19c ORACLE Database

アプリケーション・コンティニュイティ

準備のためのMAAチェックリスト

Oracleホワイト・ペーパー / 2020年7月13日

ORACLE

概要

以下のチェックリストは、Oracle Databaseのアプリケーション・コンティニュイティ機能を使用するようにお使いの環境を準備する場合に役立ちます。アプリケーション・コンティニュイティがお使いのデータベース・サービスで有効になっていない場合、あるいはお使いのアプリケーションで使用しない場合でも、このホワイト・ペーパーに記載されている内容には、可用性を維持できるようにシステムを準備する上で大きな価値があります。

次の手順を実行してください。

- データベース・サービスの使用
- URLまたは接続文字列における高可用性の構成
- 高速アプリケーション通知（FAN）の有効化
- ドレイニングに対応する推奨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
- 保護レベルの確認
- クライアントの構成
- アプリケーションとサーバーのタイムアウトの整合

データベース・サービスの使用

サービスとは、作業を管理するためのロジックを抽象化したものです。サービスは、単一のシステム・イメージによって作業を管理するため、基盤システムの複雑さはクライアントからは分かりません。アプリケーション・コンティニュイティを使用するには、サービスを使用する必要があります。このサービスは、デフォルトのデータベース・サービスやデフォルトのPDBサービスではありません。

複数のサイトを使用する場合、プライマリ・サイトのプライマリ・ロール、およびOracle Active Data Guardで開かれるサービスのスタンバイ・ロールを使用して、サービスを作成する必要があります。サービスは、ロールに基づき、サイトで自動的に起動および停止されます。

アプリケーション・コンティニュイティを使用するためのサービスを作成するには、次のようなコマンドを使用します。

注：接続文字列でRETRY_COUNTおよびRETRY_DELAYが推奨どおり設定されている場合は、failoverretryおよびfailoverdelayは必要ありません。また、ここでも表示していません。

透過的アプリケーション・コンティニュイティ

```
$ srvctl add service -db mydb -service TACSERVICE –preferred inst1 -available inst2 -failover_restore
AUTO -commit_outcome TRUE -failovertype AUTO - replay_init_time 600 -retention 86400 -notification
TRUE -drain_timeout 300 - stopoption IMMEDIATE
```

アプリケーション・コンティニュイティ

```
$ srvctl add service -db mydb -service ACSERVICE -preferred inst1 -available inst2 -failover_restore
LEVEL1 -commit_outcome TRUE -failovertype TRANSACTION - session_state dynamic -
replay_init_time 600 -retention 86400 -notification TRUE - drain_timeout 300 -stopoption IMMEDIATE
```

URLまたは接続文字列における高可用性の構成

オラクルでは、フェイルオーバー、スイッチオーバー、フォールバック、および基本的な起動時に正常に接続されるようにするために、次の接続文字列の構成を推奨しています。

tnsnames.oraファイルまたはURLで、RETRY_COUNT、RETRY_DELAY、CONNECT_TIMEOUT、TRANSPORT_CONNECT_TIMEOUTパラメータを設定し、接続リクエストがサービスの可用性を待って正常に接続されるようにします。

Oracle Real Application Clusters（Oracle RAC）およびOracle Data Guardのフェイルオーバー時間を許容できる値を使用します。

CONNECT_TIMEOUTを高い値に設定して、ログイン・ストームを回避します。値を低くすると、アプリケーションまたはプールのキャンセルや接続の再試行のために、ログイン‘合戦’が発生することがあります。

RETRY_COUNTを使用する場合は、常にRETRY_DELAYを設定します。

Oracle Database 12c Release 2（12.2）より、センチ秒（cs）またはミリ秒（ms）のいずれかの単位で待機時間を設定できます。デフォルトの単位は秒（s）です。Oracle Database Release 2のOracle JDBCドライバを使用している場合、バグのパッチ26079621を適用する必要があります。

EZCONNECTではFANの自動構成機能が回避されるため、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。

