 別名は、移行プロセス中にブローカによって管理されます。ASM ではファイルごとに 1 つの別名のみが許可されるため、別の場所を指す既存の別名を削除し、正しい場所に新しい別名を作成する必要があります。

PDBスイッチオーバーの構成

次のステップで、「DRテスト」のPDBスイッチオーバーのユースケースを構成します。

次のステップに含まれるサンプル・コマンドでは、次のCDB名とPDB名を使用します。

- CDB100 (ソースCDB)
  - PDB001、PDB002、PDB003を含みます。PDB001はスイッチオーバー用に構成されます
- CDB100_STBY (ソース・スタンバイCDB)
- CDB200 (宛先CDB)
- CDB200_STBY (宛先スタンバイCDB)

ステップ1: ソース・データベースでのPDBクラスタウェア管理サービスの抽出

CRSに追加されたソースPDB用に作成されたアプリケーションおよびエンド・ユーザー・サービスを決定します。

データベースに格納されていないデータベース・ロールなどの特定のサービス属性があるため、SRVCTL CONFIG SERVICEを使用して詳細属性をCRSから取得する必要があります。

1. プライマリPDBからサービス名を取得します(この例ではPDB001)。

```
PRIMARY_HOST $ sqlplus sys@cdb100 as sysdba
SQL> alter session set container=pdb001;
```

```
SQL> select name from dba_services;
```

2. 返されるサービス名ごとに、DATABASE_ROLEを含む構成を取得します。

```
PRIMARY_HOST $ srvctl config service -db cdb100 -s SERVICE_NAME
```

ステップ2: 空のターゲット・データベースの作成

PDBの宛先となるソースCDB (CDB100)と同じクラスタに、空のCDB (この例ではCDB200)を作成します。

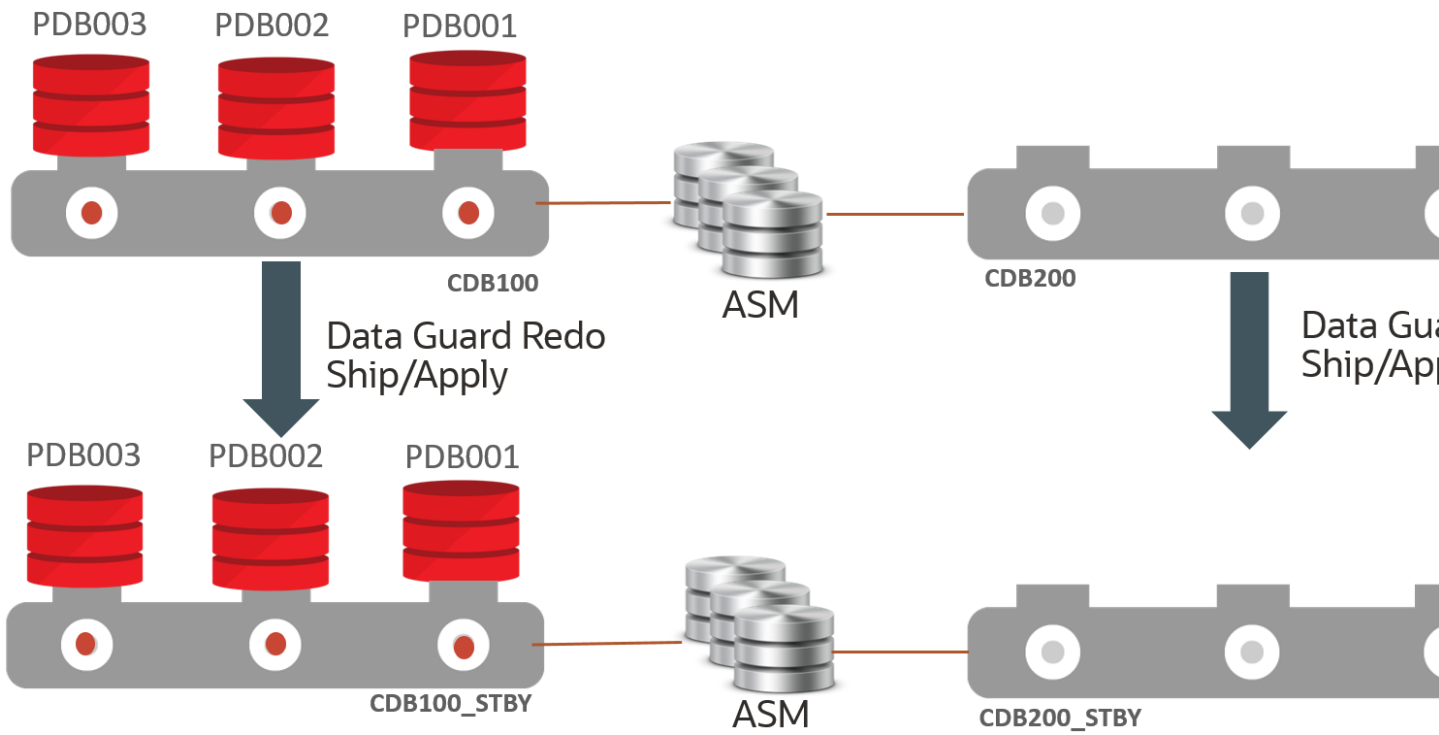
テスト期間中のPDBの使用をサポートするために、このCDBにリソースを割り当てます。

ステップ3: ターゲット・スタンバイ・データベースの作成

ターゲットCDBでOracle Data Guardを有効にして、スタンバイ・データベース(CDB200_STBY)を作成します。

スタンバイ・データベースは、ソース・スタンバイ・データベースと同じクラスタに存在する必要があります。

構成は次の図のようになります。



ステップ4: PDBの移行

PDB (PDB001)をソースCDB (CDB100)から宛先CDB (CDB200)に移行します。

1. Oracle Data Guard Brokerコマンドライン(DGMGRL)を使用して、ソース・プライマリ・データベースへの接続を開始します。

このセッションは、ソース・プライマリCDBと宛先プライマリCDBの両方のインスタンスを含むホストで実行する必要があります。セッションはsysdbaユーザーで開始する必要があります。

ブローカCLIは、プライマリCDB環境のコマンドラインから実行し、ソース・プライマリCDBへの接続中に実行する必要があります。TNS別名を使用してソース・プライマリに接続する場合は、ブローカCLIセッションと同じホストで実行されているソース・プライマリ・インスタンスに接続する必要があります。

ブローカCLIを実行する際のホストおよび環境設定では、次のSQL*Net別名にアクセスできる必要があります。

- 宛先プライマリCDB - 作成されるPDB切断マニフェスト・ファイルにプラグイン操作でアクセスできるように、この別名で、ブローカCLIセッション/ソースのプライマリ・データベース・インスタンスと同じホスト上にある宛先プライマリ・インスタンスに接続する必要があります。
- 宛先スタンバイCDB。これはスタンバイ環境の任意のインスタンスに接続できます。
- スタンバイ・サイトのASMインスタンス。これはスタンバイ環境の任意のインスタンスに接続できます。

```
PRIMARY_HOST1 $ dgmgrl sys@cdb100_prim_inst1 as sysdba
```

このセッションは、ソース・プライマリCDBと宛先プライマリCDBの両方のインスタンスを含み、sysdbaユーザーに接続されているホストで実行する必要があります。SCANではなく特定のホスト/インスタンスの組合せを使用して、目的のインスタンスに接続されるようにします。

2. DGMGRL MIGRATE PLUGGABLE DATABASEコマンドを実行します。

   STANDBY FILESキーワードが必要です。

   完全な出力を使用する例は「コマンドの完全な例と出力」を、コマンドライン引数の詳細はMIGRATE PLUGGABLE DATABASEを参照してください。

   - TDEを使用しないサンプル・コマンドの例:

   ```
   DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200
    USING '/tmp/PDB001.xml' CONNECT AS sys/password@cdb200_inst1
    STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst1
    SOURCE STANDBY CDB100_STBY DESTINATION STANDBY CDB200_STBY ;
   ```

   - TDEを使用したサンプル・コマンドの例

   ```
   DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200
    USING '/tmp/pdb001.xml' CONNECT AS sys/password@cdb200_inst1
    SECRET "some_value" KEYSTORE IDENTIFIED BY
   "destination_TDE_keystore_passwd"
    STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst1
    SOURCE STANDBY cdb100_stby DESTINATION STANDBY cdb200_stby;
   ```

コマンドが実行されると、次のようになります。

1. 資格証明および接続文字列が正しいことを確認するために、宛先データベースおよびASMインスタンスに接続します
2. 様々な事前チェックの実行 - 事前チェックが失敗した場合、コマンドは処理を停止してユーザーに制御を戻し、エラーが返され、ターゲットCDBに変更は加えられません
3. 宛先スタンバイCDBにフラッシュバック保証付きリストア・ポイントを作成します。これには、REDO適用の短い停止と開始が必要です
4. ソース・プライマリのPDBをクローズします
5. ソース・プライマリのPDBを切断します。TDEが使用中の場合は、切断操作の一部として生成されたマニフェスト・ファイ

ルにキーが含まれます

6. ソース・プライマリ・データベースのPDBをKEEP DATAFILES句で削除し、ソース・ファイルが削除されないようにします
7. PDBの削除REDOがソース・スタンバイ・データベースに適用されるのを待機します。ファイルがソース・スタンバイ・データベースによって所有されているため、削除REDOが適用されるまで待機する必要があります

   このコマンドは、最大でTIMEOUT分(デフォルトは10)待機します。REDOが適用されていない場合、コマンドは失敗し、手動でプロセスを完了する必要があります。

8. スタンバイのPDBファイルのASM別名を管理し、既存の別名を削除して、必要に応じて新しい別名を作成します。スタンバイ・ファイルが正しい場所にすでに存在する場合は、PDBのスタンバイ・コピーのすべての別名が削除されます
9. PDBを宛先のプライマリCDBに接続します。TDEが使用されている場合、キーはプラグインの一部として宛先のプライマリ・キーストアにインポートされます
10. プラグイン操作のREDOを宛先CDBに送信し適用します。この宛先CDBは、作成された別名を使用して(必要な場合)それらのファイルにアクセスし、それらをスタンバイ・データベースに組み込みます
11. REDO適用を使用してスタンバイ・ファイルが宛先スタンバイに追加されたことを検証します
12. 宛先プライマリ・データベースでPDBをオープンします
13. REDO適用を停止します
14. 宛先スタンバイ・データベースからフラッシュバック保証付きリストア・ポイントを削除します
15. TDEが有効な場合、REDO適用は停止したままになります。TDEが有効になっていない場合、REDO適用は再開されます

