Oracleホワイト・ペーパー
2010年4月

Oracle Database Resource Manager の使用

はじめに 1
第1部：リソース・マネージャの紹介 2
リソース・マネージャのシナリオ 2
リソース・マネージャの概要 3
リソース・マネージャの仕組み 3
複数のワークロードの管理 4
よくある誤解 4
第2部：健全なデータベースのためのリソース・マネージャの構成 6
インスタンス・ケージング 6
第3部：複数のワークロードのためのリソース・マネージャの構成 8
手順1：リソース・マネージャを管理するための権限の取得 8
手順3：コンシューマ・グループの作成 9
手順4：コンシューマ・グループへのセッションのマッピング 9
手順5：コンシューマ・グループの権限の追加 10
手順6：リソース・プランの作成 11
手順7：リソース・プラン・ディレクティブの追加 12
手順8：ペンディング・エリアの発行 14
手順9：リソース・プランの有効化 14
第4部：リソース・マネージャの監視とチューニング 15
リソース・マネージャの有効化 15
現在、リソース・マネージャは有効化されていますか。 15

リソース・マネージャが有効化されたのはいつですか。 15
リソース・マネージャはどのようにして有効化されましたか。 16
コンシューマ・グループへのセッションのマッピング 16
セッションはどのコンシューマ・グループにマッピングされていますか。....16
想定したコンシューマ・グループにセッションがマッピングされていないのは
なぜですか。 16
CPU管理の監視 18
データベース・インスタンスはリソース・マネージャによって
どの程度スロットリングされていますか。....18
コンシューマ・グループはリソース・マネージャによって
どの程度スロットリングされますか。....20
結論 22

はじめに

健全なデータベースを実現するうえで重要なポイントの1つは、適切なCPU負荷を維持することです。CPU負荷が過剰になると、サーバーが不安定になり、オペレーティング・システムのバグが露見することがあります。また、重要なOracleバックグラウンド・プロセスが適切なタイミングで実行されなくなり、Oracle Real Application Clusters（Oracle RAC）データベースのデータベース・インスタンスが排除されるなどの障害につながることもあります。このような状態のときは応答時間が非常に遅くなっているため、たとえば障害のあるSQLプランを特定して修正するなど、過度のCPU負荷を引き起こしている原因をデバッグすることが不可能な場合があります。

Oracle Database Resource Manager（リソース・マネージャ）を使用すると、データベースのCPU負荷を常に適切な状態に維持できるため、このような問題をすべて回避できます。リソース・マネージャでCPU負荷を管理すると、オペレーティング・システムの実行キューで待機中のＯｒａｃｌｅプロセスがない（またはほとんどない）状態になるため、すべてのCPUが完全に使用されはするものの、スラッシングは発生しなくなります。インスタンス・ケージングというデータベース機能を使用すると、CPU負荷をさらに低いレベルで管理するようにリソース・マネージャを構成できます。

リソース・マネージャを使用すると、データベースを共有している複数のワークロードまたはアプリケーションに明示的にCPUリソースを割り当てることもできます。たとえば、通常のユーザーよりも"sys"を優先したり、レポート・ジョブおよびバッチ・ジョブよりもOLTPトランザクションを優先したりすることができます。または、統合された1つのデータベース内でアプリケーションごとにCPU割当ておよび制限を構成できます。

このホワイト・ペーパーは4部構成になっています。第1部では、リソース・マネージャの概要を説明し、一般的な使用例を紹介します。第2部では、リソース・マネージャを使用してデータベース全体の負荷を管理するときに必要な唯一の手順を示します。第3部では、データベース内の複数のワークロードをリソース・マネージャで管理するための詳しい手順を説明します。第4部では、リソース・マネージャの監視方法とチューニング方法について説明します。

このホワイト・ペーパーでは、Oracle Database 11g Release 2のリソース・マネージャについて説明します。ほとんどの概念は、Oracle Database 10gおよび11g Release 1にも該当します。例外については明示的に示します。

第1部：リソース・マネージャの紹介

この項では、リソース・マネージャの一般的なシナリオと概要について説明します。

リソース・マネージャのシナリオ

リソース・マネージャは、CPU負荷が高くなると発生するデータベースの問題解決に役立ちます。最初に発生する一連のデータベースの問題は、データベース全体の負荷が過剰になったときに現れます。

- データベースの負荷が高いと、重要なバックグラウンド・プロセスに十分なリソースが割り当てられません。たとえば、LMSのリソースが不足すると、LMSのデータベース・インスタンスはRACクラスタ・グループから排除される可能性があります。または、PMONがリソース不足になると、無効になったプロセスと関係しているロックがすぐに解放されず、アプリケーションが停止します。
- データベースの負荷が高いと、サーバーが不安定になります。CPU負荷が過剰になると、オペレーティング・システムの脆弱性やバグが多数露見します。また、過度にCPU負荷がかかると、重要なアプリケーションや監視ツールをサーバー上で実行できなくなることがあります。
- データベースの負荷が高いと、DBAは問題を診断したり修正したりすることができない場合があります。不適切なSQLプランや不適切なパラレル・ヒントなどの問題が原因でデータベースの負荷が高くなった場合は、パラレル問合せの数が過剰になります。応答時間があまりに遅すぎて、DBAがログインしてデバッグすることができないことがあります。
- データベース・インスタンスが他のデータベース・インスタンスまたはアプリケーションとサーバーを共有している場合は、データベースの負荷が高いと、各インスタンスやアプリケーションのパフォーマンスおよび安定性に影響することがあります。

これらすべての事例について、リソース・マネージャでは、標準のデフォルト・プランを有効にするだけでデータベースの負荷を管理できます。この方法については第2部で説明します。

