ORACLE
Autonomous
Database Cloud

継続的な可用性

Oracle Autonomous Databaseの専用デプロイメントを使用したアプリケーションのベスト・プラクティス

ホワイト・ペーパー / 2020年9月2日

ORACLE

はじめに

計画メンテナンスや計画外停止、データベース層での負荷の不均衡に気づかれなければ、アプリケーションのサービスを継続できていると言えます。アプリケーションのベスト・プラクティス、簡単な構成、Oracle Autonomous Databaseを組み合わせれば、アプリケーションの継続的な可用性が実現します。

推奨されるアプリケーション構成のソリューションが使用できない場合の代替のアプリケーション構成については、ホワイト・ペーパーContinuous Availability Application Checklist for Continuous Service for MAA Solutions[1]を参照してください。

[1] https://www.oracle.com/technetwork/jp/database/options/clustering/applicationcontinuity/maa-continuousavailability-5169724-ja.pdf

概要

計画メンテナンスのアクティビティをアプリケーションに認識させない最適な方法は、各データベース・ワークロードのロケーションから、そのロケーションのメンテナンス期間の前に作業を透過的にドレインすることです。WebLogic Server、Universal Connection Pool（UCP）、OCIセッション・プール、ODP.NET管理対象外プロバイダなど、オラクルの接続プールと中間層は、高速アプリケーション通知（FAN[2]）に対応しているため、メンテナンス前に作業が正常にドレインされるようデータベース・サービスの移動がスケジューリングされる前に、通知を受けます。FANの通知によって、自動的にアイドル状態の接続がクローズされ、新しい接続が新しいサービス・ロケーションでオープンされます。さらに、間もなくシャットダウンされるサービス・ロケーションで、アクティブな作業を完了できる構成可能な時間が確保されます。IBM WebSphereをはじめとする大手サード・パーティのJDBC中間層は、UCP[3]を使用して構成された場合、同様の動作が可能になります。UCPを使用できないJDBCベースのアプリケーションに対しては、Oracleドライバを使用したソリューションと接続テストが用意されています。

コンポーネントや通信の障害によって引き起こされる計画外停止を認識させないようにするために、オラクルでは以下を実現しています。

通知 - 停止を不可視化する最初の手順はFANです。FANにより、停止が発生した際にクライアントが通知を受け、現在のネットワーク待機から抜け出すことができるようになります。これにより、アプリケーションがネットワーク待機のために長い時間停止することがなくなります。さらに、サービスが再び使用可能になった際に、セッションのリバランスもFANによって実行されます。

リカバリ – クライアントが通知を受けると、アプリケーション・コンティニュイティ（AC）または透過的アプリケーション・コンティニュイティ（TAC）は、新たなワークロード・ロケーションへの接続（Oracle Real Application Clusters（Oracle RAC）の場合は同じまたは別のインスタンスへの接続、Oracle Data Guardの場合はスタンバイ・サイトへの接続）を再確立し、可能な場合は処理中の（コミットされていない）作業を再実行します。新たなロケーションで処理中の作業を再実行すると、アプリケーションでは通常、障害が発生したことを認識することなく実行が継続されます。

ACまたはTACでは、割り当てられた時間内にドレインされない（現在のデータベース操作を完了する）セッションは、計画メンテナンス中も実行されます。

アプリケーション構成チェックリスト

以下のチェックリストにより、Oracle Autonomous Databaseを使用するようアプリケーションを準備できます。

Oracleサービスを使用した接続

高可用性のための接続文字列/URLの構成　高速アプリケーション通知（FAN）の使用

ドレイニングが可能な推奨のアプリケーション・プラクティスの使用

アプリケーション・コンティニュイティまたは透過的アプリケーション・コンティニュイティの有効化

[2] 高速アプリケーション通知、アプリケーション・コンティニュイティ、および透過的アプリケーション・コンティニュイティの説明については、本書の最後の「付録：追加の技術資料」に記載される資料を参照してください。

[3] メンテナンス期間と計画外停止を認識させないようにするために、UCPを使用してOracle WebLogic Server、IBM WebSphere、IBM Liberty、Apache Tomcat、およびRedhat WildFly（JBoss）を構成する方法について詳しくは、「付録：追加の技術資料」に記載される資料を参照してください。

サービスを使用した接続

各種サービスにより、基盤となるATP-Dインフラストラクチャで透明性が実現します。FAN、接続データ、透過的アプリケーション・コンティニュイティ（TAC）、アプリケーション・コンティニュイティ（AC）、スイッチオーバー、コンシューマ・グループ、およびその他の多数の機能と操作は、各種サービスを使用することを前提としています。使用するサービスによって、基盤となるData Guard環境のプライマリ・ロールまたはスタンバイ・ロールも定義されます。

オラクルのAutonomous Database Transaction Processingの専用デプロイメント（ATP-D）では、事前構成済みの5つのサービスを選択できます。どのサービスもFANとドレイニングに対応しています。TPURGENTとTPでは、ATP-D環境でTACがデフォルトで有効化されています。APIを使用して、事前構成済みのすべてのサービスのTACまたはACの設定を変更できます（付録を参照）。

Oracle Autonomous Databaseによって提供される事前構成済みのサービス

サービス名	説明	ドレイニング	FAN	TAC
TPURGENT	OLTP優先順位：最高	対応	対応	対応
TP	OLTP優先順位：普通 （メインのサービスとして使用）	対応	対応	対応
HIGH	レポートまたはバッチ （優先順位：最高）	対応	対応	
MEDIUM	レポートまたはバッチ （優先順位：中）	対応	対応	
LOW	レポートまたはバッチ （優先順位：最低）	対応	対応	