ステップ5: 移行後 - オプションのTDE構成ステップおよび適用の再起動

TDEが使用中の場合、新しいTDEキーを管理できるように、宛先スタンバイ(CDB200_STBY)のブローカMIGRATE PLUGGABLE DATABASE操作によってREDO適用が停止されます。宛先プライマリ(CDB200)のキーストアを宛先スタンバイ・キーストアにコピーし、REDO適用を開始します。

```
SOURCE_HOST $ scp DESTINATION_PRIMARY_WALLET_LOCATION/*>
 DESTINATION_HOST:DESTINATION_STANDBY_WALLET_LOCATION/
$ dgmgrl sys/password@CDB200 as sysdba
DGMGRL> edit database cdb200_stby set state='APPLY-ON';
```

ステップ6: 移行後 - サービスの有効化

PDBのアプリケーション・サービスをCluster Ready Services (CRS)に追加し、PDBに関連付けて宛先CDBでデータベース・ロールを修正し、対応するサービスをソースCDBから削除します。

1. プライマリ環境とスタンバイ環境の両方のサービスごとに、次を実行します:

```
PRIMARY_HOST $ srvctl add service -db cdb200 -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….
STANDBY_HOST $ srvctl add service -db cdb200_stby -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….
PRIMARY_HOST $ srvctl remove service -db cdb100 -s SERVICE_NAME
STANDBY_HOST $ srvctl remove service -db cdb100_stby -s SERVICE_NAME
```

2. 適切なデータベース・ロールに必要なサービスを開始します。

   a. 各PRIMARYロールのデータベース・サービスを開始します

```
PRIMARY_HOST $ srvctl start service -db cdb200 -s SERVICE_NAME
```

b. 各PHYSICAL_STANDBYロールのデータベース・サービスを開始します。

```
STANDBY_HOST $ srvctl start service -db cdb200_stby -s SERVICE_NAME
```

ステップ7: PDBロール・トランジション・テストの実行

移行の完了後、宛先CDB (CDB200)のPDBで必要なOracle Data Guardロール・トランジションまたはDRテストを実行できるようになります。ソースCDB (CDB100)内の他のPDBは影響を受けません。

また、ソースCDBと宛先CDBの両方に対するData Guardの利点(DR準備状況、データ破損用の自動ブロック・メディア・リカバリ、バインドされたリカバリ時間へのファスト・スタート・フェイルオーバー、論理破損に対する書込みの欠落検出、スタンバイへの読取りのオフロードによるスケーリングの削減、プライマリの影響の削減など)を維持します。

- DGMGRLを使用して宛先CDBに接続し、スイッチオーバーを実行します。

```
$ dgmgrl sys@cdb200 as sysdba
DGMGRL> switchover to CDB200_STBY;
```

PDBのDRテストを引き続き実行できます。

PDBのDRテストが完了したら、元に戻してPDBを元のソースCDBに戻すことができます。

- DGMGRLを使用して宛先CDB (CDB200)に接続し、スイッチバック操作を実行します。

```
$ dgmgrl sys@cdb200 as sysdba
DGMGRL> switchover to CDB200;
```

ステップ8: PDBを元のCDBに戻す

移行およびロール・トランジション・テスト後に、このCDBの元の構成に戻し、PDBを元のData Guard構成に戻し、スタンバイ・データベース・ファイルを再度自動的にメンテナンスします。Data Guard Brokerの移行では、移行プロセスの一部として削除または作成する必要がある別名が処理されます。

完全な出力の例については、「コマンドの完全な例と出力」を参照してください。

- Data Guard Brokerコマンドライン(DGMGRL)を使用して、ソース・プライマリへの接続を開始します

```
$ dgmgrl
DGMGRL> connect sys/@cdb200_inst1 as sysdba
```

  - TDEを使用しないコマンド

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB100
 USING ‘/tmp/PDB001_back.xml’ CONNECT AS sys/password@cdb100_inst1
 STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst
 SOURCE STANDBY CDB200_STBY DESTINATION STANDBY CDB100_STBY ;
```

  - TDEを使用したコマンド

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB100
 USING ‘/tmp/PDB001_back.xml’ CONNECT AS sys/password@cdb100_inst1
  SECRET "some_value" KEYSTORE
 IDENTIFIED BY "destination_TDE_keystore_passwd"
 STANDBY FILES sys/standby_asm_sys_password@standby_asm_inst
 SOURCE STANDBY CDB200_STBY DESTINATION STANDBY CDB100_STBY ;
```

ステップ9: 移行後 - サービスの有効化

PDBのアプリケーション・サービスをCluster Ready Services (CRS)に追加し、PDBに関連付けて宛先CDB (CDB100)でデータベース・ロールを修正し、対応するサービスをソースCDB (CDB200)から削除します。

- プライマリ環境とスタンバイ環境の両方のサービスごとに、次を実行します:

```
PRIMARY_HOST $ srvctl add service -db cdb100 -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….
STANDBY_HOST $ srvctl add service -db cdb100_stby -s SERVICE_NAME
 -pdb pdb001 -role [PRIMARY|PHYSICAL_STANDBY]….
<PRIMARY_HOST>PRIMARY_HOST $ srvctl remove service -db cdb200 -s SERVICE_NAME
STANDBY_HOST $ srvctl remove service -db cdb200_stby -s SERVICE_NAME
```

- 適切なデータベース・ロールに必要なサービスを開始します。

  1. 各PRIMARYロールのデータベース・サービスを開始します。

     ```
     PRIMARY_HOST $ srvctl start service -db cdb100 -s SERVICE_NAME
     ```

  2. 各PHYSICAL_STANDBYロールのデータベース・サービスを開始します。

     ```
     STANDBY_HOST $ srvctl start service -db cdb100_stby -s SERVICE_NAME
     ```

PDBフェイルオーバーのユースケース

CDB、クラスタまたはサイト障害を含む実際の障害では、常に完全なCDB Data Guardフェイルオーバー操作を利用して停止時間をバインドし、潜在的なデータ損失を減らし、管理ステップを削減する必要があるため、これは非常にまれなユースケースです。

論理的またはデータの破損が広範囲に及ぶ場合や、データベースのハングが拡大しても、CDB Data Guardロール・トランジション操作を発行する方が効率的です。ソース環境が疑わしい場合があり、根本原因の分析に時間がかかる可能性があるためです。

PDBのフェイルオーバー操作が有用なのはいつですか。アプリケーションでデータの整合性などの致命的なエラーや破損エラーが発生した場合、または単に(システム・リソースによるものではなく)適切に実行されていない場合に、PDBフェイルオーバーが有用である場合があります。ソースCDBおよび対応するPDBがまだ正常に実行されていて、スタンバイが障害の発生したターゲットPDBのエラーを受信しなかった場合、ソース・プライマリCDBの他のPDBに影響を与えることなく、障害の発生したターゲットPDBのみをスタンバイからフェイルオーバーできます。

次のプロセスでは、スタンバイの正常なPDBをソースCDBスタンバイ(CDB100_STBY)から空の宛先CDB(CDB200)に移行する障害の発生したPDBのPDBフェイルオーバーを設定する方法について説明します。移行を開始する前に、宛先CDBを作成し、ソース・スタンバイCDBと同じデータベース・オプションを持つ必要があります。宛先CDBにはPDBが含まれていません。ソースCDBと宛先CDBは同じストレージを共有し、データ・ファイルの移動は実行されません。

前提条件

使用している環境が、ユースケースの前提条件を満たしていることを確認します。

前述のPDBスイッチオーバーのユースケースにリストされている前提条件に加えて、フェイルオーバーのための次の前提条件が存在します。

- Oracleでは、移行プロセスを開始する前に、PDBにアクセスしているプライマリとスタンバイの両方のサービスを停止することをお薦めします。

  DGMGRL MIGRATE PLUGGABLE DATABASEコマンドを実行する前にPDBがプライマリでクローズされていない場合、データが失われることを示すエラーが返されます。プライマリでPDBをクローズすると、この問題が解決されます。PDBへの既存のすべての接続は、移行の一部として終了します。

宛先CDBがすでに存在し、スタンバイ・サイトで正しくパッチが適用されている場合、PDBを移動するプロセス全体を15分未満で完了できます。

追加の考慮事項

次のステップでは、ソースCDBデータベース(移行の場合はプライマリ、フェイルオーバーの場合はスタンバイ)と宛先CDBデータベースが同じストレージにアクセスできることを前提としているため、データ・ファイルのコピーは不要です。

- フェイルオーバー操作には、ソースCDBスタンバイにOracle Active Data Guardが必要です。
- ソースCDBと同じクラスタ上のPDBの宛先となる空のCDBを作成します。
- 移行を実行する前に、PDBのTEMPファイルがソースCDBスタンバイにすでに作成されていることを確認します。
- 宛先CDBが新しいOracleリリースである場合、PDBは接続されますが、移行後のタスクとして手動アップグレードを実行できるようにクローズされたままになります。
- 処理が完了したら、ソース・データベースから残りのデータベース・ファイルをクリーンアップすることが必要になる場合があります。
- 宛先CDBでの接続操作はSTANDBYS=NONEで実行されるため、移行の完了時に任意のスタンバイ・データベースでリカバリを手動で有効にする必要があります。PDBのリカバリを有効化するステップは、Oracle Multitenantでの遅延PDBリカバリおよびSTANDBYS=NONE機能の使用(ドキュメントID 1916648.1)を参照してください。

PDBフェイルオーバーの構成

DR PDBフェイルオーバーのユースケースは、次のステップで構成します。

このユースケースでは、トポロジの例に3つのPDB (PDB001、PDB002、PDB003)を持つソース・プライマリCDB100があります。CDB100には、Data Guardフィジカル・スタンバイ(CDB100_STBY)もあります。