管理が必要なワークロードがデータベース内に複数ある場合もあるでしょう。以下に例を挙げます。

- データベース上の優先順位の異なるワークロードがCPUを奪い合っている場合。たとえば、非定型の問合せが、重要なOLTPアプリケーションのパフォーマンスを阻害している、または、データウェアハウスでのデータ・ロード処理が、重要なDSS問合せを妨害しているなどです。
- データベース上の複数のアプリケーションがCPUを奪い合っている場合。たとえば、複数のアプリケーションが1つのデータベースに統合されている場合は、1つのアプリケーションのワークロードが急増して、他のアプリケーションのパフォーマンスが低下することがあります。また、アプリケーションのCPU使用率を制限して、より一貫した応答時間を確保する必要がある場合もあります。

このようなシナリオの場合は、リソース・プランを構成することでデータベース内のワークロードを管理できます。これについては第3部で説明します。

リソース・マネージャの概要

リソース・マネージャは、Oracle8iで導入されたデータベース機能です。使用できるのはEnterprise Editionのみです[1]。

リソース・マネージャを使用すると、アイドル時間の制限、リソース集中型の問合せ、並列度といったセッション動作の複数の側面を、ワークロード単位で管理できます。このホワイト・ペーパーでは、リソース・マネージャを使用したCPU管理を中心に説明します。

リソース・マネージャの仕組み

最初に、オペレーティング・システムのスケジューラの仕組みを簡単に説明します。実行可能な状態になっているプロセスは、常に数多く存在します。しかしながら、オペレーティング・システムで実行できるプロセスはCPU当たり1つのみです。他のすべてのプロセスは、オペレーティング・システムの実行キューで待機します。オペレーティング・システムのスケジューラは、1つのプロセスを短時間実行させた後、そのプロセスをスケジュールから外し、代わりに実行キューから新しいプロセスを選択して実行させます。スケジューラでは、ある種のラウンドロビン・アルゴリズムを使用して実行可能なすべてのプロセスの中からプロセスが選択されるため、すべてのプロセスの処理が進行します。

サーバーの負荷を構成しているのは、実行可能なすべてのプロセス（実行中のプロセスまたは実行キューで待機中のプロセス）です。実行可能なプロセスよりCPUの方が多いと、CPU使用率は100%未満になります。実行キューが非常に大きくなると、サーバーは過負荷になり、実行までにかなり長い時間プロセスを待機させることが必要になり、重要なプロセスはCPU不足になります。

リソース・マネージャは、データベースの負荷を特定のレベルに制御することで、CPU使用率を管理します。デフォルトでは、このレベルはCPUの数に設定されています。つまり、4CPUのサーバーの場合は、5つ以上のOracleプロセス（より具体的には、フォアグラウンドのプロセスと、CPU集中型の重要はでないバックグラウンド・プロセス）は同時に実行されなくなります。データベースの負荷が制御されるため、重要なバックグラウンド・プロセスをタイミングよく実行でき、サーバーの負荷が調整されます。

リソース・マネージャの仕組みはオペレーティング・システムのスケジューラによく似ています。リソース・マネージャを使用すると、ある時点で実行できるOracleプロセスはCPU当たり1つです。他のすべてのプロセスは、待機イベント"resmgr:cpu quantum"のもと、リソース・マネージャの内部実行キューで待機します。リソース・マネージャは、1つのOracleプロセスを一定の短時間（100ミリ秒）だけ実行させます。この一定時間が経過するか、または、このOracleプロセスが待機の状態（ロック待ちまたはI/O待ちの状態）に入ると、新しいOracleプロセスがリソース・マネージャに選択され、実行が始まります。リソース・マネージャでは、ラウンドロビン・アルゴリズムを使用して実行可能なすべてのプロセスの中からプロセスが選択されるため、すべてのプロセスの処理が進行します。

[1] Oracle Database 11gの場合は、Standard Editionでも、標準の"default_maintenance_plan"を使用してリソース・マネージャでメンテナンス・タスクを管理できます。

複数のワークロードの管理

ごく基本的な構成の場合、リソース・マネージャではすべてのセッションが同様に扱われます。ただし、コンシューマ・グループとリソース・プランを構成すれば、異なる方法でワークロードが管理されるように構成できます。

コンシューマ・グループは、1つの単位として管理されるセッションの集合です。コンシューマ・グループは、データベース内のアプリケーションごとに定義できます。または、OLTP、レポート、メンテナンスなどのワークロードの種類ごとに定義することもできます。

コンシューマ・グループ・マッピング・ルールを定義すると、セッションをコンシューマ・グループに自動的にマッピングできます。たとえば、"OLTP"サービスのセッションを"インタラクティブ"コンシューマ・グループにマッピングできます。または、ユーザー名が"sales"のセッションを"sales_app"コンシューマ・グループにマッピングできます。コンシューマ・グループ・マッピング・ルールで使用できる属性には、セッションのサービス、モジュール、アクション、Oracleユーザー名、クライアント・ユーザー名、およびプログラム名があります。

リソース・プランで指定するのは、コンシューマ・グループ間でCPUを共有する方法です。CPUの割合をコンシューマ・グループに割り当て、未使用のCPUの再分配方法を指定します。リソース・プランには、コンシューマ・グループごとに**リソース・プラン・ディレクティブ**が含まれており、ここにCPU割当てが指定されています。

よくある誤解

リソース・マネージャについてもっともよく誤解される項目をいくつか紹介します。

プロセスのバインド

リソース・マネージャでは、プロセスとCPUがバインドされません。リソース・マネージャの目的は、サーバーのCPUを管理することではなく、データベース・インスタンスのCPU使用率を制御することです。あるデータベース・インスタンスで特定のCPUセットのみ使用されるようにする必要がある場合は、論理パーティション、プロセッサ・セット、仮想マシンなどのツールを使用する必要があります。

