Oracle Maximum
Availability Architecture

高可用性Oracle Databaseでのクライアント・フェイルオーバーのベスト・プラクティス

Oracle Database 12c

Oracleホワイト・ペーパー | 2014年9月

目次

概要.....1

はじめに.....2

FANをサポートするクライアント・アプリケーションの 自動フェイルオーバー.....3

クライアント・フェイルオーバーの自動化の前提条件 – JDBC、OCIおよびODP.net.....4

ロール・ベースのデータベース・サービス.....6

JDBCクライアントの自動フェイルオーバーの構成.....7

アプリケーション継続性とトランザクション・ガードの構成.....10

自動フェイルオーバーのためのOCIクライアントの構成.....13

ODP.Netクライアントの自動フェイルオーバーの構成.....16

Oracleデータベース・サービスの構成.....17

ロール移行のタイミング.....17

FANをサポートしないクライアントの自動フェイルオーバー.....20

Active Data Guardの接続に関する追加の考慮事項.....21

ログオン・ストームの制御.....23

結論.....24

概要

The Oracle Maximum Availability Architecture (MAA).[1] is the best practices blueprint for implementing Oracle high availability technologies.高可用性の原則は、可能な限りユーザーが停止を認識しないようにすることです。Oracle Database 12cを使用したOracle MAAは、コンポーネント、システム、およびデータベースの停止の際や計画保守中に、ユーザーが中断されることなくトランザクション処理を継続できるようにする（アプリケーションはエラーを受信せずに実行中のトランザクションが完了）という点で比類のないものです。Oracle MAAのおもなコンポーネントであるOracle Real Application Cluster（Oracle RAC）、Oracle Data Guard、Oracle Active Data Guard、およびアプリケーション継続性により、この機能が有効になります。

» Oracle RACにより、複数のアクティブなOracleインスタンスを個別のサーバー上で実行でき、単一のOracle Databaseへのアクセスが共有可能になるため、高可用性（HA）とスケーラビリティが実現します。

» Data GuardおよびActive Data Guardは、シングル・インスタンスまたはOracle RAC本番データベース（プライマリ・データベース）の1つ以上の同期された物理レプリカ（スタンバイ・データベース）を作成して保持する管理、監視おおび自動化ソフトウェアを提供することにより、システムやクラスタの障害、災害、エラー、データ破損、その他の停止が発生した場合にOracleデータを保護します。何らかの理由でプライマリ・データベースが使用できなくなった場合、Data GuardおよびActive Data Guardは、高可用性を維持するためにスタンバイ・データベースに自動フェイルオーバーを実行できます。

» Oracle MAAのベスト・プラクティスおよびOracle Database 12cのイベント通知フレームワークにより、Oracle Databaseに接続されたアプリケーションを、サーバーまたはデータベースのインスタンスが停止した場合はあるRACノードから別のRACノードへ、またはデータベースが停止した場合はプライマリ・データベースからスタンバイ・データベースへ自動的に移行できます。これと同様のフレームワークが計画保守中にも使用されます。RACのローリング保守ではOracle RACデータベースのインスタンス間で、Data Guardのローリング保守ではプライマリ・データベースおよびスタンバイ・データベース間で実行されます。

» トランザクション・ガードを使用したアプリケーション継続性により、実行中のコミットされていないトランザクションが安全に再実行され、Oracleインスタンスまたはデータベースの停止がユーザーに認識されないようにします。アプリケーションはエラーを受信せず、ユーザーはトランザクションを再送信する必要がありません。

このテクニカル・ホワイト・ペーパーは、データベース管理者および開発者を対象としています。ここでは、Data GuardおよびActive Data Guardのロール移行が発生したのちに、障害が発生したプライマリ・データベースから新しいプライマリ・データベースへアプリケーション接続を自動的に移行させるためのOracle Database 12c構成のベスト・プラクティスについて説明します。また、Oracle Database 12cの新機能であるアプリケーション継続性およびトランザクション・ガードのベスト・プラクティスについても説明します。

[1] http://www.oracle.com/goto/maa

はじめに

高可用性構成におけるOracle Databaseインスタンスの予期しない障害は、次の3つの一般的なカテゴリに分類できます。

Oracle RACクラスタにおけるサーバー障害またはインスタンス障害

サーバー障害や、Oracle RACクラスタの個々のOracleインスタンスまたはデータベース・サーバーのクラッシュを引き起こす他の障害が原因で停止が起こると、障害が発生したインスタンスに接続されているクライアントに迅速に通知し、クラスタ内の存続しているインスタンスにクライアントをただちに再接続することによって可用性がリストアされます。高速アプリケーション通知（FAN）により、接続されたクライアントはTCPタイムアウトから抜け出し、透過的アプリケーション・フェイルオーバー（OCIクライアント）または高速接続フェイルオーバー（JDBCクライアント）によって、クライアントは存続しているデータベース・インスタンス上で実行されるデータベース・サービスに自動的にフェイルオーバーされます。このカテゴリの障害についての詳細なベスト・プラクティスは、『*Oracle Real Application Clusters管理およびデプロイメント・ガイド12cリリース1（12.1）』第5章「動的データベース・サービスによるワークロード管理*[2]」に、追加情報については『*My Oracle Support Note 1593712.1、Graceful Application Switchover in RAC with No Application Interruption*[3]』に記載されています。

Oracle RACを用いたOracle Database 12cの新機能は次のとおりです。

» トランザクション・ガードは、アプリケーションによって使用されるAPIで、障害の発生したセッションおよびトランザクションの最終結果を判定します。

» アプリケーション継続性は、データベース・セッションを再構築し、データベース・セッションを使用不可にするリカバリ可能なエラーの後に保留中の作業を再送信することによって、アプリケーションの停止をマスクするソリューションです。

Oracle RACでのアプリケーション継続性の使用のベスト・プラクティスについては、ホワイト・ペーパー『*Oracle Database 12cによるアプリケーション継続性*[4]』に記載されています。

サイト全体の障害

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。可用性を維持するには、ユーザーは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トするセカンダリ・サイトにリダイレクトされなければなりません。Oracle MAAのベスト・プラクティスは、スタンバイ・サイトで実行中のアプリケーション層を維持して起動時間を回避し、Data Guardを使用して本番データベースの同期コピーを維持することです。このような構成の例は、『*Oracle Fusion Middleware SOAマルチ・データセンター・アクティブ-アクティブ・デプロイメントのベスト・プラクティス*[5]』で説明されています。

サイト障害の場合は、WANトラフィック・マネージャを使用してDNSフェイルオーバーを（手動または自動で）実行し、すべてのユーザーをスタンバイ・サイトのアプリケーション層にリダイレク

[2] https://docs.oracle.com/cd/E49329_01/rac.121/b71323/hafeats.htm#BABIAICG

[3] https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1593712.1

[4] http://www.oracle.com/technetwork/jp/database/database-cloud/application-continuity-wp-12c-1966213-ja.pdf

[5] http://www.oracle.com/technetwork/jp/database/availability/fmw11gsoamultidc-aa-1998491-ja.pdf

トすると同時に、Data Guardフェイルオーバーによってスタンバイ・データベースをプライマリ本番ロールへ移行します。サイト全体のフェイルオーバーの自動化については、『*Oracle Database高可用性ベスト・プラクティス*[6]』を参照してください。

部分的なサイト障害

部分的なサイト障害が発生するのは、プライマリ・データベース（シングル・インスタンス・データベースまたはOracle RACデータベースのすべてのノード）が使用できなくなる一方で、プライマリ・データベースに接続されたアプリケーション層が損傷していない場合です。

可用性を維持するためには、Data Guardフェイルオーバーの完了後に、アプリケーション層を新しいプライマリ・データベースへリダイレクトすることが必要です。このユースケースでは、スタンバイ・データベースは存続しているアプリケーション層から離れた場所にあり、そこではデータベース・フェイルオーバーの発生後にリモート接続を使用して許容可能なパフォーマンスを提供できます。

Oracle RACのユースケースと同様に、FANにより、接続されたクライアントはTCPタイムアウトから抜け出し、透過的アプリケーション・フェイルオーバー（OCIクライアント）または高速接続フェイルオーバー（JDBCクライアント）によって、アプリケーションは新しいプライマリ・データベースに自動的にフェイルオーバーされます。このプロセスは、Oracle RACデータベースを含むData Guard構成またはシングル・インスタンスの、Oracle Restart[7]を使用するOracle RAC以外のデータベースに適用されます。Oracle RACと同様に、アプリケーション継続性とトランザクション・ガードの利点はすべて、Data Guardスイッチオーバー（計画されたイベント）、およびData Guardの自動フェイルオーバー（ファスト・スタート・フェイルオーバー）が使用された場合のデータ損失ゼロのフェイルオーバー（計画外停止）にも当てはまります。

このドキュメントの残りの部分では、この停止のカテゴリにおける新しいプライマリ・データベースへの自動アプリケーション・フェイルオーバーに関する構成のベスト・プラクティスについて説明します。