LDAPやtnsnames.oraといった中央のロケーションでTNS/URLを保守します。プロパティ・ファイルやプライベート・ロケーションにTNS/URLを分散させないでください。保守が極めて困難になります。一元管理されるロケーションを使用すると、標準の形式、チューニング、サービス設定を保持するのに役立ちます。

次に示すのは、12.2以降のすべてのOracleドライバに推奨される接続文字列です。特定の値がチューニングされる場合があります。

```
Alias (or URL) = (DESCRIPTION =
(CONNECT_TIMEOUT= 90)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
 (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    (ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521))) (ADDRESS_LIST =
    (LOAD_BALANCE=on)
    (ADDRESS = (PROTOCOL = TCP)(HOST=secondary-scan)(PORT=1521)))
 (CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

Oracle Database 12c Release 1（12.1）のJDBC接続では、以下を使用する必要があります。特定の値がチューニングされる場合があります。また、バグ用のパッチ19154304（パッチ・リクエスト18695876）をRETRY_COUNT機能に適用する必要があります。

```
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)(RETRY_COUNT=20)(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=secondary-scan)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = gold-cloud)))
```

高速アプリケーション通知（FAN）の有効化

FANを有効にする必要があります。FANは、アプリケーションのフェイルオーバーを中断する場合の必須コンポーネントです。ノードまたはネットワークが停止した場合は、アプリケーションをリアルタイムで中断する必要があります。FANを有効化しておかないと、ノード、ネットワーク、サイトの障害など、ハードの物理的障害が発生した場合にアプリケーションがハングします。

- FANは、Oracle Real Application Clustersに付属しているデフォルトの機能で、自動構成されて有効化されています。アプリケーションが接続されると、FANはURLまたはTNS接続文字列を読み取り、クライアントで自動構成されます。上記のURL形式を使用することは、FANを自動構成する上で重要です（別の形式を使用すると、FANを自動構成できなくなります）。

Oracle Database 18cおよびOracleクライアント18c以降、次の重要な2つの機能強化が図られています。

- 計画メンテナンス（停止イベント）では、FANは帯域内でドライバに直接送信されます
- Oracle DatabaseおよびOracleクライアント・ドライバは、接続テストおよびリクエスト境界でドレイニングを行います

サービスを使用することで、FANはデータベースとクラスタ・サーバーで有効化されます。

クライアント固有の手順について詳しくは、クライアントの構成を参照してください。FANを使用するためにアプリケーション・コードを変更する必要はありません。

ドレイニングに対応する推奨プラクティスの使用

ベスト・プラクティスに従って計画メンテナンスを行う場合は、アプリケーション・サーバーを再起動する必要はありません。

計画メンテナンスを行う場合は、メンテナンスを開始する前に、進行中の作業を完了させる時間を確保することを推奨します。作業をドレイニングすることでこれを行います。割り当てられた時間内に完了しないリクエストに対しては、選択したフェイルオーバー・ソリューションと組み合わせてドレイニングを使用します。割り当てられた時間内にドレイニングが完了しなかったセッションについては、選択したフェイルオーバー・ソリューションによってリカバリが試行されます。

ドレイニングのためのクライアントの手順

推奨アプローチ

計画メンテナンスを認識させないためのソリューションとしては、FANに対応したOracle接続プールを使用する方法を推奨します。Oracleプールは、ドレイニング、再接続、MAAシステム内でのリバランシングといったライフサイクル全体に対応しています。メンテナンスの進行に合わせてセッションの移動とリバランシングが行われ、完了するとまた移動とリバランシングが行われます。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変更などを行う必要は一切ありません。

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、現在のアクションの操作が完了したら接続をプールにチェックインするのがベスト・プラクティスです。これは、アプリケーションを最高のパフォーマンスで実行するために重要ですが、実行時、メンテナンス時、フェイルオーバー・イベントの発生時に作業をリバランスするためにも重要です。

サード・パーティのJavaベースのアプリケーション・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。
このアプローチは、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・サーバーでサポートされています。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。

代替アプローチ – Oracle Database 19cによるドレイニングまたは Oracle Driver 19cによるドレイニング

Oracleプールを使用できない場合やFANを使用しない場合でも、Oracle Database 18c以降およびOracleクライアント・ドライバ18c以降では、セッションがドレイニングされます。サービスが再配置された場合や停止した場合、またはOracle Data Guardにスイッチオーバーされた場合、Oracle DatabaseとOracleクライアント・ドライバは、次のルールに従って、接続をリリースするための安全な場所を検索します。

- 接続を検証する標準アプリケーション・サーバー接続テスト
- 接続を検証するカスタムSQLテスト
- リクエストの境界の有効化と現在のリクエストの終了

追加した接続テストや有効化した接続テストは、ビューDBA_CONNECTION_TESTSで確認できます。サービス、プラガブル・データベース、非コンテナ・データベースへの接続テストを追加、削除、有効化、無効化できます。例：