スタンバイCDBと同じ環境で、ソースPDBのうちのいずれかの新しいホストになる読取り/書込みデータベースである新しいCDB (CDB200)を作成します。

ステップ1: ソース・データベースでのPDBクラスタウェア管理サービスの抽出

CRSに追加されたソースPDB用に作成されたアプリケーションおよびエンド・ユーザー・サービスを決定します。

データベースに格納されていないデータベース・ロールなどの特定のサービス属性があるため、SRVCTL CONFIG SERVICEを使用して詳細属性をCRSから取得する必要があります。

1. プライマリPDBからサービス名を取得します(この例ではPDB002)。

```
PRIMARY_HOST $ sqlplus sys@cdb100 as sysdba
SQL> alter session set container=pdb002;
SQL> select name from dba_services;
```

2. 返されるサービス名ごとに、DATABASE_ROLEを含む構成を取得します。

```
PRIMARY_HOST $ srvctl config service -db cdb100 -s SERVICE_NAME
```

ステップ2: 空のターゲット・データベースの作成

ソース・スタンバイCDB (CDB100_STBY)と同じクラスタに、PDB (PDB002)の宛先となる空のCDB (この例ではCDB200)を作成します。

このCDBにリソースを割り当てて、PDBがこのCDBに残っている間はPDBの使用をサポートします。

ステップ3: 空のターゲット・データベースのOracle Data Guard構成の作成

Data Guard Brokerが新しいCDB (CDB200)にアクセスできるようにするには、Data Guard構成の一部である必要があります。この構成は、プライマリ・データベースのみで構成できます。

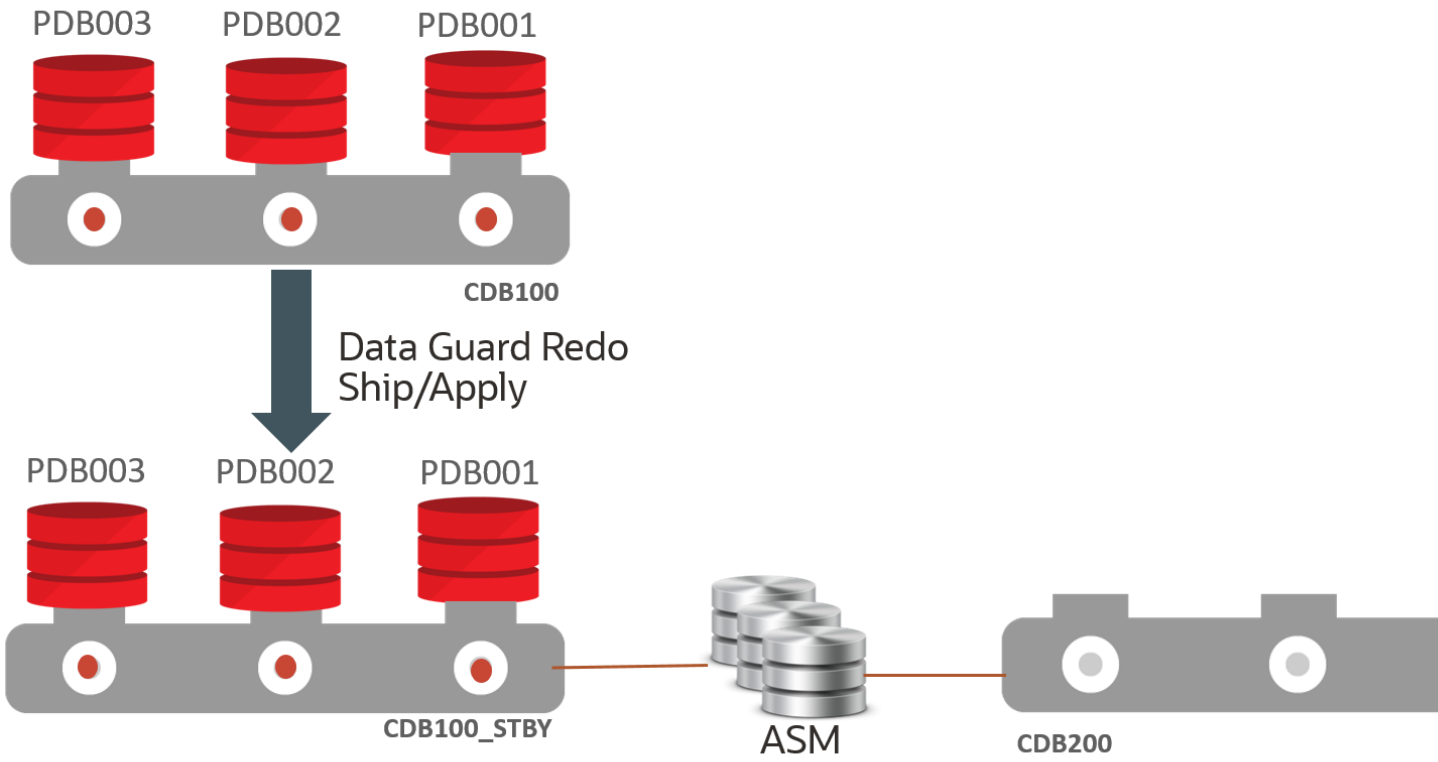
1. ブローカ用にデータベースを構成します。

```
STANDBY_HOST $ sqlplus sys@cdb200 as sysdba
SQL> alter system set dg_broker_config_file1='+DATAC1/cdb200/dg_broker_1.dat';
SQL> alter system set dg_broker_config_file2='+DATAC1/cdb200/dg_broker_2.dat';
SQL> alter system set dg_broker_start=TRUE;
```

2. 構成を作成し、データベースをプライマリとして追加します。

```
STANDBY_HOST $ dgmgrl
DGMGRL> connect sys@cdb200 as sysdba
DGMGRL> create configuration failover_dest as primary database is cdb200
 connect identifier is 'cdb200';
DGMGRL> enable configuration;
```

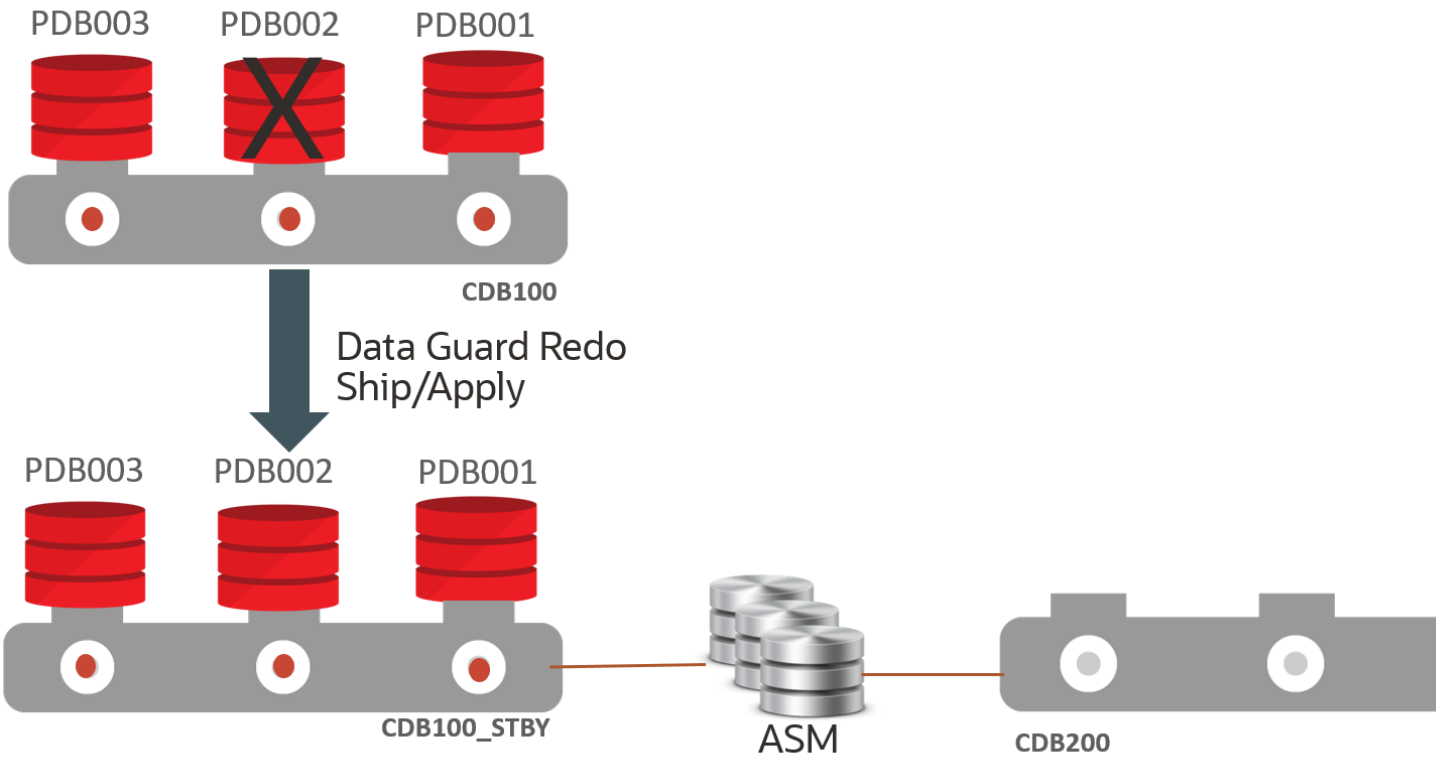
構成は次の図のようになります。



この図では、ソース・プライマリCDB (CDB100)およびすべてのPDBが正常に実行されています。ソース・スタンバイCDB (CDB100_STBY)をActive Data Guardモードで実行して、他のPDBに影響を与えずに切断操作を正常に実行できるようにする必要があります。宛先CDB (CDB200)は現在空です。

次の図に示すように、ソースのプライマリPDB (PDB002)の1つで障害が発生し、長いリカバリ期間が必要であるが、障害は他

のPDB (PDB001およびPDB003)には影響せず、ソースCDBのスタンバイはエラーなしでREDOを引き続き適用するとします。



この構成では、スタンバイ・サイト(CDB100_STBY)のPDB002のファイルを使用して宛先CDB (CDB200)に接続し、読取り/書込みアプリケーション・アクセスをリストアしてから、ソース・プライマリCDB (CDB100)から障害のあるPDB (PDB002)を削除します。ネイティブの切断には読取り/書込みCDBが必要であり、このシナリオではスタンバイから抽出しているため、これはネイティブの切断操作ではありません。