注：特に記述のない場合、このホワイト・ペーパーでData Guardと表現するものはすべて、Oracle Data Guard（Oracle Database Enterprise Editionに含まれる機能）およびActive Data Guard（Oracle Enterprise Editionのオプション・ライセンスに含まれる高度な機能）の両方を指します。

FANをサポートするクライアント・アプリケーションの自動フェイルオーバー

大まかに言うと、Data Guard構成におけるクライアント・フェイルオーバーの自動化には、Data Guardフェイルオーバーの一部として、新しいプライマリ・データベースへのデータベース・サービスの再配置が含まれます。これによって、障害が発生したことをクライアントに通知してTCPタイムアウトから抜け出させ、新しいプライマリ・データベースへクライアントをリダイレクトします。

ここでは、Data Guard構成においてロールベースのデータベース・サービスを作成する方法につい

[6] http://otndnld.oracle.co.jp/document/products/oracle10g/102/doc_cd/server.102/B31489-01/outage.htm#417267

[7] https://docs.oracle.com/cd/E49329_01/server.121/b71301/restart.htm#BABGIGDB

て説明します。これには、OCIおよびJDBC、OLE DB、およびODP.Netアプリケーション・クライアントを有効化してFAN通知を受信し、新しいプライマリ・データベースへ迅速に再接続するための詳細な構成手順が含まれます。

WeblogicやFusion Middlewareなどのパッケージ化されたミドルウェア・アプリケーションでは、ここで説明するのと同じ基本構成がアプリケーション固有の構成をいくつか追加した状態で使用されます。これらのパッケージ化されたアプリケーションの構成の詳細は、このホワイト・ペーパーの範囲に含まれません。詳細については、以下を参照してください。

» *Client and Application Failover Validation Matrix、My Oracle Support Note 1617163.18*[8]

» *Oracle WebLogic Serverと高可用性 Oracle Database：オラクルの統合Maximum Availabilityソリューション*[9]

» *Oracle Database 12cによるアプリケーション継続性*[10]

ご使用のアプリケーション・クライアントがFANをサポートしていない場合は、このホワイト・ペーパーの項「FANをサポートしないクライアント・アプリケーションの自動フェイルオーバー」を参照してください。

クライアント・フェイルオーバーの自動化の前提条件 - JDBC、OCIおよびODP.net

このホワイト・ペーパーで説明するベスト・プラクティスでは、Oracle Database 12.1.0.2以上を必要とします。

データベース・サービスは、このホワイト・ペーパーで説明するアプリケーション・フェイルオーバー・ベスト・プラクティスの基盤となります。データベース・サービスについて十分に理解していない場合は、先に進む前に『*Oracle Real Application Clusters管理およびデプロイメント・ガイド 12cリリース1（12.1）*』の「*第5章動的データベース・サービスによるワークロード管理*」[11]を参照してください。シングル・インスタンス・データベース環境（Oracle RAC以外）については、サービスの管理方法を理解するためにOracle Restart[12]ドキュメントを参照してください。

注：Oracle RestartのOracle Database 12cドキュメントには、この機能は非推奨であると記載されています。このホワイト・ペーパーが書かれた時点では、代換機能がリリースされたあとに最低1回リリースされるまでは、Oracle Restartのサポートやバグ修正を廃止する予定はありませんでした。これは、Oracle Restartの機能を必要とする顧客は、代換機能が発表されていなくても確信をもってデプロイを継続できることを意味します。Oracle Restartのステータスに関する最新情報については、My Oracle Support Note 1584742.1を参照してください。

[8] https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1617163.1

[9] http://www.oracle.com/technetwork/jp/database/features/availability/wlsdatasourcefordataguard-1645083-ja.pdf

[10] http://www.oracle.com/technetwork/jp/database/database-cloud/application-continuity-wp-12c-1966213-ja.pdf

[11] https://docs.oracle.com/cd/E49329_01/rac.121/b71323/hafeats.htm#BABIAICG

[12] http://docs.oracle.com/cd/E57425_01/121/ADMIN/restart.htm#BABGIGDB

このホワイト・ペーパーで説明するベスト・プラクティスでは、Data Guard Brokerを使用してData Guard構成を管理する必要があります[13]。また、Data Guard Brokerはデータベースへの接続をクリーンアップし、新しい本番データベースに再接続するために、クライアント・アプリケーションにFANイベントを送信する役割を担っています。Data Guardのフェイルオーバーの実行後、Data Guard BrokerはOracle Clusterware（またはシングル・インスタンス・データベース向けのOracle Restart）と連携して、新しいプライマリ・データベースにロールベースのサービスを適切にフェイルオーバーします。

データベース・フェイルオーバー・プロセスの完全な自動化には、Data Guardファスト・スタート・フェイルオーバー（FSFO）[14]をお薦めします。FSFOによって、プライマリ・データベースが使用できなくなった場合にフェイルオーバーを実行する際に管理者の操作の必要がなくなります。FSFOは、データ保護と可用性の目標を確実に達成する、緊密に制御されたプロセスです。FSFOでは、スタンバイ・データベースへのフェイルオーバーにアプリケーション継続性の使用を必要とします。

注：Oracle Database 12cの新機能のOracle Global Data Services（Oracle GDS）[15]では、Data GuardやOracle GoldenGateなどの、ワークロード・ロードバランシング向けのグローバル・メカニズムやレプリケートされたデータベース向けのデータベース・サービス管理が提供されます。Oracle GDSは、Oracle RACおよびData Guardと同様のイベント通知フレームワークを使用すると同時に、イベント処理やサービス管理のための独自のメソッドを提供します。ここでは、Data Guard Broker、Oracle Clusterware、またはこのホワイト・ペーパーで説明したようにOracle Restartは使用されません。Oracle GDSには、このホワイト・ペーパーで説明するものと同様のベスト・プラクティスが多く含まれますが、Oracle GDSはクライアント・フェイルオーバーの自動化への代替アプローチを表しており、これはこのホワイト・ペーパーの対象範囲外です。

FANイベントを受信してこれに対応するには、クライアント・アプリケーションおよび接続プールが以下の要件を満たす必要があります。

JDBCアプリケーション

- 暗黙的接続キャッシュが有効になっていること。
- データベースに接続するためのサービス名がアプリケーションで使用されていること。
- 基礎になるデータベースがOracle RACである、または基礎になるデータベースにOracle Restart機能（シングル・インスタンス・データベース向け）が含まれていること。
- Oracle Notification Serviceが構成されており、JDBCが実行されているノード上で使用できること。
- JDBCインスタンスが実行されているJava仮想マシン（JVM）で、ORACLE_HOMEを指すようにoracle.ons.oraclehomeが設定されていること。

詳しくは、『Oracle Universal Connection Pool for JDBC開発者ガイド』[16]を参照してください。

[13] http://docs.oracle.com/cd/E57425_01/121/DGBKR/toc.htm

[14] http://docs.oracle.com/cd/E57425_01/121/DGBKR/sofo.htm#i1005572

[15] http://www.oracle.com/technetwork/database/features/availability/global-data-services-1949717.html

[16] http://docs.oracle.com/cd/E57425_01/121/JJUCP/rac.htm#CHDCDFAC

OCIアプリケーション

» Oracle RAC環境にOracle Clusterwareが設定され有効化されている、またはシングル・ノード（Oracle RAC以外）データベースにOracle Restartが設定され有効になっていること。

» アプリケーションがスレッド・ライブラリにリンクしていること。

» OCI環境がOCI_EVENTSモードおよびOCI_THREADEDモードで作成されていること。詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』[17]を参照してください。

ODP.Net

» ネームスペース：Oracle.DataAccess.Client、アセンブリ：Oracle.DataAccess.dll

» Microsoft .NET Frameworkバージョン2.0以上

詳細は、『Oracle Database Administrator's Guide』[18]を参照してください。

ロール・ベースのデータベース・サービス

Data Guard 11gリリース2（11.2）以降では、各サービスにデータベース・ロール[-l {[PRIMARY] | [PHYSICAL_STANDBY] | [LOGICAL_STANDBY] | [SNAPSHOT_STANDBY]}]を割り当てることで、プライマリ・データベースとスタンバイ・データベースでのデータベース・サービスの起動を自動的に制御できます[19]。サービスの管理ポリシーがAUTOMATICに設定されており、サービスに割り当てられたロールのいずれかが現在のデータベース・ロールと一致している場合、データベースの起動時にデータベース・サービスが自動的に開始されます。

これらのサービスは、Server Control Utilityを使用して、Data Guard構成に含まれるすべてのデータベース上でまったく同じように構成される必要があります。次の例では、データベースAustinがプライマリ・ロール（-l PRIMARY）にある場合、oltpworkloadという名前のサービスがアクティブになるよう構成されています。同じサービスがスタンバイ・データベースHouston上にも構成されており、Houstonがプライマリ・ロールとして機能する場合はいつでもこのサービスが開始されるようになっています。