```
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> set linesize 120

SQL> SELECT * FROM DBA_CONNECTION_TESTS
```

Oracleプールまたは接続テストとともにFANを使用する場合のクライアント固有の構成については、クライアント構成を参照してください。

ドレイニングのためのサーバーの手順

Oracle Databaseに接続するサービスは、接続テストを行うように構成し、ドレイニングに使用できる時間をdrain_timeoutに指定し、ドレイニング・タイムアウトの終了後に適用されるstopoptionにIMMEDIATEを指定します。
SRVCTLで管理する停止、再配置、スイッチオーバーの各コマンドには、drain_timeoutおよびstopoptionスイッチが含まれており、サービスで設定された値を必要に応じてオーバーライドできます。

保守コマンドは、以下で説明するコマンドと類似しています。Oracle Fleet Patching and Provisioning（Oracle FPP）などのOracleツールでは、これらのコマンドを使用します。これらのコマンドを使用してドレイニングを開始します。必要に応じて追加オプションを含めます。詳しくは、My Oracle Support（MOS）NoteのDoc ID 1593712.1を参照してください。

Oracle RAC上のデータベース、ノードまたはpdb別にすべてのサービスを再配置する

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

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

***inst1*という名前のインスタンス（任意のインスタンス）でGOLDという名前のサービスを停止する**

srvctl stop service –db myDB –service GOLD –instance inst1 -drain_timeout <timeout> -stopoption <stop_option>

すべてのインスタンスでGOLDという名前のサービスを停止する

srvctl stop service –db myDB –service GOLD -drain_timeout <timeout> -stopoption <stop_option>

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

データベース・サービスで設定されるアプリケーション・コンティニュイティには、アプリケーションに応じて、次の2つの構成のいずれかが使用されます。

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、停止を認識させないようにする機能です。Oracleデータベース12.1以降、Javaベースのシン・アプリケーションで使用でき、Oracle Database 19c以降は、オープン・ソース・ドライバ（Node.js、Pythonなど）をサポートするOracle Call Interface（OCI）およびODP.NETベースのアプリケーションでも使用できます。アプリケーション・コンティニュイティを有効にすると、セッションの状態やトランザクションの状態などがわかっている時点からセッションがリカバリされて再構築されます。処理中のすべての作業がアプリケーション・コンティニュイティによって再構築されます。フェイルオーバーが行われるときに実行時間がわずかに長くなりますが、アプリケーションはそれまでと同様に動作し続けます。アプリケーション・コンティニュイティの標準モードは、Oracle接続プールを使用したOLTPアプリケーションに適しています。

透過的アプリケーション・コンティニュイティ（TAC）

Oracle Database 19c以降、セッションとトランザクションの状態を透過的に追跡および記録する透過的アプリケーション・コンティニュイティ機能が導入され、リカバリ可能な停止からデータベース・セッションをリカバリできるようになりました。アプリケーションの透過的アプリケーション・コンティニュイティを有効にするためには、アプリケーションの知識もアプリケーションのコード変更も必要ありません。アプリケーションを透過的にフェイルオーバーするため、アプリケーションからユーザー・コールが発行されたときのセッション状態の使用状況を取得して分類した状態追跡情報が使用されます。

	TAC	AC
アプリケーションの実装方法が分からない場合	対応	非対応
ステートレスのアプリケーションの場合（Oracle REST、Oracle APEX、マイクロサービスなど）	対応	対応（推奨）
トランザクションを実行するアプリケーションの場合	対応	対応
Oracleセッションの状態（一時LOB、一時表、PL/SQLグローバル変数）を使用するアプリケーションの場合	対応	対応 非対応（静的モードの場合）
Oracle接続プールを使用しないアプリケーションの場合	対応	非対応
副次的影響（ファイル転送や電子メールなどの外部アクション）があるアプリケーションの場合	対応 副次的影響は再実行されません	対応 次的影響を再実行するかどうかをカスタマイズできます