ステップ4: 失敗したPDBのサービスの停止

必須ではありませんが、移行するPDB (PDB002)に関連するソース・プライマリ・データベースとスタンバイ・データベースの両方ですべてのサービスを停止します。

次のコマンドは、CRSで定義されているすべてのサービスを停止しますが、PDBはクローズしません。

```
SOURCE_PRIMARY $ srvctl stop service -d CDB100 -pdb PDB002
SOURCE_PRIMARY $ srvctl stop service -d CDB100_STBY -pdb PDB002
```

ステップ5: スタンバイからのPDBのフェイルオーバー

スタンバイCDB (CDB100_STBY)から宛先CDB (CDB200)に、障害のあるPDB (PDB002)をフェイルオーバーします。

1. ソース構成スタンバイ・データベース(CDB100_STBY)に接続するDGMGRLセッションを開始します。

   次のようなものを使用して、ソース・スタンバイ・データベースにSYSDBAとして接続する必要があります。

   ```
   $ dgmgrl
   ```

```
DGMGRL> connect sys@cdb100_stby_inst1 as sysdba
```

2. DGMGRL MIGRATE PLUGGABLE DATABASEコマンドを実行してフェイルオーバーを実行します。

> ノート:
>
> DGMGRL FAILOVER コマンドの形式は、MIGRATE PLUGGABLE DATABASE コマンドと同じです。
>
> フェイルオーバー操作には STANDBY FILES キーワードを使用しないでください。
>
> データ損失が検出され(最初の SYSTEM 表領域スタンバイ・データ・ファイルのヘッダーの SCN がプライマリ内のファイルの対応する SCN より小さい)、IMMEDIATE が指定されていない場合、MIGRATE PLUGGABLE DATABASE コマンドは失敗します。最も一般的な理由は、プライマリ CDB の PDB がまだオープンしており、フェイルオーバーを試行する前にプライマリの PDB をクローズする必要があるためです。
>
> SCN の相違を解決するか、IMMEDIATE 句でデータ損失を受け入れる必要があります。

3. PDBをフェイルオーバーします

完全な出力の例については、「コマンドの完全な例と出力」を参照してください。

CONNECT別名は、ブローカCLIセッション/ソース・スタンバイ・データベース・インスタンスと同じホスト上にある宛先プライマリ・インスタンスに接続し、作成されるPDB切断マニフェスト・ファイルに接続操作でアクセスできるようにする必要があります。

> ノート:
>
> 次の例では、ブローカが CDB200_INST1 インスタンスに接続しようとすると、宛先 CDB (CDB200) の SYSDBA パスワードの入力を求められます。

- TDE対応でない環境の場合:

```
DGMGRL> migrate pluggable database PDB002 to container CDB200
 using '/tmp/PDB002.xml>' connect as sys@"CDB200_INST1";
```

- TDE対応の環境の場合:

```
DGMGRL> migrate pluggable database PDB002 to container CDB200
 using '/tmp/PDB002.xml>' connect as sys@"CDB200_INST1"
 secret "some_value"
 keystore identified by "destination_keystore_password"
 keyfile '/tmp/pdb002_key.dat'
 source keystore identified by "source_keystore_password";
```

> ノート:
>
> TDE 環境では、SECRET、KEYSTORE、KEYFILE または SOURCE KEYSTORE がコマンドラインで指定されていない場合、MIGRATE PLUGGABLE DATABASE コマンドは失敗します。

宛先への接続が確立されると、コマンドは次のようになります。

1. フェイルオーバー操作に必要なすべての検証を実行します
2. TDEが有効な場合、ソース・スタンバイ・キーストアからPDBのTDEキーをエクスポートします
3. ソース・スタンバイで実行中のREDO適用を停止します
4. DBMS_PDB.DESCRIBEコマンドを使用して、コマンドで指定された場所のスタンバイにマニフェストを作成します
5. ソース・スタンバイでPDBのリカバリを無効にします
6. TDEが有効な場合は、TDEキーを宛先CDBキーストアにインポートして、プラグインを正常に実行できるようにします
7. スタンバイのデータ・ファイル(NOCOPY句)およびSTANDBYS=NONEを使用して、宛先データベースにPDBを接続します。
8. 宛先プライマリ・データベースのすべてのインスタンスでPDBをオープンします
9. TDEが有効な場合は、PDBのコンテキストでADMINISTER KEY MANAGEMENT USE KEYを発行して、インポートされたキーとPDBを関連付けます。
10. ソース・プライマリからPDBを切断します。切断でエラーが発生した場合、手動でクリーンアップを実行するようにメッセージが提示されます
11. 切断が成功した場合は、KEEP DATAFILES句を使用してソース・プライマリからPDBを削除します。これにより、すべてのソース・スタンバイ・データベースのPDBも削除されます。

ステップ6: 移行後 - サービスの有効化

PDBのアプリケーション・サービスをCluster Ready Services (CRS)に追加し、PDBに関連付けて宛先CDBでデータベース・ロールを修正し、対応するサービスをソースCDBから削除します。

1. プライマリ環境とスタンバイ環境の両方のサービスごとに、次を実行します:

```
DESTINATION_PRIMARY_HOST $ srvctl add service -db cdb200 -s SERVICE_NAME
 -pdb pdb002 -role [PRIMARY|PHYSICAL_STANDBY]….
SOURCE_PRIMARY_HOST $ srvctl remove service -db cdb100 -s SERVICE_NAME
SOURCE_STANDBY_HOST $ srvctl remove service -db cdb100_stby -s SERVICE_NAME
```

2. 適切なデータベース・ロールに必要なサービスを開始します。

   各PRIMARYロールのデータベース・サービスを開始します

```
DESTINATION_PRIMARY_HOST $ srvctl start service -db cdb200 -s SERVICE_NAME
```

ステップ7: PDBのバックアップ

将来リカバリできるように、宛先CDB (CDB200)のPDBをバックアップします。

```
DESTINATION_PRIMARY_HOST $ rman
RMAN> connect target sys@cdb200
RMAN> backup pluggable database pdb002;
```

ステップ8: PDBのリカバリのオプションでの有効化

Oracle Multitenantでの遅延PDBリカバリおよびSTANDBYS=NONE機能の使用(ドキュメントID 1916648.1)のステップに従って、スタンバイ・データベースでのPDBのリカバリを有効にし、可用性および障害時リカバリの要件を確立します。

ステップ9: オプションでの移行

「PDBスイッチオーバーの構成」の移行ステップを参照してください。

エラーの解決

宛先プライマリCDBへの接続が成功したが、宛先スタンバイでファイルが見つからないなどの問題がある場合、宛先CDBスタン

バイ・データベースで作成されたGRPを使用して解決に役立てることができます。

ブローカは、GRPを削除せずに実行を終了するスタンバイでエラーを検出した場合、エラーの解決に使用できます。GRP名は、CLIコマンド実行の出力に表示されます。

この方法を使用する前に、前提条件セクションのすべてのパッチが適用されていることを確認します。

1. 自動的に起動しないように、Data Guard BrokerでのREDO適用をオフにします

   ```
   DGMGRL> edit database CDB200_STBY set state='APPLY-OFF';
   ```

2. 宛先CDBスタンバイをマウント・モードで再起動し、RAC環境で1つのインスタンスのみが実行されていることを確認します。

   - Oracle RACの場合

     ```
     $ srvctl stop database –d cdb200_stby –o immediate
     $ srvctl start instance –d cdb200_stby –i cdb200s1 –o mount
     ```

   - SIDBの場合

     ```
     SQL> shutdown immediate
     SQL> startup mount
     ```

3. 宛先CDBスタンバイ・データベースのPDBに接続し、PDBのリカバリを無効にします。

   ```
   SQL> alter session set container=pdb001;
   SQL> alter pluggable database disable recovery;
   ```

4. 宛先CDBスタンバイ・データベースのCDB$rootに接続し、スタンバイ・データベースをフラッシュバックします。

   ```
   SQL> alter session set container=cdb$root;
   SQL> flashback database to restore point <GRP from execution>;
   ```

5. REDO適用の失敗の原因となった問題を修復します(ASM別名がないなど)。

6. CDBスタンバイでマウント・モードのままにして、REDO適用を開始します。

   ```
   SQL> recover managed standby database disconnect;
   ```

   REDO適用では、新しく接続されたPDBのすべてのファイルの再スキャンを含め、GRPからのすべてのREDOの適用が開始されます。フラッシュバックGRPは、宛先CDBスタンバイを、PDBがスタンバイに認識されていないポイントにロールバックするため、PDBのリカバリの無効化もバックアウトされます。

ステップ1から6は、すべてのファイルがスタンバイに追加され、追加のREDOが適用されるまで、必要な回数繰り返すことができます。この時点で、次のようになります。

1. リカバリを停止します

   ```
   DGMGRL> edit database CDB200_STBY set state='APPLY-OFF';
   ```

2. 宛先CDBスタンバイ・データベースのCDB$rootに接続し、宛先スタンバイ・データベースからGRPを削除します。

   ```
   SQL> drop restore point <GRP from execution>;
   ```

3. REDO適用を再開します

   ```
   DGMGRL> edit database CDB200_STBY set state='APPLY-ON';
   ```

問題が続いており、問題解決中にCDBスタンバイ・データベースがスタンバイ内の追加PDBの保護を維持する必要がある場合:

- 前述のようにPDBのリカバリを無効にします

- CDBスタンバイ内の他のPDBを保護するためにREDO適用を再開します
- Oracle Multitenantでの遅延PDBリカバリおよびSTANDBYS=NONE機能の使用(ドキュメントID 1916648.1)のリカバリの有効化のステップに従って、失敗したPDBのリカバリを有効にします。
- 宛先CDBスタンバイからGRPを削除します。

テスト中に、解決できないスタンバイで繰返しエラーが発生した場合:

1. スタンバイでのREDO適用のためのPDB操作のデバッグを有効にします。

   ```
   SQL> alter system set "_pluggable_database_debug"=256 comment='set to help
   debug PDB plugin issues for PDB100, reset when done' scope=both;
   ```