リソース・マネージャは、重要性の低いCPU集中型のプロセスでない限り、バックグラウンド・プロセスを管理しません。重要なバックグラウンド・プロセスではCPUが適切に使用されるため、CPU消費量が過剰になったり急増したりする原因になることはありません。重要なプロセスの大半は迅速にCPUにアクセスする必要があるため、これらのプロセスがスロットリングの対象になった場合は、データベース全体のパフォーマンスに悪影響が及びます。

スレッドCPU

最近のサーバーには、ハイパースレッド・プロセッサまたはCMTプロセッサが使用されているものが多数あります。このようなサーバーの場合は、1つのCPUコアに複数のスレッド（CPU）が含まれることがあります。たとえば、Xeonプロセッサには2つのスレッドが含まれ、各スレッドがCPUとしてカウントされます。UltraSparc T2コアには8つのスレッドが含まれ、各スレッドがCPUとしてカウントされます。

デフォルトでは、リソース・マネージャはサーバー上の合計スレッド数に基づいてデータベースの負荷を管理します。この数は、オペレーティング・システムから報告されるCPU数と同じです。この仕組みがあるため、データベース・インスタンスはサーバーを十分に活用できます。データベースの負荷を下げる必要がある場合は、インスタンス・ケージングを有効化する必要があります。

Oracle RAC

RAC環境でリソース・マネージャを使用すると、各データベース・インスタンスが別個に管理されます。各データベース・インスタンスで実行されるアプリケーションを考慮した独自のリソース・プランを使用して、データベース・インスタンス1つ1つを構成できます。ただし、ほとんどの配置では、RACデータベースのすべてのインスタンスに同じリソース・プランが構成されています。

第2部：健全なデータベースのためのリソース・マネージャの構成

データベースの負荷を管理できるようにリソース・マネージャを構成するのに必要な手順はただ1つ、resource_manager_planパラメータを"DEFAULT_PLAN"に設定するだけです（この変更を永続的にするには、init.oraまたはSPFILEも更新する必要があります）。以下に例を示します。

```
SQL> alter system set resource_manager_plan = 'DEFAULT_PLAN' sid = '*';
```

標準の"DEFAULT_PLAN"を使用してリソース・マネージャを有効化すると、次のメリットが得られます。

- フォアグラウンド・プロセスから過剰に負荷がかかることが原因でPMON、LMSなどの重要なバックグラウンド・プロセスがCPU不足になることがない。
- SYSTEMおよびSYSがもっとも高い優先順位でスケジュールされる。一般ユーザーがCPUを大量に使用するアクティビティを実行したことで、SYSTEMやSYSの応答時間に影響が及ぶことがなくなり、いつでもSYSTEMまたはSYSでログインしてデータベースの問題をデバッグできるようになります。
- 自動メンテナンス・タスクがもっとも低い優先順位でスケジュールされる[2]。これらのタスクに含まれるのは、オプティマイザ統計の収集、自動セグメントおよびSQLチューニング・アドバイザの実行です。したがって、これらのタスクが他のセッションとCPUを奪い合うことがなくなります。ただし、データベースのワークロードが低い場合はこれらのメンテナンス・タスクがスケジュールされ、未使用になる残りのCPUリソースが消費されます。

インスタンス・ケージング[3]

リソース・マネージャでは、すべてのCPUが使用されるようにデータベースの負荷が管理されますが、次のように、状況によってはデータベース・インスタンスが使用するサーバーのCPUを100%未満にすることが必要な場合もあります。

- 他のアプリケーションまたはデータベース・インスタンスと、サーバーを共有している場合。このデータベース・インスタンスが使用できるサーバーのCPUを一部のみにする必要があります。
- 統合されたサーバーで、複数のデータベース・インスタンスがホストされている場合。1つのデータベース・インスタンスのCPU使用率を制限し、ユーザーに対する応答時間をより一定にさせる必要があります。または、データベース・インスタンスのCPU使用率を、顧客の支払う金額に基づいて制限する必要があります。

[2] DEFAULT_PLANを編集すれば、自動メンテナンス・タスクへの割当および優先順位を変更できます。
[3] インスタンス・ケージングはOracle Database 11g Release 2以降で使用できます。

- CPU使用率のレベルが低い方がサーバーが安定する場合。データベース・インスタンスの負荷をこの低いレベルに維持する必要があります。

このような場合は、cpu_countパラメータを、リソース・マネージャが使用するCPU数に設定することで、データベースの負荷を軽くできます。たとえば、8CPUのサーバーでリソース・マネージャを使用し、アクティブなOracleプロセスが6以下になるようにデータベースの負荷を制限する（したがって、6つ以下のCPUを使用する）必要がある場合は、リソース・マネージャを有効化し、cpu_countパラメータを6に設定します。

詳細は、インスタンス・ケージングに関する次のOracleホワイト・ペーパーを参照してください。

http://www.oracle.com/technology/products/manageability/database/pdf/owp_instance_caging.pdf.