データベース・リクエストを再実行するアプリケーション・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機能には、いくつかの制約があります。これらの制約について詳しくは、Oracleドキュメント[1]を参照してください。

アプリケーション・コンティニュイティを使用する場合の手順

開発者はこれらの手順に従う必要があります。データベース構成については、データベース管理者のサポートを依頼してください。

1 最新のOracleドキュメントを参照してください。バージョン19cの最新ドキュメントは以下で参照できます。
https://docs.oracle.com/en/database/oracle/oracle- database/19/racad/ensuring-application-continuity.html#GUID-2400FAAD-0BB2-48AF-B1F6-358EBA724028

接続プールへの接続の返却

アプリケーションで使用する接続は、リクエストのたびにOracle接続プールに返却する必要があります。アプリケーションで必要になったときだけ接続をチェックアウトするのがベスト・プラクティスです。接続をプールに返却せずに、保持し続けることはできません。したがってアプリケーションでは、接続をチェックアウトし、処理が完了したらすぐにその接続をチェックインします。そうすることで、他のスレッドや、自分のスレッドで再び必要になったときに接続を使用できます。

Oracle Universal Connection Pool（Oracle UCP）やOCIセッション・プールといったOracle接続プール、ODP.Net管理対象外プロバイダ、またはWebLogic Active GridLinkを使用している場合にこのプラクティスに従うと、リクエストの境界も埋め込まれ、キャプチャの再開と終了を安全に実行できる場所をアプリケーション・コンティニュイティで識別できるようになります。これはアプリケーション・コンティニュイティでは必須のプラクティスであり、透過的アプリケーション・コンティニュイティでは推奨のプラクティスです。

透過的アプリケーション・コンティニュイティでは、プールが使用されていない場合や再実行が無効化されている場合もリクエスト境界が検出されます。Oracle Database 19cで境界が検出される条件は以下のとおりです。

- オープンなトランザクションがない
- カーソルがステートメント・キャッシュに戻されたか、取り消されている
- リストア不可能なセッション状態が存在しない（リクエスト間のセッション状態の消去を参照）

データベース・リクエスト間のセッション状態を消去するのがベスト・プラクティスです。以下のセクションを参照してください。

サービスでのFAILOVER_RESTOREの構成

属性FAILOVER_RESTOREをデータベース・サービスで設定する必要があります。

リクエスト以外で接続にセッションの状態を意図的に設定し、この状態を想定してリクエストを行う場合は、再実行の前にこの状態を作成し直して再実行する必要があります。

サービスでFAILOVER_RESTOREを設定すると、もっとも一般的なセッション状態が自動的にリストアされます。Oracle Database 19c RU 6以降、変更可能なすべてのシステム・パラメータはウォレットを使用してフェイルオーバーでリストアされるようになりました（Oracleドキュメント『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」を参照してください）。ALTER SESSIONコマンドを使用して設定する場合にウォレットが必要なパラメータの一例は、USE_STORED_OUTLINESです。パラメータがinit.oraで指定された場合や、ログオン・トリガーなどのカスタムの方法を使用して指定された場合は、操作は不要です。

ACではFAILOVER_RESTORE=LEVEL1を、TACではFAILOVER_RESTORE=AUTOを使用します。接続確立時にカスタムの値を構成するには以下を使用します。

- ログオン・トリガー
- 接続初期化コールバックまたはUCPラベル（Javaの場合）、もしくはTAFコールバック（OCI、ODP.Net、オープン・ソース・ドライバの場合）
- Oracle Universal Connection PoolまたはOracle WebLogic Serverの接続ラベル付け

アプリケーションで使用される可変値の有効化

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、可変関数の元の結果を保持できるようになっています。元の値が保持されず、再実行時に異なる値がアプリケーションに返される場合は、再実行が拒否されます。Oracle Database 19cではSQLの可変値が自動的に保持されます。

PL/SQLの可変値が必要な場合、またはOracle Database 19c以前のデータベース・バージョンを使用している場合は、可変値を構成しますが、このときアプリケーション・ユーザーにはGRANT KEEPを使用し、シーケンス所有者にはKEEP句を使用します。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。

例：