以下を参考にバッチ作業のサービスを選択してください。

HIGH：CPU_COUNTと同じ並列度で問合せが実行されます。同時に実行される問合せは3つまでで、その後は文のキューイングが発生します。

MEDIUM：並列度4で問合せが実行されます。同時に実行できる問合せの最大数は（CPU_COUNT x 1.25）です。

LOW：問合せは順番に実行されます。同時問合せが（2 x CPU_COUNT）を超えるとキューイングが開始されます。

高可用性のための接続文字列/URLの構成

オラクルでは、Oracle Autonomous Databaseに接続する場合に以下の接続文字列/URLの構成を推奨しています。オラクルの供給するウォレットに組み込まれる接続文字列は、この方法で構成されています。EZCONNECTには高可用性機能がないため、クライアントで簡易接続ネーミングを使用しないでください。

なお、以下で指定しているstandby-scanの参照先は、Oracle Active Data Guard構成で指定したスタンバイ・サイトで使用可能なSCANアドレスです。ドライバは、まずプライマリ・サイトへの接続を試み、サービスが使用可能でない場合は、スタンバイのサービスへの接続を試みます。どちらのサイトであれサービスに接続すると、バージョン12.2以降のOracleドライバは、そのサービスを提供するTNSアドレス・リストを記憶し、このサイトを優先させます。

バージョン12.2以降のすべてのOracleドライバで、次の接続文字列を使用します。

```
Alias (or URL) =
(DESCRIPTION =
(CONNECT_TIMEOUT= 90)(RETRY_COUNT=50)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
 (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
 (ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=standby-scan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME = ATP-D SERVICE)))
```

バージョン12.1以前のOracleドライバを使用したJDBC接続では、次のようになります。

```
Alias (or URL) =
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)(RETRY_COUNT=50)(RETRY_DELAY=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
(ADDRESS_LIST =
   (LOAD_BALANCE=on)
   (ADDRESS = (PROTOCOL = TCP)(HOST=standby-scan)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = ATP-D SERVICE)))
```

高速アプリケーション通知の使用

FANでは、サービスの停止時または再開時に、アプリケーションに即座に通知を送信します。FANを使用しなければ、ハードウェア障害やネットワーク障害が発生した場合にアプリケーションがTCP/IPタイムアウトでハングし、リソースが再開されたときにリバランスが行われない可能性があります。OracleプールおよびOracleアプリケーション・サーバーはすべて、FANを使用します。サード・パーティのJAVAアプリケーション・サーバーでは、UCPを使用してFANを有効にできます。

FANを使用するためにアプリケーションの変更は不要です。構成の変更のみが必要となります。計画メンテナンス中にサービスを継続させるには、FANを以下のいずれかと併用します。

- Oracleプール
- UCPとサード・パーティJDBCアプリケーション・サーバー
- 最新のOracleクライアント・ドライバ

計画外停止中にサービスを継続させるには、FANを以下のいずれかと併用します。

- アプリケーション・コンティニュイティ
- 透過的アプリケーション・コンティニュイティ

FANのサポート対象

FANのイベントを統合できるものは次のとおりです。

- Oracle Fusion MiddlewareおよびOracle WebLogic Server
- Oracle Data Guard Broker
- JDBC Thinインタフェースおよび Oracle Call Interface（OCI）向けOracle JDBC Universal Connection PoolまたはOracle JDBCドライバ
- 非管理対象プロバイダおよび管理対象プロバイダ向けODP.NET接続プール
- Oracle Tuxedo
- SQL*Plus
- Python、Node.js、PHPなどの言語向けOracle Databaseドライバ
- Global Data Services
- Oracle JDBC Universal Connection Poolを使用するサード・パーティのJDBCアプリケーション・サーバー
- リスナー

クライアントでFANを有効にする手順：

前述したTNSエイリアスまたはURLを使用します。Oracle Database 12c以降のクライアント・ドライバを使用する場合は、FANイベントを受信するためのOracle Notification Service（ONS）サブスクリプションが、この接続文字列によりクライアントで自動構成されます。古いドライバを使用する場合、構成の詳細については、付録に掲載されているFANのホワイト・ペーパーを参照してください。ONSによってデータベース層とクライアント層間にセキュアな通信パスが提供され、サービスの可用性（コンポーネントの停止または起動）やランタイム・ロードバランシングに関するアドバイスをクライアントに通知できるようになるため、通常運用時に作業の配置が改善されます。

クライアントに応じて、次のようにアプリケーション構成プロパティでFANを有効にします。

Universal Connection PoolまたはJDBCシン・ドライバ（12.2以降）

プロパティ`FastConnectionFailoverEnabled`を設定します。

WebLogic Active GridLink for Oracle RAC

FANと高速接続フェイルオーバーはデフォルトで有効化されます。

Oracle WebLogic Server、IBM WebSphere、IBM Liberty、Apache Tomcat、Red Hat WildFly（JBoss）、JDBCアプリケーション

接続プールの代わりにUniversal Connection Poolを使用します。

ODP.Netクライアント（管理対象プロバイダおよび管理対象外プロバイダ）

ODP.Net 12.1以前を使用する場合、接続文字列に“`HA events = true;pooling=true`”を設定します。

Oracle Call Interface（OCI）クライアントおよびOCIベースのドライバ

ネイティブ設定を使用しないOracle Call Interface（OCI）クライアントでは、oraacces.xmlファイルを使用し、eventsをtrueに設定できます。