2. 前述のステップに従って、宛先CDBスタンバイ・データベースをフラッシュバックします。
3. REDO適用を再開します。

新しい障害の後、REDO適用を実行していたスタンバイ・ホストからREDO適用トレース・ファイル(..../trace/<SID>_pr*.trc)を収集し、バグをオープンします。

デバッグが完了すると、次のようになります。

1. パラメータをリセットしてデバッグをオフにします。

   ```
   SQL> alter system reset "_pluggable_database_debug" scope=spfile;
   ```

2. CDBスタンバイ・データベースをバウンスします。

リファレンス

次の例では、DGMGRL MIGRATEコマンドの一部として、コマンドを実行する際に表示されるものとは異なる出力が生成される場合があることに注意してください。これは、お使いの環境でDGMGRLが事前チェックを実行して検出したPDBやアイテムの状態が異なることによりますまた、Oracleはバグ修正を含むメッセージ・ファイルを送信しないため、完全なメッセージを表示するかわりに、次のようなメッセージを受信する可能性があります。

```
Message 17241 not found; product=rdbms; facility=DGM
```

これは、エラーや問題ではなく、表示するテキストがメッセージ・ファイルにないことを意味します。すべてのメッセージが、すべての修正を含む最初のリリースで完全に表示されます。

コマンドの完全な例と出力

次に、コマンドの例と出力を示します。

例30-1 TDEを使用しない移行

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200
 USING '/tmp/PDB001.xml' CONNECT AS sys/password@cdb200_inst1
 STANDBY  FILES sys/standby_asm_sys_passwd@standby_asm_inst1
 SOURCE STANDBY CDB100_STBY DESTINATION STANDBY CDB200_STBY ;
Beginning migration of pluggable database PDB001.
Source multitenant container database is CDB100.
Destination multitenant container database is CDB200.
Connecting to "+ASM1".
Connected as SYSASM.
Stopping Redo Apply services on multitenant container database cdb200_stby.
The guaranteed restore point "<GRP name>" was created for multitenant container
database "cdb2001_stby".
Restarting redo apply services on multitenant container database cdb200_stby.
```

```
Closing pluggable database PDB001 on all instances of multitenant container database
CDB100.
Unplugging pluggable database PDB001 from multitenant container database cdb100.
Pluggable database description will be written to /tmp/pdb001.xml
Dropping pluggable database PDB001 from multitenant container database CDB100.
Waiting for the pluggable database PDB001 to be dropped from standby multitenant
container database cdb100_stby.
Creating pluggable database PDB100 on multitenant container database CDB200.
Checking whether standby multitenant container database cdb200_stby has added all
data files for pluggable database PDB001.
Opening pluggable database PDB001 on all instances of multitenant container database
CDB200.
The guaranteed restore point "<GRP_name>" was dropped for multitenant container
database "cdb200_stby".
Migration of pluggable database PDB001 completed.
Succeeded.
```

例30-2 TDEを使用した移行

```
DGMGRL> MIGRATE PLUGGABLE DATABASE PDB001 TO CONTAINER CDB200 USING ‘/tmp/pdb001.xml’
 CONNECT AS sys/password@cdb200_inst1 SECRET "some_value"
 KEYSTORE IDENTIFIED BY "destination_TDE_keystore_passwd"
 STANDBY FILES sys/standby_ASM_sys_passwd@standby_asm_inst1
 SOURCE STANDBY cdb100_stby DESTINATION STANDBY cdb200_stby;
Master keys of the pluggable database PDB001 to need to be migrated.
Keystore of pluggable database PDB001 is open.
Beginning migration of pluggable database PDB001.
Source multitenant container database is cdb100.
Destination multitenant container database is cdb200.
Connecting to "+ASM1".
Connected as SYSASM.
Stopping Redo Apply services on multitenant container database cdb200_stby.
The guaranteed restore point "..." was created for multitenant container database
"cdb200_stby".
Restarting redo apply services on multitenant container database cdb200_stby.
Closing pluggable database PDB001 on all instances of multitenant container database
cdb100.
Unplugging pluggable database PDB001 from multitenant container database cdb100.
Pluggable database description will be written to /tmp/pdb001.xml
Dropping pluggable database PDBT001 from multitenant container database cdb100.
Waiting for the pluggable database PDB001 to be dropped from standby multitenant
container
database cdb100_stby.
Creating pluggable database PDB1001 on multitenant container database cdb200.
Checking whether standby multitenant container database cdb200_stby has added all
data files for pluggable database PDB001.
Stopping Redo Apply services on multitenant container database cdb200_stby.
Opening pluggable database PDB001 on all instances of multitenant container database
cdb400.
The guaranteed restore point "..." was dropped for multitenant container database
"cdb200_stby".
Please complete the following steps to finish the operation:
1. Copy keystore located in <cdb200 primary keystore location> for migration
destination primary database to <cdb200 standby keystore location> for migration
destination standby database.
2. Start DGMGRL, connect to multitenant container database cdb200_stby, and issue
command "EDIT DATABASE cdb200_stby SET STATE=APPLY-ON".
3. If the clusterware is configured on multitenant container databases cdb200 or
cdb200_stby, add all non-default services for the migrated pluggable database in
cluster ready services.
Migration of pluggable database PDB001 completed.
Succeeded.
```

例30-3 TDEを使用しないフェイルオーバー

```
DGMGRL> migrate pluggable database PDB002 immediate to container CDB200
```

```
 using '/tmp/<pdb002.xml>';
Username: USERNAME@cdb200
Password:
Connected to "cdb200"
Connected as SYSDBA.
Beginning migration of pluggable database pdb002.
Source multitenant container database is cdb100_stby.
Destination multitenant container database is cdb200.
Connected to "cdb100"
Closing pluggable database pdb002 on all instances of multitenant container database
cdb100.
Continuing with migration of pluggable database pdb002 to multitenant container
database cdb200.
Stopping Redo Apply services on source multitenant container database cdb100_stby.
Succeeded.
Pluggable database description will be written to /tmp/pdb002.xml.
Closing pluggable database pdb002 on all instances of multitenant container database
cdb100_stby.
Disabling media recovery for pluggable database pdb002.
Restarting redo apply services on source multitenant container database cdb100_stby.
Succeeded.
Creating pluggable database pdb002 on multitenant container database cdb200.
Opening pluggable database pdb002 on all instances of multitenant container database
cdb200.
Unplugging pluggable database pdb002 from multitenant container database cdb100.
Pluggable database description will be written to /tmp/pdb002_temp.xml.
Dropping pluggable database pdb002 from multitenant container database cdb100.
Unresolved plug in violations found while migrating pluggable database pdb002 to
multitenant container database cdb200.
Please examine the PDB_PLUG_IN_VIOLATIONS view to see the violations that need to be
resolved.
Migration of pluggable database pdb002 completed.
Succeeded.
```

例30-4 TDEを使用したフェイルオーバー

ノート: 出力内のORA-46655エラーは無視できます。

```
DGMGRL> migrate pluggable database PDB002 to container CDB200
 using '/tmp/PDB002.xml>' connect as sys@”CDB200” secret “some_value”
 keystore identified by “destination_keystore_password” keyfile ‘/tmp/pdb002_key.dat’
 source keystore identified by “source_keystore_password”;
Connected to "cdb200"
Connected as SYSDBA.
Master keys of the pluggable database PDB002 need to be migrated.
Keystore of pluggable database PDB002 is open.
Beginning migration of pluggable database PDB002.
Source multitenant container database is adg.
Destination multitenant container database is cdb200.
Connected to "cdb1001"
Exporting master keys of pluggable database PDB002.
Continuing with migration of pluggable database PDB002 to multitenant container
database cdb200.
Stopping Redo Apply services on multitenant container database adg.
Pluggable database description will be written to /tmp/PDB002.xml.
Closing pluggable database PDB002 on all instances of multitenant container database
adg.
Disabling media recovery for pluggable database PDB002.
Restarting redo apply services on multitenant container database adg.
Unplugging pluggable database PDB002 from multitenant container database cdb100.
Pluggable database description will be written to /tmp/ora_tfilSxnmva.xml.
Dropping pluggable database PDB002 from multitenant container database cdb100.
Importing master keys of pluggable database PDB002 to multitenant container database
cdb200.
Creating pluggable database PDB002 on multitenant container database cdb200.
```

```
Opening pluggable database PDB002 on all instances of multitenant container database
cdb200.
ORA-46655: no valid keys in the file from which keys are to be imported
Closing pluggable database PDB002 on all instances of multitenant container database
cdb200.
Opening pluggable database PDB002 on all instances of multitenant container database
cdb200.
Please complete the following steps to finish the operation:
If the Oracle Clusterware is configured on multitenant container database CDB200, add
all non-default services for the migted pluggable database in Cluster Ready Services.
Migration of pluggable database PDB002 completed.
Succeeded.
```

キーワード定義

DGMGRL MIGRATEコマンドのキーワードを次に説明します。

構文

