Oracle® Database
パフォーマンス・チューニング・ガイド
19c

F16140-05(原本部品番号:E96347-06)

2022年9月

タイトルおよび著作権情報

Oracle Databaseデータベース・パフォーマンス・チューニング・ガイド 19c

F16140-05

原本協力著者: Glenn Maxey

原本著者: Rajesh Bhatiya, Immanuel Chan, Lance Ashdown

原本協力者: Hermann Baer、Deba Chatterjee、Maria Colgan、Mikael Fries、Prabhaker Gongloor、Kevin Jernigan、Sue K.Lee、William Lee、David McDermid、Uri Shaft、Oscar Suro、Trung Tran、Sriram Vrinda、Yujun Wang

目次

はじめに

この章には、次の項目が含まれます。

対象読者

このマニュアルは、Oracle Databaseの運用、メンテナンスおよびパフォーマンスを担当するデータベース管理者(DBA)を対象としています。このマニュアルでは、Oracle Databaseのパフォーマンス・ツールを使用して、データベース・パフォーマンスを最適化する方法を説明しています。また、データベースの初期の作成でのパフォーマンスのベスト・プラクティスを説明するとともに、パフォーマンス関連の参照情報を示します。

関連項目:

- SQLパフォーマンスを最適化およびチューニングする方法の詳細は、Oracle Database SQLチューニング・ガイドを参照してください
- Oracle Enterprise Manager Cloud Control (Cloud Control)を使用してデータベースのパフォーマンスをチューニングする方法を学習するには、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

ドキュメントのアクセシビリティについて

Oracleのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWebサイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートを購入したオラクル社のお客様は、My Oracle Supportを介して電子的なサポートにアクセスできます。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info)か、聴覚に障害のあるお客様は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

関連ドキュメント

このマニュアルを読む前に、次のマニュアルの内容を理解しておく必要があります。

- Oracle Database概要
- Oracle Database管理者ガイド

- Oracle Multitenant管理者ガイド
- Oracle Database 2日でデータベース管理者
- Oracle Database 2日でパフォーマンス・チューニング・ガイド

データ・ウェアハウス環境のチューニング方法を学習するには、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください。

表記規則

このマニュアルでは次の表記規則を使用します。

規則	意味
太字	太字は、操作に関連する Graphical User Interface 要素、または本文中で定義されている用語および用語集に記載されている用語を示します。
イタリック体	イタリックは、ユーザーが特定の値を指定するプレースホルダ変数を示します。
固定幅フォント	固定幅フォントは、段落内のコマンド、URL、サンプル内のコード、画面に表示されるテキスト、または入力するテキストを示します。

このリリースの『Oracle Databaseパフォーマンス・チューニング・ガイド』の変更内容

「はじめに」の内容は次のとおりです。

- Oracle Databaseリリース19c, バージョン19.1の変更内容
- Oracle Databaseリリース18c, バージョン18.1の変更内容
- Oracle Database 12cリリース2 (12.2)の変更内容
- Oracle Database 12cリリース1 (12.1.0.2)での変更点
- Oracle Database 12cリリース1 (12.1.0.1)での変更点

Oracle Databaseリリース19c, バージョン19.1の変更内容

Oracle Databaseリリース19c, バージョン19.1のOracle Databaseパフォーマンス・チューニング・ガイドの変更内容は次のとおりです。

新機能

このリリースの新機能は次のとおりです。

- Memoptimizeされた行ストア - 高速収集

  Memoptimizeされた行ストアの高速収集機能により、モノのインターネット(IoT)アプリケーションなどのアプリケーションからの高頻度の単一行データの挿入処理が最適化されます。

  高速収集の使用を参照してください

- プラガブル・データベース(PDB)での自動データベース診断モニター(ADDM)のサポート

  ADDMを使用してPDBのAWRデータを分析し、パフォーマンスに関連する問題を特定して解決できるようになりました。

  マルチテナント環境でのADDMの使用を参照してください

サポート対象外の機能

次の機能は、このリリースではサポートされなくなりました。

- Oracle Streams

  Oracle Database 19c以降では、Oracle Streams機能はサポートされません。Oracle GoldenGateを使用して、Oracle Streamsのすべてのレプリケーション機能を置き換えます。

Oracle Databaseリリース18c, バージョン18.1の変更内容

Oracle Databaseリリース18c, バージョン18.1のOracle Databaseパフォーマンス・チューニング・ガイドの変更内容は次のとおりです。

新機能

このリリースの新機能は次のとおりです。

- Memoptimizeされた行ストア

  Memoptimizeされた行ストアにより、MEMOPTIMIZE FOR READ句で指定した表に対する高速読取りが有効になります。この機能は、モノのインターネット(IoT)のアプリケーションなど、主に主キーの値に基づいて非常に高頻度に表に問い合せるアプリケーションに特に有効です。

  Memoptimizeされた行ストアによる高パフォーマンス・データ・ストリーミングの有効化を参照してください

Oracle Database 12cリリース2 (12.2)での変更点

Oracle Database 12c リリース2 (12.2)の『Oracle Databaseパフォーマンス・チューニング・ガイド』の変更内容は次のとおりです。

新機能

このリリースの新機能は次のとおりです。

- プロセスごとのPGA制限

  PGAメモリーを過剰に消費するランナウェイ問合せによって、Oracle Databaseでパフォーマンスに関する重大な問題が発生する可能性があります。マルチテナント・コンテナ・データベース(CDB)では、このタイプの問合せによって、すべてのプラガブル・データベース(PDB)のパフォーマンスが影響を受けます。この問題を回避するために、特定のコンシューマ・グループ内の各セッションが使用できるPGAメモリー量の絶対制限を指定できるようになりました。

  リソース・マネージャを使用したプログラム・グローバル領域のサイズ設定を参照してください

- 事前起動されるサーバー・プロセス

  Oracle Databaseでは、専用ブローカ接続モードが有効化されたとき、またはスレッド化実行モードが有効化されたときに、サーバー・プロセスのプールが事前起動されるようになりました。この機能によって、これらのデータベース操作モードでのクライアント接続のパフォーマンスが向上します。

  事前起動プロセスによるクライアント接続のパフォーマンスの向上を参照してください

- 自動ワークロード・リポジトリ(AWR)によるマルチテナント環境のサポート

  マルチテナント環境内のPDBの自動ワークロード・リポジトリ(AWR)データを取得できるようになりました。この機能によって、マルチテナント環境内のPDBのパフォーマンス・チューニングが可能になります。

  マルチテナント環境内の自動ワークロード・リポジトリの管理を参照してください

- 自動ワークロード・リポジトリ(AWR)によるOracle Active Data Guardスタンバイ・データベースのサポート

  Oracle Active Data Guardスタンバイ・データベースの自動ワークロード・リポジトリ(AWR)データを取得できるようになりました。この機能によって、Oracle Active Data Guardスタンバイ・データベースのパフォーマンス・チューニングが可能になります。

  Active Data Guardスタンバイ・データベースでの自動ワークロード・リポジトリの管理を参照してください

- Direct NFS Clientのパフォーマンス向上のための機能 - Parallel NFSおよびDirect NFS Dispatcher

Parallel NFSはNFSバージョン4.1で導入された機能であり、Oracle Direct NFS Clientではオプションで提供される機能です。この機能を使用すると、スケーラビリティの高い分散NASストレージを構築でき、I/Oパフォーマンスが向上します。

Direct NFSのディスパッチャ機能によって、データベース・インスタンスからNFSサーバーに作成される複数のTCP接続が統合されます。大規模なデータベース・デプロイメントでは、Direct NFSディスパッチャを使用すると、スケーラビリティおよびネットワーク・パフォーマンスが向上します。

I/Oパフォーマンス向上のためのDirect NFS Clientの使用を参照してください

- サービス指向のバッファ・キャッシュ・アクセスの最適化

  クラスタ管理のサービスによって、Oracle Real Application Clusters (Oracle RAC)の様々なデータベース・インスタンス全体にワークロードが割り当てられます。また、これらのサービスはOracle RACのデータベース・インスタンスのデータにアクセスし、そのデータをリクエストしているアプリケーションに提供します。サービス指向のバッファ・キャッシュ・アクセスの最適化機能によって、これらのサービスが頻繁にアクセスするサービス固有のデータをOracle RACでキャッシュできるようになったため、サービスのデータ・アクセス時間が改善されます。また、このデータ依存キャッシングにより、複数のOracle RACデータベース・インスタンス上に存在するデータへのアクセス時の応答時間がより一貫性のあるものになります。この機能はOracle RACで永続的に有効化されます。

Oracle Database 12cリリース1 (12.1.0.2)の変更内容

Oracle Database 12cリリース1 (12.1.0.2)の『Oracle Databaseパフォーマンス・チューニング・ガイド』の変更内容は次のとおりです。

新機能

このリリースの新機能は次のとおりです。

- インメモリー列ストア

  インメモリー列ストア(IM列ストア)は任意のSGA領域です。ここには、迅速にスキャンできるように最適化された列形式で、表のコピー、パーティションおよびその他のデータベース・オブジェクトが格納されています。IM列ストアは、分析、データ・ウェアハウスおよびオンライン・トランザクション処理(OLTP)のアプリケーションのデータベース・パフォーマンスを促進します。

  インメモリー列ストアによる問合せパフォーマンスの改善を参照してください。

- インメモリー列ストアの管理性のサポート

  SQL監視レポート、ASHレポートおよびAWRレポートが、様々なインメモリー操作の統計を表示するようになりました。

- 強制全データベース・キャッシュ・モード

  強制全データベース・キャッシュ・モードでは全データベースをメモリー内にキャッシュできるため、全表スキャンの実行時やLOBのアクセス時にパフォーマンスが大幅に向上する可能性があります。

  「データベース・キャッシュ・モードの構成」を参照してください。

Oracle Database 12cリリース1 (12.1.0.1)での変更点

Oracle Database 12cリリース1 (12.1.0.1)の『Oracle Databaseパフォーマンス・チューニング・ガイド』の変更内容は次

のとおりです。

新機能

このリリースの新機能は次のとおりです。

- リアルタイムADDM

  リアルタイムADDMを使用すると、応答しないデータベースやハングしているデータベースの問題を、データベースを再起動せずに分析および解決できます。

  「リアルタイムADDM分析」を参照してください。

- プログラム・グローバル領域(PGA)のサイズ制限

  PGA_AGGREGATE_LIMIT初期化パラメータを使用すると、PGAメモリー使用量に強い制限を指定できます。Oracle Databaseは、PGAメモリーを最も消費しているセッションやプロセスを終了することで、PGAサイズがこの制限を越えないようにします。

  「プログラム・グローバル領域のサイズ制限」を参照してください。

その他の変更

このリリースでの追加変更は次のとおりです。

- 新しいマニュアル

  『Oracle Databaseパフォーマンス・チューニング・ガイド』は、このリリースで大幅な改訂が行われています。その結果、マニュアルの構造が大きく変わりました。現在、このマニュアルに含まれているのは、データベース自体に関係するチューニング・トピックのみです。SQL関連のチューニング・トピックはすべて、Oracle Database SQLチューニング・ガイドに移動しました。

第I部 データベース・パフォーマンスの基本

この部は、次の章で構成されています。

- パフォーマンス・チューニングの概要
- パフォーマンスを考慮した設計と開発
- パフォーマンス改善方法
- パフォーマンスを考慮したデータベースの構成

1 パフォーマンス・チューニングの概要

この章では、パフォーマンス・チューニングの概要を説明します。この章には、次の項があります。

- パフォーマンス・チューニングの概要
- パフォーマンス・チューニング機能およびツールの概要

パフォーマンス・チューニングの概要

このガイドは、Oracle Databaseのパフォーマンス・チューニングに関する情報を提供します。この項には次の項目があります。

- パフォーマンス計画
- インスタンスのチューニング
- SQLチューニング

関連項目:

Oracle Enterprise Manager Cloud Control (Cloud Control)を使用してデータベースのパフォーマンスをチューニングする方法を学習するには、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