Python、Node.js、およびPHPではネイティブ・オプションを使用できます。PythonおよびNode.jsでは、接続プールの作成時にイベント・モードを設定できます。PHPでは、`php.ini`を編集してエントリ`oci8.events=on`を追加します。

SQL*PlusではFANがデフォルトで有効になります。

ATP-D環境では、ONSによってオプションのTLS（ウォレットベースの）認証が提供されます。ウォレット構成は、付録に記載されるように、アプリケーションの種類（JDBCまたはOracle Call Interface）に応じて特定のルールに従う必要があります。

ドレイニングが可能な推奨のアプリケーション・プラクティスの使用

アプリケーションの用法としては、必要なときに接続をチェックアウトし、現在のアクションが完了したら接続をプールに戻すのがベスト・プラクティスです。このプラクティスは、実行時の作業のリバランス、および作業をドレインするメンテナンス期間において、良いパフォーマンスを達成するために重要です。お使いのアプリケーションがこのプラクティスにどの程度従っているかを確認するには、「TACまたはACを使用する場合の保護レベルの把握」の統計セクションを参照してください。

オラクルでは、計画メンテナンスを認識させないために、FANに対応したOracle接続プールを使用することを推奨しています。アプリケーションでOracleプールとFANが使用され、リクエストとリクエストの間にプールに接続が戻される場合、ユーザーに影響はありません。FANを使用するために、アプリケーションに変更を加える必要はありません。

Oracle接続プールが計画停止時間にFANイベントを受信すると、ドレインされるインスタンスですべての接続がマークされます。チェックイン済みの接続は、再利用されないよう即座にクローズされます。使用中の接続は、プールに戻されるとクローズされます。これにより、すべての接続が時間とともに正常にクローズされます。

Javaベースのサード・パーティ製アプリケーション・サーバーを使用している場合、もっとも効果的にドレイニングやフェイルオーバーを行うには、プールされたデータソースの代わりにUCPを使用します。Oracle WebLogic Server、IBM WebSphere、IBM Liberty、Apache Tomcat、Red Hat WildFly（JBoss）、Spring、Hibernateなど、多くのアプリケーション・サーバーでこのアプローチがサポートされます。これらのアプリケーション・サーバーでUCPを使用する方法については、オラクルや他のプロバイダ（IBMなど）が発行しているホワイト・ペーパーを参照してください。UCPをデータソースとして使用すると、高速接続フェイルオーバー、ランタイム・ロードバランシング、アプリケーション・コンティニュイティ、透過的アプリケーション・コンティニュイティといった、十分に動作保証されたUCPの機能を使用できます。

オプション1：透過的アプリケーション・コンティニュイティの有効化

TACにより、セッションとトランザクションの状態が透過的に追跡および記録されるため、リカバリ可能な停止の直後にデータベース・セッションをリカバリできます。Autonomous Databaseに対応した適切なサービスを選択すると、TACが有効化されます。

透過的アプリケーション・コンティニュイティのサポート対象

Oracle Autonomous Databaseの透過的アプリケーション・コンティニュイティは、次のクライアントに対応しています。

最新のクライアント・ドライバを使用することを強く推奨します。Oracle Database 19c以降のクライアント・ドライバは、TACを完全にサポートしています。

- Oracle JDBC Replay Driver 18c以降。これは、Oracle Database 18cに付属する、アプリケーション・コンティニュイティのためのJDBCドライバ機能です
- Oracle Universal Connection Pool（UCP）18c以降とOracle JDBC Replay Driver 18c以降
- Oracle WebLogic Server Active GridLink、またはUCPとOracle JDBC Replay Driver 18c以降を使用したサード・パーティのJDBCアプリケーション・サーバー
- Oracle JDBC Replay Driver 18c以降を使用したJava接続プールまたはスタンドアロンJavaアプリケーション
- Oracle Call Interface Session Pool 19c以降
- SQL*Plus 19c（19.3）以降
- プールされたODP.NET管理対象外ドライバ18c以降（12.2以降では“Pooling=true”がデフォルト）

- 19c以降のOCIドライバを使用したOracle Call Interfaceベースのアプリケーション

透過的アプリケーション・コンティニュイティを使用する場合の手順

「付録：フェイルオーバーのためのサービス属性の有効化」の参照
サポート対象のクライアントの使用（前述したサポート対象を参照）
接続プールへの接続の返却

アプリケーションで以下のいずれかの接続を使用している場合は、アプリケーションを変更しなくてもリクエストの境界を識別できます。

- Oracle接続プールからの接続
- Oracle JDBC Replay Driver 18c以降からの接続
- 19c以降を使用したOracle Call Interfaceベースのアプリケーションからの接続

接続プールを使用する場合、各リクエストが完了すると、接続はアプリケーションによってプールに返却される必要があります。オラクルでは、アプリケーションで必要な場合のみ接続をチェックアウトすることを推奨しています。使用していない接続を保持しているのは適切ではありません。また、記載されるドライバを使用した透過的アプリケーション・コンティニュイティでは、境界がどこに追加されるかが検出され、独自の境界が作成されます。

FAILOVER_RESTOREの使用

透過的アプリケーション・コンティニュイティを有効化すると、事前設定されたセッション状態が自動的にリストアされます。標準設定の他に事前設定のセッション状態が必要になった場合は、コールバック[4]（UCPラベル）を登録してその状態をリストアすることができます。一時表やSYS_CONTEXTといった複雑なセッション状態をリストアすることが必要になった場合は、アプリケーション・コンティニュイティを使用すると柔軟に対応できます。