```
DGMGRL> MIGRATE PLUGGABLE DATABASE pdb-name
TO CONTAINER dest-cdb-name
USING XML-description-file
CONNECT AS { /@dest-cdb-connect-identifer |
 dest-cdb-user/dest-cdb-password@dest-cdb-connect-identifier}
[SECRET “secret” KEYSTORE IDENTIFIED BY ( EXTERNAL STORE | wallet-password) ;]
STANDBY FILES { /@asm-instance-connect-identifer |
 sysasm-user/sysasm-password@asm-instance-connect-identifier}
SOURCE STANDBY source-standby-cdb-name
DESTINATION STANDBY dest-standby-cdb-name
[TIMEOUT timeout]
```

これらは、PDBの移行コマンドで使用されるキーワード定義です

- pdb-name - 移行対象のPDBの名前。
- dest-cdb-name - 移行対象のPDBを受け入れるCDBの一意のデータベース名。
- XML-description-file - 移行対象のPDBの説明が含まれているXMLファイル。このファイルは、MIGRATE PLUGGABLE DATABASEコマンドで実行されたSQL文によって自動的に作成されます。このファイルの場所には、ソースと宛先の両方のプライマリ・データベース・インスタンスで直接アクセスできる必要があります。コマンドの実行前に存在することはできません。
- dest-cdb-user - 宛先CDBへのSYSDBAアクセス権があるユーザーの名前。
- dest-cdb-password - dest-cdb-userに指定されているユーザー名に関連付けられたパスワード。
- dest-cdb-connect-identifier - 宛先CDBへの接続に使用するOracle Net接続識別子。
- secret - ソースPDBのエクスポートされた暗号化キーを含むエクスポート・ファイルを暗号化するために使用される単語。この句は、TDE対応環境でのみ必要です。
- keyfile - ソースPDBのエクスポートされた暗号化キーを含むデータ・ファイル。このファイルは、フェイルオーバー・ユースケースにおいてMIGRATE PLUGGABLE DATABASEコマンドで実行されたSQL文によって作成されます。このファイルの場所には、ソース・スタンバイ・インスタンスおよび宛先プライマリ・インスタンスによって直接アクセスできる必要があります。
- wallet-password - 暗号化キーを含む宛先CDBキーストアのパスワード。これは、ソースPDBがTDE対応環境でパスワード・キーストアを使用して暗号化された場合に必要です。
- asm-instance-connect-identifier - ソース・スタンバイ・データベース・ファイルがあるASMインスタンスへの接続識別子。
- sysasm-user - ASMインスタンスに対するSYSASM権限があるユーザー。
- sysasm-password - sysasm-userのパスワード。

- source-standby-cdb-name - 移行ソースCDBのスタンバイ・データベースのDB_UNIQUE_NAME。
- dest-standby-cdb-name - 移行先CDBのスタンバイ・データベースのDB_UNIQUE_NAME。
- timeout - 移行中に宛先スタンバイ・データベースでデータ・ファイルが取得されるまでの待機時間のタイムアウト値(秒)。これはオプションです。TIMEOUT句を省略した場合のデフォルトは5分です。

メッセージ

次に、DGMGRL MIGRATE関数によって生成される可能性があるメッセージのリストを示します。

一般的な処理の場合

17180 - 「移行操作を開始する前に、プラガブル・データベース%sをオープンする必要があります。」

17217 - 「ソース・マルチテナント・コンテナ・データベース(%(1)s)が、%(2)s以外のバージョンのOracleを実行する物理スタンバイである場合、移行を実行できません。」

17235 - 「プラガブル・データベース%sを切断できなかった理由を調査してください。」

17236 - 「問題を解決し、データベース%sからプラガブル・データベースを手動で切断して削除してください。」

17237 - 「プラガブル・データベース%sの移行が完了しました。」

17238 - 「プラガブル・データベース%sの移行が警告付きで完了しました。」

17239 - 「プラガブル・データベース%sの移行に失敗しました。」

17240 - 「プラガブル・データベース%(1)s (マルチテナント・コンテナ・データベース%(2)s)のメディア・リカバリが無効化されています。」

17241 - 「警告: ソースまたは宛先のマルチテナント・コンテナ・データベースで、ローカルUNDOが有効化されていません。」

17242 - 「スナップショットの子または親であるため、プラガブル・データベース%sからの移行はできません。」

17243 - 「より高いバージョンのOracleが稼働しているデータベースに移行されたため、プラガブル・データベース%sをオープンできませんでした。」

17244 - 「プラガブル・データベースをオープンする前に適切なアップグレード手順を実行してください。」

17245 - 「指定されたファイルの場所(%s)はアクセスできません。」

17246 - 「ファイル名が指定されていませんでした。」

17247 - 「無効なファイル名(%s)が指定されました。」

17248 - 「ラグの解消後にコマンドを再試行するか、IMMEDIATEオプションを使用してデータ損失を無視します。」

17249 - 「プラガブル・データベース%(1)s (マルチテナント・コンテナ・データベース%(2)s)のメディア・リカバリが無効化されています。」

17250 - 「警告: ソースまたは宛先のマルチテナント・コンテナ・データベースで、ローカルUNDOが有効化されていません。」

17251 - 「スナップショットの子または親であるため、プラガブル・データベース%sからの移行はできません。」

TDEサポートの追加の場合

17413 - 「プラガブル・データベース%sのキーストアのオープンに失敗しました。」

17414 - 「プラガブル・データベース%sのキーストアがオープンしていません。」

17415 - 「プラガブル・データベース%sのキーストア・パスワードは必須です。」

17416 - 「マルチテナント・コンテナ・データベース%sのキーストア・パスワードは必須です。」

17427 - 「プラガブル・データベース%sのキーストア・ステータスをフェッチできません。」

17428 - 「プラガブル・データベース%sのキーストアがオープンしています。」

17429 - 「プラガブル・データベース%sのキーストア・パスワードが正しくありません。」

17430 - 「マルチテナント・コンテナ・データベース%sのキーストア・パスワードが正しくありません。」

スタンバイ・ファイルのサポートの場合

17510 - 「スタンバイ・データベース¥"%s¥"は、ASMインスタンスを使用していないか、それに接続されていません。」

17511 - 「ソース・マルチテナント・コンテナ・データベースのスタンバイ・データベースは、宛先マルチテナント・コンテナ・データベースとは異なるASMディスク・グループを使用しています。」

17512 - 「移行宛先のスタンバイ・データベースの初期化パラメータDB_FILE_NAME_CONVERTがNULLではありません。」

17513 - 「移行ソースのスタンバイ・データベースの初期化パラメータSTANDBY_FILE_MANAGEMENTがAUTOではありません。」

17514 - 「マルチテナント・コンテナ・データベース%sは、フィジカル・スタンバイ・データベースではありません。」

17515 - 「データ・ファイル%sのASM別名が予期された場所にありません。」

17516 - 「Data Guard Broker構成のマルチテナント・コンテナ・スタンバイ・データベースを指定する必要があります。」

17517 - 「ソース・マルチテナント・コンテナ・データベースがスタンバイ・データベースの場合、データ・ファイルを再利用できません。」

17518 - 「ASM別名%sは、予期された場所にないASMファイルを参照しています。」

17519 - 「保証付きリストア・ポイント¥"%(1)s¥"は、マルチテナント・コンテナ・データベース¥"%(2)s¥"で作成されました。」

17520 - 「保証付きリストア・ポイント¥"%(1)s¥"は、マルチテナント・コンテナ・データベース¥"%(2)s¥"で削除されました。」

17521 - 「SYSASMとして接続しました。」

17522 - 「マルチテナント・コンテナ・データベース¥"%(1)s¥"は、データ・ファイル¥"%(2)s¥"の検出に失敗しました。」

17523 - 「マルチテナント・コンテナ・データベース¥"%s¥"は、不安定な状態です。」

17524 - 「マルチテナント・コンテナ・データベース¥"%(1)s¥"は、リストア・ポイント¥"%(2)s¥"を使用してリストアできます。」

17525 - 「REDO適用がマルチテナント・コンテナ・データベース¥"%s¥"で停止または失敗しました。」

17530 - 「スタンバイ・マルチテナント・コンテナ・データベース%(1)sは、プラガブル・データベース%(2)sのすべてのデータ・ファイルの追加に失敗しました。」

17532 - 「プラガブル・データベース%(1)sのスタンバイ・マルチテナント・コンテナ・データベース%(2)sからの削除に失敗しました。」

17533 - 「指定されたファイル(%s)は存在していない必要があります。」

17534 - 「パスが指定されませんでした。」

17536 -「プラガブル・データベース%sのキーストア・モードをフェッチできません。」

17537 -「KEYFILEおよびSOURCE IDENTIFIED BY句が必要です。」

17539 -「プラガブル・データベース%(1)sのマスター・キーをマルチテナント・コンテナ・データベース%(2)sにインポートしています。」

サンプルOracle Database Net Services接続別名

ブローカ・セッションの開始時に、DGMGRLから次のNet Services接続別名にアクセスできる必要があります。これは、デフォルトのtnsnames.oraの場所を使用するか、DGMGRLを起動する前に環境でTNS_ADMINを設定することで可能です。

PDBスイッチオーバー

次の例のホスト名は、Oracle単一クライアント・アクセス名(SCAN)のホスト名を参照します。ソース・データベースと宛先データベースは同じホストに存在する必要があるため、ホスト名に重複があります。いずれの場合も、接続文字列はデータベースのcdb$rootに接続する必要があります。

ソース・プライマリ・データベース

```
CDB100 =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-primary-scan-name>)
                (PORT = <source-primary-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-primary-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
              )
             )
          )
```

ソース・プライマリ・データベースのローカル・インスタンス

```
CDB100_INST1 =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-primary-scan-name>)
                (PORT = <source-primary-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-primary-cdb$root-service-name>)
              (INSTANCE_NAME = <source-primary-local-instance-name>)
            )
          )
```

宛先プライマリ・データベース

```
CDB200=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
```

```
        (HOST= <source-primary-scan-name>)
        (PORT= <source-primary-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)))
```

宛先プライマリ・ローカル・インスタンス

これは、dgmgrlが実行されている同じホスト上のインスタンスに接続する必要があります

```
CDB200_INST1=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-primary-scan-name>)
        (PORT= <source-primary-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)
        (INSTANCE_NAME = <destination-primary-local-instance-name>)
      )
    )
```

ソース・スタンバイ・データベース

```
CDB100_STBY =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-standby-scan-name)
                (PORT = <source-standby-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-standby-cdb$root-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
              )
             )
          )
```

宛先スタンバイ・データベース

```
CDB200_STBY =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-standby-scan-name>)
                (PORT = <source-standby-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <destination-standby=cdb$root-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
              )
             )
          )
```

スタンバイ環境ASM

これは、ソース・スタンバイおよび宛先スタンバイのそれぞれ1つのインスタンスと同じホストで実行されているASMインスタンスに接続する必要があります

```
STANDBY_ASM_INST1=
    (DESCRIPTION=
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST = <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= +ASM)
        (INSTANCE_NAME=<ASM_instance_name>)
      )
    )
```

PDBフェイルオーバー

ソース・プライマリ・データベース

```
CDB100 =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-primary-scan-name>)
                (PORT = <source-primary-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-primary-cdb$root-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
              )
             )
          )
```

ソース・スタンバイ・データベース