第3部：複数のワークロードのためのリソース・マネージャの構成

この項では、1つのデータベース内の複数のワークロードを管理できるようにリソース・マネージャを構成する方法を説明します。手順は9つで、一部はオプションです。

1 リソース・マネージャを管理するための権限の取得

2 ペンディング・エリアの作成

3 コンシューマ・グループの作成

4 コンシューマ・グループへのセッションのマッピング

5 コンシューマ・グループの権限の追加

6 リソース・プランの作成

7 リソース・プラン・ディレクティブの追加

8 ペンディング・エリアの発行

9 リソース・プランの有効化

手順1：リソース・マネージャを管理するための権限の取得

リソース・マネージャの管理には、"ADMINISTER_RESOURCE_MANAGER"権限が必要です。この権限は"SYS"に自動的に付与されます。次のPL/SQLコマンドは、この権限をユーザー"scott"に付与しています。

```
SQL> exec dbms_resource_manager_privs.grant_system_privilege( -
grantee_name => 'scott', -
admin_option => true);
```

手順2：ペンディング・エリアの作成

ペンディング・エリアの作成は、コンシューマ・グループおよびリソース・プランを定義または変更するときの最初の手順です。ペンディング・エリアは、リソース・マネージャを構成するための一時作業領域です。ペンディング・エリアでの変更内容は、ペンディング・エリアを発行する（手順8）までは表示されません。

次のコマンドを実行してペンディング・エリアを作成します。

```
SQL> exec dbms_resource_manager.create_pending_area();
```

ペンディング・エリア内の変更は、次のコマンドでいつでも破棄できます。

```
SQL> exec dbms_resource_manager.clear_pending_area();
```

手順3：コンシューマ・グループの作成

Oracleデータベースには、複数のコンシューマ・グループが事前に構成されています。独自のコンシューマ・グループを作成する前に、事前構成済みのコンシューマ・グループのいずれかが自社の要件を満たすかどうか検討してください。

表1－事前構成済みのコンシューマ・グループ

コンシューマ・グループ名	説明
INTERACTIVE_GROUP	インタラクティブなOLTP操作用のコンシューマ・グループ
BATCH_GROUP	バッチ操作用のコンシューマ・グループ
DSS_GROUP	意志決定支援システム（DSS）問合せ用のコンシューマ・グループ
DSS_CRITICAL_GROUP	重要なDSS問合せ用のコンシューマ・グループ
ETL_GROUP	データ・ロード・ジョブ、またはETLジョブ用のコンシューマ・グループ
SYS_GROUP	システム管理者用のコンシューマ・グループ
LOW_GROUP	優先順位が低いセッション用のコンシューマ・グループ
OTHER_GROUPS	すべてのセッション用のデフォルト・コンシューマ・グループ

これらの事前構成済みコンシューマ・グループでは自社のワークロードを説明できない場合、または追加のコンシューマ・グループを作成する必要がある場合は、次のPL/SQLコマンドを使用してください。この例では、"SALES_APP"という名前のコンシューマ・グループを作成します。

```
SQL> exec dbms_resource_manager.create_consumer_group( -
        'SALES_APP', -'
        Sessions for the Sales Application');
```

手順4：コンシューマ・グループへのセッションのマッピング

セッションを自動的にコンシューマ・グループに配置するには、マッピング・ルールを使用します。たとえば、次のルールは、"BATCH"サービスに含まれるセッションを"BATCH_GROUP"コンシューマ・グループにマッピングします。

```
SQL> exec dbms_resource_manager.set_consumer_group_mapping( -
        attribute => dbms_resource_manager.service_name, -
        value => 'BATCH', -
        consumer_group => 'BATCH_GROUP');
```

マッピング・ルールで使用できるセッション属性を次の表にリストします。

表2－コンシューマ・グループ・マッピング・ルールの作成に使用できるセッション属性

セッション属性	説明
service_module_action	サービス名、モジュール名、およびアクション名の組合せ。書式は次のとおりです。 `service_name.module_name.action_name`
service_module	サービス名とモジュール名の組合せ。書式は次のとおりです。 `service_name.module_name`
module_name_action	モジュール名とアクション名の組合せ。書式は次のとおりです。 `module_name.action_name`
module_name	現在実行中のアプリケーションのモジュール名。
service_name	クライアントが接続を確立するときに使用したサービス名。
oracle_user	Oracleデータベースのユーザー名。
client_program	サーバーにログオンするときに使用されたクライアント・プログラム名。
client_os_user	クライアントのオペレーティング・システム・ユーザー名。
client_machine	クライアントの接続元のコンピュータ名。
client_id	クライアント識別子。

dbms_session.switch_consumer_group()を使用すると、セッションを明示的にコンシューマ・グループにマッピングできます。

switch_time、switch_io_megabytes、switch_io_reqs、またはswitch_estimateの各ディレクティブを使用すると、CPUまたはI/Oの消費量または想定実行時間に基づいて、あるコンシューマ・グループから別のコンシューマ・グループに自動的にセッションが切り替わるように構成することもできます。

手順5：コンシューマ・グループの権限の追加

あるコンシューマ・グループに切り替えるには、ユーザーまたはロールに権限が付与されている必要があります。次のPL/SQLコマンドを使用すると、ユーザー"scott"をコンシューマ・グループ"INTERACTIVE_GROUP"に切り替えることができます。

```
SQL> exec dbms_resource_manager_privs.grant_switch_consumer_group( -
        grantee_name => 'scott', -
        consumer_group => 'INTERACTIVE_GROUP', -
        grant_option => FALSE);
```

次のPL/SQLコマンドを使用すると、どのユーザーでも"BATCH_GROUP"コンシューマ・グループに切り替えることができます。

```
SQL> exec dbms_resource_manager_privs.grant_switch_consumer_group
        ( -grantee_name => 'public', -
        consumer_group => 'BATCH_GROUP', -
        grant_option => FALSE);
```

手順6：リソース・プランの作成

Oracleデータベースには、複数のリソース・プランが事前に構成されています。独自のリソース・プランを作成する前に、事前構成済みのリソース・プランのいずれかが自社の要件を満たすかどうか検討してください。

表3ー事前構成済みのリソース・プラン

リソース・プラン名	説明
DEFAULT_PLAN	汎用のプラン。ハングや想定外の重負荷をDBAがデバッグできるように、SYSを最優先します。
DEFAULT_MAINTENANCE_PLAN	メンテナンス・ウィンドウ用のプラン。一部のCPUリソースを自動メンテナンス・タスクに割り当てます。
MIXED_WORKLOAD_PLAN	複合ワークロード（インタラクティブとバッチ）を管理するためのプラン。
DSS_PLAN	DSSワークロード（DSS、ETL、およびバッチ）を管理するためのプラン。
ETL_CRITICAL_PLAN	DSS_PLANと似ていますが、DSSよりETLを優先します。