同様に、データベースAustinまたはHoustonがスタンバイ・データベース・ロール（-l PHYSICAL_STANDBY）として機能する場合、reportsという名前の2番目のサービスが開始されるよう構成されています。reportsサービスはActive Data Guardを使用して、リアルタイムのレポーティングを提供します（プライマリ・データベースから受信したREDOをスタンバイ・データベースに適用している間、スタンバイ・データベースは読取り専用でオープンされます）。

[17] https://docs.oracle.com/cd/E16338_01/appdev.112/e50264/oci09adv.htm#i441210

[18] http://download.oracle.com/docs/cd/E11882_01/server.112/e17120/restart002.htm#ADMIN13196

[19] http://download.oracle.com/docs/cd/E11882_01/rac.112/e16795/hafeats.htm#RACAD7126

JDBCクライアントの自動フェイルオーバーの構成

前提条件

» Universal Connection Pool（UCP）が有効になっていること（UCP 12.1.0.2以上）。

» データベースに接続するためのサービス名がアプリケーションで使用されていること。

» Oracle Notification Serviceが構成されており、JDBCが実行されているノード上で使用できること。

» JDBCインスタンスが実行されているJava仮想マシン（JVM）で、ORACLE_HOMEを指すようにoracle.ons.oraclehomeが設定されていること。

構成

1. 高速接続フェイルオーバー（FCF）を有効にし、setONSConfigurationプロパティを使用して、プライマリ・クラスタおよびスタンバイ・クラスタに対するすべてのONSデーモンに接続するようにJDBCアプリケーションを構成します。setONSConfigurationプロパティは、プライマリおよびスタンバイのすべてのONSデーモンを指している必要があります。

```
pds.setONSConfiguration("nodes=adczatdb01:6200,adczatdb02:6200,slcc17adm01:6200,slc c17adm02:6200");
pds.setFastConnectionFailoverEnabled(true);
```

2. デフォルトでは、JDBCアプリケーションは、setONSConfigurationプロパティから3つのホストを無作為に選択し、それらの3つのONSデーモンへの接続を作成します。このデフォルトは、すべてのONSデーモンへの接続が作成されるように変更する必要があります。これを実行するには、JDBCアプリケーションの起動時に、以下のプロパティを構成内のONSデーモンの総数に設定します。

```
java -Doracle.ons.maxconnections=4
```

3. JDBCクライアントでは、oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STRプロパティを設定する必要があります。このプロパティによって、障害発生時にJDBCクライアントはADDRESS_LISTを迅速に横断できます。たとえば、使用できないホストにクライアントが接続しようとした場合、接続の試行はSQLnetDef.TCP_CONNTIMEOUT_STRプロパティで指定された時間に制限され、そののちにクライアントはADDRESS_LIST内の次のホストに接続を試みます。この動作は、ADDRESS_LIST内にある各ホストに対して接続が作成されるまで継続されます。ほとんどの環境では、プロパティの値を3秒に設定すれば十分です。SQLnetDef.TCP_CONNTIMEOUT_STRプロパティは、Universal Connection Pool上ではなく、データソース上で設定する必要があることに注意してください。

```
Properties prop = new Properties();
prop.put(oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR, ""+3000); // 3000ms
pds.setConnectionProperties(prop);
```

4. SCANロードバランシングから適切な動作を取得できるように以下を設定します。

```
// need to set oracle.jdbc.thinForceDNSLoadBalancing
prop.put("oracle.jdbc.thinForceDNSLoadBalancing","true");
```

5. JDBCクライアントを、各サイトのSCANアドレスが順番に記載され、既存のサービスに接続するためのアドレス・リストを含む接続記述子を使用するように構成します。JDBCシック・クライアントを使用する場合は、TAFとJDBC FCFの両方を構成しないでください。

以下のURLでプライマリとスタンバイのサイトを検索し、オーバーヘッドがほとんど発生しない適切なサービスを探します。アプローチとしてこのURL構成をお薦めします。

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setUser("system");
 pds.setPassword("oracle");
 String dbURL =
 "jdbc:oracle:thin:@" +
 "(DESCRIPTION=" +
"(FAILOVER=on)" +
"(ADDRESS_LIST=" +
 "(LOAD_BALANCE=on)" +
 "(CONNECT_TIMEOUT=3)(RETRY_COUNT=3)" +
 "(ADDRESS=(PROTOCOL=TCP)(HOST=prmy-scan)(PORT=1521))”+
"(ADDRESS=(PROTOCOL=TCP)(HOST= stby-scan)(PORT=1521)))" +
 "(CONNECT_DATA=(SERVICE_NAME=oltpworkload)))"
 System.out.println("Url=" + dbURL);
 pds.setURL(dbURL);
```

プライマリがこれまでにセカンダリ・サイト上でほとんど実行されず、可能な限り迅速に接続をつなげる必要がある場合は、以下のURLを使用します。

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setUser("system");
pds.setPassword("oracle");
 String dbURL =
 "jdbc:oracle:thin:@" +
 "(DESCRIPTION_LIST=" +
 “(LOAD_BALANCE=off)” +
 "(FAILOVER=on)" +
 "(DESCRIPTION=" +
 "(CONNECT_TIMEOUT=3)(RETRY_COUNT=3)" +
 "(ADDRESS_LIST=" +
 "(LOAD_BALANCE=on)" +
 "(ADDRESS=(PROTOCOL=TCP)(HOST=prmy-scan)(PORT=1521)))" +
 "(CONNECT_DATA=(SERVICE_NAME=oltpworkload)))" +
 "(DESCRIPTION=" +
 "(ADDRESS_LIST=" +
 "(LOAD_BALANCE=on)" +
 "(ADDRESS=(PROTOCOL=TCP)(HOST= stby-scan)(PORT=1521)))" +
 "(CONNECT_DATA=(SERVICE_NAME=oltpworkload))))";
 System.out.println("Url=" + dbURL);
 pds.setURL(dbURL);
```

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、上記のURLは、プライマリが現在稼働しているstby-scanを使用する前に、強制的にすべての接続を古いprmy-scanに通過させることに注意してください。

上記のURLを使用して新しい接続が実行される場合、次のロジックが使用されます。

» Oracle NetはDNSを参照してprmy-scanを解決し、合計3つのIPアドレスを取得します。

» Oracle Netは3つのIPアドレスから1つを無作為に選択し、接続を試行します。3つすべてのIPアドレスに対して4回（最初の試行と上記例のRETRY_COUNTの合計数）の試行が行われます。

» プライマリ・サイトへの接続が成功しない場合、DNSを参照してstby-scanを解決し、3つのアドレスを取得します。

» prmy-scanに対して実行されたものと同じシーケンスが、stby-scanに対して実行されます。

以下に、上記のエイリアスで使用されるOracle Netパラメータの追加情報を列記します。

» LOAD_BALANCEは、DESCRIPTION_LISTに対してのみデフォルトではONです。このパラメータは、DESCRIPTION内のアドレス・リストに対してデフォルトではOFFです。これをSCANベースのアドレスに対してONに設定すると、新しい接続がDNSによって解決されるSCANベースの3つのIPアドレスのうちの1つに無作為に割り当てられることを意味します。

» FAILOVERパラメータのデフォルト値は、DESCRIPTION内のアドレス・リストに対してONです。これは、これらの3つのIPアドレスが接続記述子に明示的に表示された場合と同様に3つのSCAN IPアドレスに影響します。つまり、最初に無作為に割り当てられたSCAN IPアドレスに対する初回接続のリクエストが失敗した場合、接続は別のSCAN IPアドレスにフェイルオーバーし、アドレス・リスト全体に繰り返されるまでその動作を継続します。このパラメータは、新しい接続のみに関連することに注意してください。既存の接続のフェイルオーバーは、個別のFAILOVER_MODEパラメータによって制御される、TAFによって処理されます。

» CONNECT_TIMEOUTパラメータは、リクエストされたサービスを提供するデータベース・インスタンスへ接続する時間を示すもので、リスナーへのTCP接続を確立する時間を含みます。タイムアウト時間は、ADDRESS_LIST内の各ADDRESS、およびホスト名がマッピングされている各IPアドレスに適用できます。CONNECT_TIMEOUTパラメータを、次のアドレスにスキップする前に、アドレスからのレスポンスを待機するための最大時間（秒単位）に設定します。お薦めの設定は3秒で、これはほとんどの場合で許容可能です。このパラメータを3秒未満に設定しないでください。

  sqlnet.oraの同等のグローバル・パラメータは、SQLNET.OUTBOUND_CONNECT_TIMEOUTです。すべての接続文字列に対して同じタイムアウト値で十分な場合は、グローバル・パラメータを設定するよりも簡単です。そうでない場合は、おのおのの接続文字列に対して個別に設定できます。