パフォーマンス計画

このドキュメントの他の部分に進む前に、「データベース・パフォーマンスの基本」を参照してください。オラクル社では、長年にわたる設計およびパフォーマンス経験に基づき、パフォーマンスに関する方法論を設計しました。このトピックでは、システム・パフォーマンスを大幅に改善できるアクティビティについて説明します。内容は次のとおりです。

- 投資の選択肢について
- スケーラビリティについて
- システム・アーキテクチャ
- アプリケーション設計の原則
- ワークロードのテスト、モデル化および実装
- 新規アプリケーションのデプロイ

インスタンスのチューニング

「データベース・パフォーマンスの診断およびチューニング」では、Oracleデータベース・インスタンスのチューニングおよび最適化に関連のある要素について説明します。

インスタンスのチューニングを検討する場合、データベースの初期の設計で、パフォーマンスの問題の原因となるボトルネックを回避するよう注意します。さらに、次の点も考慮する必要があります。

- データベース構造体へのメモリーの割当て

- データベースの様々な部分のI/O要件の判断
- データベースのパフォーマンスを最適化するためのオペレーティング・システムのチューニング

データベース・インスタンスをインストールおよび構成した後、パフォーマンス関連の問題をチェックするために動作しているデータベースを監視する必要があります。

パフォーマンスの原理

パフォーマンス・チューニングでは、システムの初期構成に対して異なる(ただし関連性がある)方法を必要とします。システムの構成では、初期システムの構成が機能的なものになるように整理された手順に従ってリソースを割り当てます。

チューニングを開始するには、最も影響のあるボトルネックを識別し、適切な変更を行ってそのボトルネックの影響を低減するかまたは排除します。チューニングは通常、システムが本番開始前か、または稼働状態になった後で事後的に行われます。

ベースライン

チューニングでは、実証されたパフォーマンス・ベースラインを使用して、パフォーマンスの問題が生じたときに比較するのが最も効果的な方法です。データベース管理者(DBA)の多くは、自分のシステムを熟知し、ピークの使用期間を簡単に識別できます。たとえば、ピーク期間は10.00amから12.00pmである場合、また1.30pmから3.00pmである場合もあります。これには、深夜の12.00amから6.00amまでのバッチ・ウィンドウが含まれることがあります。

サイトでこのようなピーク時間帯を識別し、このような高負荷の時間帯のパフォーマンス・データを収集するモニタリング・ツールをインストールすることが重要です。アプリケーションがQAサイクル中の初期のトライアル段階にある時点から、データ収集を構成することが最適です。それ以外の場合は、システムが最初に稼働したときに、このデータ収集を構成する必要があります。

収集されたベースライン・データには、次の内容が含まれていることが理想的です。

- アプリケーション統計(トランザクション・ボリューム、レスポンス時間)
- データベース統計
- オペレーティング・システム統計
- ディスクI/O統計
- ネットワーク統計

自動ワークロード・リポジトリでは、ベースラインは将来比較するために保持されるスナップショットの範囲により識別されます。「自動ワークロード・リポジトリ」を参照してください。

症状および問題点

パフォーマンス・チューニングの一般的な誤りは、ある問題の症状を現実の問題自体であると思い違いをすることです。多くのパフォーマンス統計はこの症状を示すこと、およびこの症状を識別することが修正を実施するために十分なデータではないことを認識することが重要です。次に例を示します。

- 低速な物理I/O

  一般に、この原因はディスクの構成が適切ではないことにあります。しかし、チューニングが適切ではないSQLから発行された、これらのディスク上の大量の不要な物理I/Oが原因になっている可能性もあります。

- ラッチの競合

インスタンスを再構成して、ラッチの競合をチューニングできることはほとんどありません。むしろ、ラッチの競合は通常、アプリケーションの変更により解決されます。

- 過剰なCPU使用率

  過剰なCPU使用率は通常、システム上にアイドル状態のCPUがほとんどないことを意味します。この原因として、システムのサイズ設定が不適切であること、SQL文がチューニングされていないこと、またはアプリケーション・プログラムが不十分である可能性があります。

チューニングの時期

チューニングには次の2種類があります。

- プロアクティブな監視
- ボトルネックの解消

プロアクティブな監視

プロアクティブな監視は通常、定期的にスケジュールされた間隔で行われます。この場合、システム動作とリソースの使用量が変化したかどうかを識別するために複数のパフォーマンス統計が調べられます。プロアクティブな監視は、プロアクティブなチューニングとも考えられます。

通常は、進行中の重大な問題が監視により明らかにならないかぎり、監視によりシステムの構成が変化することはありません。状況によっては、経験豊富なパフォーマンス・エンジニアが統計のみで潜在的な問題点を識別できますが、通常はパフォーマンスの低下を伴います。

明らかなパフォーマンスの低下がないときに、事前のアクションとしてシステムを試行したり微調整することは危険なアクティビティであり、不必要にパフォーマンスを低下させる可能性があります。システムを微調整することは事後チューニングと考えられ、事後チューニングのステップに従う必要があります。

監視は通常、より大規模な容量計画の調査の一環です。容量計画ではリソース使用状況を調べて、さらにアプリケーションが使用されている方法の変化、およびアプリケーションがデータベース・リソースとホスト・リソースを使用している方法の変化を調べます。

ボトルネックの解消

チューニングは通常、パフォーマンスの問題の修正を意味します。ただし、チューニングは、分析、設計、コーディング、生産およびメンテナンスの各段階を通じて、アプリケーションのライフサイクルの一部である必要があります。多くの場合、チューニング段階は、データベースが稼働段階に入るまで残されます。この時点で、チューニングは事後対応処理になり、最も重要なボトルネックを識別し、修正します。

チューニングの目的は通常、リソース使用量を減らしたり、操作を完了する経過時間を減らすことにあります。いずれの場合も、目標は特定のリソースの有効利用を向上することにあります。一般に、パフォーマンスの問題は特定のリソースの過剰使用によって発生します。リソースの過剰使用は、システムのボトルネックです。ボトルネックと潜在的な修正を識別するには、様々な複数の段階があります。これらについては、これ以降の項で説明します。

競合の様々な形態は症状であり、次のいずれかを変更することで修正できることに注意してください。

- アプリケーションまたはアプリケーションの使用方法の変更
- Oracleの変更
- ホスト・ハードウェア構成の変更

多くの場合、ボトルネックを解決する最も効果的な方法は、アプリケーションを変更することです。

SQLチューニング

SQLでは、問合せを発行し、データが戻されるため、多くのアプリケーション・プログラマはSQLをメッセージ言語として認識しています。しかし、クライアント・ツールでは非効率的なSQL文が生成される場合がよくあります。したがって、データベースSQL処理エンジンについて理解することは、最適なSQLを作成するために必要です。このことは特に、大量トランザクション処理システムについて言えます。

通常、オンライン・トランザクション処理(OLTP)アプリケーションにより発行されたSQL文は、比較的少数の行で一度に処理されます。索引が正確に必要な行を示す場合、正確な計画を作成して、可能な最短のパスを使用して効率的に行にアクセスできます。意思決定支援システム(DSS)環境では、表の行のほとんどにアクセスする場合が多いため、選択性は重要視されません。そのような状況では、全表スキャンが一般的であり、索引は使用しません。このマニュアルは、主として、OLTPアプリケーションを中心に説明しています。

関連項目:

- SQL文のチューニングおよび最適化のプロセスの詳細は、『Oracle Database SQLチューニング・ガイド』を参照してください
- 意思決定支援システム(DSS)および混合環境の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください

問合せオプティマイザおよび実行計画

OracleデータベースでSQL文を実行すると、問合せオプティマイザは、問合せに指定されたオブジェクトの参照および条件に関連する様々な要素を考慮した後、最も効率的な実行計画を判断します。この判断は、SQL文の処理で重要なステップであり、実行時間が大きく変化します。

評価プロセスでは、問合せオプティマイザにより、システム上に収集された統計が確認され、最適なデータ・アクセス・パスおよびその他の考慮事項が判断されます。問合せオプティマイザの実行計画を、SQL文に挿入されたヒントで上書きできます。

パフォーマンス・チューニング機能およびツールの概要

効果的なデータの収集と解析は、パフォーマンスの問題を識別して修正するために不可欠です。Oracle Databaseには、パフォーマンス・エンジニアがデータベースのパフォーマンスに関する情報の収集に使用できる様々なツールがあります。データの収集に加え、パフォーマンスの監視、問題の診断およびアプリケーションのチューニングのためのツールもあります。

Oracle Databaseの収集および監視機能は、大部分が自動で動作し、Oracleバックグラウンド・プロセスによって管理されます。自動統計収集機能と自動パフォーマンス機能を有効にするには、STATISTICS_LEVEL初期化パラメータをTYPICALまたは

ALLに設定する必要があります。収集ツールおよびチューニング・ツールの出力は、Oracle Enterprise Manager Cloud Control (Cloud Control)、またはAPIとビューを使用して管理および表示できます。使いやすく、様々な自動化された監視および診断ツールの利点があることから、Cloud Controlをお薦めします。

関連項目:

- Cloud Controlを使用してOracle Databaseを管理する方法を学習するには、『Oracle Database 2日でデータベース管理者』を参照してください
- Cloud Controlを使用してデータベースのパフォーマンスをチューニングする方法を学習するには、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください
- DBMS_ADVISOR、DBMS_SQLTUNE、DBMS_AUTO_SQLTUNEおよびDBMS_WORKLOAD_REPOSITORYパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- STATISTICS_LEVEL初期化パラメータについては、『Oracle Databaseリファレンス』を参照してください

自動パフォーマンス・チューニング機能

Oracle Databaseの自動パフォーマンス・チューニング機能には、次のものがあります。

- 自動ワークロード・リポジトリ(AWR)では、問題の検出および自己チューニングを目的として、パフォーマンス統計が収集、処理および保守されます。「自動ワークロード・リポジトリ」を参照してください。
- 自動データベース診断モニター(ADDM)では、Oracleデータベースにおいて考えられるパフォーマンス上の問題について、AWRによって収集された情報が分析されます。「自動データベース診断モニターの概要」を参照してください。
- SQLチューニング・アドバイザでは、SQL文を変更しないでSQL文を素早く効率的に最適化することが可能です。Oracle Database SQLチューニング・ガイドを参照してください。
- SQLアクセス・アドバイザでは、マテリアライズド・ビュー、索引およびマテリアライズド・ビュー・ログについてアドバイスが提供されます。Oracle Database SQLチューニング・ガイドを参照してください。
- End to End Application Tracingでは、特定のユーザー、サービスまたはアプリケーション・コンポーネントに関して、システム上の過剰なワークロードが識別されます。Oracle Database SQLチューニング・ガイドを参照してください。
- 障害となっている問題が検出されると、サーバー生成アラートにより自動的に通知が提供されます。サーバー生成アラートを使用したデータベース操作の監視方法を学習するには、『Oracle Database管理者ガイド』を参照してください。
- Oracle Enterprise Manager Cloud Control (Cloud Control)から、インスタンスのメモリーを最適化するためのメモリー・アドバイザなど、その他のアドバイザを起動できます。通常、メモリー・アドバイザが使用されるのは、データベースの自動メモリー管理が設定されていない場合です。その他のアドバイザは、平均リカバリ時間(MTTR)の最適化、セグメントの縮小およびUNDO表領域の設定に使用されます。Cloud Controlで使用可能なアドバイザについて学習するには、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください。
- Cloud Controlの「データベース・パフォーマンス」ページには、リアルタイム監視および診断に関するホスト、インスタンス・サービス時間およびスループット情報が表示されます。このページは、選択した間隔で自動的に、または手動でリフレッシュするように設定できます。データベースの「パフォーマンス」ページについて学習するには、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください。