リソース・プランで管理されているコンシューマ・グループは、次のコマンドで確認できます（この例では、MIXED_WORKLOAD_PLAN）。

```
SQL> select group_or_subplan from dba_rsrc_plan_directives where plan =
        'MIXED_WORKLOAD_PLAN';
```

次の手順では、これらの事前構成済みプランに対してコンシューマ・グループを追加したり削除したりします。

新しいリソース・プランの作成から開始する場合は、次のPL/SQLコマンドを使用できます。これは、"DB_CONSOLIDATE_PLAN"という名前のリソース・プランを作成する例です。

```
SQL> exec dbms_resource_manager.create_plan( -
        'DB_CONSOLIDATE_PLAN', -
        'Plan for database consolidation');
```

手順7：リソース・プラン・ディレクティブの追加

リソース・プラン・ディレクティブには、コンシューマ・グループに割り当てるCPU量を指定します。dbms_resource_manager.create_plan_directive()をコールすれば、新しいディレクティブを作成できます。update_plan_directiveまたはdelete_plan_directive()をコールすれば、ディレクティブを更新したり削除したりすることができます。

パラメータmgmt_p1には、コンシューマ・グループに割り当てるCPUの割合を指定します。たとえば、アプリケーションが4つあり、次のようにCPUを割り当てる必要があるとします。OTHER_GROUPSというデフォルト・コンシューマ・グループのディレクティブを用意する必要がある点に注意してください。

コンシューマ・グループ	mgmt_p1
APP_1	40%
APP_2	25%
APP_3	25%
APP_4	5%
OTHER_GROUPS	5%

この例のPL/SQLは次のとおりです。

```
SQL> exec dbms_resource_manager.create_plan_directive( -
        'DB_CONSOLIDATE_PLAN',
        'APP_1', -
        'Percentage of CPU for APP_1', -
        mgmt_p1 => 40);

SQL> exec dbms_resource_manager.create_plan_directive( -
        'DB_CONSOLIDATE_PLAN', -
        'APP_2', -
        'Percentage of CPU for APP_2', -
        mgmt_p1 => 25);
```

その他

どのコンシューマ・グループにも使用されていないCPUがある場合はいつでも、CPUを必要とするコンシューマ・グループにCPUが再分配されます。上の例では、実行中のコンシューマ・グループがAPP_1のみであれば、APP_1にCPUが100%割り当てられます。したがって、mgmt_p1には、コンシューマ・グループへの割当てが保証されているCPU量を指定します。コンシューマ・グループが消費できるCPUの最大量は100%です。

CPUの割当て量に応じてアプリケーション所有者に課金している場合に、支払われる金額より多くのCPUが消費されないようにするには、CPU使用率を制限する必要があります。使用率は、max_utilization_limitディレクティブを使用して制限できます[4]。次の例は、各コンシューマ・グループの割当てを5%まで超過できるようにしています。

コンシューマ・グループ	mgmt_p1	max_utilization_limit
APP_1	40%	45%
APP_2	25%	30%
APP_3	25%	30%
APP_4	5%	10%
OTHER_GROUPS	5%	

この例のPL/SQLは次のとおりです。

```
SQL> exec dbms_resource_manager.create_plan_directive( -
        'DB_CONSOLIDATE_PLAN', -
        'APP_1', -
        'Percentage of CPU for APP_1', -
        mgmt_p1 => 40, max_utilization_limit => 45);

SQL> exec dbms_resource_manager.create_plan_directive( -
        'DB_CONSOLIDATE_PLAN', -
        'APP_2', -
        'Percentage of CPU for APP_2', -
        mgmt_p1 => 25, max_utilization_limit => 30);
```

その他

場合によっては、優先順位が上のコンシューマ・グループで割当て分が使用されていない場合にのみ、別のコンシューマ・グループにCPUリソースが割り当てられるようにすることがあります。プランのレベルを使用すれば、未使用のCPUの再割当て方法を指定できます。レベル1のCPUの未使用分または未割当て分は、レベル2のコンシューマ・グループに使用されますが、その際にはmgmt_p2ディレクティブが使用されます。レベル2のCPUの未使用分または未割当て分は、レベル3のコンシューマ・グループに使用され、その際に使用されるのはmgmt_p3ディレクティブです。以下、同様に再割当てされます。

たとえば、バッチ・コンシューマ・グループが1つあり、このコンシューマ・グループにCPUの5%を消費させ、他のアプリケーションが消費していないCPUがあればそれも消費させるとします。これを実現するプランは次のようになります。

[4] "max_utilization_limit"はOracle Database 11g Release 2以降で使用できます。

コンシューマ・グループ	mgmt_p1	mgmt_p2
APP_1	35%	
APP_2	25%	
APP_3	25%	
APP_4	5%	
BATCH_GROUP	5%	100%
OTHER_GROUPS	5%	

この例のPL/SQLは次のとおりです。

```
SQL> exec dbms_resource_manager.create_plan_directive( -
        'DB_CONSOLIDATE_PLAN', -
        'BATCH_GROUP', -
        'CPU for batch jobs', -
        mgmt_p1 => 5, mgmt_p2 => 100);
```

手順8：ペンディング・エリアの発行

リソース・プランを構成したら、次のPL/SQLを使用して変更内容をデータベースに保存できます。現在アクティブになっているリソース・プランを編集している場合は、変更がすぐに有効化されます。

```
SQL> exec dbms_resource_manager.submit_pending_area();
```