» RETRY_COUNTパラメータは、新しい接続の試行が終了するまでにアドレス・リストが繰り返される回数を指定します。デフォルト値は0です。たとえば、SCANについてFAILOVER = onの場合、このRETRY_COUNTパラメータの値を2に設定すると、接続が終了するまでに3つのSCAN IPアドレスが3回繰り返されることになります（3*3=9回の接続の試行）。

» 接続の初回リクエストが発生すると、最初に無作為に割り当てられたIPアドレスがそのリクエストに対応しようとし、そのあとに残りの2つのIPアドレスが続きます（この動作は、FAILOVERパラメータによって制御されます）。

» その後再試行が行われ、3つのIPアドレスのリストがあと2回試行されます。RETRY_COUNTは、接続文字列のDESCRIPTIONレベルでのみサポートされ、グローバル（sqlnet.oraなど）レベルではサポートされません。

アプリケーション継続性とトランザクション・ガードの構成

アプリケーション継続性が登場する以前は、あらゆるエラーの検出、マスク、および対応は開発者の責任でした。多くの場合、これを実現することは困難で、アプリケーション全体のエラー、タイムアウト、そして生産性の低下につながることがありました。アプリケーション継続性は、実行中のトランザクションおよびリクエストをリカバリし、エンドユーザーおよびアプリケーションから停止を自動的にマスクすることによって、アプリケーション開発者をこの責任から解放します。リカバリはすべてアプリケーションに対して透過的に実行されるため、エンドユーザーはわずかな実行の遅れに気づくだけです。これは、計画停止と計画外停止の両方に適用されます。

Oracle Database 12cでのアプリケーション継続性は、Oracle RACインスタンス間、またはData Guardのプライマリおよびスタンバイ・データベース間でローリング方式で実行される計画保守の停止をマスクするために使用されます。アプリケーション継続性は、Oracle RACインスタンスまたはData Guardファスト・スタート・フェイルオーバー（自動データベース・フェイルオーバー）によってMaximum Availability（データ損失ゼロのフェイルオーバー）で構成されたData Guardプライマリ・データベースの計画外停止もマスクします。Data Guardフェイルオーバーにアプリケーション継続性を使用するには、ソース・データベースとターゲット・データベースの両方が Oracle 12.1.0.2以降にあることが必要です。

アプリケーション継続性は以下で利用できます。

- Oracle JDBC Replay Driver 12c以降。これは、Oracle Database 12cに付属する、アプリケーション継続性のためのJDBCドライバ機能です。これ以降は"再実行ドライバ"と呼びます（今後のリリースではOCIサポートが計画されています）。
- Oracle Universal Connection Pool、Oracle WebLogic Server 12c（12.1.2）以降、およびOracle JDBC- Replay Driver 12c以降を使用するサード・パーティのJava接続プールまたはスタンドアロンJavaアプリケーション。
- 12.1.0.2以降では、IBM WebSphere、Apache Tomcatなど、プールされた接続インタフェースを使用する標準のサード・パーティのJavaアプリケーション・サーバー。
- プールされたデータソースとしてUniversal Connection Poolをサポートする、標準のサード・パーティのJavaアプリケーション・サーバー。IBM WebSphere、Apache Tomcat、RedHat JBossなど。
- Oracle JDBC- Replay Driver 12c以降を使用し、その独自のリクエスト境界を組み込む、スタンドアロンのJavaアプリケーション、またはサード・パーティのJava接続プール

アプリケーション継続性[20]は、トランザクション・ガード[21]を使用して、最後のトランザクションがコミットされたかどうかを確実に判断します。トランザクション・ガードがないと、停止後にアプリケーションやユーザーによって操作が再試行され、重複したり、順序が正しくなかったりするトランザクションをコミットすることで、論理的破損が発生する可能性があります。

以下のオプションから1つを選択し、ユーザーの構成に基づいて、Oracle JDBC 12c Replay Driverを使用してアプリケーション継続性を構成します。

[20] http://www.oracle.com/technetwork/jp/database/database-cloud/application-continuity-wp-12c-1966213-ja.pdf

[21] http://www.oracle.com/technetwork/database/database-cloud/private/transaction-guard-wp-12c-1966209.pdf

Oracle UCP 12cの構成

» UCP PoolDataSourceに対して、Oracle JDBC 12c再実行データソースを接続ファクトリとして構成します。
setConnectionFactoryClassName("oracle.jdbc.replay.OracleDataSourceImpl");

Oracle WebLogic Server 12cの構成

» Oracle WebLogic Server管理コンソールを使用して、Oracle 12c JDBC再実行データソースを構成します。

スタンドアロンJavaアプリケーションまたはサード・パーティ接続プールの構成

» プロパティ・ファイル内またはThin JDBCアプリケーション内で、Oracle JDBC 12c再実行データソースを構成します。
replay datasource=oracle.jdbc.replay.OracleDataSourceImpl

高可用性のための接続の構成（フェイルオーバーおよびフェイルバック）

» データベースのREMOTE_LISTENER設定に、クライアント接続に使用するすべてのURLに関するADDRESS_LIST内のアドレスが含まれている必要があります。

  » SCAN名を使用しているURLがある場合は、REMOTE_LISTENERSにそのSCAN名が含まれている必要があります。

  » ホストVIPのADDRESS_LISTを使用しているURLがある場合は、REMOTE_LISTENERSに、すべてのSCAN VIPとすべてのホストVIPを含むADDRESSリストが含まれている必要があります。

» URL内でRETRY_COUNT、CONNECT_TIMEOUTの各パラメータを設定し、新しい着信接続要求の再試行を許可します。この属性に関する詳細は、このホワイト・ペーパーの「JDBCアプリケーションの構成」の項を参照してください。

例

```
  "jdbc:oracle:thin:@"+
  "(DESCRIPTION=" +
"(FAILOVER=on)" +
  "(ADDRESS_LIST=" +
  "(LOAD_BALANCE=on)" +
  "(CONNECT_TIMEOUT=3)(RETRY_COUNT=3)" +
"(ADDRESS=(PROTOCOL=TCP)(HOST=prmy-scan)(PORT=1521))"+ "(ADDRESS=(PROTOCOL=TCP)(HOST= s
tby-scan)(PORT=1521)))" +
  "(CONNECT_DATA=(SERVICE_NAME=oltpworkload)))"
```

アプリケーション継続性向けのサービスの構成

» アプリケーション継続性を使用するには、SRVCTL / GDSCTLを使用してサービス属性を設定します。

» FAILOVER_TYPEをTRANSACTIONに設定すると、アプリケーション継続性が有効化されます。

» COMMIT_OUTCOMEをTRUEに設定して、トランザクション・ガードを有効化します（必須）。

» 次のサービス属性を確認します。

  » REPLAY_INITIATION_TIMEOUT：再実行の開始を許可する期間（秒）を設定します（80、300、1800秒など。この期間が過ぎると再実行がキャンセルされます）。このタイマーはbeginRequestの時点で開始します（デフォルトは300秒）。

  » FAILOVER_RETRIES：再実行のそれぞれの試行における接続の再試行回数を指定します（デフォルトの再試行回数は30回。再実行ドライバに適用されます）。

  » FAILOVER_DELAY：接続の再試行から次の再試行までの間隔（秒）を指定します（デフォルトは10秒。再実行ドライバに適用されます）。

  » AQ_HA_NOTIFICATIONS：TRUEに設定すると、FANが有効化されます（デフォルトはTRUE）。

  » 例

  srvctl add service -db mts -service oltpworkload -role PRIMARY -notification TRUE -session_state dynamic -failovertype transaction -failovermethod basic - commit_outcome TRUE -failoverretry 30 -failoverdelay 10 -replay_init_time 900 - clbgoal SHORT -rlbgoal SERVICE_TIME -preferred mts1,mts2-retention 3600 – verbose

リソース割当てのチェック

» 必要となるメモリ・リソースとCPUリソースがシステムに搭載されていることを確認します。

» メモリ：JDBC再実行ドライバでは、データベース・リクエストの終了までコールが保持されるため、ベースのJDBCドライバよりも多くのメモリを消費します。保持されるコール数が少ない場合は、再実行ドライバによるメモリ消費量はベース・ドライバとほぼ同じです。リクエストの終了時に、コールはガベージ・コレクタに解放されます。このアクションは、コールのクローズ時に解放されるベース・ドライバとは異なります。

» 良好なパフォーマンスを得るために、十分なメモリがある場合は、4～8GB（またはそれ以上）のメモリを仮想マシン（VM）に割り当ててください。たとえば、4GBの場合は、- Xms4096mと設定します。

» CPU：JDBC再実行ドライバでは、プロキシ・オブジェクトの作成、キューの管理、ガベージ・コレクションのために追加のCPUが使用されます。サーバーでは妥当性確認管理のために追加のCPUが使用されます。CPUオーバーヘッドは赤で表示されます。

自動フェイルオーバーのためのOCIクライアントの構成

前提条件

» Oracle RAC環境にOracle Clusterwareが設定され有効化されている、またはシングル・ノード（Oracle RAC以外）データベースにOracle Restartが設定され有効になっていること。

» アプリケーションがスレッド・ライブラリにリンクしていること。

» OCI環境がOCI_EVENTSモードおよびOCI_THREADEDモードで作成されていること。

構成

以下のように、OCI_EVENTSパラメータを使用して環境を初期化することにより、OCIクライアントのFANを有効化します。OCIEnvCreate(...OCI_EVENTS...)