アプリケーションでの可変値の使用の有効化

可変関数とは、実行されるたびに新しい値を返す可能性のある関数です。SYSDATE、SYSTIMESTAMP、SYS_GUID、sequence.NEXTVALについては、元の結果を保持できるようになっています。アプリケーション・コンティニュイティ19c以降では、SQLの可変値が自動的に保持されるため、何も行う必要はありません。PL/SQLの可変値が必要な場合は、DBAはGRANT KEEP権限を発行する必要があります。KEEP権限が付与されると、関数の元の結果が再実行時に適用されます。

以下に例を示します。

```
SQL> GRANT [KEEP DATE TIME | KEEP SYSGUID] … TO USER
```

```
SQL> GRANT KEEP SEQUENCE mySequence TO myUser ON sequence.object
```

副次的影響の無効化

[4] コールバックの登録方法については、ホワイト・ペーパー『*Continuous Availability Application Checklist for Continuous Service for MAA Solutions*』（https://www.oracle.com/technetwork/jp/database/options/clustering/applicationcontinuity/maa-continuousavailability-5169724-ja.pdf）を参照してください。

副次的影響とは、メールの送信、ファイルの転送、TCPの使用といった外部アクションです。透過的アプリケーション・コンティニュイティにより、副次的影響は検出され、再実行されません。副次的影響の再実行が必要な場合は、アプリケーション・コンティニュイティを使用すると柔軟に対応できます。

オプション2：アプリケーション・コンティニュイティの有効化

アプリケーション・コンティニュイティはカスタマイズが可能で、副次的影響の再実行を選択したり、透過的アプリケーション・コンティニュイティでは実行されないフェイルオーバー時の複雑なコールバックを追加したりできます。アプリケーション・コンティニュイティは、Oracle 12cのドライバ（JDBCシンまたはOracle Call Interface）を使用している場合、副次的影響またはコールバックを使用してカスタマイズしたい場合、もしくはセッション期間中の一時表などの状態を使用しており、リクエスト全体でクリーンアップを行わないアプリケーションがある場合に使用します。

アプリケーション・コンティニュイティのサポート対象

Oracle Database 19cのアプリケーション・コンティニュイティでは、以下のクライアントをサポートしています。

- 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・コンティニュイティのためのJDBCドライバ機能です。
- Oracle Universal Connection Pool（UCP）12c以降とOracle JDBC Replay Driver 12c以降
- Oracle WebLogic Server Active GridLink、およびUCPとOracle JDBC Replay Driver 12c以降を使用したサード・パーティのJDBCアプリケーション・サーバー
- Oracle JDBC Replay Driver 12c以降とリクエスト境界またはプールされたデータソースを使用したJava接続プールまたはスタンドアロンJavaアプリケーション
- Oracle Call Interface Session Pool 12cリリース2以降を使用するアプリケーションおよび言語ドライバ
- SQL*Plus 19.3以降
- プールされたODP.NET管理対象外ドライバ12cリリース2以降（12.2以降では“Pooling=true”、“Application Continuity=true”がデフォルト）

アプリケーション・コンティニュイティを使用する場合の手順

「付録：フェイルオーバーのためのサービス属性の有効化」の参照

サポート対象のクライアントの使用（サポート対象を参照）

プールへの接続の返却

アプリケーションでOracle接続プールまたはリクエスト境界をサポートするサード・パーティのJDBCプールを使用している場合は、アプリケーションを変更しなくてもリクエスト境界を識別できます。Oracleプールを使用し、リクエストとリクエストの間に接続をOracleプールに返すのがベスト・プラクティスです。オラクルでは、アプリケーションで必要な場合のみ接続をチェックアウトすることを推奨しています。使用していない接続を保持しているのは適切ではありません。

`FAILOVER_RESTORE`の使用

`FAILOVER_RESTORE=LEVEL1`に設定すると、もっとも一般的な状態が自動的にリストアされます。アプリケーションで、標準設定の他にセッション状態を事前設定する場合は、コールバック（UCPラベル）を登録してそれらの状態をリストアする必要があります。以下を使用します。

`FAILOVER_RESTORE=LEVEL1`（サービスで設定）

- 接続初期化コールバック（Javaの場合）または（古い）TAFコールバック（Oracle Call Interfaceの場合）
- Oracle Universal Connection PoolまたはWebLogic Serverの接続ラベル付け

アプリケーションでの可変値の使用の有効化（可変値の「透過的アプリケーション・コンティニュイティの有効化」セクションを参照）
副次的影響を再実行するかどうかの判断

副次的影響とは、メールの送信、ファイルの転送、TCPの使用といった外部アクションです。アプリケーション・コンティニュイティでは、再実行を無効にしない限り、副次的影響は再実行されます。再実行すべきではない外部アクションがリクエストに含まれる場合は、アプリケーション・コンティニュイティが有効になっていない接続をそのリクエストで使用するか、disableReplay () API（Javaの場合）またはOCIRequestDisableReplay()（Oracle Call Interfaceの場合）を使用して、そのリクエストの再実行を無効にします。すべての副次的影響を再実行することを望まない場合は、透過的アプリケーション・コンティニュイティを使用します。

TACまたはACを使用する場合の保護レベルの把握

リクエスト境界と保護レベルに関する統計情報を使用してカバレッジのレベルを監視することで、アプリケーションによって接続がプールに返されているかどうか、アプリケーションがどの程度保護されているかを把握します。

アプリケーション・コンティニュイティを使用する場合はシステム、セッション、サービスから統計情報が収集されるため、保護レベルを監視できます。統計情報は、バージョン19の後期よりV$SYSSTAT、V$SESSTAT、およびV$SERVICE_STATSで利用できます。これらの統計情報は、自動ワークロード・リポジトリ（AWR）に保存され、AWRレポートとASHビューで確認できます。