プランにエラーがあれば、このコマンドは失敗し、問題点を説明するエラーが表示されます。

手順9：リソース・プランの有効化

これで、リソース・プランは定義されたので、定義したリソース・プランの名前を使用してresource_manager_planパラメータを設定すれば、プランを有効化できます。

```
SQL> alter system set resource_manager_plan = 'DB_CONSOLIDATION_PLAN' sid='*';
```

ジョブ・スケジューラ・ウィンドウに対応する期間に対してリソース・プランを有効化することもできます。この例では、DB_CONSOLIDATION_PLANは、MONDAY_WINDOWが開くと有効化され、MONDAY_WINDOWが閉じると無効化されます。

```
SQL> exec dbms_scheduler.set_attribute('MONDAY_WINDOW', 'RESOURCE_PLAN',-
        'DB_CONSOLIDATION_PLAN');
```

第4部：リソース・マネージャの監視とチューニング

この項では、リソース・マネージャの監視方法、デバッグ方法、およびチューニング方法を説明します。この項はQ&A形式になっています。質問は、リソース・マネージャの有効化、コンシューマ・グループへのセッションのマッピング、CPU管理の監視のカテゴリに分類されています。

リソース・マネージャの有効化

リソース・マネージャは、"resource_manager_plan"パラメータを設定するか、dbms_scheduler PL/SQLパッケージを使用してリソース・プランとスケジューラ・ウィンドウを関連付ける方法で有効化できます。この項では、リソース・マネージャが有効化されているかどうかを判定する方法を説明します。

現在、リソース・マネージャは有効化されていますか。

現在、リソース・マネージャが有効化されているかどうかを判定するには、次のコマンドを実行します。

```
SQL> select name, cpu_managed from v$rsrc_plan where is_top_plan = 'TRUE';
```

この問合せを実行すると、現在のリソース・プランの名前が戻されます。行が戻されなければ、現在、リソース・マネージャは無効化されています。リソース・マネージャでCPUが管理されていれば、列"cpu_managed"に表示されます[5]。

リソース・マネージャが有効化されたのはいつですか。

リソース・マネージャの使用履歴を確認するには、次のコマンドを実行します。

```
SQL> select name,
        to_char(start_time, 'MON DD HH24:MI') start_time,
        to_char(end_time, 'MON DD HH24:MI') end_time,
        window_name
        from v$rsrc_plan_history order by start_time;
```

この問合せを実行すると、プランの名前、そのプランが有効化された時刻と無効化された時刻、およびそのプランを有効化するのに使用されたスケジューラ・ウィンドウ（ある場合）が戻されます。各行は、リソース・マネージャが有効化または無効化された操作の1回分に相当します。最大16回分の履歴が保持されます。

[5] "cpu_managed"は、Oracle Database 11g Release 2以降で使用できます。どのリリースでも、現在のリソース・プランにCPUディレクティブが含まれているかどうかを確認すれば、リソース・マネージャがCPUを管理しているかどうかを判定できます。

リソース・マネージャはどのようにして有効化されましたか。

前述の問合せを実行すると、リソース・プランを自動的に有効化したスケジューラ・ウィンドウがwindow_nameに示されます。window_nameがNULLの場合は、"resource_manager_plan"パラメータを使用してリソース・プランが有効化されています。

リソース・プランに関連付けられているスケジューラ・ウィンドウを確認するには、次のコマンドを実行します。

```
SQL> select window_name, resource_plan, active
        from dba_scheduler_windows
        where resource_plan is not null and enabled = 'TRUE';
```

あるスケジューラ・ウィンドウの"active"がTRUEの場合は、現在、スケジューラ・ウィンドウが開いており、それのリソース・プランが有効化されています。

コンシューマ・グループへのセッションのマッピング

dbms_resource_manager.set_consumer_group_mapping()プロシージャを使用すれば、セッションを自動的にコンシューマ・グループにマッピングできます。dbms_session.switch_consumer_group()プロシージャを使用すれば、セッションがマッピングされているコンシューマ・グループを手動で切り替えることもできます。

この項では、セッションが適切なコンシューマ・グループにマッピングされているかどうかを判定する方法を説明します。

セッションはどのコンシューマ・グループにマッピングされていますか。

次の問合せを実行すると、すべてのセッションの現在のコンシューマ・グループが戻されます。

```
SQL> select sid, resource_consumer_group from v$session;
```

想定したコンシューマ・グループにセッションがマッピングされていないのはなぜですか。

いくつかの理由により、想定したコンシューマ・グループにセッションがマッピングされないことがあります。

権限がない

セッションがマッピングされているコンシューマ・グループを切り替えるには、そのコンシューマ・グループに切り替えるための権限が、セッションのユーザーまたはロールに付与されている必要があります。あるコンシューマ・グループにセッションがマッピングされていても、そのコンシューマ・グループに切り替えるための権限がセッションに付与されていなければ、マッピング・ルールは無視されます。

次の問合せを実行すると、すべてのコンシューマ・グループの権限が表示されます。特定のユーザー（権限受領者）の権限を確認するには、where句を指定します。

```
SQL> select grantee, granted_group from DBA_RSRC_CONSUMER_GROUP_PRIVS
        order by granted_group;
```

次のSQLコマンドを実行すると、あるコンシューマ・グループに切り替えるための権限がすべてのセッションに付与されます。この例では、コンシューマ・グループ'BATCH_GROUP'を使用します。

```
SQL> exec dbms_resource_manager_privs.grant_switch_consumer_group( -
        'public', 'BATCH_GROUP', FALSE);
```

コンシューマ・グループがアクティブでない

セッションがマッピングされているコンシューマ・グループ、または手動で切り替えた先のコンシューマ・グループが現在のリソース・プランに含まれていない場合、そのセッションはデフォルト・コンシューマ・グループ"OTHER_GROUPS"に切り替えられます。