1. OCIクライアント・アプリケーションをスレッド・ライブラリlibthreadまたはlibpthreadとリンクさせます。
2. 以下の例と同様のコードを使用して、イベントが発生したかどうかをアプリケーションがチェックできるようにします。

```
void evtcallback_fn(ha_ctx, eventhp)
...
printf("HA Event received.\n");
  if (OCIHandleAlloc( (dvoid *)envhp, (dvoid **)&errhp, (ub4) OCI_HTYPE_ERROR,
            (size_t) 0, (dvoid **) 0))
    return;
  if (retcode = OCIAttrGet(eventhp, OCT_HTYPE_EVENT, (dvoid *)&srvhp, (ub4 *)0,
                OCI_ATTR_HA_SRVFIRST, errhp))
    checkerr (errhp, (sword)retcode;
  else {
     printf("found first server handle.\n");
     /*get associated instance name */
     if (retcode = OCIAttrGet(srvhp, OCI_HTYPE_SERVER, (dvoid *)&instname,
                    (ub4 *)&sizep, OCI_ATTR_INSTNAME, errhp))
       checkerr (errhp, (sword)retcode);
     else
       printf("instance name is %s.\n", instname);
```

3. クライアントおよびアプリケーションでは、以下の例に示すように、高可用性イベントが発生するたびに起動するコールバックを登録できます。

```
/*Registering HA callback function */
  if (checkerr(errhp, OCIAttrSet(envhp, (ub4) OCI_HTYPE_ENV,
                 (dvoid *)evtcallback_fn, (ub4) 0,
                 (ub4)OCI_ATTR_EVTCBK, errhp)))
  {
    printf("Failed to set register EVENT callback.\n");
    return EX_FAILURE;
  }
  if (checkerr(errhp, OCIAttrSet(envhp, (ub4) OCI_HTYPE_ENV,
                     (dvoid *)evtctx, (ub4) 0,
                     (ub4)OCI_ATTR_EVTCTX, errhp)))
  {
    printf("Failed to set register EVENT callback context.\n");
    return EX_FAILURE;
  }
return EX_SUCCESS;
```

イベントのコールバックとコンテキストを登録したのち、OCIは高可用性イベントが発生するたびに登録された関数を1回コールします。

4. OCIアプリケーションでデータベースへの接続に使用されるOracle Netエイリアスを構成します。Oracle Netエイリアスでは、プライマリおよびスタンバイの両方のSCANホスト名を指定する必要があります。新しい接続の作成中に最高のパフォーマンスを得るには、Oracle NetエイリアスがDESCRIPTION_LISTに対してLOAD_BALANCE=OFFを保持し、リストの順番通りにDESCRIPTIONが上から下へ試行されるようにします。この構成では、2番目のDESCRIPTIONは、1番目のDESCRIPTIONに対する接続の試行がすべて失敗した場合にのみ試行されます。

```
SALES=
 (DESCRIPTION_LIST=
  (LOAD_BALANCE=off)
  (FAILOVER=on)
  (DESCRIPTION=
(CONNECT_TIMEOUT=5)(TRANSPORT_CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
    (ADDRESS_LIST=
      (LOAD_BALANCE=on)
      (ADDRESS=(PROTOCOL=TCP)(HOST=prmy-scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=oltpworkload)))
  (DESCRIPTION=
(CONNECT_TIMEOUT=5)(TRANSPORT_CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
    (ADDRESS_LIST=
      (LOAD_BALANCE=on)
      (ADDRESS=(PROTOCOL=TCP)(HOST=stby-scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=oltpworkload))))
```

上記のOracle Netエイリアスを使用して新しい接続が実行される場合、次のロジックが使用されます。

» Oracle NetはDNSを参照してprmy-scanを解決し、合計3つのIPアドレスを取得します。

» Oracle Netは3つのIPアドレスから1つを無作為に選択し、接続を試行します。このIPアドレスへの接続試行に対して3秒以内（TRANSPORT_CONNECT_TIMEOUT）に応答がない場合、次のIPアドレスが試行されます。3つすべてのIPアドレスに対して4回（最初の試行と上記例のRETRY_COUNTの合計数）の試行が行われます。

» プライマリ・サイトへの接続が成功しない場合、DNSを参照してstby-scanを解決し、3つのアドレスを取得します。

» prmy-scanに対して実行されたものと同じシーケンスが、stby-scanに対して実行されます。

以下に、上記のエイリアスで使用されるOracle Netパラメータの追加情報を列記します。

» LOAD_BALANCEは、DESCRIPTION_LISTに対してのみデフォルトではONです。このパラメータは、DESCRIPTION内のアドレス・リストに対してデフォルトではOFFです。これをSCANベースのアドレスに対してONに設定すると、新しい接続がDNSによって解決されるSCANベースの3つのIPアドレスのうちの1つに無作為に割り当てられることを意味します。

» FAILOVERパラメータのデフォルト値は、DESCRIPTION内のアドレス・リストに対してONです。これは、これらの3つのIPアドレスが接続記述子に明示的に表示された場合と同様に3つのSCAN IPアドレスに影響します。つまり、最初に無作為に割り当てられたSCAN IPアドレスに対する初回接続のリクエストが失敗した場合、接続は別のSCAN IPアドレスにフェイルオーバーし、アドレス・リスト全体に繰り返されるまでその動作を継続します。このパラメータは、新しい接続のみに関連することに注意してください。既存の接続のフェイルオーバーは、個別のFAILOVER_MODEパラメータによって制御される、TAFによって処理されます。

CONNECT_TIMEOUTパラメータは、リクエストされたサービスを提供するデータベース・インスタンスへ接続する時間を示すもので、リスナーへのTCP接続を確立する時間を含みます。TCPの継続時間は、TRANSPORT_CONNECT_TIMEOUTによって制御され、このデフォルト値は60秒です。両方のタイムアウトを指定する場合、CONNECT_TIMEOUTの値をTRANSPORT_CONNECT_TIMEOUTよりも少し大きく設定することをお薦めします。タイムアウト時間は、ADDRESS_LIST内の各ADDRESS、およびホスト名がマッピングされている各IPアドレスに適用できます。CONNECT_TIMEOUTパラメータを、次のアドレスにスキップする前に、アドレスからのレスポンスを待機するための最大時間（秒単位）に設定します。お薦めの設定は最低3秒で、これはほとんどの場合で許容可能です。sqlnet.oraの同等のグローバル・パラメータは、SQLNET.OUTBOUND_CONNECT_TIMEOUTです。すべての接続文字列に対して同じタイムアウト値で十分な場合は、グローバル・パラメータを設定するよりも簡単です。そうでない場合は、おのおのの接続文字列に対して個別に設定できます。

TRANSPORT_CONNECT_TIMEOUTと同等のグローバル・パラメータは、TCP.CONNECT_TIMEOUTです。これらのパラメータは、プロトコルがTCPの場合にのみ適用できます。

» RETRY_COUNTパラメータは、新しい接続の試行が終了するまでにアドレス・リストが繰り返される回数を指定します。デフォルト値は0です。たとえば、SCANについてFAILOVER = onの場合、このRETRY_COUNTパラメータの値を2に設定すると、接続が終了するまでに3つのSCAN IPアドレスが3回繰り返されることになります（3*3=9回の接続の試行）。

- 接続の初回リクエストが発生すると、最初に無作為に割り当てられたIPアドレスがそのリクエストに対応しようとし、そのあとに残りの2つのIPアドレスが続きます（この動作は、FAILOVERパラメータによって制御されます）。
- その後再試行が行われ、3つのIPアドレスのリストがあと2回試行されます。RETRY_COUNTは、接続文字列のDESCRIPTIONレベルでのみサポートされ、グローバル（sqlnet.oraなど）レベルではサポートされません。

ODP.Netクライアントの自動フェイルオーバーの構成

前提条件

上記の「自動フェイルオーバーのためのOCIクライアントの構成」の項の説明に従ってOracle Netエイリアスを構成します。

追加構成

1. FAN高可用性イベントをサブスクライブすることによって、ODP.NET接続プールの高速接続フェイルオーバーを有効にします。高速接続フェイルオーバーを有効にするには、以下の例に示すように、"HA Events=true"および"pooling=true"を接続文字列に含めます。ここでは、user_nameはデータベース・ユーザーの名前、password はそのユーザーのパスワードです。

   ```
   con.ConnectionString =
       "User Id=user_name;Password=password;Data Source=sales;" +
       "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
        "HA Events=true;Incr Pool Size=5;Decr Pool Size=2";
   ```