```
SQL> GRANT KEEP DATE TIME to scott;

SQL> GRANT KEEP SYSGUID to scott;

SQL> GRANT KEEP SEQUENCE mySequence to scott on mysequence.myobject;
```

副次的影響

データベース・リクエストに電子メール送信やファイル送信などの外部コールが含まれている場合、これは副次的影響です。

副次的影響は外部アクションのため、把握することが重要です。外部アクションはトランザクションではないため、ロールバックされません。再実行時に副次的影響を再実行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、ジャーナルの入力や電子メールの送信といった副次的影響を再実行することが選択されています。アプリケーション・コンティニュイティでは、リクエストやユーザー・コールの再実行が明示的に無効化されていない限り、副次的影響は再実行されます。一方、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有効なため、TACでは保守的なアプローチが取られ、副次的影響は再実行されません。キャプチャは無効化されており、次にTACによって暗黙的な境界が作成されたときに再有効化されます。

リクエスト間のセッション状態の消去

アプリケーションによって接続が接続プールに返却されても、フェッチ状態のカーソルとそのセッションで設定されたセッション状態は、消去するための操作を行わない限りそのまま残ります。お使いのアプリケーションが状態を設定している場合、後でそのデータベース・セッションを再利用する際にリークを避けるために、カーソルをステートメント・キャッシュに戻し、アプリケーションに関連するセッション状態を消去するのがベスト・プラクティスです。

セッション状態を消去すると、TACで境界を検出できます。

PL/SQLグローバル変数を消去するにはDBMS_SESSION.RESET_PACKAGEを、一時表を消去するにはTRUNCATEを使用します。また、コンテキストを消去し、カーソルをステートメント・キャッシュに戻して取り消すには、SYS_CONTEXT.CLEAR_CONTEXTを使用します。この操作を行うと、状態のリークが回避され、後で接続プールを使用した際にカーソルからフェッチされることがなくなります。

ステートレスのアプリケーション（Oracle REST、Oracle APEX、マイクロサービスなど）では、状態を消去するのがベスト・プラクティスです。

問合せでのORDER BYまたはGROUP BYの使用

アプリケーション・コンティニュイティによって、再実行時にアプリケーションには確実に同じデータが表示されます。同じデータをリストアできない場合、アプリケーション・コンティニュイティでは再実行が許可されません。SELECTでORDER BYまたはGROUP BYが使用されている場合は、順序が保存されます。Oracle RAC環境では、問合せオプティマイザはほとんどの場合同じアクセス・パスを使用するため、同じ順序で結果が表示されます。アプリケーション・コンティニュイティは、AS OF句が許可されている場合は、内部でAS OF句を使用して同じ問合せ結果を戻します。AS OFはトランザクションやPL/SQL内では使用されません。

SQL*Plusの考慮事項

SQL*Plusは通常、何かを試す際の主要ツールです。当然ながらSQL*Plusは、本番環境で使用される実際のアプリケーションを反映していないため、常に本物のアプリケーション・テスト一式を使用してフェイルオーバー計画をテストし、保護レベルを測定した方が良いでしょう。SQL*Plusはプールされるアプリケーションではないため、明示的なリクエスト境界はありません。一部のアプリケーションでは、たとえばレポートなどでSQL*Plusを使用します。フェイルオーバーと一緒にSQL*Plusを使用する場合は、次の手順を確認してください。

1. FANはSQL*Plusでは常に有効化されています。ONSエンド・ポイントを自動構成する上記のTNSを使用します。
2. SQL*plusを使用する上で重要なのは、データベースに対するラウンドトリップを最小限に抑えることです（https://blogs.oracle.com/opal/sqlplus-12201- adds-new-performance-features）。
3. SQL*PlusはOracle 19c以降、TACでサポートされています。最適な結果を得るには、大きい配列サイズ（1000など）を設定します。serveroutputを有効化しないようにしてください。有効化すると、リストアできないセッション状態が生成されます。（この制限を排除するには、リリースノートを参照してください。）

4. SQL*PlusはOracle 12.2以降、ACでサポートされています。ACには暗黙的な境界がないため、最初のコミット後に再有効化されません。

エンド・ツー・エンド（e2e）のトレース

アプリケーション・コンティニュイティの統計情報およびACCHKは、サーバーごとに分離されるほか、モジュールとアクションごとにも分離されます。各種サービスを使用するでしょうが、アプリケーションではモジュールとアクションを使用して作業を指定するのが良いプラクティスです。モジュールとアクションのタグを使用している場合、EMのTop Consumers、AWR、統計情報、ACCHKで報告されます。モジュールとアクションを設定するには、PL/SQLのBMS_APPLICATION_INFOではなく、ドライバが提供するAPIを使用します。APIはローカル・ドライバ・コールであるため、パフォーマンスがより優れているためです。

保護レベルの確認

リクエストの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V$SESSTAT、V$SYSSTATで確認でき、Oracle Database 19cではV$SERVICE_STATSで確認できます。
統計情報は自動ワークロード・リポジトリに保存され、自動ワークロード・リポジトリ・レポートで確認できます。
（Oracle Database 18cを使用した場合の統計情報では、_request_boundaries=3を設定します）。

サービスごとに保護履歴を報告するには、たとえば以下を実行します。