マッピング・ルールを使用してコンシューマ・グループにセッションが割り当てられている場合は、次の問合せを使用すると、マッピング・ルールで選択されたコンシューマ・グループ、使用されたマッピング属性、セッションが元々マッピングされていたコンシューマ・グループを確認できます。マッピングされたコンシューマ・グループが元のコンシューマ・グループと異なる場合は、マッピングされたコンシューマ・グループはリソース・プランに含まれていなかったことになります。

```
SQL> select r.sid,
        r.mapped_consumer_group,
        r.mapping_attribute,
        c.consumer_group original_consumer_group
        from v$rsrc_session_info r, dba_rsrc_consumer_groups c
        where r.orig_consumer_group_id = c.consumer_group_id;
```

コンシューマ・グループが切り替わった

セッションの現在のコンシューマ・グループが元のコンシューマ・グループと一致しない場合は、セッションのコンシューマ・グループが切り替わっています。次の問合せを実行すると、自動的に切り替えられたセッションを検出できます。

```
SQL> select r.sid,
        c1.consumer_group original_consumer_group,
        c2.consumer_group current_consumer_group
        from v$rsrc_session_info r, dba_rsrc_consumer_groups c1,
        dba_rsrc_consumer_groups c2
        where r.orig_consumer_group_id = c1.consumer_group_id
        and r.current_consumer_group_id = c2.consumer_group_id
        and r.orig_consumer_group_id != r.current_consumer_group_id;
```

さまざまな理由でセッションのコンシューマ・グループが切り替わることがあります。

1. 元のコンシューマ・グループが現在のリソース・プランに含まれていない。この場合は、現在のコンシューマ・グループは"OTHER_GROUPS"になります。
2. 自動切替えディレクティブである"switch_time"、"switch_io_megabytes"、または"switch_io_reqs"がリソース・プランに含まれる場合は、セッションが自動的に切り替えられることがあります。

次の問合せを実行すると、現在のリソース・プランに自動切替えディレクティブが含まれるかどうかが表示されます。

```
SQL> select plan, group_or_subplan, switch_time, switch_io_megabytes, switch_io_reqs
        from dba_rsrc_plan_directives
        where plan in (select name from v$rsrc_plan)
         and (switch_time is not null
              or switch_io_megabytes is not null
              or switch_io_reqs is not null);
```

コンシューマ・グループを手動で切り替えた

dbms_session.switch_consumer_group()、dbms_resource_manager.switch_consumer_group_for_user()、またはdbms_resource_manager.switch_consumer_group_for_sess()の各プロシージャを使用してセッションのコンシューマ・グループを手動で切り替えた場合、コンシューマ・グループはすぐに切り替わりません。セッションで次のコールが実行されたときに、コンシューマ・グループが切り替わります。

CPU管理の監視

この項では、リソース・マネージャによるCPUの管理状況を監視する方法を説明します。前半は、データベース・インスタンス全体でのリソース・マネージャの効果を監視する方法を説明します。後半は、コンシューマ・グループごとにリソース・マネージャの効果を監視する方法を説明します。

データベース・インスタンスはリソース・マネージャによってどの程度スロットリングされていますか。

Oracleには、CPUリソースの管理状況を監視するためのさまざなビューが用意されています。

待機イベント別のスロットリング量

リソース・マネージャによるスロットリングは、待機イベント"resmgr:cpu quantum"で監視できます。これらの待機は、AWRレポートを生成して"Foreground Wait Events"を調査すれば確認できます。

リソース・マネージャは、"scheduler"という、スロットリング待機とリソース・マネージャ関連の同時実行性待機の両方を含む待機クラスを使用して監視することもできます。"scheduler"待機のほとんどは、スロットリングのための待機です。

"scheduler"待機クラスの待機は、次の問合せを使用して監視できます。"dbtime_in_wait"には、リソース・マネージャの待機で費やされたデータベース時間の割合が示されます。"time_waited"には、実際の待機時間がマイクロ秒単位で表示されます。

```
SQL> select to_char(h.begin_time, 'HH:MI') time,
        h.average_waiter_count, h.dbtime_in_wait, h.time_waited
        from v$waitclassmetric_history h, v$system_wait_class c
        where h.wait_class_id = c.wait_class_id and c.wait_class = ' SCHEDULER'
        order by h.begin_time;
```

データベース・インスタンスのワークロードがサーバーのCPU容量をはるかに超えている場合は、上位待機イベントの1つとして"resmgr:cpu quantum"が表示されます。Oracleプロセスはリソース・マネージャで厳しくスロットリングされますが、データベース・インスタンスの全体的なスループットは、リソース・マネージャが無効化されている場合と同じになる点に注意してください。リソース・マネージャを使用しない場合は、"resmgr:cpu quantum"で費やされる時間は、代わりにオペレーティング・システムの実行キューでの待機として費やされます。AWRレポートでは、実行キューの上位待機は、サーバーのロード数によってのみ示されます。

1分当たりのスロットリング量

スロットリング量も、リソース・マネージャのビューで監視できます[6]。次の問合せを実行すると、次の項目を1分単位で比較できます。

- total：サーバー上の使用可能なCPUタイムの量
- db_total：このデータベース・インスタンスが使用できるCPUタイムの量（インスタンス・ケージングが有効化されている場合は、この値の方が少なくなります）
- consumed：消費されたCPUタイムの量
- throttled：プロセスがスロットリングされた時間の量

すべて、時間は秒単位で表示されます。

```
SQL> select to_char(begin_time, 'HH:MI') time,
        60 * (select value from v$osstat where stat_name = 'NUM_CPUS') total,
        60 * (select value from v$parameter where name = 'cpu_count') db_total,
        sum(cpu_consumed_time) / 1000 consumed,
        sum(cpu_wait_time) / 1000 throttled
        from gv$rsrcmgrmetric_history
        group by begin_time order by begin_time;
```