出力は次のようになります。

統計情報	合計	1秒あたり	トランザクションあたり
cumulative begin requests	1,500,000	14,192.9	2.4
cumulative end requests	1,500,000	14,192.9	2.4
cumulative user calls in request	6,672,566	63,135.2	10.8
cumulative user calls protected	6,672,566	63,135.2	10.8

ヒント：保護レベルをPDBによって、または履歴データを使用してレポートするには、付録のSQL使用例を参照してください。以下の場合に、TACまたはACが有効になってアプリケーションを保護します。

- Cumulative user calls in requestの値とcumulative user calls protectedの値が等しい
- かつ、上記の値がゼロでない

この保護レベルはデータベース内部で測定されます。このレベルの保護を達成するには、クライアントで問合せにORDER BY句を使用し、FAILOVER_RESTOREの対象ではない状態が含まれている場合は、セッションの初期状態を事前設定することが必要になる場合があります。

上記の例は、開始リクエストと終了リクエストの数が増加していることを示しています。数自体は、アプリケーションが接続プールに対してチェックアウトおよびチェックインを実行する頻度や、TACの使用時にデータベースで検出可能なリクエスト境界によって異なります。これらの値が増加する速度は、リクエストが送信される速度によって異なります。以下を使用すると、保護対象のユーザー・コールの割合を計算できます。

保護対象のコールの割合 = cumulative user calls protected / cumulative user calls in request × 100

保護対象のコールの割合が100 %未満になる可能性があります。JDBC具象クラスを使用している、副次的影響が無効化されている、リカバリ不能な状態が使用されている、または特定のリクエストに対してアプリケーション・コンティニュイティを無効化するようにアプリケーションで選択されている場合などです。アプリケーションの保護が100 %でない場合は、サイトでORAchk[5]コンポーネントのacchkを使用して、アプリケーション内のどの箇所で保護が100 %未満になっているのかを把握できます。影響を評価してから、アドバイザに従うのか措置を何も実施しないのかをマネジメントが判断できます。

[5] ORAchkユーティリティのダウンロード、構成、実行について詳しくは、My Oracle Support Note 1268927.2を参照してください。

クライアントの構成

JDBC Thinドライバの場合のチェックリスト

1. すべての推奨パッチがクライアントに適用されていることを確認します。MOS Note『*Client Validation Matrix for Application Continuity*』（*Doc ID 2511448.1*）を参照してください。

2. 保護およびパフォーマンスのためにJDBCステートメント・キャッシュを使用します。

最大の保護とパフォーマンスを達成するには、アプリケーション・サーバーのステートメント・キャッシュの代わりにJDBCドライバのステートメント・キャッシュを使用します。これによりドライバは、リクエストの終了時にステートメントがクローズされてメモリが解放されることを認識できます。

JDBCステートメント・キャッシュを使用するには、接続プロパティ`oracle.jdbc.implicitStatementCacheSize`（`OracleConnection.CONNECTION_PROPERTY_IMPLICIT_STATEMENT_CACHE_SIZE`）を使用します。キャッシュ・サイズの値はopen_cursorsの数と同じです。以下に例を示します。

`oracle.jdbc.implicitStatementCacheSize=`*`nnn`* ここで、*nnn*は通常、10～100の値であり、アプリケーションで維持されているオープン・カーソルの数と同じになります。

3. ガベージ・コレクタをチューニングします。

多くのアプリケーションでは、ガベージ・コレクタのデフォルト・チューニングで十分です。大量のデータを返すアプリケーションや保持するアプリケーションの場合は、2G以上といった高い値を使用できます。以下に例を示します。

```
java -Xms3072m -Xmx3072m
```

Javaの初期ヒープ・サイズ（ms）のメモリ割当てと最大ヒープ・サイズ（mx）のメモリ割当てを同じ値に設定することを推奨します。これにより、メモリ・ヒープの増加および縮小で、システム・リソースが使用されません。

4. JDBC具象クラス

JDBCアプリケーションについては、推奨されないoracle.sql具象クラス（BLOB、CLOB、BFILE、OPAQUE、ARRAY、STRUCT、またはORADATA）はサポートされません（MOS Note 1364193.1『*New JDBC Interfaces*』を参照）。アプリケーションに問題がないことを確認するには、クライアントでORAchk -acchkを使用します[6]。Oracle JDBCシン・ドライバのバージョン18c以降、JDBC Replay Driverの制限付き具象クラスのリストは次のとおりに削減されています。`oracle.sql.OPAQUE`、`oracle.sql.STRUCT`、`oracle.sql.ANYDATA`。

5. 高速接続フェイルオーバー（FCF）を構成します。

クライアント・ドライバ12c以降の場合

- ONSの自動構成に推奨されている接続文字列/URLを使用します。
- ons.jar（およびオプションのWALLET jar、osdt_cert.jar、osdt_core.jar、oraclepki.jar）がCLASSPATHに存在することを確認します。
- プール・プロパティまたはドライバ・プロパティfastConnectionFailoverEnabled=trueを設定します。

[6] ORAchk –acchkについて詳しくは、ブログ記事Using Orachk to Clean Up Concrete Classes for Application Continuityを参照してください。

- サード・パーティのJDBCプールの場合は、UCPを推奨します。
- ONS用にポート6200を開きます（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場合もあります）。

クライアント・ドライバが12cより以前の場合は、以下のアドレスを使用します。
oracle.ons.nodesをXXX01:6200、XXX02:6200、XXX03:6200となるように設定します。