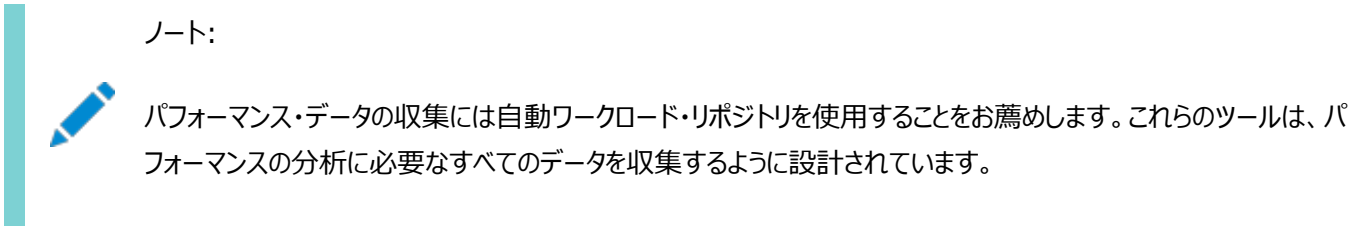
その他のOracle Databaseツール

この項では、パフォーマンス問題の調査に使用できるその他のOracle Databaseツールについて説明します。

V$パフォーマンス・ビュー

V$ビューは、すべてのOracle Databaseパフォーマンス・チューニング・ツールで使用されるパフォーマンスの情報ソースです。

V$ビューは、インスタンスの起動時に初期化されたメモリー構造に基づいています。メモリー構造およびそれらを表すビューは、インスタンスの存続期間中、Oracle Databaseにより自動的に管理されます。



ノート:

パフォーマンス・データの収集には自動ワークロード・リポジトリを使用することをお薦めします。これらのツールは、パフォーマンスの分析に必要なすべてのデータを収集するように設計されています。

関連項目:

- V$パフォーマンス・ビューを使用して、データベース・パフォーマンスの問題を診断する方法の詳細は、「パフォーマンス・ビューを使用したインスタンスのチューニング」を参照してください
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください

2 パフォーマンスを考慮した設計と開発

最適なシステム・パフォーマンスは設計段階で決まり、その効果は使用しているシステムが存続するかぎり持続します。初期の設計段階でパフォーマンス問題を慎重に検討すると、本番環境でのチューニングが容易になります。

この章の構成は、次のとおりです。

- オラクル社の新しい方法論
- 投資の選択肢について
- スケーラビリティについて
- システム・アーキテクチャ
- アプリケーション設計の原則
- ワークロードのテスト、モデル化および実装
- 新規アプリケーションのデプロイ

オラクル社の新しい方法論

コンピュータ・システムの規模が拡大して複雑になり、ビジネス・アプリケーションでのインターネットの役割が重要になるに従い、システムのパフォーマンスはますます重要になっています。オラクル社では、この状況にあわせて、設計およびパフォーマンスに関する長年の経験に基づいてパフォーマンス方法論を作成しました。この方法論は、システム・パフォーマンスを大幅に向上させる、明瞭で簡潔なアクティビティについて説明したものです。

パフォーマンス計画は、その効果によって異なります。業務システムや意思決定支援システムなどのようにシステムの目的が異なると、求められるパフォーマンス・スキルも異なります。このマニュアルでは、データベース設計者、管理者またはパフォーマンス・エンジニアが重点的に考慮する必要がある事柄について説明します。

システムのパフォーマンスは、設計してシステムに組み込むものです。偶然にパフォーマンスがよくなるわけではありません。パフォーマンスの問題は、通常、システム・リソースの競合、またはシステム・リソースを使い切ったことが原因で発生します。システム・リソースを使い切ると、システムを拡張してパフォーマンス・レベルを向上させることができません。ここで説明する新しいパフォーマンス方法論は、データベースの慎重な計画と設計に基づいており、システム・リソースを使い切ったことが原因で停止時間が発生するのを防止します。リソースの競合を除去することによって、ビジネス要件を満たすレベルまでシステムをスケーラブルにすることができます。

投資の選択肢について

高性能のプロセッサ、メモリーおよびディスク・ドライブが比較的安価に入手できることから、安易なシステム・リソースの追加購入によって、パフォーマンスを改善しようとする傾向があります。多くの場合、新しいCPU、メモリーまたはディスク・ドライブの増設によって、確かにパフォーマンスは一時的には改善されます。しかし、ハードウェア増設によるパフォーマンスの改善は、目の前の問題の短期的解決にすぎません。アプリケーションに対する需要と負荷が増加し続けると、すぐに同じ問題に直面する可能性があります。

状況によっては、ハードウェアを増設してもシステムのパフォーマンスがまったく改善されない場合もあります。システム設計が不適切な場合、追加のハードウェアをいくつ割り当ててもパフォーマンスは改善されません。ハードウェアを追加購入する前に、アプリケーション内にシリアライズやシングル・スレッドがないことを確認してください。長期的には、各ビジネス・トランザクションで使用する物理リソース数の観点からみて、アプリケーションの効率を上げるほうが一般的には効果的です。

スケーラビリティについて

スケーラブルという用語は、開発環境の様々な状況で使用されます。次の項では、アプリケーション設計者とパフォーマンス・エンジニアを対象にしたスケーラビリティについて説明します。

このセクションでは、次のトピックについて説明します。

- スケーラビリティとは
- システムのスケーラビリティ
- スケーラビリティを妨げる要因

スケーラビリティとは

スケーラビリティとは、より多くのワークロードを処理するためのシステムの能力で、それと比例してシステム・リソースの使用が増大します。

スケーラブルなシステムでは、ワークロードが2倍になると、使用するシステム・リソースも2倍になります。これは当たり前のようですが、システム内で競合が発生すると、元のワークロードに対し、リソースの使用が2倍よりも多くなる場合があります。

リソースの競合によってスケーラビリティが低くなる例を次に示します。

- ユーザー数の増加に伴い、アプリケーションでかなりの同時実行性管理が要求される場合
- ロック・アクティビティが増加した場合
- データ整合性に関するワークロードが増加した場合
- オペレーティング・システムのワークロードが増加した場合
- データ量の増加に伴い、トランザクションでデータ・アクセスの増加が要求される場合
- SQLと索引の不適切な設計が原因で、同じ戻り行数に対する論理I/O数が増加した場合
- データベース・オブジェクトのメンテナンス時間が長くなったことで、可用性が低下した場合

アプリケーションのワークロードが増加してもこれ以上のスループットの向上は不可能という点までシステム・リソースを使い切った場合、そのアプリケーションは拡張性がありません。このようなアプリケーションでは、スループットが固定化し、レスポンス時間が長くなります。

リソースを使い切った例を次に示します。

- ハードウェアを使い切った場合
- 大量トランザクションでの表スキャンによって、ディスクI/O不足が発生する場合
- 過剰なネットワーク・リクエストによって、ネットワークとスケジューリングにボトルネックが発生する場合

- メモリー割当てによって、ページングとスワッピングが発生する場合
- プロセスやスレッドの過剰な割当てによって、オペレーティング・システムのスラッシングが発生する場合

このことから、アプリケーション設計者は、ユーザー数やデータ量に関係なく同一リソースを使用するように設計し、限界を超える負荷をシステム・リソースに与えないようにする必要があります。

システムのスケーラビリティ

インターネット経由でアクセスできるアプリケーションでは、パフォーマンスや可用性の要件がさらに複雑になります。

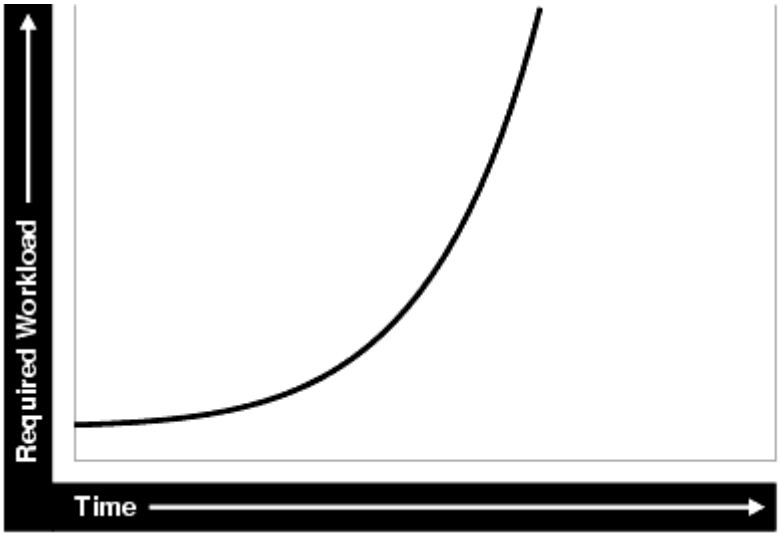
インターネット専用に設計および作成されたアプリケーションもありますが、一般会計アプリケーションなどの典型的なバック・オフィス・アプリケーションであっても、データの一部またはすべてをオンラインで利用できるようにすることが必要な場合があります。

インターネット時代のアプリケーションの特徴は、次のとおりです。

- 1日24時間、1年365日の可用性
- 同時ユーザー数が予測不可能で正確な把握が困難
- 容量の計画が困難
- あらゆるタイプの問合せが選択可能
- 多層アーキテクチャ
- ステートレスなミドルウェア
- 短い開発期間
- 最小限のテスト時間

次の図は、需要が増大するときの典型的なワークロード成長曲線を示しています。アプリケーションは、ワークロードの増大に伴い拡張できる必要があります。また、増大する需要をサポートするためにハードウェアが追加されたときにも拡張できる必要があります。設計に失敗すると、ハードウェア・リソースの増設や再設計に関係なく、実装が限界に達する可能性があります。

図2-1 ワークロード成長曲線



アプリケーションは非常に短期間での開発が要求され、テストや評価の時間も限定されています。しかし、一般的に、不適切な設計は、後でシステムの再構築および再実装が必要になることを意味します。アーキテクチャや実装に既知の制限があるアプリケーションをインターネット上にデプロイした場合、ワークロードが需要予測を超過すると、実際に障害が発生する可能性があり

ます。ビジネスの観点からは、低いパフォーマンスは顧客を失うことにつながります。Webユーザーの場合、7秒以内にレスポンスがないと、ユーザーの興味は二度と喚起できません。

多くの場合、新規の実装に移行するためにシステムを停止する間のコストも含めて、システムを再設計するコストは、最初からシステムを適切に構築する場合のコストを上回ります。ここでの教訓は単純明快です。開発の当初からスケーラビリティに留意して設計と実装を行うということです。

スケーラビリティを妨げる要因

アプリケーションを作成するとき、設計者とアーキテクトは、可能なかぎり完全なスケーラビリティに近づけることを目指す必要があります。これは線形のスケーラビリティと呼ばれ、システムのスループットがCPUの数に比例するものです。

スケーラビリティを直線的に高めることは、設計者の制御の及ばない部分があるため、実際には不可能です。しかし、アプリケーションの設計や実装に可能なかぎりスケーラブルにしておくと、現在も、将来においても、ハードウェア・コンポーネントの拡張やCPUテクノロジの発展とともにパフォーマンス目標を達成できます。

線形のスケーラビリティを妨げる要因としては次のものが考えられます。

- 不適切なアプリケーションの設計、実装および構成

  アプリケーションは、スケーラビリティに最も大きく影響します。次に例を示します。

  - スキーマ設計が不適切であると、SQLにコストがかかり、拡張性を持ちません。
  - トランザクション設計が不適切であると、ロックおよびシリアライズの問題が発生します。
  - 接続管理が不適切であると、レスポンス時間が長くなり、システムの信頼性が低下します。

  問題となるのは、設計のみではありません。アプリケーションの物理的な実装が弱点になる場合があります。次に例を示します。

  - システムが誤ったI/O方針のまま本番環境で使用される場合。
  - テスト時に生成された実行計画とは異なる実行計画が本番環境で使用される場合。
  - 実行時のメモリーの解放を十分に考慮せずに、大量のメモリー割当てを行うメモリー集中型アプリケーションで、メモリーが過剰に使用される可能性がある場合。
  - 非効率的なメモリー使用やメモリー・リークによって、動作中の仮想メモリー・サブシステムに高いストレスがかかる場合。このようなストレスは、パフォーマンスや可用性に影響を与えます。

- ハードウェア・コンポーネントの誤ったサイズ指定

  ハードウェア価格の低下に伴い、どのハードウェア・コンポーネントについても容量計画が不適切で問題になるということは少なくなっています。ただし、容量が大きすぎると、システムでワークロードが増大したときに、スケーラビリティの問題が隠されてしまう場合があります。