```
CDB100_STBY =
          (DESCRIPTION =
            (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
            (ADDRESS =
                (PROTOCOL = TCP)
                (HOST = <source-standby-scan-name>)
                (PORT = <source-standby-listener-port>)
            )
            (CONNECT_DATA =
              (SERVER = DEDICATED)
              (SERVICE_NAME = <source-standby-cdb$root-service-name>)
              (FAILOVER_MODE =
                  (TYPE = select)
                  (METHOD = basic)
              )
             )
          )
```

ソース・スタンバイ・データベースのローカル・インスタンス

これは、dgmgrlが実行されている同じホスト上のインスタンスに接続する必要があります

```
CDB100_STBY_INST1=
    (DESCRIPTION=
```

```
      (CONNECT_TIMEOUT=120)(TRANSPORT_CONNECT_TIMEOUT=90)(RETRY_COUNT=3)
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <source-standby-cdb$root-service-name>)
        (INSTANCE_NAME = <source-standby-local-instance-name>)
      )
    )
```

宛先プライマリ・データベース

```
CDB200=
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)))
```

宛先プライマリ・ローカル・インスタンス

これは、dgmgrlが実行されている同じホスト上のインスタンスに接続する必要があります

```
CDB200_INST1=
    (DESCRIPTION=
      (ADDRESS=
        (PROTOCOL= TCP)
        (HOST= <source-standby-scan-name>)
        (PORT= <source-standby-listener-port>))
      (CONNECT_DATA=
        (SERVER= DEDICATED)
        (SERVICE_NAME= <destination-primary-cdb$root-service-name>)
        (INSTANCE_NAME = <destination-primary-local-instance-name>)
      )
    )
```

第IX部 Oracle CloudまたはオンプレミスでのサイトOracle Cloudまたはオンプレミスでのサイト全体の切替え

31 Oracle Cloudまたはオンプレミスでのサイト全体の切替え

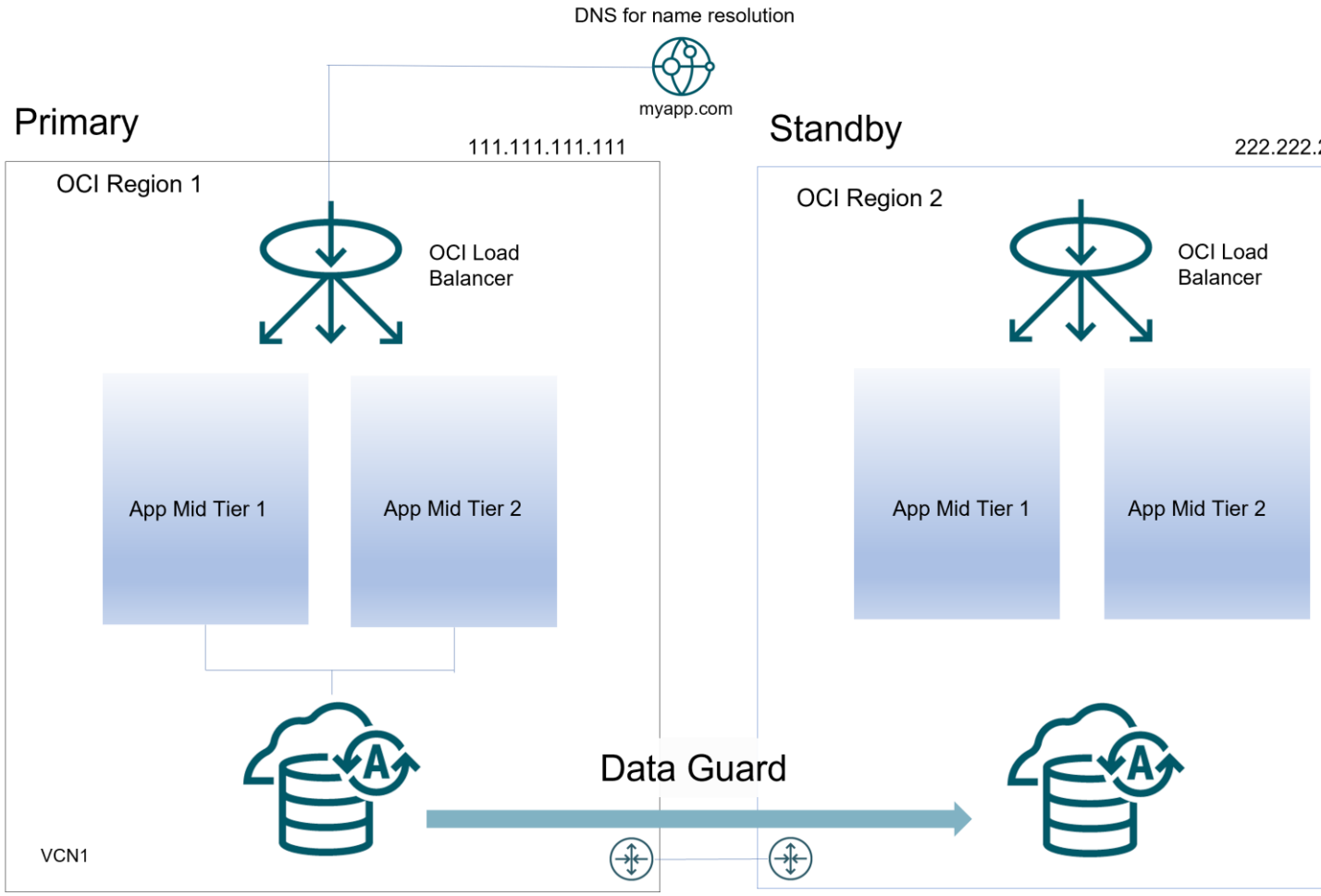
完全サイト障害またはサイト全体の障害が発生すると、アプリケーション層とデータベース層の両方が使用できなくなります。可用性を維持するため、本番データベースの冗長アプリケーション層と同期コピーをホスティングするセカンダリ・サイトに、ユーザーをリダイレクトする必要があります。MAAのベスト・プラクティスは、Data Guardを使用して本番データベースの同期コピーを保持することです。サイト障害が発生すると、WANトラフィック・マネージャまたはロード・バランサを使用してDNSフェイルオーバーが(手動または自動で)実行されて、すべてのユーザーがスタンバイ・サイトのアプリケーション層にリダイレクトされます。その間に、Data Guardフェイルオーバーによってスタンバイ・データベースがプライマリ本番ロールに遷移します。

通常のランタイム操作では、次のことが行われます。

1. クライアント・リクエストは、プライマリ・サイトのクライアント層に入り、WANトラフィック・マネージャによって転送されます。
2. クライアント・リクエストは、アプリケーション・サーバー層に送信されます。
3. リクエストは、アクティブなロード・バランサを介してアプリケーション・サーバーに転送されます。
4. リクエストは、データベース・サーバー層に送信されます。
5. アプリケーション・リクエストは、必要に応じてOracle RACインスタンスにルーティングされます。
6. レスポンスは、同様の経路でアプリケーションとクライアントに戻されます。

次に、サイト・スイッチオーバー前のネットワーク・ルートを示します。

図31-1 スイッチオーバー前のサイト

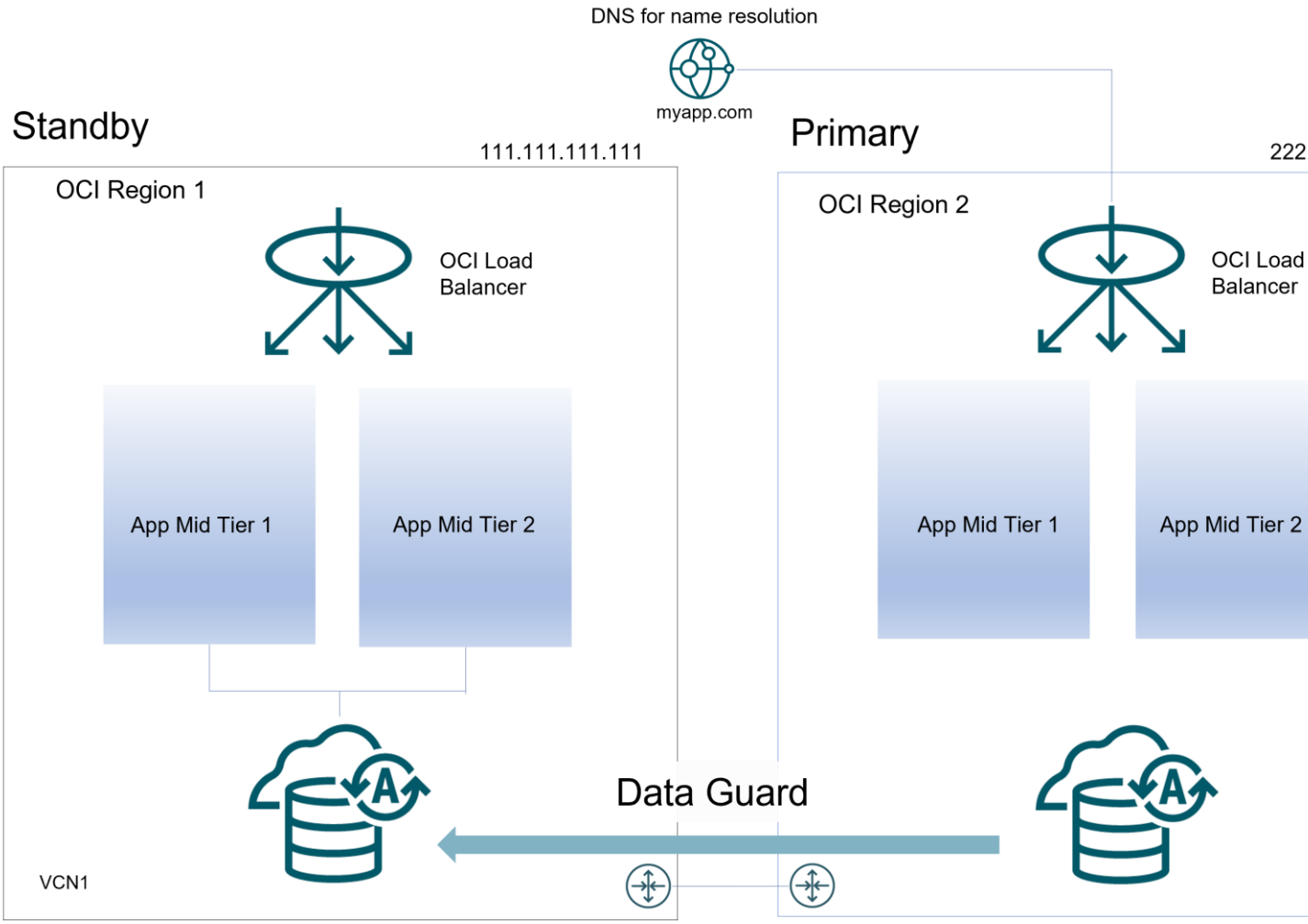


次のステップでは、サイト・スイッチオーバーの影響について説明します。

1. 管理者は、プライマリ・データベースをセカンダリ・サイトにフェイルオーバーまたはスイッチオーバーします。この操作は、Data Guardファスト・スタート・フェイルオーバーを使用している場合は自動的に処理されます。専用ハードウェア上のAutonomous Databaseは、Data Guardファスト・スタート・フェイルオーバーをサポートしています。
2. 管理者は、セカンダリ・サイトの中間層アプリケーション・サーバーを起動します(実行されていない場合)。障害が発生したサイトに同じ中間層アプリケーション・サーバーが存在する場合、それらを利用できることがあります。
3. セカンダリ・サイトのワイドエリア・トラフィック・マネージャの選択は、サイト全体の障害に対して自動的に行うことができます。
4. セカンダリ・サイトのワイドエリア・トラフィック・マネージャがセカンダリ・サイトのロード・バランサの仮想IPアドレスを返し、後続の再接続でクライアントが自動的に方向付けされます。このシナリオでは、サイト・フェイルオーバーは自動ドメイン・ネーム・システム(DNS)フェイルオーバーによって実行されます。

次の図は、サイト・フェイルオーバー後のネットワーク・ルートを示しています。クライアントまたはアプリケーション・リクエストは、セカンダリ・サイトのクライアント層に入り、プライマリ・サイトと同じ経路をたどります。

図31-2 スイッチオーバー後のサイト



フェイルオーバーは、クライアントのWebブラウザにも依存します。ほとんどのブラウザ・アプリケーションでは、一定期間DNSエントリがキャッシュされます。したがって、停止の発生時に進行中だったセッションは、キャッシュ・タイムアウトの期限切れまでフェイルオーバーされない可能性があります。このようなクライアントでサービスを再開するには、ブラウザを一度終了して再起動します。

リージョン間のロール遷移の実行

次に、Oracle Public Cloudを活用する例を示します。ただし、同様のステップをオンプレミスまたはハイブリッド・クラウドのシナリオで実行できます。

別のリージョンへのフェイルオーバー

フェイルオーバー操作は、プライマリ・サイトが使用できなくなったときに実行され、通常は計画外の操作です。元のプライマリ・データベースで障害が発生し、プライマリ・データベースを適時にリカバリできる可能性がない場合は、スタンバイ・データベースのプライマリ・データベースへのロール・トランジションを実行できます。プライマリ・データベースに障害が発生した時点でプライマリ・データベースとターゲット・スタンバイ・データベースに一貫性があったかどうかによって、データが消失する場合があります。

DR構成で手動フェイルオーバーを実行するには、次のステップに従います。

1. DNS名をスイッチオーバーします。

   システムで使用される名前をホストするDNSサーバーで必要なDNSプッシュを実行するか、クライアントのファイル・ホスト解決を変更して、システムのフロント・エンド・アドレスがsite2のロード・バランサによって使用されるパブリックIPを指すようにします。DNSが外部フロント・エンド解決(OCI DNS、商用DNSなど)に使用されるシナリオでは、適切なAPIを使用して変更をプッシュできます。OCI DNSでこの変更をプッシュする例:

次に、ordscsdroci.domainexample.comなどのフロント・エンドDNSエントリをサイト1ロード・バランサのパブリックIPアドレス(111.111.111.123など)に更新するOCIクライアント・スクリプトを示します。