Oracle Call Interface（OCI）ドライバの場合のチェックリスト

1. すべての推奨パッチがクライアントに適用されていることを確認します。MOS Note『*Client Validation Matrix for Application Continuity*』（*Doc ID 251148.1*）を参照してください。
2. `OCIStmtPrepare`を`OCIStmtPrepare2`に置き換えます。`OCIStmtPrepare()`は12.2以降、非推奨になっています。すべてのアプリケーションで`OCIStmtPrepare2()`を使用する必要があります。TACおよびACでは`OCIStmtPrepare`を使用できますが、このステートメントは再実行されません。

   https://docs.oracle.com/en/database/oracle/oracle-database/19/lnoci/deprecated-oci-functions.html#GUID-FD74B639-8B97-4A5A-BC3E-269CE59345CA
3. OCIベースのアプリケーションにFANを使用するには、以下を実行します。

- ATP-Dでサービスに`aq_ha_notifications`を事前設定します。
- ONSの自動構成に推奨されている接続文字列を使用します。
- `oraaccess.xml`で`auto_config`、`events`、および`wallet_location`（オプション）を設定します（付録を参照）。
- アプリケーションをO/Sクライアント・スレッド・ライブラリにリンクさせます。
- ONS用にポート6200を開きます（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場合もあります）。

クライアント・ドライバが12cより前の場合は、oraccess.xmlに指定されているアドレスをコピーします。

ODP.NET管理対象外のプロバイダ・ドライバの場合のチェックリスト

1. すべての推奨パッチがクライアントに適用されていることを確認します。MOS Note『Client Validation Matrix for Application Continuity』（Doc ID 251148.1）を参照してください。
2. Oracle Call Interfaceベースのアプリケーションに FANを使用するには、以下を実行します。

- ATP-Dでサービスに`aq_ha_notifications`を事前設定します。
- auto-onsに推奨されている接続文字列を使用します。
- `oraaccess.xml`で`onsConfig`および`wallet_location`（オプション）を設定します（付録を参照）。
- ONS用にポート6200を開きます（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場合もあります）。
- 次の接続文字列で、FANを設定します。

```
"user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true;"
```

- （オプション）次の接続文字列で、ラインタイム・ロードバランシングを設定します。

```
"user id=oracle; password=oracle; data source=HA; pooling=true; HA events=true; load balancing=true;"
```

結論

Oracle Autonomous Databaseは、高可用性を実現するようにユーザーに代わって構成および管理されています。ユーザーが追加の構成や管理を行う必要はありません。

以下の簡単な手順を実行することで、アプリケーションで継続的可用性を達成できます。

- それぞれのSLAに適したATP-Dサービスを選択する
- 高速アプリケーション通知（FAN）を構成する
- アプリケーションに推奨されている接続文字列を使用する
- ドレイニングを最適化するアプリケーション・ベスト・プラクティスを使用する
- 継続的なサービスを達成するために、透過的アプリケーション・コンティニュイティまたはアプリケーション・コンティニュイティを使用する

上記の5つの簡単な手順に従うことで、計画メンテナンス作業時の停止が不要になり、計画外のイベントが発生した場合もトランザクションの失敗やサービスの中断がほとんどなくなります。

付録：フェイルオーバーのためのサービス属性の有効化

事前構成されたサービスのTPURGENTおよびTPでは、透過的アプリケーション・コンティニュイティはデフォルトであり、何も行う必要はありません。マーケTPURGENTおよびTPを使用する場合、FAILOVER_RESTOREのDEFAULT値はAUTOです。

汎用パッケージのDBMS_APP_CONT_ADMINを使用することで、サービスで提供されるフェイルオーバー・タイプを変更できます。このAPIを使用すると、アプリケーション・コンティニュイティ、透過的アプリケーション・コンティニュイティ、または透過的アプリケーション・フェイルオーバー（TAF）を有効化することも、フェイルオーバーを完全に無効化することもできます。新しいセッションでは新しいフェイルオーバー・タイプが使用されます。

これらのプロシージャを使用するには、PDBADMINロールが付与されている必要があります。
サービスで透過的アプリケーション・コンティニュイティを有効化するには以下を実行します。

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAC(`HIGH`);
```

サービスでアプリケーション・コンティニュイティを有効化するには以下を実行します。

```
execute DBMS_APP_CONT_ADMIN.ENABLE_AC(`TPURGENT`);
```

サービスでTAF SELECTを有効化するには以下を実行します。

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF(`LOW`);
```

サービスでTAF BASICを有効化するには以下を実行します。

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF(`MEDIUM`, `SESSION`);
```

サービスでフェイルオーバーを無効化するには以下を実行します。

```
execute DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER(`HIGH`);
```

サービスを変更せずにTAFを使用する場合は、TAFの古いクライアント側構成を接続文字列で使用します。

```
(FAILOVER_MODE=(TYPE=select)(METHOD=basic)(OVERRIDE=TRUE))
```

付録：ランタイム・ロードバランシングのためのサービス属性の有効化

OLTPおよびバッチ・アプリケーションをロードバランスするには、ランタイム・ロードバランシングを使用します。これは、複数のインスタンスで同時に実行されているサービスにのみ適しています。NONEがデフォルト設定です。これらのプロシージャを使用するには、PDBADMINロールが付与されている必要があります。

OLTP型のアプリケーションでは、作業の送信にサービスのレスポンス時間が使用される場合にランタイム・ロードバランシングを有効化します。