```
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;
```

このレポート問合せにより、以下のような結果が得られます。

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

acchkカバレッジ・レポートの実行

Orachkのacchk機能では、保護レベルの詳細についてのレポートが提供されます。acchkを使用するには、以下の手順に従います。

データベースのトレース機能をオンにします。

ワークロードの実行前に、必要な情報がトレース・ファイルに含まれるように、テスト用のOracle Databaseサーバー上で以下の文をDBAとして実行します。

```
SQL> alter system set events='10602 trace name context forever, level 28:
trace[progint_appcont_rdbms]:10702 trace name context forever, level 16';
```

ア
プリケーション関数を次々と実行します。アプリケーション関数についてレポートを作成するには、アプリケーション関数を実行する必要があります。実行するアプリケーション関数の数が多いほど、カバレッジ分析の情報が詳細になります。

完了時にイベントをオフにするには、以下を実行します。

```
SQL> > alter system set events='10602 trace name context forever, off:
trace[progint_appcont_rdbms] off : 10702 trace name context forever, off';
```

L

Oracle ORAchkを使用して、収集したデータベース・トレースを分析し、保護レベルを報告します。保護されていない場合は、リクエストが保護されない理由を報告します。次のコマンドを使用します。

```
./orachk -acchk -javahome /scratch/nfs/jdk1.8.0_171 –apptrc $ORACLE_BASE/diag/rdbms/myDB/myDB1/trace
```

コマンドライン引数	シェル環境変数	使用方法
-javahome JDK8dirname	RAT_JAVA_HOME	JAVA_HOMEディレクトリを参照する必要があります。
-apptrc dirname	RAT_AC_TRCDIR	カバレッジを分析するには、データベース・サーバー・トレース・ファイルが1つ以上あるディレクトリ名を指定します。トレース・ディレクトリは通常、以下になります。 $ORACLE_BASE/diag/rdbms/{DB_UNIQUE_NAME}/$ORACLE_SID/trace

クライアントの構成

アプリケーション・コンティニュイティおよびJDBCアプリケーション

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. プロパティ・ファイルまたはコンソールでOracle JDBC再実行データソースを構成します。
3. プロパティ・ファイルまたはコンソールでOracle JDBC再実行データソースを構成します。

    a. Oracle Universal Connection Poolの場合

    UCP PoolDataSourceに対して、Oracle JDBC再実行データソースをコネクション・ファクトリとして構成します。setConnectionFactoryClassName(“oracle.jdbc.replay.OracleDataSourceImpl”);または setConnectionFactoryClassName(“oracle.jdbc.replay.OracleXADataSourceImpl”);またはプロパティ・ファイルでこれらを優先設定

    b. WebLogic Serverの場合、Oracle WebLogic Server管理コンソールを使用して、次のローカルの再実行ドライバまたはXA再実行ドライバを選択します。
    Active GridLinkアプリケーション・コンティニュイティ接続向けのOracle Driver（Thin） Active GridLinkアプリケーション・コンティニュイティ接続向けのOracle Driver（Thin XA）

    c. 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場合