2. ODP.NET接続プールでロードバランシング・イベントを利用するには、ConnectionStringのロードバランシング属性をTRUEに設定します（デフォルトはFALSEです）。これは、接続時に実行できます。これは、接続プールを使用している場合、またはプーリング属性がTRUE（デフォルト）に設定されている場合にのみ実行できます。

   次の例に、てロードバランシングを有効にするConnectionStringの構成方法を示します。

   ```
   con.ConnectionString =
       "User Id=user_name;Password=password;Data Source=odpapp;" +
       "Min Pool Size=10;Connection Lifetime=120;Connection Timeout=60;" +
       "Load Balancing=true;Incr Pool Size=5;Decr Pool Size=2";
   ```

3. FANイベントとHAイベントを有効にするようにサービスを構成します。これについては、このホワイト・ペーパーの次の項で説明します。

Oracleデータベース・サービスの構成

プライマリ・ホストおよびスタンバイ・ホスト上で、このホワイト・ペーパーで示す例を使用して、データベースへの接続にアプリケーションが使用するサービス（oltpworkload）を作成します。また、データベースが'PRIMARY'データベース・ロールである場合に、このデータベース上でサービスが実行されるように関連付けて作成します。

注：FANを有効にしたJDBCデータベース・サービスに対して、透過的アプリケーション・フェイルオーバー（TAF）を有効化しないでください。

以下に、前述のJDBCアプリケーションに対するサービスと同様のサービスを作成する例を示します。

» プライマリ・クラスタ

```
srvctl add service -db austin -service oltpworkload -preferred ssa1,ssa2,ssa3,ssa4
-role PRIMARY -notifications TRUE -tafpolicy NONE –failovermethod NONE - failoverdelay 0 -failoverretry 0
```

» スタンバイ・クラスタ

```
srvctl add service -db houston -service oltpworkload -preferred ssa1,ssa2,ssa3,ssa4
-role PRIMARY -notifications TRUE -tafpolicy NONE –failovermethod NONE - failoverdelay 0 -failoverretry 0
```

OCIおよびODP.Netアプリケーションで使用されるサービスについては、上記のsrvctlコマンドをわずかに変更し、透過的アプリケーション・フェイルオーバーをサービス・レベルで有効にします。

» プライマリ・クラスタ

```
srvctl add service -db austin -service oltpworkload -preferred ssa1,ssa2,ssa3,ssa4
-role PRIMARY -notifications TRUE -tafpolicy BASIC –failovermethod BASIC - failoverdelay 3 -failoverretry 90
```

» スタンバイ・クラスタ

```
srvctl add service -db houston -service oltpworkload -preferred ssa1,ssa2,ssa3,ssa4
-role PRIMARY -notifications TRUE -tafpolicy BASIC –failovermethod BASIC - failoverdelay 3 -failoverretry 90
```

ロール移行のタイミング

Oracle Data Guardのロール移行が完了するまでの時間はOracle Databaseが新しくリリースされるたびに短縮されてきました。新規のプライマリ・データベースがオープンして、新規のトランザクションを処理できるようになったのち、余計な時間がかかり、重要でないタスクをフェイルオーバー後まで延期するプロセスの手順を除外する拡張機能によってこれは実現しました。

ロール移行プロセスは、単一のDDLを使用してスイッチオーバーおよびフェイルオーバーをOracle Database 12cで実行することによって明らかに簡素化できます。次に例を示します。

» 計画イベントの場合：SQL> ALTER DATABASE SWITCHOVER TO <target db>

» 計画外イベントの場合：SLQ> ALTER DATABASE FAILOVER TO <target db>

Oracle Database 12cでは、Data Guardスタンバイが、スイッチオーバーが実際に実行される前にス

イッチオーバーが成功する状態にあることを迅速に確認するための機能も改善されています。

» 新規のスイッチオーバー文の確認オプションを使用してSQL*Plusから確認する方法

SQL> ALTER DATABASE SWITCHOVER TO <target db> VERIFY;

» Data Guard Brokerでは、拡張された一連の条件の評価に、SQL*Plusを使用して検証できるよりもさらに包括的な検証が実行されます。Data Guard Brokerコマンドは次のとおりです。

DGMGRL> VALIDATE DATABASE [VERBOSE] database-name;[22]

Oracle MAAのベスト・プラクティスでは、Data Guard構成の管理にData Guard Broker[23]を使用することを常にお薦めします。Data Guard Brokerは、コマンドライン・インタフェースのDGMGRL経由、またはEnterprise Managerを使用して利用します。フェイルオーバー操作は、手動で、またはファスト・スタート・フェイルオーバーを使用して自動的に実行できます。これらの機能は、Data Guard Brokerに含まれます。

データベース・スイッチオーバーのタイミング

Oracle MAAテストでは、さまざまな数のアクティブなデータベース接続を用いて、シングル・インスタンス（Oracle RAC以外）とOracle RACデータベースの両方についてスイッチオーバーのタイミングを検証しました。また、テストでは、アクティブなスタンバイ・データベースで読取り専用の接続が発生した場合に必要とされる追加時間を計測しました。図1に示す結果からは、スイッチオーバー処理の各フェーズ時の経過時間が分かります。

Oracle RACプライマリ（読取り/書込みでオープン）とアクティブ・スタンバイ（読取り専用でオープン）の両方で1,000の接続があるもっとも複雑なテスト・ケースでは、オリジナルのプライマリ・データベースがクローズしてから新しいプライマリが新規接続に対してオープンするまでにおよそ45秒かかりました。

マウント・モード（Active Data Guardなし）におけるシングル・インスタンスのスタンバイというよりシンプルなテスト・ケースでは、スイッチオーバーはおよそ12秒で完了しました。

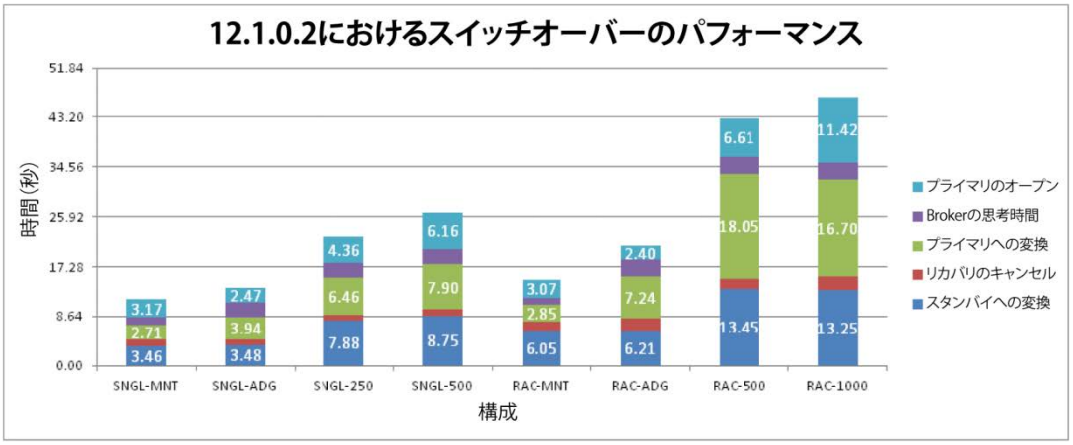

図1：Data Guardによるスイッチオーバーのパフォーマンス - スイッチオーバー処理の各フェーズにおけるタイミングの詳細

[22] https://docs.oracle.com/cd/E49329_01/server.121/b71305/dgmgrl.htm#i80635

[23] https://docs.oracle.com/cd/E49329_01/server.121/b71305/toc.htm

データベース・フェイルオーバーのタイミング

フェイルオーバーに関して、一連の同様のテストを実行しました。突然の停止では、スタンバイをプライマリ・ロールに移行するために必要とされる処理が少ないため、フェイルオーバーはスイッチオーバーよりも速く完了します。図2に示した結果からは、フェイルオーバー処理の各フェーズの経過時間が分かります。

Oracle RACプライマリ（読取り/書込みでオープン）とアクティブ・スタンバイ（読取り専用でオープン）の両方で1,000の接続がある複雑な同じテスト・ケースでは、オリジナルのプライマリ・データベースの停止が検出されてから新しいプライマリが新規接続に対してオープンするまでにおよそ26秒かかりました。

マウント・モード（Active Data Guardなし）におけるシングル・インスタンスのスタンバイというよりシンプルなテスト・ケースでは、フェイルオーバーはおよそ8秒で完了しました。