```
execute DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING_GOAL(`TPURGENT`, `SERVICE_TIME`);
```

バッチ型のアプリケーションでは、作業の送信にサービスのスループットが使用される場合にランタイム・ロードバランシングを有効化します。

```
execute DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING_GOAL(`HIGH`, `THROUGHPUT`);
```

実行時のロードバランシングを無効化するには以下を実行します。

```
execute DBMS_APP_CONT_ADMIN.SET_LOAD_BALANCING_GOAL(`HIGH`, `NONE`);
```

付録：PDB、サービス、履歴によってレポートされる保護レベル

```
set lines 85
col Service_name format a30 trunc heading "Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select con_id, total_requests,
total_calls,total_protected,total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select  s.con_id, s.name, s.value
  FROM   GV$CON_SYSSTAT s, GV$STATNAME n
  WHERE s.inst_id    = n.inst_id
  AND   s.statistic# = n.statistic#
  AND   s.value     != 0 )
pivot(
  sum(value)
  for name in ('cumulative begin requests' as total_requests, 'cumulative end requests' as
Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user
calls protected by Application Continuity' as total_protected)
))
order by con_id;
```

PDBによる保護のレポートを作成するには、以下の例を使用します。

サービスによる保護のレポートを作成するには、以下の例を使用します。

```
set pagesize 60
set lines 120
col Service_name format a30 trunc heading "Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select con_id, service_name,total_requests,
total_calls,total_protected,total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select   a.con_id, a.service_name, c.name,b.value
  FROM    gv$session a, gv$sesstat b, gv$statname c
  WHERE   a.sid          = b.sid
  AND     a.inst_id      = b.inst_id
  AND     b.value       != 0
  AND     b.statistic# = c.statistic#
  AND     b.inst_id      = c.inst_id
  AND     a.service_name not in ('SYS$USERS','SYS$BACKGROUND'))
pivot(
  sum(value)
  for name in ('cumulative begin requests' as total_requests, 'cumulative end requests' as
Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user calls
protected by Application Continuity' as total_protected) ))
order by con_id, service_name;
```

過去3日間の保護履歴に関してレポートを作成するには、以下の例を使用します。

```
set lines 85
col Service_name format a30 trunc heading"Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests" col
Total_protected format 9,999,9999 heading "Calls Protected" col
Protected format 999.9 heading "Protected %"

set lines 85
col Service_name format a30 trunc heading"Service"
break on con_id skip1
col Total_requests format 999,999,9999 heading "Requests"
col Total_calls format 9,999,9999 heading "Calls in requests"
col Total_protected format 9,999,9999 heading "Calls Protected"
col Protected format 999.9 heading "Protected %"

select   a.instance_number,begin_interval_time, total_requests, total_calls, total_protected,
total_protected*100/NULLIF(total_calls,0) as Protected
from(
select * from
(select   a.snap_id, a.instance_number,a.stat_name, a.value
  FROM    dba_hist_sysstat a
  WHERE     a.value       ! = 0 )
pivot(
  sum(value)
  for stat_name in ('cumulative begin requests' as total_requests, 'cumulative end requests'
as Total_end_requests, 'cumulative user calls in requests' as Total_calls, 'cumulative user
calls protected by Application Continuity' as total_protected)
)) a,
dba_hist_snapshot b
where a.snap_id=b.snap_id
and a.instance_number=b.instance_number
and begin_interval_time>systimestamp - interval '3' day
order by a.snap_id,a.instance_number;
```

付録：オプションのウォレットを含むFAN用のクライアント構成

ATP-Dを使用している場合、ウォレットに基づく認証はFANのオプションです。TNS接続にも同じウォレットを使用します。

JDBCアプリケーションの場合

1. アプリケーションのCLASSPATHに以下のjarファイルが存在することを確認します。

   （ons.jar、osdt_cert.jar、osdt_core.jar、oraclepki.jar）

2. 次のいずれかの方法でFAN用のウォレットを指定します。

   - 自動構成されるONSをウォレットで使用するには、以下のJavaシステム・プロパティを設定します。

     “-Doracle.ons.walletfile=/replace this with host path/onswallet”
     “-Doracle.ons.walletpassword=myONSWalletPassword”

     これらは、プール単位または接続単位では設定できないことに注意してください。

   - ONSを明示的に設定するには、次のいずれかを行います。

     - UCP XML構成ファイルを使用して明示的に設定します。以下に例を示します。

```
<!--?xml version="1.0" encoding="UTF-8"? -->
<ucp-properties>
  <connection-pool
    connection-pool-name="UCP_pool1"
    user="dbuser"
    password="dbuserpasswd"
    connection-factory-class-name="oracle.jdbc.pool.OracleDataSource"
    initial-pool-size="10"
    min-pool-size="5"
    max-pool-size="15"
    validate-connection-on-borrow="true"
    connection-wait-timeout="900"
    max-connections-per-service="50"
    sql-for-validate-connection="select 1 from dual"
    url="jdbc:oracle:thin:@(DESCRIPTION =(CONNECT_TIMEOUT= 120)(RETRY_COUNT=20)
(RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST =(LOAD_BALANCE=on)(ADDRESS
= (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))(ADDRESS_LIST
=(LOAD_BALANCE=on)(ADDRESS = (PROTOCOL = TCP)(HOST=standby-
scan)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME = MY-SERVICE)))"
    fastConnectionFailoverEnabled="true"
    onsConfiguration="nodes=primary-scanhost:6200,secondary-
scanhost:6200\nwalletfile=/replace_with_host_path/onswallet\nwalletpassword=myWalletP assword">
  </connection-pool>
</ucp-properties>
```

     - setONSConfiguration()へのコールを使用して、UCP内でプログラムによって設定します。以下に例を示します。