- ソフトウェア・コンポーネントの制限

  すべてのソフトウェア・コンポーネントに、スケーラビリティおよびリソース使用上の制限があります。このことは、アプリケーション・サーバー、データベース・サーバーおよびオペレーティング・システムでも同様です。アプリケーションの設計では、ソフトウェアの処理能力を超えた要求をしないでください。

- ハードウェア・コンポーネントの制限

ハードウェアは、完全にはスケーラブルになりません。ほとんどのマルチプロセッサ・コンピュータでは、限られた数のCPUでほぼ線形の拡張性を達成できますが、ある数を超えると、CPUの追加でパフォーマンスが全体的には向上しても、その数に見合った向上はありません。さらに続けて追加していくと、ある時点からパフォーマンスは向上しなくなり、むしろ低下する場合もあります。この動作は、ワークロードとオペレーティング・システムの設定に深く関連しています。

ノート:

前述の要因は、スケーラブルでないシステムのチューニングにおいてオラクル社のサーバー・パフォーマンス・グループが得た経験を基にしたものです。

システム・アーキテクチャ

システムのアーキテクチャには、主要な部分が2つあります。

- ハードウェア・コンポーネントとソフトウェア・コンポーネント
- 要件に合った正しいシステム・アーキテクチャの構成

ハードウェア・コンポーネントとソフトウェア・コンポーネント

システム・アーキテクチャには主に、ハードウェア・コンポーネントとソフトウェア・コンポーネントが含まれます。

- ハードウェア・コンポーネント
- ソフトウェア・コンポーネント

ハードウェア・コンポーネント

今日の設計者とアーキテクトは、多層環境の各層でハードウェアのサイズ指定と容量計画を行う必要があります。バランスの取れた設計を実現するのは、アーキテクトの責任です。これは、橋の設計に似ています。橋の設計者は、橋に関する様々なペイロードや構造上の要件をすべて考慮する必要があります。橋の強度は、最も弱いコンポーネントの強度にしかなりません。このため、橋は、すべてのコンポーネントがその設計上の限界に同時に達するように、バランスよく設計されています。

次に、システムの主要なハードウェア・コンポーネントを示します。

CPU

CPUは1つ以上使用されており、その処理能力は、携帯端末に見られるような単純なCPUのものから高性能サーバーのCPUのものまで様々です。他のハードウェア・コンポーネントのサイズは、通常、システム上のCPUの倍数になります。

メモリー

データベース・サーバーとアプリケーション・サーバーには、データをキャッシュしたり、長時間のディスク・アクセスを避けるために、十分な量のメモリーが必要です。

I/Oサブシステム

I/Oサブシステムは、クライアントPCのハード・ディスクから高性能なディスク・アレイまで様々あります。ディスク・アレイは、毎秒数千回のI/Oを実行できるうえ、複数のI/Oパスおよびホット・プラガブルなミラー化ディスクという冗長性から、可用性も得られます。

ネットワーク

システム内のコンピュータは、すべて、モデム回線から高速社内LANに到るまでのネットワークに接続しています。ネットワーク仕様に関する主な考慮点は、帯域幅(通信量)と待機時間(速度)です。

ソフトウェア・コンポーネント

コンピュータに共通のハードウェア・コンポーネントがあるように、アプリケーションにも共通の機能コンポーネントがあります。ソフトウェアの開発を機能コンポーネントごとに分割することで、アプリケーションの設計やアーキテクチャがより理解しやすくなります。システムのコンポーネントの中には、アプリケーションの実装の高速化や、共通コンポーネントの再作成の防止のために購入された既成のソフトウェアによって実行されるものもあります。

ソフトウェア・コンポーネントとハードウェア・コンポーネントの違いは、ハードウェア・コンポーネントが1つのタスクしか実行しないのに対して、ソフトウェアはその1つ1つが様々なソフトウェア・コンポーネントの役割を実行できるということです。たとえば、ディスク・ドライブはデータの格納と取出ししか行いませんが、クライアント・プログラムはユーザー・インタフェースを管理し、ビジネス・ロジックを実行できます。

ほとんどのアプリケーションに、次に関するソフトウェア・コンポーネントが含まれています。

ユーザー・インタフェース

ユーザー・インタフェースは、アプリケーション・ユーザーの目に触れることが最も多いコンポーネントです。次のような機能があります。

- ユーザーへの画面の表示
- ユーザー・データの収集とビジネス・ロジックへのデータの転送
- データ入力の検証
- アプリケーションのレベル間または状態間のナビゲーション

ビジネス・ロジック

このコンポーネントは、アプリケーション機能の中心となるコア・ビジネス・ルールを実装します。このコンポーネントでのエラーは、修復に相当なコストを要する場合があります。このコンポーネントは、宣言型アプローチとプロシージャ型アプローチの組合せで実装されます。宣言アクティビティの例としては、一意キーと外部キーの定義があります。また、プロシージャ・ベースのロジックの例としては、値引計画の実装があります。

このコンポーネントの一般的な機能は、次のとおりです。

- リレーショナル表構造へのデータ・モデルの移行
- リレーショナル表構造での制約の定義
- ビジネス・ルールを実装するプロシージャ型ロジックのコーディング

ユーザーのリクエストを管理するリソース

このコンポーネントは、すべてのソフトウェアに実装されます。ただし、アプリケーション設計の影響を受けるリクエストやリソースがある一方で、影響を受けないリクエストやリソースもあります。

マルチ・ユーザー・アプリケーションでは、ユーザー・リクエストによるリソース割当てのほとんどは、データベース・サーバーまたはオペ

レーティング・システムで処理されます。ただし、ユーザー数や使用パターンが不明または急増している大規模アプリケーションの場合、システム・アーキテクトはどのソフトウェア・コンポーネントも過負荷や不安定にならないように事前に対処する必要があります。

このコンポーネントの一般的な機能は、次のとおりです。

- データベースとの接続管理
- 効率的なSQLの実行(カーソルとSQLの共有)
- クライアント状態情報の管理
- ハードウェア・リソース間でのユーザー・リクエストのロード・バランシング
- ハードウェアおよびソフトウェア・コンポーネントの操作目標の設定
- タスクの非同期実行のための永続キューイング

データおよびトランザクション

このコンポーネントは、主にデータベース・サーバーとオペレーティング・システムが管理します。

このコンポーネントの一般的な機能は、次のとおりです。

- ロックとトランザクション・セマンティクスを使用した、データへの同時アクセス
- 索引およびメモリー・キャッシュを使用した、データへの最適化アクセス
- ハードウェア障害時のデータ変更の記録
- データに対して定義されているルールの適用

要件に合った正しいシステム・アーキテクチャの構成

初期のシステム・アーキテクチャを構成する作業は、反復的な処理です。システム・アーキテクトは、予算やスケジュールの制約内でシステム要件を満たす必要があります。対話型ユーザーがデータベースの内容に基づいてビジネスの取引や意思決定を行うシステムの場合、ユーザー要件が主体のアーキテクチャになります。システム上に対話型ユーザーがほとんどいない場合は、プロセス主体のアーキテクチャになります。

対話型ユーザー・アプリケーションの例を、次に示します。

- 経理アプリケーションや会計帳簿アプリケーション
- 注文入力システム
- 電子メール・サーバー
- Webベースの小売販売アプリケーション
- 取引システム

プロセス主体のアプリケーションの例を、次に示します。

- 公共料金支払システム
- 不正検出システム
- ダイレクト・メール

多くの点で、プロセス主体のアプリケーションのほうが、ユーザー・インタフェース要素がないのでマルチ・ユーザー・アプリケーションよりも設計が容易です。しかし、目的がプロセス指向であるため、大量のデータや様々な成功要因の扱いに慣れていないシステム・アーキテクトは混乱することがあります。プロセス主体のアプリケーションでは、ユーザー・ベースのアプリケーションとデータ・ウェアハウス・アプリケーションの両方で使用されるスキルを利用します。したがって、このマニュアルでは、対話型ユーザー向けの新しいシステム・アーキテクチャについて、重点的に説明します。

ノート:

システム・アーキテクチャの生成は、決定論的なプロセスではありません。ビジネス要件、テクノロジの選択肢、既存のインフラストラクチャやシステム、さらに予算や人員などの実際の物理リソースなどを慎重に考慮する必要があります。

次の質問事項は、システム・アーキテクチャの完全なガイドではありませんが、システム・アーキテクチャに関する思考力を高めます。これらの質問では、アーキテクチャ、実装しやすさ、さらにシステムの全体的なパフォーマンスと可用性に対して、ビジネス要件がどのように影響を及ぼすかを示しています。次に例を示します。

- 何人のユーザーをサポートするシステムが必要ですか。

  ほとんどのアプリケーションは、次のいずれかのカテゴリに該当します。

  - あまり使用されない専用のコンピュータでごく少数のユーザーをサポートする場合

    このタイプのアプリケーションでは、通常、ユーザーは1人です。この場合のアプリケーション設計の重点は、レスポンス時間を短くし、しかもアプリケーションの管理を最小限に抑えることで、シングル・ユーザーの生産性をできるだけ高くすることにあります。このようなアプリケーションのユーザーは、相互に介入することはほとんどなく、リソースの競合も最小限に抑えられます。

  - 中規模から大規模の数の社内ユーザーが共有アプリケーションを使用する場合

    このタイプのアプリケーションでは、ユーザー数は、社内でシステムを実際に使用してビジネスを行う従業員の数に限定されます。したがって、ユーザー数は予測可能です。ただし、信頼性のあるサービスを提供することが、ビジネスにとっては必須です。ユーザーはリソースを共有する必要があるため、アプリケーションを設計する際には、大量のシステム・ロード下でのレスポンス時間、各セッションで使用されるリソースの増加および将来の拡張のための余地に重点を置きます。

  - 無数のユーザーがインターネット上に存在する場合

    このタイプのアプリケーションでは、すべてのシステム・コンポーネントがそれぞれの限界を超えないように設計する必要があります。ボトルネックが作成されると、システムが停止するか不安定になります。このようなアプリケーションでは、複合的なロード・バランシング、ステートレスなアプリケーション・サーバーおよび効率的なデータベース接続管理が必要です。さらに、システムの過負荷が原因でユーザー・リクエストが満たされない場合にユーザーがフィードバックを得られるように、統計情報およびガバナーを使用します。

- ユーザーとの対話方式は何ですか。

  ユーザー・インタフェースの選択肢は、単純なWebブラウザからカスタム・クライアント・プログラムまで様々です。

- ユーザーはどこに位置しますか。

  ユーザー間の距離は、ネットワーク待機時間に対処するためのアプリケーションの設計方法に影響します。また、ユーザーの位置は、1日の中でピークの時間帯、つまりバッチ機能やシステム・メンテナンス機能を実行できない時間帯の決

定にも影響を与えます。

- ネットワーク速度はどの程度ですか。

  ネットワーク速度は、データ量に影響を与え、アプリケーション・サーバーやデータベース・サーバーとのユーザー・インタフェースの対話特性にも影響を与えます。対話主体のユーザー・インタフェースでは、キー・ストロークごとまたはフィールド・レベルの妥当性チェックごとにバックエンド・サーバーと通信します。対話が少ないインタフェースは、画面ごとに送受信を行うモデルで機能します。通信速度が遅いネットワークでは、対話主体のユーザー・インタフェースを使用しても高速のデータ入力速度は実現できません。

- ユーザーがアクセスするデータ量はどれくらいで、そのデータの中で読取り専用データが占める割合はどの程度ですか。

  オンラインでの問合せのデータ量は、表や索引の設計からプレゼンテーション層にいたる設計のあらゆる局面に影響を与えます。データベースのサイズによってユーザーのレスポンス時間が影響されないように設計する必要があります。アプリケーションが主として読取り専用の場合は、アプリケーション・サーバーのローカル・キャッシュへのレプリケーションとデータ配分が有効な選択肢になります。また、これにより、主要トランザクション・サーバーでのワークロードが削減されます。

- ユーザー・レスポンス時間の要件は何ですか。

  ユーザー・タイプに関する考慮が重要です。ユーザーが経営者で、瞬時に意思決定を行うために正確な情報を必要とする場合は、ユーザー・レスポンス時間の妥協は許されません。データ入力操作を行うユーザーなど、他のタイプのユーザーの場合は、それほど高いパフォーマンスは必要としません。