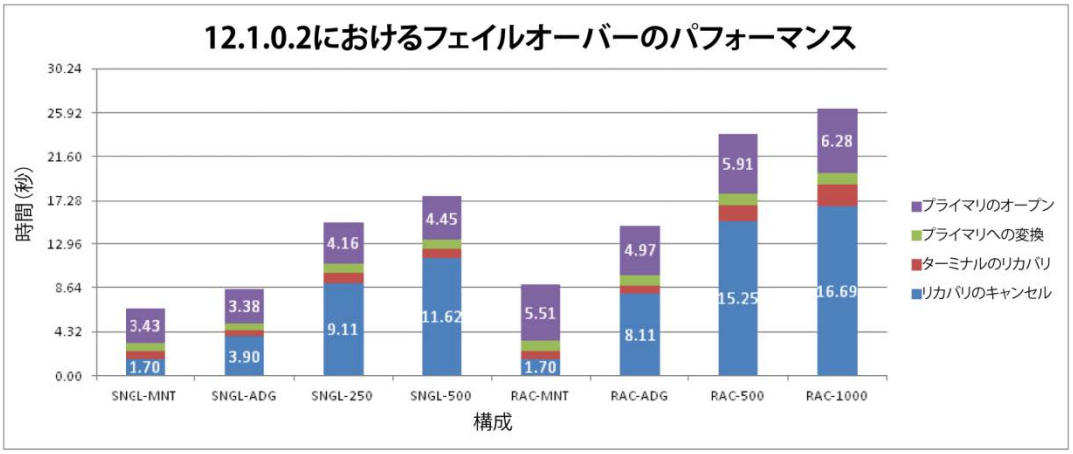

図2：Data Guardによるフェイルオーバーのパフォーマンス - スイッチオーバー処理の各フェーズにおけるタイミングの詳細

アプリケーション継続性によるエンド・ツー・エンドのアプリケーション・フェイルオーバーのタイミング

最後のテストは、Data Guardのフェイルオーバー時に、アプリケーション継続性を用いてアプリケーション接続を移行するために必要な時間を計測するために行われました。このテスト・ケースでは、アプリケーション接続性はエラーを受信せず、実行中のトランザクションは新規のプライマリ・データベースで自動的に再実行されます。

図3は、アクティブ接続が1,000ある場合の、Oracle RACデータベースでのData Guardによるフェイルオーバーのテスト時におけるアプリケーション・スループットの内容とセッション数を示したものです（スタンバイ・データベースでの読取り専用接続はありませんでした）。

このテスト・シナリオでは、次の一連のイベントが使用されました。

1. プライマリのOracle RACデータベースの停止をシミュレートするために、シャットダウンの中断が使用されました。図3は、データベース・スループット（1秒あたりのトランザクション/TPS）

およびアクティブ接続数がゼロになる速度を示しています。

2. スタンバイ・データベースを読取り/書込みモードでオープンされたプライマリ・データベース・ロールへ変換するデータ損失ゼロのフェイルオーバーは24秒で完了します。
3. 新規のプライマリは、プライマリ・ロールに適したすべてのデータベース・サービスを自動的に開始します。
4. 同時に、新規のプライマリは、失敗したプライマリに接続をドロップし、再接続するようにクライアントに通知するFANイベントを公開します。1,000の接続すべてでこのプロセスは5秒で完了し、スループットは停止前のレベルに迅速にリストアされます。
5. 停止が発生してから1,000の接続すべてが新規プライマリでアクティブになるまでの全体の経過時間は29秒です（ステップ2から4までの合計）。
    - » Data Guardによる自動フェイルオーバーを使用する場合、全体の経過時間はFastStartFailoverThreshold秒で延長できることに注意してください。推奨される最小のしきい値は6秒です。
6. アプリケーション継続性の再実行ドライバは、停止発生時に実行中だったトランザクションをすべて自動的に送信します。停止は、ユーザーには成功したトランザクションのレスポンスの遅延として現れ、エラーを生成したり、ユーザーに再送信を要求したりすることはありません。

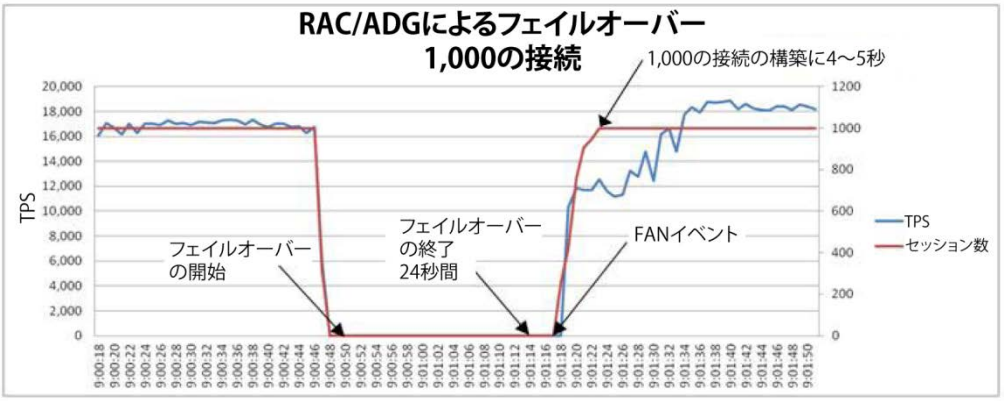

図3：アプリケーション継続性によるエンド・ツー・エンドのアプリケーション・フェイルオーバーのタイミング

FANをサポートしないクライアントの自動フェイルオーバー

次の項では、FANイベントをサポートしないアプリケーションに自動フェイルオーバーを構成する方法について説明します。これには、いくつかのオラクルのアプリケーション（Siebel、Oracle E-Business Suiteなど）や、FANサポートをまだ実装していないアプリケーション・サーバーなどが含まれます。たとえこれらのケースでFANイベントが使用できなくても、タイムアウトおよびアプリケーションのリトライを使用することによって、効率的なフェイルオーバー用にアプリケーションを構成できます。

FANをサポートしないOCIクライアント

OCIアプリケーションについては、「自動フェイルオーバーのためのOCIクライアントの構成」の項で説明した手順に従います。ただし、FANサポートの有効化に関する手順（ステップ1から3）は除きます。クライアントがFANをサポートしていない場合、OSレベルまたはOracle Netレベルで TCPタイムアウトを使用して、クライアント接続をTCPハングから抜け出させます。

1. アプリケーション・レイヤーを実行するホスト上で効率的なTCPタイムアウトが得られるようにオペレーティング・システムを構成します。OSのTCPタイムアウトは、データベース・レイヤーがフェイルオーバーを実行し、データベース・サービスが開始される所要時間に設定する必要があります。TCPタイムアウトを適切に構成する手順については、オペレーティング・システムのマニュアルを参照してください。
2. アプリケーションの再実行は、「OCIデータベース・サービスの構成」の項で説明したように、サーバー側のTAFを構成することによって自動化されます。アプリケーションでTAFを使用できない場合は、例外時の再接続ロジックを用いてアプリケーションを構成する必要があります。たとえば、セッションが接続プールから切断の原因となる例外を受け取ると（ORA-3113エラーなど）、アプリケーションは自動的にそのセッションへの再接続を試みる必要があります。再接続の試行は、データベース・レイヤーをフェイルオーバーし、アプリケーション・サービスをオンラインに戻すまでにかかる時間と同じだけ継続するように構成する必要があります。
3. TAFは、既存のセッションに対する再実行のみ自動化できることに注意してください。新しい接続の試行がエラーを受信したときに再実行を自動化するには、Oracle Net DESCRIPTION_LISTの一部であるRETRY_COUNTパラメータを使用します。このパラメータは、新しい接続の試行が終了するまでにADDRESSリストが繰り返される回数を指定します。

FANをサポートしないJDBCクライアント

1. アプリケーション・レイヤーを実行するホスト上で効率的なTCPタイムアウトが得られるようにオペレーティング・システムを構成します。OSのTCPタイムアウトは、データベース・レイヤーがフェイルオーバーを実行し、データベース・サービスが開始される所要時間に設定する必要があります。TCPタイムアウトを適切に構成する方法については、オペレーティング・システムのマニュアルを参照してください。
2. アプリケーション内の再接続ロジックを例外時に適切に対処できるように構成します。たとえば、セッションが接続プールから切断の原因となる例外を受け取ると（ORA-3113エラーなど）、アプリケーションは自動的にそのセッションへの再接続を試みる必要があります。再接続の試行は、データベース・レイヤーをフェイルオーバーし、アプリケーション・サービスをオンラインに戻すまでにかかる時間と同じだけ継続するように構成する必要があります。

Active Data Guardの接続に関する追加の考慮事項

Oracle Active Data Guardは、いくつかの高度な機能を提供することで基本的なData Guardの機能を拡張します。Active Data Guardの機能の一つに、スタンバイがプライマリ・データベースから受け取った変更を連続的に適用しながら、フィジカル・スタンバイ・データベースへの読取り専用アクセスを有効にする機能があります。そのため、フィジカル・スタンバイ・データベースは、ディザスタ・リカバリを提供できると同時に、プライマリ・データベースから非定型の問合せ、Webベースのアクセス、レポート作成、およびバックアップをオフロードすることによって、パフォーマンス、スケーラビリティ、および投資収益率を向上させます。これによって、フェイルオーバーが発生した場合に追加の考慮事項が発生します。たとえば、スタンバイ・ロールにあるときに実行中だった読取り専用ワークロードと、フェイルオーバーが発生した場合に引き継がれてプライマリ本番データベースになる追加のワークロードの両方をサポートするのに十分な容量があるかを判断する必要があります。