```
oci dns record rrset update
 --config-file /home/opc/scripts/.oci_ordscsdr/config
 --zone-name-or-id "domainexample.com"
 --domain "ordscsdroci.domainexample.com"
 --rtype "A"
 --items
'[{"domain":"ordscsdroci.domainexample.com","rdata":"111.111.111.123","rtype":"
A","ttl":60}]'
 --force
```

2. データベースをフェイルオーバーします。

   Oracle Cloud:

   Oracle Control Planeを使用して、Data Guardスイッチオーバーまたはフェイルオーバー操作を発行します。

   オンプレミス:

   セカンダリ・データベース・ホストのData Guard Brokerを使用してフェイルオーバーを実行します。ユーザーoracleとして:

```
[oracle@drdbwlmp1b ~]$ dgmgrl sys/your_sys_password@secondary_db_unqname
DGMGRL> failover to “secondary_db_unqname”
```

3. セカンダリ・サイトのサーバーを起動します。

   セカンダリ・アプリケーション・サーバーを再起動します。

スイッチオーバー

スイッチオーバーは、管理者が2つのサイトのロールを元に戻す計画操作です。ロールは、プライマリからスタンバイ、およびスタンバイからプライマリに変更されます。これは手動スイッチオーバーと呼ばれます。手動スイッチオーバーを実行するには、次のステップに従います。

1. 保留中の構成変更を伝播します。

   データベース以外のファイルの場合は、rsyncまたはオブジェクト・ストレージ・サービス(OSS)を使用してセカンダリ・サイトにレプリケートできます。

2. プライマリ・サイトのサーバーを停止します。

   スクリプトを使用して、プライマリ・サイトの管理対象サーバー/中間層を停止します。

3. DNS名をスイッチオーバーします

   システムで使用される名前をホストするDNSサーバーで必要なDNSプッシュを実行するか、クライアントのファイル・ホスト解決を変更して、システムのフロント・エンド・アドレスがサイト2のロード・バランサによって使用されるパブリックIPを指すようにします。DNSが外部フロント・エンド解決(OCI DNS、商用DNSなど)に使用されるシナリオでは、適切なAPIを使用して変更をプッシュできます。

   次の例では、OCI DNSでこの変更をプッシュします。

   OCIクライアント・スクリプトは、フロント・エンドDNSエントリ(ordscsdroci.domainexample.comなど)をsite1ロード・バランサのパブリックIPアドレス(111.111.111.123など)に更新します。

```
oci dns record rrset update
 --config-file /home/opc/scripts/.oci_ordscsdr/config
```

```
 --zone-name-or-id "domainexample.com"
 --domain "ordscsdroci.domainexample.com"
 --rtype "A"
 --items
'[{"domain":"ordscsdroci.domainexample.com","rdata":"111.111.111.123","rtype":"
A","ttl":60}]'
 --force
```

DNSエントリのTTL値が、スイッチオーバーの有効なRTOに影響することに注意してください。TTLの値が大きい(20分など)と、DNSの変更がクライアントで有効になるのに、それと同じ時間がかかります。TTLの値を小さくすると、この処理は短時間で行われますが、クライアントがDNSをチェックする頻度が高くなるため、オーバーヘッドの原因になる可能性があります。DNSの変更前に、TTLを一時的に小さい値(1分など)に設定することをお薦めします。その後、変更を行い、スイッチオーバーの手順が完了したら、TTLを通常の値に設定しなおします。

4. データベース・スイッチオーバーを実行します。

   Oracle Cloud:

   Oracle Control Planeを使用して、Data Guardスイッチオーバー操作を発行します。

   オンプレミス:

   プライマリ・データベース・ホストのData Guard Brokerを使用してスイッチオーバーを実行します。

   ユーザーoracleとして:

```
$ dgmgrl sys/your_sys_password@primary_db_unqname
DGMGRL> switchover to “secondary_db_unqname”
```

5. セカンダリ・サイトのサーバーを起動します(新しいプライマリ)。

   セカンダリ管理対象サーバーおよび中間層を再起動します。

サイト全体のスイッチオーバーのベスト・プラクティス

次のベスト・プラクティスを使用することをお薦めします。

- プライマリ・サイトとスタンバイ・サイトで同じ構成を維持します。プライマリ・システムに適用される変更は、セカンダリ・システムでも実行する必要があるため、プライマリ・システムとセカンダリ・システムの両方が同じ構成になります。例: プライマリ・ロード・バランサの変更、オペレーティング・システムへの変更など。
- 定期的なスイッチオーバーを実行して、セカンダリ・サイトの状態を検証します。
- プライマリ・サーバーを停止する前に、停止時間を必要としないスイッチオーバー関連アクティビティを実行します。たとえば、config_replica.shスクリプトに基づくWLS構成レプリケーションには停止時間は必要ないため、プライマリ・システムの稼働中にこれを実行できます。その他の例には、スタンバイ・サイトでの停止ホストの起動があります。
- アプリケーション・サーバーの再起動のために必要な場合は、管理対象サーバー/中間層をパラレルに停止および起動します。
- DNSのフロントエンド更新は顧客に依存します。適切なDNSエントリで(少なくともスイッチオーバー操作中に)小さいTTL値を使用して、更新の時間を短縮します。スイッチオーバーが終了すると、TTLを元の値に戻すことができます。
- また、OCIロード・バランサは、サーバーが稼働していることを認識し、リクエストの送信を開始するのに時間がかかります。通常は、OCIロード・バランサのヘルス・チェックの頻度に応じて数秒かかります。チェックに使用する間隔を短くするほど、サーバーが稼働していることが早くわかるようになります。ただし、使用間隔が短すぎると注意が必要です。ヘルス・チェッ

クが大量のチェックの場合は、バックエンドが過負荷になる可能性があります。

サイト全体のスイッチオーバーの詳細情報

前のトピックでは、サイト全体のフェイルオーバーを一般的な方法で説明しています。特定のアプリケーションのサイト全体のフェイルオーバーの詳細は、次のソースを参照してください。

- Oracle Cloud Infrastructure MarketplaceのSOA Suiteでの障害時リカバリ
- Oracle Cloud InfrastructureのOracle WebLogic Serverの障害時リカバリ
- フル・スタック障害時リカバリ
- Oracle Cloud Infrastructureフル・スタック障害時リカバリ