- ユーザーは1日24時間のサービスを期待していますか。

  取引が1日24時間行われている今日のインターネット・アプリケーションの場合、24時間連続稼働は必須です。ただし、1つのタイム・ゾーンでのみ稼働する企業システムの場合は、終業後にシステムを停止できます。この終業後の停止時間を利用して、バッチ処理やシステム管理を実行できます。このような場合は、連続稼働システムでない方が経済的です。

- 変更はすべてリアルタイムで行う必要がありますか。

  ユーザーのレスポンス時間内にトランザクションを実行する必要があるか、またはキューに入れて非同期で実行できるかを判断することが重要です。

次の質問は二次的な質問です。これらは設計にも影響を及ぼしますが、予算や実装しやすさの面に、より大きく影響します。次に例を示します。

- データベースの規模はどの程度ですか。

  データベースの規模は、データベース・サーバーのサイズ決定に影響します。大規模データベースを使用するサーバーでは、ワークロードから判断されるサイズよりも大型のコンピュータを用意することが必要になる場合があります。これは、大規模データベースに伴う管理オーバーヘッドが、主としてデータベース・サイズに比例するためです。表および索引が大きくなると、許容できる時間内に表を再編成したり索引を作成する必要があるため、必要なCPUの数もそれに比例して増加します。

- ビジネス・トランザクションで必要とされるスループットはどの程度ですか。

- 可用性に関する要件は何ですか。

- このアプリケーションを作成して管理するためのスキルはありますか。

- 予算上の制約からどのような妥協を求められますか。

アプリケーション設計の原則

この項では、アプリケーション作成時に必要な次の設計上の決定事項を説明します。

- アプリケーション設計の簡潔さ
- データのモデル化
- 表および索引の設計
- ビューの使用
- SQLの実行効率
- アプリケーションの実装
- アプリケーション開発の傾向

アプリケーション設計の簡潔さ

アプリケーションは、設計と開発を伴う他の製品となんら変わりはありません。通常、適切に設計された構造、コンピュータおよびツールは、信頼性があり、使用方法やメンテナンスが容易で、コンセプトもシンプルです。ごく一般的に表現すると、設計が適切に見えるものは、実際に適切に設計されているといえます。アプリケーションの作成では、この原則を常に覚えておいてください。

次の設計の問題を考慮してください。

- 表の設計が複雑で、誰も完全に理解できない場合、その表の設計は不適切であるといえます。
- SQL文が非常に長く、どのオプティマイザでもリアルタイムに効率よく最適化できない場合、そのSQL文、基になるトランザクションまたは表の設計が不適切であるといえます。
- 表に索引があり、同じ列が繰り返し参照される場合は、索引の設計が不適切であるといえます。
- 問合せを送信したときに、オンライン・ユーザーにとって必要な迅速なレスポンス能力がなかった場合は、ユーザー・インタフェースまたはトランザクションの設計が不適切であるといえます。
- ソフトウェアの多くの層で、データベース・コールがアプリケーション・ロジックから離れて抽象化される場合は、ソフトウェアの開発方法が不適切であるといえます。

データのモデル化

データのモデル化は、リレーショナル・アプリケーションの適切な設計に重要です。モデル化によって、ビジネス・プラクティスを迅速に表現する必要があります。正しいデータ・モデルに関して様々な議論が展開される場合があります。重要なことは、最も頻繁に行われるビジネス・トランザクションの影響を受けるエンティティをモデル化の主な対象にすることです。モデル化フェーズでは、あまり重要でないデータ要素のモデル化に時間を取られることがあり、開発の準備期間が延長される結果になります。モデル化ツールを使用すると、スキーマ定義をすばやく生成できます。また、短期間でプロトタイプを作成する必要がある場合に便利です。

表および索引の設計

表の設計は、主に、主要トランザクションの柔軟性とパフォーマンスの間でバランスを取る作業です。データベースの柔軟性を保持しながら予想外のワークロードに対処できるようにするには、表の設計をデータ・モデルとほぼ同じようにする必要があり、最低でも第3正規形に正規化しておく必要があります。ただし、ユーザーが必要とする特定のトランザクションでは、パフォーマンス向

上のために選択的な非正規化が求められることがあります。

この技法の例としては、事前に結合された表の格納、導出列の追加および集計値があります。Oracle Databaseでは、クラスタ化機能およびマテリアライズド・ビュー機能によって、集計データおよび事前結合データに関する多数のストレージ・オプションが提供されています。これらの機能によって、より簡潔な表の設計を最初から採用できます。

ここでも、最適のパフォーマンスを実現できるように、ビジネス上重要な表に重点的にリソースを使用する必要があります。重要でない表の場合は、アプリケーション開発を迅速に進めるためにも、設計に簡略な方法を採用できます。ただし、プロトタイプおよびテストの結果、重要でない表でパフォーマンスの問題が発生する場合は、ただちに設計を修正する必要があります。

索引の設計も、アプリケーション設計者が生成したSQLに基づく、主として反復的なプロセスです。ただし、主キー制約を適用する索引や個人名など既知のアクセス・パターンに対する索引を作成することから開始することも可能です。アプリケーションの開発が進み、実際のデータ量でテストを実行するときは、適切な索引を作成して特定の問合せのパフォーマンスを改善する必要があります。新しい索引を作成する際には、索引の設計に関する次の事項を検討してください。

- 索引への列の追加または索引構成表の使用
- 異なる索引タイプの使用
- 索引のコストの算出
- 索引内のシリアライズ
- 索引内の列の順序付け

索引への列の追加または索引構成表の使用

問合せをスピードアップする簡単な方法の1つは、実行計画から表アクセスを排除して論理I/Oの数を減らすことです。これは、問合せによって参照されるすべての列を索引に追加することで実現できます。これらの列は、選択リスト列、および結合またはソートに必要な列です。この方法は、時間がかかるI/Oを削減して、オンライン・アプリケーションのレスポンス時間を短縮する場合に特に役立ちます。これは、適切なサイズのデータを使用してアプリケーションを初めてテストするときに最も効果を発揮します。

この技法を最も積極的に使用しているのが、索引構成表(IOT)の作成です。ただし、IOTのリーフ・サイズの増加がI/O削減効果を妨げないように注意する必要があります。

異なる索引タイプの使用

選択できる索引のタイプはいくつかあり、それぞれ特定の状況に応じた利点があります。各タイプの索引に関連したパフォーマンスの考慮点を、次のリストに示します。

Bツリー索引

標準的な索引タイプです。主キー索引および選択的な索引に最も適しています。連結索引として使用すると、索引列順にソートされたデータを取得するのにBツリー索引を使用できます。

ビットマップ索引

これらの索引は、比較的少ない個別値を持つ列に適しています。この場合、Bツリー索引を追加する利点が制限される可能性があります。これらの索引は、DMLアクティビティおよび非定型フィルタリング・パターンが少ないデータ・ウェアハウス・アプリケー

ションに適しています。ビットマップ索引を列に結合すると、最小限のI/Oで効率的なAND操作およびOR操作が可能になります。さらに、この索引は圧縮技法によって最小限のI/Oで多数のROWIDを生成できます。ビットマップ索引は、索引内で問合せを満たすことができるため、COUNT()を使用した問合せで特に効果的です。

ファンクション索引

この索引では、ベース・データ上のファンクションから導出された値に対するBツリー経由でアクセスできます。ファンクション索引ではNULLの使用に制限があり、問合せオプティマイザを有効にしておく必要があります。

ファンクション索引は、複合列への問合せで導出される結果を生成する場合や、データベースへのデータの格納方法における制限を克服する場合に、特に役立ちます。その例として、(販売価格 - 値引)×数量から算出された特定の値を超える注文の明細項目(表内の列)の問合せがあります。別の例としては、UPPERファンクションをデータに適用した、大/小文字を区別しない検索があります。

パーティション索引

グローバル索引のパーティション化によって、パーティション・プルーニングが索引アクセス内で発生し、I/Oを削減できます。レンジ・パーティション化またはリスト・パーティション化を適切に定義すると、正しい索引パーティションが高速で索引スキャンされるため、問合せ時間を大幅に短縮できます。

逆キー索引

この索引は、挿入アプリケーションでの索引のホット・スポットを除去するように設計されています。この索引は挿入パフォーマンスに優れていますが、索引レンジ・スキャンに使用できないという制限があります。

索引のコストの算出

索引構造の作成とメンテナンスにはコストがかかり、ディスク領域、CPUおよびI/O容量などのリソースを消費します。設計者は、索引のメンテナンスにかかるコストより、索引の使用による利点が上回るように設計する必要があります。

索引のメンテナンスにかかるコストを見積る簡単な目安があります。それは、索引キーに対してINSERT、DELETEまたはUPDATEを実行することでメンテナンスされる索引では、索引ごとに、表に対する実際のDML操作で使用されるリソースの3倍のリソースが必要になる、というものです。したがって、3つの索引を持つ表へのINSERTは、索引のない表へのINSERTよりも約10倍遅くなります。DMLの場合、特にINSERT操作が多いアプリケーションでは、問合せとINSERT操作のパフォーマンスとの間のバランスを取る必要があるため、索引の設計は十分に検討する必要があります。

関連項目:

索引の使用状況の監視方法を学習するには、『Oracle Database管理者ガイド』を参照してください

索引内のシリアライズ

順序またはタイムスタンプを使用して索引付きキー値を生成すると、データベースのホット・スポット問題が生じ、レスポンス時間やスループットに影響を与える場合があります。これは、通常、キーが単調に増加することによるもので、結果として索引は適度に増加します。この問題を回避するには、索引の範囲全体に対して挿入されるようなキーを生成して、ワークロードのスケーラビ

リティを拡張します。これを行うには、次の方法をいずれでも使用できます。

- 逆キー索引を使用
- ハッシュ・パーティション索引を使用
- 循環する順序の接頭辞を順序値に使用
- スケーラブルな順序を使用

関連項目:

スケーラブルな順序の詳細は、Oracle Database管理者ガイドを参照してください。

索引内の列の順序付け

設計者は、索引作成の規則を柔軟に定義する必要があります。状況に応じて、次のいずれかの方法で索引内のキーに順序を付けます。

- 選択頻度の高い列から順序を付けます。これによって、必要なROWIDに最小限のI/Oで最も速くアクセスできるため、通常はこの方法を使用します。この方法は、主として主キーや選択頻度の高いレンジ・スキャンに使用します。
- 列に順序を付け、データをクラスタ化またはソートしてI/Oを削減します。大規模なレンジ・スキャンでは、通常、選択頻度の低い順に列に順序を付けるか、取り出す順にデータをソートすると、I/Oを削減できます。

ビューの使用

ビューを使用すると、アプリケーションの設計がスピードアップされ簡潔になります。簡単なビュー定義により、データの取出し、表示、収集および格納処理を優先するプログラマが、複雑なデータ・モデルから解放されます。

ただし、ビューは、クリーンなプログラミング・インタフェースを提供する一方で、最適とはいえないリソース集中型の問合せの原因になる場合があります。ビューの最も不適切な使用例は、ビューが他の複数のビューを参照し、その複数のビューが問合せ内で結合されている場合です。多くの場合、開発者はビューを使用せずに表から直接問合せを満たすことができます。ビューが持つ本来の特性により、通常は、オプティマイザによる最適な実行計画の生成が困難になります。

SQLの実行効率

システム開発の設計およびアーキテクチャ・フェーズでは、アプリケーション開発者がSQLの実行効率を理解していることが重要です。この目標を達成するには、開発環境が次の特性をサポートしている必要があります。

- データベース接続の適切な管理

  データベースへの接続は、コストが高くスケーラブルでない操作です。このため、データベースへの同時接続数はできるだけ少なくする必要があります。アプリケーションの初期化時にユーザーが1人接続しているという単純なシステムが理想的です。しかし、Webベース・アプリケーションや多層アプリケーションでは、複数のアプリケーション・サーバーが使用されてユーザーへのデータベース接続が多重化しているため、接続数を少なくするのは困難です。このようなタイプのアプリケーションでは、データベース接続をプールし、ユーザー・リクエストごとに接続が再確立されないように設計する必要があります。