```
pds.setONSConfiguration("nodes=primary-scanhost:6200,secondary-
scanhost:6200\nwalletfile=/replace_this_with_host_path/onswallet\nwalletpassword=myWa
lletPassword");
```

Oracleドライバのバージョン12.2以降を使用したOracle Call Interface（OCI）アプリケーションの場合

oraaccess.xmlファイルの<default_parameters>セクションに以下を追加します。

```
<default_parameters>
       （その他の設定がこのセクションに含まれる場合があります）
   <events>
      true
   </events>
   <ons>
      <auto_config>true</auto_config>
      <wallet_location>/my_path/onswallet</wallet_location>
   </ons>
</default_parameters>
```

<wallet_location>パスには、ウォレットが含まれているディレクトリの名前を指定する必要があります。

<hosts>、<max_connections>、<subscription_wait_timeout>などのその他のパラメータは、oraaccess.xmlのonsセクションで設定できます。

ネイティブ・イベントの設定制御をサポートしているドライバでは、<events>セクションを省略して、代わりにドライバ設定を使用できます。

デフォルトでは、ONSが失敗した場合でもデータベースへのアプリケーション接続は正常に行われます。そのため、アプリケーションの実行を続行できます。ONSの初期構成を検証するために、ONSが機能していないときにアプリケーションがデータベースに接続しようとした場合に、強制的にエラーをスローさせることができます。oraaccess.xmlに、<ons>と同じレベルでセクションを追加します。これは、診断の目的に限って使用する必要があります。

```
<fan>
   <subscription_failure_action>
       error
   </subscription_failure_action>
</fan>
```

ネットワーク・ファイルtnsnames.oraおよびsqlnet.oraと同じディレクトリにoraaccess.xmlファイルを配置します。たとえば、Oracle Instant Clientを使用する場合、これらのファイルをデフォルト・ディレクトリnetwork/adminに配置できます。または、アクセス可能な別のディレクトリにすべてのネットワーク構成ファイルを配置することもできます。配置したら、環境変数TNS_ADMINをそのディレクトリの名前に設定します。

ODP.Net管理対象プロバイダ

application.configファイルを使用してONS構成とウォレット・ロケーションを指定します。以下に例を示します。

```
<oracle.manageddataaccess.client>
    <version number="*">
      <onsConfig mode="remote">
        <settings>
          <setting name="Protocol" value="TCPS" />
          <setting name="WALLET_LOCATION" value="C:\myPath\ONS_SSLWallet" />
        </settings>
        <ons database="atp01db">
          <add name="nodeList" value="racNode1:6205,racNode2:6205,racNode3:6205" />
        </ons>
      </onsConfig>
    </version>
</oracle.manageddataaccess.client>
```

付録：Oracleライブラリの追加の技術資料

高速アプリケーション通知

http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/learnmore/fastapplicationnotification12c-2538999.pdf

JAVAアプリケーション・サーバーにUCPを埋め込む方法

『WLS UCP Datasource』（https://blogs.oracle.com/weblogicserver/wls-ucp-datasource）

『Design and Deploy WebSphere, IBM Liberty Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP』https://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-websphere-2409214.pdf）

「Reactive programming in microservices with MicroProfile on Open Liberty 19.0.0.4」
（https://openliberty.io/blog/2019/04/26/reactive-microservices-microprofile-19004.html#oracle）

『Design and deploy Tomcat Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP』
（http://www.oracle.com/technetwork/database/application-development/planned-unplanned-rlb-ucp-tomcat-2265175.pdf）

『Using Universal Connection Pool with WildFly (JBoss) AS』（https://blogs.oracle.com/developers/post/using-universal-connection-pooling-ucp-with-jboss-as）

アプリケーション・コンティニュイティ

『Application Continuity with Oracle Database12c Release 2』
（http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/overview/application-continuity-wp-12c-1966213.pdf）

『アプリケーション・コンティニュイティの確保』（https://docs.oracle.com/cd/E96517_01/racad/ensuring-application-continuity.html#GUID-C1EF6BDA-5F90-448F-A1E2-DC15AD5CFE75）

透過的アプリケーション・コンティニュイティ

https://docs.oracle.com/cd/E96517_01/adfns/high-availability.html#GUID-96599425-9BDA-483C-9BA2-4A4D13013A37

トランザクション・ガード

『Oracle Database 12c Release 2でのトランザクション・ガード』
（https://www.oracle.com/technetwork/jp/database/database-cloud/private/transaction-guard-wp-12c-1966209-ja.pdf）

ORACLE CORPORATION

Worldwide 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+ 1.800.ORACLE1 FAX
＋ 1.650.506.7200
oracle.com

オラクルの情報を発信しています

+1.800.ORACLE1までご連絡いただくか、oracle.comをご覧ください。北米以外の地域では、oracle.com/contactで最寄りの営業所をご確認いただけます。

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

Integrated Cloud Applications & Platform Services

Copyright © 2020, 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、その内容に誤りがないことを保証するものではなく、また、口頭による明示的保証や法律による黙示的保証を含め、商品性ないし特定目的適合性に関する黙示的保証および条件などのいかなる保証および条件も提供するものではありません。オラクル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。0920

ホワイト・ペーパー・タイトル 継続的な可用性：Oracle Autonomous Databaseの専用デプロイメントを使用したアプリケーションのベスト・プラクティス
2019年5月
著者：Troy AnthonyおよびCarol Colrain
貢献者：Lawrence To、Ian Cookson、Hector Pujol、Hairong Qin

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