    プロパティ・ファイル内またはシン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。datasource=oracle.jdbc.replay.OracleDataSourceImpl（非XA向け）または datasource=oracle.jdbc.replay.OracleXADataSourceImpl（XA向け）

4. JDBCステートメント・キャッシュを使用します。

    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、ドライバはステートメントが取り消されることを認識できるようになるため、リクエストの終了時にメモリを解放できます。
    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。例：
    oracle.jdbc.implicitStatementCacheSize=*nnn* ここで、*nnn*は通常、50～200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。

5. 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。

6. コミット

    JDBCアプリケーションでAUTOCOMMITを使用する必要がない場合は、アプリケーションまたは接続のプロパティでAUTOCOMMITを無効にします。Apache Tomcat、IBM WebSphere、IBM Liberty、Red Hat WildFly（JBoss）などのサード・パーティ製アプリケーション・サーバーにUCPまたは再実行ドライバが埋め込まれている場合、この作業は重要です。
    UCP PoolDataSource接続プロパティを使用してautoCommitをfalseに設定します。

    ```
    connectionProperties="{autoCommit=false}"
    ```

7. JDBC具象クラス
   JDBCアプリケーションについては、非推奨のoracle.sql具象クラス（BLOB、CLOB、BFILE、OPAQUE、ARRAY、STRUCT、またはORADATA）はアプリケーション・コンティニュイティで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クライアントで認識するには、ORAchk -acchkを使用します。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。

8. 高速接続フェイルオーバー（FCF）と呼ばれるJava向けのFAN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。
   - サード・パーティのJDBCプールの場合は、Universal Connection Poolが推奨されます。
   - 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）。

   推奨の接続文字列を使用できない場合は、以下の設定を使用して手作業でクライアントを構成します。

   oracle.ons.nodes =XXX01:6200, XXX02:6200, XXX03:6200

アプリケーション・コンティニュイティおよびOCIベースのクライアント

OCIベースのクライアントには、Oracle 19c以降、Node.js、Python、SODAなどが含まれます。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 2511448.1*）を参照してください。
2. サポートされるステートメントの全一覧については、ドキュメントを参照してください。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およびサポートされないその他のOCI APIは許可されますが、これら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/high-availability-in-oci.html#GUID-D30079AC-4E59-4CC3-86E8-6487A4891BA2

   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。

- サービスでaq_ha_notificationsを設定します。
- 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。