- カーソルの適切な使用および管理

ユーザー接続のメンテナンスも、システムでの解析アクティビティの最小化にとっては同じように重要です。解析とは、SQL文を解釈し、そのSQL文の実行計画を作成する処理です。この処理には、構文検査、セキュリティ検査、実行計画の生成、共有プールへの共有構造のロードなど、多くのフェーズがあります。解析操作には、次の2種類があります。

- ハード解析

  SQL文が初めて送信され、共有プール内に一致するものがない場合です。ハード解析は、解析に関連するすべての操作を実行するため、最もリソース集中型であり、スケーラブルではありません。

- ソフト解析

  SQL文が初めて送信され、共有プール内に一致が見つかった場合です。別のユーザーが以前に実行した結果が一致することがあります。SQL文は共有されるため、パフォーマンスが向上します。ただし、ソフト解析では、システム・リソースを消費する構文検査やセキュリティ検査が必要であり、理想的とはいえません。

解析はできるだけ最小限にする必要があるため、アプリケーション開発者は、SQL文を1回解析し、そのSQL文を複数回実行するようにアプリケーションを設計してください。これは、カーソルを使用して行います。経験のあるSQLプログラマであれば、カーソルのオープンと再実行の概念を理解しています。

アプリケーション開発者は、SQL文が共有プール内で共有されるようにする必要もあります。この目標を達成するには、問合せの中で実行ごとに変化する部分をバインド変数として表します。これを行わないと、SQL文は1回解析された後、他のユーザーから再利用されない可能性があります。SQLの共有を確実にするには、SQL文ではバインド変数を使用し文字列リテラルは使用しないようにします。次に例を示します。

文字列リテラルがある文は、次のとおりです。

```
SELECT * FROM employees
  WHERE last_name LIKE 'KING';
```

バインド変数がある文は、次のとおりです。

```
SELECT * FROM employees
  WHERE last_name LIKE :1;
```

次の例は、単純なOLTPアプリケーションでのテスト結果です。

```
Test                          #Users Supported
No Parsing all statements           270
Soft Parsing all statements         150
Hard Parsing all statements          60
Re-Connecting for each Transaction   30
```

このテストは、4台のCPUを搭載したコンピュータで実行されました。システム上のCPUの数が増えると、差異も大きくなります。

アプリケーションの実装

開発環境およびプログラミング言語の選択は、開発チームのスキルと、アプリケーションの指定時に決定したアーキテクチャにより決定します。ただし、簡単なパフォーマンス管理規則がいくつかあり、これに従うと、スケーラブルで高パフォーマンスのアプリケーションの実現につながります。

1. ソフトウェア・コンポーネントに適した開発環境を選択し、その環境によってパフォーマンスに関する設計が制限されないようにしてください。設計が制限される場合は、選択した言語または環境が不適切であるといえます。

- ユーザー・インタフェース

  プログラミング・モデルには、HTML生成からウィンドウ・システムの直接コールまで様々なモデルがあります。開発方法では、ユーザー・インタフェース・コードのレスポンス時間に重点を置く必要があります。HTMLまたはJavaをネットワークで送信する場合は、ネットワーク通信量や対話を最小限に抑えるようにしてください。

- ビジネス・ロジック

  JavaやPL/SQLなどのインタプリタ型言語は、ビジネス・ロジックのエンコードには理想的です。これらの言語は完全に移植可能であるため、ロジックのアップグレードが比較的簡単にできます。どちらの言語も構文が豊富で、読みやすく解析しやすいコードを作成できます。ビジネス・ロジックで複雑な数学関数が必要な場合は、コンパイラ型バイナリ言語が必要になる場合があります。ビジネス・ロジック・コードは、クライアント・コンピュータ、アプリケーション・サーバーおよびデータベース・サーバーに配置できます。ただし、アプリケーション・サーバーに配置するのが最も一般的です。

- ユーザー・リクエストとリソース割当て

  プログラミング言語の影響はほとんど受けませんが、データベース接続およびカーソルの管理をマスクするツールおよび第4世代言語では、非効率的なメカニズムが使用されることがあります。このようなツールや環境を評価するときは、データベース接続モデルおよびカーソルやバインド変数の使用方法をチェックしてください。

- データ管理とトランザクション

  これに関しては、プログラミング言語による影響はほとんどありません。

2. ソフトウェア・コンポーネントを実装するときは、その機能を実装するようにし、他のコンポーネントに関連付けられている機能は実装しないでください。他のコンポーネントの機能を実装すると、設計や実装が最適でなくなります。これはすべてのコンポーネントに当てはまります。

3. 機能のギャップをそのまま放置したり、検討中のソフトウェア・コンポーネントを設計、実装またはテストで使用しないでください。多くの場合、ギャップは、アプリケーションを展開するか、実際のボリュームでテストするまで検出されません。このギャップは、通常、アーキテクチャまたは当初のシステム仕様が不適切であることを示します。データのアーカイブ・モジュールおよびパージ・モジュールは、初期のシステム設計、作成および実装時には最も無視されやすいモジュールです。

4. プロシージャ型ロジックを実装するときは、C、JavaまたはPL/SQLなどの手続き型言語で実装するようにします。データ・アクセス(問合せ)またはデータ変更(DML)を実装するときは、SQLを使用します。これは、プロシージャ型コードとデータ・アクセス(非プロシージャ型SQL)コードが混在しているビジネス・ロジック・モジュール特有の規則です。SQLアクセスにプロシージャ型ロジックを使用することも考えられます。しかし、これはリソースを多く消費するSQLになる傾向があります。DECODEケース文を含むSQL文は、多くの場合、最適化が必要です。大量のOR述語または集合演算子(UNIONやMINUSなど)を含む文も同様です。

5. 頻繁にアクセスし、変更がほとんどなく、繰り返し取り出すのにコストがかかるデータは、キャッシュします。ただし、キャッシュ・メカニズムは使いやすくして、元の方法でデータにアクセスするよりもコストが実際に低くなるようにしてください。これは、頻繁に使用するデータ値はリモート・データ・ストアまたはコストがかかるデータ・ストアから繰り返し取り出すよりも、キャッシュするかローカルに格納する必要があるすべてのモジュールにあてはまります。

   ローカルにキャッシュするデータの一般的な候補例をいくつか示します。

   - 本日の日付。SELECT SYSDATE FROM DUALが、データベースにかかるワークロードの60%以上を占めることもあります。
   - 現行ユーザー名。

- 税率、値引率、位置情報など、繰り返し使用するアプリケーション変数および定数。
- ローカルでのデータ・キャッシュは、さらに、アプリケーション・サーバー中間層へのローカル・データ・キャッシュの作成へ発展させることができます。これにより、中央のデータベース・サーバーの負荷が減ります。ただし、ローカル・キャッシュを作成するときは、パフォーマンス上の利点がなくなるほどキャッシュが複雑にならないように注意してください。
- ローカル順序の生成。

キャッシュを使用する場合は、設計上の意味を考慮してください。たとえば、ユーザーが午前0時に接続して日付がキャッシュされた場合、ユーザーの日付値は無効になります。

6. コンポーネント間のインタフェースを最適化し、すべてのコンポーネントが最もスケーラブルな構成で使用されるようにします。この規則に説明はほとんど不要で、すべてのモジュールとインタフェースに当てはまります。
7. 外部キー参照を使用します。アプリケーションから参照整合性を適用するのはコストがかかります。外部キー参照は、子の列値を親から選択してその存在を確認することで、メンテナンスできます。Oracleが提供する外部キー制約(SQLを使用しない)は高速で、しかも簡単に宣言でき、ネットワーク・トラフィックを作成しません。
8. End to End Application Tracingで使用するアクション名およびモジュール名の設定を検討してください。設定すると、ワークロードの問題を柔軟にトレースできます。

アプリケーション開発の傾向

今日のアプリケーション開発には、2つの大きな特徴があります。1つは、コンパイラ型のCまたはC++アプリケーションにかわってJavaの使用が増えていること、もう1つは、スキーマ設計に影響を与えるオブジェクト指向手法の使用が増えていることです。

Javaはコードの移植性に優れ、高い可用性をプログラマに提供します。ただし、Javaにはパフォーマンスに影響することがいくつかあります。Javaはインタプリタ型言語であるため、C言語などのコンパイラ型言語に比べてロジックの実行速度が遅くなります。その結果、クライアント・コンピュータでのリソース使用量が増加します。このため、強力なCPUをクライアント・コンピュータや中間層コンピュータに設置する必要があり、プログラマは効率的なコードを作成するようにさらに注意を払う必要があります。

Javaはオブジェクト指向言語であるため、ビジネス・ロジックを実行しないクラスへのデータ・アクセスの分離を奨励します。この結果、プログラマは、使用するデータ・アクセス・メソッドの効率を認識せずにそのメソッドを呼び出す可能性があります。このため、データベース・アクセスが最小限になり、データベースへのインタフェースには最も簡潔で単純なインタフェースが使用される可能性があります。

このタイプのソフトウェア設計では、常にすべてのWHERE述語が問合せに効率的に組み込まれるとは限らず、行のフィルタ処理はJavaプログラムで実行されます。これはかなり非効率的です。さらに、DML操作、特にINSERT操作では単一のINSERTが実行され、配列インタフェースは使用できません。これが、プロシージャ・コールによりさらに非効率的になることがあります。データベースとのデータのやり取りでは、実際のデータベース・コールよりも多くのリソースが使用されます。

一般に、最善の総合的トランザクション設計を実現するには、データ・アクセス・コールをビジネス・ロジックの次に配置するのが最適です。

プログラミング・レベルでのオブジェクト指向の採用は、Oracleサーバー内にオブジェクト指向データベースを作成することに発展しています。これは、オブジェクト構造のBLOBへの格納および効果的な索引付きカード・ファイルとしてのデータベースの使用から、Oracle Databaseオブジェクト・リレーショナル機能の使用まで、様々な形で現れています。

オブジェクト指向アプローチをスキーマ設計に採用する場合は、リレーショナル・ストレージ・モデルの柔軟性が失われないようにします。多くの場合、スキーマ設計でのオブジェクト指向アプローチにより、データ構造にかなりの非正規化が生じ、相当なメンテナンスが必要になり、オブジェクトに関連付けられたREFポインタも必要になります。このような設計は、多くの場合、リレーショナル格納方式に置き換わる前の階層データベースやネットワーク・データベースの設計に戻ることを意味します。

要約すると、データベースにデータを長期間格納する場合、また同一スキーマに対する非定型問合せまたはアプリケーション開発をある程度予期している場合は、リレーショナル格納方式が最善のパフォーマンスと柔軟性をもたらす可能性があります。

ワークロードのテスト、モデル化および実装

この項では、ワークロードの見積り、モデル化、実装およびテストについて説明します。このセクションでは、次のトピックについて説明します。

- データのサイズ設定
- ワークロードの見積り
- アプリケーションのモデル化
- 設計のテスト、デバッグおよび検証

データのサイズ設定

不適切なサンプル・セットを使用すると、可変長データを扱うときにサイズの見積りを誤ることがあります。データ量の増加に伴い、キーの長さも大幅に長くなり、列サイズの見積りに変更が生じます。

システムは、本番稼働が始まると、データベース規模の拡大、特に索引の増加は予想が困難になります。表は時間とともに増大し、索引は、キーの生成、挿入パターンおよび行の削除というアプリケーションの個々の動作によって変化します。最悪のケースは、昇順キーを使用して行を挿入してから、一部の行を残してほとんどの行を表の左側から削除した場合です。これによって、ギャップと無駄な領域が残ります。索引をこのように使用している場合は、オンラインの索引再作成機能の使用方法を理解している必要があります。

DBAは、オブジェクトごとに領域割当てを監視し、増大して制御できなくなる可能性のあるオブジェクトを探します。アプリケーションを十分に理解することで、急速にまたは予測を超えて増大する可能性のあるオブジェクトを発見できます。これは、あらゆるシステムのパフォーマンス計画と可用性計画の両方で重要なことです。本番データベースを実装するときは、対話型ユーザーがアプリケーションを使用しているときに領域管理が最小限になるように設計する必要があります。これは、データ・セグメント、一時セグメントおよびロールバック・セグメントのすべてに当てはまります。