消費されたCPUの方が合計CPUより少ない場合、このデータベース・インスタンスではサーバーが過度に活用されていません。データベースの合計CPUより"throttled"の方が多い場合、データベースはCPUの制約を受けています。

スロットリングされたセッション数の1分当たり平均

スロットリング量は、スロットリングされた平均セッション数として数値化することもできます。次の問合せを実行すると、下記の項目を1分単位で比較できます。

[6] v$rsrcmgrmetricは、Oracle Database 11g Release 1以降で使用できます。

- num_cpus：CPUの数
- num_db_cpus：このデータベース・インスタンスで使用できるCPUの数（インスタンス・ケージングが有効化されている場合は、この数の方が小さくなります）
- avg_running：実行中のOracleセッションの平均数
- avg_throttled：スロットリングされたOracleセッションの平均数

```
SQL> select to_char(begin_time, 'HH:MI') time,
       (select value from v$osstat where stat_name = 'NUM_CPUS') num_cpus,
       (select value from v$parameter where name = 'cpu_count') num_db_cpus,
       sum(cpu_consumed_time) / 60000 avg_running,
       sum(cpu_wait_time) / 60000 avg_throttled
       from v$rsrcmgrmetric_history
       group by begin_time order by begin_time;
```

スロットリングされたセッションの平均数がCPUの数と等しい場合は、実行を待機しているセッションがCPUごとに平均して1つあったことになります。これは、過負荷のデータベース・インスタンスとサーバーの例になるでしょう。

リソース・マネージャで厳しくスロットリングしている場合、次にすることは何ですか。

リソース・マネージャで大幅なスロットリングをしている場合は、データベース・インスタンスがCPUの制約を受けます。スロットリングされているワークロードのパフォーマンスが不十分な場合は、リソース・プランをチューニングし、そのコンシューマ・グループに割り当てるCPUを増やす必要があります。このチューニング・プロセスについては、後半で説明します。すでにそのコンシューマ・グループにすべて、またはほとんどのCPUが割り当てられている場合は、CPUを追加するしか方法はありません（もちろん、アプリケーションとデータベースのチューニングが済んでいると仮定した場合です）。

コンシューマ・グループはリソース・マネージャによってどの程度スロットリングされますか。

いったんリソース・プランを有効化したら、コンシューマ・グループのパフォーマンスを監視し、必要に応じてCPU割当てを調整する必要があります。

次の問合せを実行すると、各コンシューマ・グループの次の項目を1分単位で比較できます。

- total：サーバー上の使用可能なCPUタイムの量
- db_total：このデータベース・インスタンスが使用できるCPUタイムの量（インスタンス・ケージングが有効化されている場合は、この値の方が少なくなります）
- consumed：このコンシューマ・グループが消費したCPUタイムの量
- cpu_utilization：このデータベースで使用できるCPUのうち、このコンシューマ・グループが消費した割合
- throttled：このコンシューマ・グループがスロットリングされた時間の量

すべて、時間は秒単位で表示されます。

```
SQL> select to_char(begin_time, 'HH:MI') time,
        consumer_group_name,
        60 * (select value from v$osstat where stat_name = 'NUM_CPUS') total,
        60 * (select value from v$parameter where name = 'cpu_count') db_total,
        cpu_consumed_time / 1000 consumed,
        cpu_consumed_time / (select value from v$parameter where name =
          'cpu_count') / 600 cpu_utilization,
        cpu_wait_time / 1000 throttled from v$rsrcmgrmetric_history
        order by begin_time;
```

なぜコンシューマ・グループのCPU使用率はリソース・プランと一致しないのですか。

上の問合せを実行すると、コンシューマ・グループごとの実際のCPU使用率が表示されます。リソース・マネージャを監視する場合、通常は、この値とリソース・プランの割当て（mgmt_p1、mgmt_p2、mgmt_p3など）を比較します。

すべてのコンシューマ・グループに、リソース・プランの割当て量を完全に使い切るだけの負荷がかかっていれば、CPU使用率はリソース・プランの割当て量と一致するはずです。ただし、CPU使用率は、さまざまな理由でリソース・プランの割当て量と一致しないことがあります。

- あるコンシューマ・グループの使用率がリソース・プランの割当てよりも高くなることがあります。これは、他のコンシューマ・グループがそれぞれの割当てを使用できなかったため、その割当て分が、このコンシューマ・グループに再割当てされたからです。
- 割当て分を完全に消費しきれなかったために、コンシューマ・グループの使用率がリソース・プランの割当てより低くなることがあります。この場合、スロットリングされた時間はゼロ（またはほぼゼロ）になっているはずです。
- コンシューマ・グループに、リソース・プランの割当てよりも低いmax_utilization_limitが設定されている場合。

結論

Oracle Database Resource Managerを使用してデータベースのワークロードを管理すれば、適切な負荷でサーバーを実行し続けることができ、重要なバックグラウンド・プロセスがリソース不足になることもなく、DBAは常にログインしてデータベースの問題をデバッグできます。

複数のコンシューマ・グループにCPUを割り当てるリソース・プランを有効化することで、各コンシューマ・グループのCPU使用率も管理できます。

CPUの管理はリソース・マネージャの機能の1つにすぎません。リソース集中型の問合せ、パラレル問合せ、アイドル・セッション、およびUNDO領域の管理にリソース・マネージャを役立てる方法を理解するには、『Oracle Database管理者ガイド』を参照してください。I/Oの管理にリソース・マネージャを役立てる方法は、『Oracle Exadata Storage Server Software User's Guide』を参照してください。



Oracle Database Resource Managerの使用
2010年4月
著者：Sue K. Lee
Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.
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
www.oracle.com



Copyright © 2010, 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は米国Oracle Corporation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。

0109