```
<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>/path/onswallet</wallet_location>
   </ons>
</default_parameters>
```

- オープン・ソースを含む多くのアプリケーションは、スレッド化されます。スレッド化されない場合は、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）。

推奨の接続文字列を使用できない場合は、以下の設定を使用して手作業でクライアントを構成します。

ネイティブ設定を使用しないOracle Call Interface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。

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 2511448.1）を参照してください。
2. 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。

- サービスでaq_ha_notificationsを設定します。
- 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onsConfigおよびwallet_location（オプション）を設定します。
- 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）。
- 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;“

アプリケーションとサーバーのタイムアウトの整合

アプリケーション・レベルのタイムアウトが、基盤システムの検出およびリカバリのための時間として設定されているタイムアウトより短いと、基盤となるリカバリと再実行を完了するための時間が不足します。時間の整合性が保たれていないと、システムのリカバリが完了する前にアプリケーション・コンティニュイティによる再実行が開始され、成功するまで再実行が繰り返し試行される可能性や、リクエストがタイムアウトしてアプリケーションまたはユーザーにエラーが返される可能性があります。

リソース・マネージャは、長時間実行されているSQLを停止、隔離、または降格するため、およびSQLが最初から実行されないようにするための推奨機能です。ハングしたシステムや完全に停止したシステムにREAD_TIMEOUTを使用する場合は、READ_TIMEOUTをリカバリ・タイムアウトより大きい値にする必要があります。小さい値のREAD_TIMEOUTを使用することは推奨されません。大量に再試行が行われ、途中で中断が発生する可能性があります。

アプリケーションでREAD_TIMEOUTやHTTP_REQUEST_TIMEOUT、または何らかのカスタム・タイムアウトを使用することを検討してください。続いて、次の指針を適用します。

READ_TIMEOUT ＞　EXADATAの特別なノード・エビクション（FDDN）（2秒）

READ_TIMEOUT ＞　MISSCOUNT（デフォルトは30秒、12c Grid Infrastructureでは変更可能）

READ_TIMEOUT ＞ Data Guard Observer: FastStartFailoverThreshold（デフォルトは30秒、変更可能）

FastStartFailoverThreshold > MISSCOUNT（2回以上）

READ_TIMEOUT > FAST_START_MTTR_TARGET（多くのシステムでは15または30秒を選択）

READ_TIMEOUT > Oracle SQL*NET level: (RETRY_COUNT+1) * RETRY_DELAY

READ_TIMEOUT < Replay_Initiation_Timeout（サービス上で変更可能、デフォルトは300秒）

リクエストのキャンセルが早すぎないようにするため、アプリケーションのタイムアウト値を次の最大値より大きい値にする必要があります。

(MISSCOUNT（またはFDNN） + FAST_START_MTTR_TARGET), (FastStartFailoverThreshold + FAST_START_MTTR_TARGET + 開く時間)

その他の資料

アプリケーション・コンティニュイティのOracle Technology Network（OTN）ホームページ
https://www.oracle.com/jp/database/technologies/high-availability/app-continuity.html

アプリケーション・コンティニュイティ

『Continuous Availability, Application Continuity for the Oracle Database』

(https://www.oracle.com/technetwork//database/options/clustering/applicationcontinuity/applicationcontinuityformaa-6348196.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/en/database/oracle/oracle-database/18/racad/ensuring-application-continuity.html#GUID-C1EF6BDA-5F90-448F-A1E2-DC15AD5CFE75）

『Oracle Database 12c Release 2のアプリケーション・コンティニュイティ』
（http://www.oracle.com/technetwork/database/options/clustering/applicationcontinuity/overview/application-continuity-wp-12c-1966213.pdf）

『Graceful Application Switchover in RAC with No Application Interruption』

My Oracle Support（MOS）Note：Doc ID 1593712.1

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 Applications for Planned, Unplanned Database Downtimes and Runtime Load Balancing with UCP』（http://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 JBoss AS』（https://blogs.oracle.com/dev2dev/using-universal-connection-pooling-ucp-with-jboss-as）

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

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

2020年5月
著者：Carol Colrain、Troy Anthony、Ian Cookson
共著者：

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