ワークロードの見積り

含まれる変数の数を考えると、容量計画およびテストのためのワークロードの見積りは非常に困難です。しかし、設計者は、CPU、メモリーおよびディスク・ドライブを搭載したコンピュータを指定して、最終的にはアプリケーションを展開する必要があります。サイズ設定に使用する技法はいくつかあり、それぞれに利点があります。サイズを設定するときは、次の方法を使用して意思決定プロセスを検証し、サポート用ドキュメントを準備することをお薦めします。

類似システムからの推定

これは完全に経験に基づくアプローチで、特性が類似しており、パフォーマンスが把握されている既存のシステムを基礎システムとして使用します。このシステムの仕様を、サイズ設定の担当者が既知の相違点に応じて変更します。このアプローチでは、既存のシステムと関連する部分は参考になりますが、相違点のある部分を扱うときにはほとんど参考になりません。

このアプローチは、巨大なビル、船舶、橋梁、石油掘削装置などの大規模エンジニアリング分野のほとんどすべてで、エンジニアリング・プロジェクトのコストを見積るときに使用されています。参照システムがサイズの点で計画しているシステムと大きな差がある場合は、一部のコンポーネントが設計上限を超えてしまう可能性があります。

ベンチマーク

ベンチマーク・プロセスは、リソースと時間をかなり消費し、必ずしも正確な結果が得られるとはかぎりません。開発初期またはプロトタイプの段階でアプリケーションをシミュレーションすると、実際の本番システムの場合とは異なるものを計測する危険性があります。予想外かもしれませんが、オラクル社では長年にわたりデータベース開発組織とともに顧客アプリケーションのベンチマークを行っていますが、ベンチマーク・アプリケーションと実際の本番システムとの間に信頼に値する相関関係はまだ認められていません。これは、開発プロセスでアプリケーションの非効率な点がいくつか組み込まれてしまうことが主な原因です。

しかし、ベンチマークは、許容可能なレベルの精度でシステムのサイズを設定するためには有効に利用されています。特に、ベンチマークは、システムの負荷が最大になったときの実際のI/O要求数の判断やリカバリ処理のテストに効果があります。

ベンチマークでは、その性質上、システムの全コンポーネントに対してそれぞれの上限までストレスがかけられます。すべてのコンポーネントにストレスがかけられるため、ベンチマーク中にアプリケーションの設計および実装のエラーがすべて明らかになります。また、ベンチマークでは、データベース、オペレーティング・システムおよびハードウェア・コンポーネントもテストされます。ほとんどのベンチマークが急いで実行されるため、システム・コンポーネントが失敗すると障害や問題が発生すると考えてください。ベンチマークは実施者にストレスがかかる作業であり、ベンチマークから最大限の結果を得るには豊富な経験が必要です。

アプリケーションのモデル化

アプリケーションのモデル化は、複雑な数学的モデル化から、封筒の裏を使用するような典型的な単純計算まで多岐にわたります。どちらの方法にもメリットがあり、一方は高い精度を目標とし、もう一方は大まかな見積りを作成します。デメリットは、どちらも実装エラーや効率の悪さが考慮されないことです。

見積りやサイズ設定のプロセスは、正確性に欠ける技法です。しかし、このプロセスを精査することで、ある程度知的な見積りを行うことができます。見積りプロセス全体としては、不適切なSQL、索引の設計またはカーソル管理によるアプリケーションの非効率は考慮されていません。サイズ設定担当者は、アプリケーションの非効率性に対応する余裕を組み入れる必要があります。パフォーマンス・エンジニアは、非効率性を検出し、見積りを現実的なものにします。アプリケーションの非効率性を検出する方法は、Oracleのパフォーマンス改善方法で説明しています。

設計のテスト、デバッグおよび検証

テスト・プロセスは、主に機能テストと安定性テストで構成されます。このプロセスの途中で、パフォーマンス・テストが実施されます。

アプリケーションのパフォーマンス・テストを実行するときの簡単な規則を説明したリストを次に示します。適切に文書化した場合、このリストは、アプリケーション稼働後の本番アプリケーションと容量計画プロセスにとって重要な情報を提供します。

- 自動データベース診断モニター(ADDM)とSQLチューニング・アドバイザを使用した設計の検証
- 現実的なデータ量とデータ配分によるテスト

  すべてのテストは、データが完全に含まれている表を使用して行う必要があります。テスト用データベースには、データ量や表のカーディナリティという点で本番システムと同様のデータを入れておく必要があります。本番と同様の索引をすべて作成し、スキーマ統計を正しく入力します。
- 正しいオプティマイザ・モードの使用

  すべてのテストは、本番で使用するオプティマイザ・モードで実行する必要があります。Oracle Databaseのすべての研究開発の取組みは、問合せオプティマイザに重点が置かれています。したがって、問合せオプティマイザの使用をお薦めします。
- シングル・ユーザーのパフォーマンスのテスト

  アイドル状態または使用率の低いデータベースで、シングル・ユーザーのパフォーマンスが許容範囲にあるかテストします。シングル・ユーザーのパフォーマンスが理想的な条件下で許容範囲に達しない場合、実際の条件下で複数のユーザーが許容可能なパフォーマンスを得ることはできません。
- 全SQL文の計画の取得と文書化

  各SQL文の実行計画を取得します。このプロセスを使用して、オプティマイザの実行計画が最適かどうか、およびSQL文の相対コストがCPU時間と物理I/O数の観点から把握されているかどうかを検証します。このプロセスは、将来において多くのチューニングおよびパフォーマンスの必要な頻繁に使用されるトランザクションを識別するのに役立ちます。
- マルチ・ユーザーのテスト

  ユーザーのワークロードやプロファイルがまだ完全に定量化されていない可能性があるため、このプロセスを正確に実行するのは困難です。しかし、DML文を実行するトランザクションをテストして、ロックの競合やシリアライズの問題がないことは確認する必要があります。
- 正しいハードウェア構成でのテスト

  できるだけ本番システムに近い構成でテストを行います。実際的なシステムの使用は、ネットワーク待機時間、I/Oサブシステム帯域幅およびプロセッサのタイプと速度についてテストする場合に特に重要です。このアプローチを使用しなかった場合、潜在的なパフォーマンスの問題を正しく分析できません。
- 安定した状態でのパフォーマンスの測定

  ベンチマークを行うときは、安定した状態でパフォーマンスを測定することが重要です。各ベンチマークの実行には開始フェーズが必要です。このフェーズでは、ユーザーがアプリケーションに接続し、アプリケーションでの作業を徐々に開始します。このプロセスによって、安定した状態になる前に、頻繁にキャッシュされるデータをキャッシュに初期化し、解析などの1回のみ実行する操作を完了しておくことができます。同様に、ベンチマークの実行後には終了フェーズが必要です。このフェーズでは、リソースをシステムから解放し、ユーザーは作業を終了して接続を切断します。

新規アプリケーションのデプロイ

アプリケーションのデプロイに関連する主要な設計面での決定事項について次に説明します。

- ロールアウトの方法
- パフォーマンス・チェックリスト

ロールアウトの方法

新しいアプリケーションをロールアウトするときは、次の2つの方法が一般的です。

- ビッグ・バン・アプローチ: 全ユーザーが一度に新しいシステムに移行します。
- トリクル・アプローチ: ユーザーは既存のシステムから新しいシステムに徐々に移行します。

いずれの方式にもメリットとデメリットがあります。ビッグ・バン・アプローチでは、必要な規模でアプリケーションを十分にテストしておく必要がありますが、旧システムは完全に使用されなくなるため、旧システムからのデータの変換と旧システムとの同期が最小限で済みます。トリクル・アプローチでは、ワークロードの増加に伴いスケーラビリティの問題をデバッグできますが、移行中に旧システムとの間でデータを相互に移行する必要性が発生することがあります。

いずれの方法も、それぞれ移行実施中にシステムの停止につながるリスクがあるため、どちらかを推奨することは困難です。トリクル・アプローチでは、実際のユーザーが新しいアプリケーションに移行するにつれてユーザー・プロファイルを作成できるので、システムを再構成しても、その影響を受けるのは移行済ユーザーのみです。この方式は、初期の段階で移行したユーザーの作業に影響を与えますが、サポート・サービスの負荷は限定されます。これは、スケジュール外の停止が発生しても、影響を受けるユーザーの割合が小さいことを意味します。

新規アプリケーションのロールアウト方法は、ビジネスごとに判断してください。採用するどのアプローチにも、それぞれ固有の長所と短所があります。テストを重ねて、そのテストから得られた知識が増えるほど、最善のロールアウト方式を判断できます。

パフォーマンス・チェックリスト

本番での最適なパフォーマンスの可能性を高め、アプリケーションの迅速なデバッグを可能にするタスクのリストを作成すると、ロールアウトの際に役立ちます。次を実行します。

1. 本番データベース用の制御ファイルを作成するときは、MAXINSTANCES、MAXDATAFILES、MAXLOGFILES、MAXLOGMEMBERSおよびMAXLOGHISTORYをロールアウトで予測されるよりも大きい値に設定することで、データベースの成長に対応します。この方法では、ディスク領域の使用量が増えて制御ファイルも大きくなりますが、後でこれらを緊急に拡張する必要が生じたときに時間を節約できます。
2. ブロック・サイズをアプリケーションの開発時に使用した値に設定します。本番同様のデータ量でテストが完了し、現在のSQL実行計画が正しい場合は、スキーマ統計を開発またはテスト環境から本番データベースにエクスポートします。
3. 最小限の初期化パラメータを設定します。他のパラメータはデフォルトのままにしておくのが理想的です。チューニングをさらに行う必要がある場合は、システムに負荷がかかったときに明らかになります。
4. データベース・オブジェクトの記憶領域オプションを設定して、ブロック競合の管理に備えます。INSERT/UPDATE/DELETE操作が頻繁に発生する表と索引を作成するには、自動セグメント領域管理を使用する必要があります。自動UNDO管理を使用して、ロールバック・セグメントの競合を回避します。
5. すべてのSQL文が最適であることを検証し、そのリソース使用を把握します。
6. データベースに接続するミドルウェアとプログラムの接続が効率的に管理され、ログオンとログオフが繰り返し発生しないことを確認します。
7. SQL文でカーソルが効率的に使用されていることを確認します。データベースでは、各SQL文を1回解析したら、それを複数回実行します。これが正しく行われない原因として最も一般的なものは、バインド変数が正しく使用されていないことか、WHERE句の述語が文字列リテラルとして送信されていることです。プリコンパイラを使用してアプリケーションを開発する場合は、アプリケーションをプリコンパイルする前に、MAXOPENCURSORS、HOLD_CURSORおよびRELEASE_CURSORの各パラメータのデフォルト値からの再設定が必要です。

8. すべてのスキーマ・オブジェクトが開発環境から本番データベースに正しく移行されていることを確認します。これには、表、索引、順序、トリガー、パッケージ、プロシージャ、ファンクション、Javaオブジェクト、シノニム、権限付与およびビューが含まれます。テスト中に変更を加えた場合は、その変更が本番システムにも加えられていることを確認します。
9. システムをロールアウトした直後に、データベースとオペレーティング・システムの統計用ベースライン・セットを設定します。最初の統計データにより、設計およびロールアウト・プロセスで設定した前提事項が検証または訂正されます。
10. 最初のボトルネックの予測を開始し(不可避のボトルネック)、Oracleのパフォーマンス改善方法に従ってパフォーマンスを改善します。

3 パフォーマンス改善方法

この章では、Oracle Databaseのパフォーマンス改善方法について説明します。この章の内容は次のとおりです。

- Oracleのパフォーマンス改善方法
- パフォーマンスの緊急の問題に対処する方法

Oracleのパフォーマンス改善方法

Oracleのパフォーマンス方法論は、Oracleデータベースのパフォーマンス問題の識別に役立ちます。これには、ボトルネックの識別とその修正が含まれます。システムの変更は、ボトルネックの存在を確認した後にのみ行うことをお薦めします。