初回の接続

読取り専用の問合せおよびレポート作成アプリケーションは、データベースがスタンバイ・ロールで機能するときはいつでも読取り専用サービスを開始するデータベース・サービス・ディスクリプションを作成することによって、Active Data Guardスタンバイに送信できます。ロールベースのサービスは、Data Guard BrokerおよびCRS（シングル・インスタンスのデータベースの場合はOracle Restart）によって制御されるため、Data Guardのロール移行の発生時に常にサービスをデータベース・ロールに適するように開始/終了させることができます。

Data Guard構成に接続されるすべてのアプリケーションは、DESCRIPTION_LIST内にプライマリ・データベースとスタンバイ・データベースの両方のSCAN名を含むOracle Netエイリアスを使用して接続する必要があります。プライマリ・ロールのデータベースに接続する読取り/書込みアプリケーションが使用するOracle Netエイリアスは、最初のDESCRIPTIONにプライマリSCAN名を記述する必要があります。一方で、スタンバイ・ロールのデータベースに接続する読取り専用アプリケーションが使用するOracle Netエイリアスは、最初のDESCRIPTIONにスタンバイSCAN名を記述する必要があります。

計画外停止での既存の接続の管理

安定した状態では、プライマリ・アプリケーションはプライマリ・データベースに接続され、読取り専用のレポート作成アプリケーションはアクティブ・スタンバイ・データベースに接続されます。ロールの移行が発生すると、アクティブ・スタンバイ・データベースに接続されたレポート作成アプリケーションのユーザーは、フェイルオーバー・プロセスの一環として接続が切断されます。スタンバイがプライマリ・データベース・ロールに移行すると、プライマリ・データベース・サービスが開始し、ユーザーはこのホワイト・ペーパーで前述したプロセスを使用して、新しいプライマリ・データベースへ自動的に接続を開始します。

スタンバイ・データベース上で実行される、レポート作成アプリケーションの接続およびサービスのフェイルオーバーは、手動で実行されます。フェイルオーバーの次は、レポート作成アプリケーションを送信するデータベースを決定します。選択肢は次のとおりです。

- 新規のプライマリ・データベースに、プライマリ・アプリケーション接続とレポート作成アプリケーション接続の両方をサポートするだけの十分な容量がある場合
  - プライマリ・データベース上でレポート作成データベース・サービスを手動で起動
  - サービスが使用可能になったら、レポート作成アプリケーションを再起動して接続を確立
- プライマリ・データベースに、プライマリ・アプリケーションとレポート作成アプリケーションの両方をサポートするだけの容量が十分にない場合、他のActive Data Guardスタンバイ・データベース上でレポート作成データベース・サービスを起動
  - Data Guard構成に複数のActive Data Guardスタンバイ・データベースがない場合は、古いプライマリ・データベースを新規の同期化されたスタンバイ・データベースとして回復させます[24]。これは、フラッシュバック・データベースがフェイルオーバーよりも先に有効になっており、停止の原因が、障害が発生したプライマリの回復のためにこのデータベースを使用することを妨げない場合に、迅速で簡単なプロセスです。Data Guardのファスト・スタート・フェイルオーバーが使用された場合、問題が解決してデータベースがマウ

[24] https://docs.oracle.com/cd/E49329_01/server.121/b71305/cli.htm#BGBBDEBC

ントされると、オブザーバ・プロセスは以前のプライマリをスタンバイ・データベースとして自動的に回復させます[25]。新規のスタンバイが操作可能になったら、読取り専用データベース・サービスを起動してレポート作成アプリケーションを接続します。

注：Oracle Database 12cで導入されたOracle Global Data Services（Oracle GDS）では、読取り専用サービスを再配置するための自動化を追加で提供することにより、この項で説明した、読取り専用サービスを手動で再起動する手順が省かれます。Oracle GDSはこのホワイト・ペーパーの対象外です。Oracle GDSに関する追加情報は、Oracle Technology Networkを参照してください[26]。

ログオン・ストームの制御

接続プールは小規模にすることを強くお薦めしますが、接続が多数ある場合は、チューニングによりログオン・ストームを制御できます。次に説明するチューニングは、単一のExadataコンピュート・ノード上で実行されており、5,000の同時接続を含むデフォルトのTCPキューサイズを使用した場合、クライアントの80%が接続に失敗し、残りのクライアントは46秒間で接続できたことを示しています。簡単なTCPキューサイズ・チューニングのあとは、クライアント側のエラーは発生せず、すべてのクライアントが6〜8秒間で接続できました。

» OSレベルでListenバックログを増加させます。再起動せずに新規の値を有効にするには、以下をrootとして実行します。

```
echo 8000 > /proc/sys/net/core/somaxconn
```

再起動後もこの値を永続化するには、/etc/sysctl.confにも追加します。

```
net.core.somaxconn=6000
```

» リスナーのキューサイズを増やします。リスナーの実行元のOracleホームでsqlnet.oraを更新し、キューサイズ・パラメータを増やします。

```
TCP.QUEUESIZE=6000
```

これはsqlnet.oraの新しいグローバル・パラメータで、RACの場合は、リスナーの実行元の関連するGI/DBホームで設定する必要があります。

おのおののリスナー・アドレスには、前のパラメータもありました（QUEUESIZE=）。これも同様に動作し、CRS以外のマネージド・リスナー向けに使用できます。ただし、Oracle RAC 11.2以降のリリースでは、CRSエージェントがリスナー・アドレスを動的に管理し、キューサイズはsrvctlではサポートされません。これがOracle RAC 11.2.0.4でTCP.QUEUESIZEが追加された理由で、以前のリリースではワンオフ・パッチ経由で使用できます。両方のパラメータが設定された場合、リスナー・アドレスの値が優先されます。

[25] https://docs.oracle.com/cd/E49329_01/server.121/b71305/sofo.htm#CHDIBEBC

[26] http://www.oracle.com/technetwork/database/features/availability/global-data-services-1949717.html

Rate_limitは反対で、上述のチューニングとともに使用できます。システムがチューニングによって多数の同時接続をサポートできる場合、rate limitを有効化する必要はありません。それ以外の場合は、マシン機能に従ってrate_limitを適切な値に設定できます。rate_limitが役立つシナリオは（設計された当初の目的の1つ）は、サーバー上の二次的なエラーや悪影響につながるデータベース・ボトルネックがある場合です。

結論

Oracle Maximum Availability Architecture（Oracle MAA）は、オラクルの高可用性テクノロジーを実装するためのベスト・プラクティス構想です。The core principle of high availability is to make outages as transparent as possible to the user.Oracle Database 12cを使用したOracle MAAは、コンポーネント、システム、およびデータベースの停止の際もユーザーがトランザクション処理を継続できるようにする（アプリケーションはエラーを受信せずに実行中のトランザクションが完了）という点で比類のないものです。Oracle MAAのおもなコンポーネントであるOracle Real Application Cluster（Oracle RAC）、Oracle Data Guard、Oracle Active Data Guard、およびアプリケーション継続性により、この機能が有効になります。

このテクニカル・ホワイト・ペーパーでは、Data GuardおよびActive Data Guardのロール移行が発生したのちに、障害が発生したプライマリ・データベースから新しいプライマリ・データベースへアプリケーション接続を自動的に移行させるためのOracle Database 12c構成のベスト・プラクティスについて説明しました。アプリケーション継続性は、RACノードの移行やData Guardによるフェイルオーバーに続いて実行中のトランザクションを保存するという利点を追加で提供することにより、停止による影響をユーザーからマスキングしてサービス品質を向上させます。

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

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Hardware and Software, Engineered to Work Together

Copyright © 2014, Oracle and/or its affiliates.All rights reserved.本文書は情報提供のみを目的として提供されており、ここに記載されている内容は予告なく変更されることがあります。本文書は一切間違いがないことを保証するものではなく、さらに、口述による明示または法律による黙示を問わず、特定の目的に対する商品性もしくは適合性についての黙示的な保証を含み、いかなる他の保証や条件も提供するものではありません。オラクル社は本文書に関するいかなる法的責任も明確に否認し、本文書によって直接的または間接的に確立される契約義務はないものとします。本文書はオラクル社の書面による許可を前もって得ることなく、いかなる目的のためにも、電子または印刷を含むいかなる形式や手段によっても再作成または送信することはできません。

OracleおよびJavaはOracleおよびその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標です。

IntelおよびIntel XeonはIntel Corporationの商標または登録商標です。すべてのSPARC商標はライセンスに基づいて使用されるSPARC International, Inc.の商標または登録商標です。AMD、Opteron、AMDロゴおよびAMD Opteronロゴは、Advanced Micro Devicesの商標または登録商標です。UNIXは、The Open Groupの登録商標です。0914

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