パフォーマンスの改善は、本質的に反復的なプロセスです。このため、最初のボトルネックを取り除いても、別のボトルネックが明らかになる可能性があり、パフォーマンスがすぐには改善されないことがあります。また、シリアライズ・ポイントが効率の悪い共有メカニズムに移動したことによって、パフォーマンスが低下する場合もあります。経験を重ね、ボトルネックの除去方法に厳密に従うことにより、アプリケーションをデバッグしてスケーラブルにすることができます。

パフォーマンスの問題は、通常、スループットの不足または許容範囲を超えたユーザー/ジョブ・レスポンス時間(あるいはその両方)が原因で発生します。問題は、アプリケーション・モジュール間でローカルに発生する場合も、システム全体にわたる場合もあります。

データベース統計やオペレーティング・システム統計を調べる前に、システムで最も重要なコンポーネントからのフィードバックを取得することが重要です。つまり、システムのユーザーと、アプリケーション・コストを最終的に負担する人々からのフィードバックです。ユーザーからの典型的なフィードバックには、次のような報告が含まれます。

- オンライン・パフォーマンスがあまりに低いため、部下が業務を遂行できない。
- 請求作業にかかる時間が長すぎる。
- Webトラフィックが多くなると、レスポンス時間が許容できないほど遅くなり、このままでは顧客を失ってしまう。
- 現在、1日に5000件の取引を行っているが、システムが限界を超えている。来月は全ユーザーにロールアウトする予定で、取引数は4倍になる見込みである。

このような率直なフィードバックがあると、パフォーマンスの改善作業を成功させる重要な要因を容易に設定できます。パフォーマンスの目標およびパフォーマンス・エンジニアにとっての最終基準を決定することで、パフォーマンス・プロセスの管理がすべてのレベルで簡潔になり円滑に運びます。このような重要な成功要因は、システム統計ではなく現実的なビジネス目標の観点から定義した方が効果的です。

前述した典型的なユーザー・フィードバックに対する実際のビジネス目標を、次にいくつか示します。

- 請求作業は、3時間で1,000,000件を処理する必要がある。
- Webサイトのピーク時に、1回のページ・リフレッシュのレスポンス時間は5秒以内である必要がある。
- システムでは、8時間で25,000件の取引を処理可能にする。

最終的な満足度は、システム・パフォーマンスに関するユーザーの認識で判断されます。パフォーマンス・エンジニアの役割は、パフォーマンスを低下させるボトルネックを取り除くことです。ボトルネックの原因としては、限られた共有リソースの非効率的な使用、

またはシリアライズの原因となる共有リソースの不適切な使用が考えられます。すべての共有リソースにはかぎりがあるため、パフォーマンス・エンジニアの目標は、共有リソースを効率的に活用して、ビジネス処理件数を最大にすることです。高いレベルでは、データベース・サーバー全体を共有リソースとみなすことができます。逆に、低いレベルでは、1台のCPUやディスクを共有リソースとみなすことができます。

Oracleのパフォーマンス改善方法は、パフォーマンス目標を達成するまで、または目標の達成が不可能と判断されるまで適用できます。これは非常に反復的なプロセスです。必然的に、データベースのパフォーマンスにほとんど影響のない調査も発生します。重大なボトルネックを正確かつ迅速に特定するスキルを開発するには、時間と経験が必要です。ただし、利用できるデータや統計を軽視すると、たとえ経験が豊かな技術者でもその経験が裏目に出ることがあります。このような技術者は、根拠のない思い込みと慣習でデータベースをチューニングしようとします。これは、データベースのチューニング方法としては非常に危険でコストもかかり、成功するとは考えられません。

自動データベース診断モニター(ADDM)は、パフォーマンス改善方法の各部を実装し、統計を分析して、重大なパフォーマンスの問題の自動診断を提供します。ADDMを使用すると、システム・パフォーマンスの改善に伴う所要時間を大幅に短縮できます。

システムは多様かつ複雑であるため、パフォーマンス分析における絶対的な規則は不可能です。本質的に、Oracleのパフォーマンス改善方法は、作業の方法を定義するものであり、確定した一連の法則を定義するものではありません。ボトルネックの検出における唯一の法則は、法則がないということです。優れたパフォーマンス・エンジニアは、提供されたデータを利用し、様々な角度から検討してパフォーマンスの問題を判断します。

Oracleのパフォーマンス改善方法のステップ

1. 次に示す初期標準チェックを実行します。
    a. ユーザーから率直なフィードバックを取得します。パフォーマンス・プロジェクトの適用範囲、最終的なパフォーマンス目標、今後のパフォーマンス目標を決定します。このプロセスは、今後の容量計画にとってのキーです。
    b. パフォーマンスが良いときと悪いときの両方で、オペレーティング・システム、データベースおよびアプリケーションの統計をフルセットでシステムから取得します。すべての統計を取得できない場合は、可能な範囲で取得します。使用できる統計がないのは、犯罪捜査で証拠がないのと同じです。証拠がないと、捜査が困難になり時間もかかります。
    c. ユーザー・パフォーマンスに関係のあるすべてのコンピュータのオペレーティング・システムが健全であることをチェックします。オペレーティング・システムの健全性をチェックすることにより、フルに使用されているハードウェアやオペレーティング・システムのリソースを探します。過剰使用のリソースを症状としてリストし、後で分析します。さらに、すべてのハードウェアでエラーや診断症状がないことをチェックします。
2. Oracle Databaseで最もよく見られる誤りの上位10項目をチェックし、それらが問題であるかどうかを判断します。問題となる可能性がある場合は、症状としてリストし、後で分析します。これらの項目をリストに含めるのは、問題となる可能性が高いためです。ADDMでは、これらの上位10項目のうち9項目が自動的に検出されレポートされます。
3. 症状を手がかりにして、システムで発生している状況の概念モデルを作成し、パフォーマンスの問題の原因を把握します。「パフォーマンスを概念的にモデル化する際の意思決定プロセスの例」を参照してください。
4. 一連の修正処理とシステムに対して予想される動作を提示し、アプリケーションに対して最も有効な処理から順に適用します。ADDMにより、各推奨事項と予測されるメリットが生成されます。パフォーマンス改善作業の原則は、変更は一度に1つのみとし、変更前後の差異を測定することです。ただし、システム停止時間に制約があり、この方法を忠

実に実行できないことがあります。一度に複数の変更を適用する場合は、それぞれの変更を切り離して、それぞれの変更の効果を個別に検証できるようにします。

5. 変更により期待された効果があることを検証し、ユーザーがパフォーマンスの改善を認識したかどうかを確認します。ユーザーが認識しない場合は、さらにボトルネックを探し、アプリケーションをより正確に把握できるまで、概念モデルの微調整を続けます。
6. パフォーマンス目標が達成されるまで、または他の制約によって達成が不可能になるまで、前述の最後の3つのステップを繰り返します。

この方法によって、最大のボトルネックが特定され、パフォーマンスの改善に対する客観的アプローチが使用されます。重要なことは、アプリケーションの効率を高め、リソース不足とボトルネックを取り除くことにより、パフォーマンスを大幅に改善することです。このプロセスでは、インスタンスのチューニングにより最低限のパフォーマンスの向上(10%未満)が期待でき、アプリケーションの効率の悪さを切り離すことで大幅なパフォーマンスの向上(100%以上)が期待できます。

パフォーマンスを概念的にモデル化する際の意思決定プロセスの例

概念的なモデル化は、ほとんど決定論的なプロセスです。しかし、パフォーマンス・チューニングの経験を積むと、実際に規則が存在しないことの良さに気がつきます。統計を解析して最適な意思決定を行うには、柔軟で機敏なアプローチが必要になります。

迅速かつ簡単なパフォーマンス・チューニングへのアプローチには、ADDMを使用します。ADDMを使用すると、Oracleシステムが自動的に監視され、パフォーマンスの問題が発生した場合は解決のための推奨事項が提供されます。たとえば、データベース管理者がユーザーからシステム・パフォーマンスが低下したという苦情を受け取ったとします。データベース管理者は、ただ最新のADDMレポートを調べ、問題を解決するにはどの推奨事項を実装すればよいかを確認します。

次のステップは、パフォーマンス・エンジニアが自動診断機能を使用せずにボトルネックを調べる方法を示しています。これらのステップは、あくまでも手動プロセスのガイドラインです。パフォーマンス・エンジニアは、経験を積みながら、それを関連するステップに追加していきます。この分析では、オペレーティング・システム統計とデータベース統計が収集されていることが前提です。

1. 負荷がまったくないか少ないコンピュータで、シングル・ユーザーに対するレスポンス時間またはバッチ実行時間は許容できる範囲ですか。

   許容範囲外の場合は、アプリケーションのコードまたは設計が最適でないと考えられます。また、これは、システム・リソースを共有するマルチ・ユーザーの状況では、まったく受け入れられません。このような場合は、アプリケーションの内部統計を取得し、SQLトレースとSQL計画の情報を取得します。開発者とともに、データ、索引およびトランザクションのSQL設計に関する問題を調査し、バッチ処理およびバックグラウンド処理の遅延の可能性について調査します。

2. CPUがすべて使用されていますか。

   カーネル使用率が40%を超えた場合は、ネットワーク転送、ページング、スワッピングまたはプロセス・スラッシングについてオペレーティング・システムを調べます。引き続き、ユーザー領域でのCPU使用率をチェックし、CPUを消費するデータベース以外のジョブ(バックアップ、ファイル変換、印刷キューなど)がシステム上で発生して、共有CPUリソース量を制限していないかどうかを確認します。データベースがCPUのほとんどを使用していることを確認した後、CPU使用率が上位のSQLを調べます。これらの文が、今後の分析の基礎になります。SQLおよびSQLを送信するトランザクションが、最適に実行されているかどうかをチェックします。Oracle Databaseでは、V$SQLおよびV$SQLSTATSにCPU統計が示されます。

関連項目:

V$SQLおよびV$SQLSTATSの詳細は、Oracle Databaseリファレンスを参照してください

アプリケーションが最適で、SQLが効率的に実行されている場合は、一部の作業をピーク時以外に再スケジュールするか、大型のコンピュータの使用を検討してください。

3. この段階で、CPUリソースが完全には使用されていないのに、システム・パフォーマンスがまだ不十分ですか。

   不十分な場合は、サーバー内にシリアライズされた、スケーラブルでない動作があります。サーバーからWAIT_EVENTS統計を取得し、最大のシリアライズ・ポイントを確認します。シリアライズ・ポイントがない場合、問題はデータベースの外にある可能性が高く、これを重点的に調査する必要があります。WAIT_EVENTSをなくすには、アプリケーションSQLを修正し、データベース・パラメータをチューニングします。このプロセスは非常に反復的で、WAIT_EVENTSを系統的にドリルダウンして、シリアライズ・ポイントを取り除く技術を必要とします。

Oracleシステムにおける誤りの上位10項目

この項では、Oracleデータベースで最もよく見受けられる誤りをリストします。Oracleのパフォーマンス改善方法に従うことで、これらの誤りはすべて回避できます。これらの誤りをシステム内で見つけた場合は、アプリケーション内でパフォーマンス改善の効果のある箇所を再設計します。

1. 不適切な接続管理

   アプリケーションで、データベースとの対話ごとに接続と切断が行われることがあります。この問題は、アプリケーション・サーバー内のステートレス・ミドルウェアで多く発生します。この問題がパフォーマンスに与える影響の大きさは2桁以上違い、まったくスケーラブルではありません。

2. カーソルと共有プールの不適切な使用

   カーソルを使用しないと、解析が繰り返し行われることになります。バインド変数を使用しないと、すべてのSQL文のハード解析が行われます。この問題がパフォーマンスに与える影響は甚だしく、まったくスケーラブルではありません。カーソルをオープンするバインド変数とともにカーソルを使用し、繰り返し実行します。動的SQLを生成するアプリケーションは疑ってみます。

3. 不適切なSQL

   不適切なSQLとは、アプリケーション要件よりも多くのリソースを使用するSQLです。たとえば、意思決定支援システム(DSS)による問合せが24時間以上実行されている場合や、オンライン・アプリケーションからの問合せに1分以上かかる場合などがあります。大量のシステム・リソースを消費するSQLを検査し、改善の可能性を調査します。ADDMは、高負荷のSQLを識別します。SQLチューニング・アドバイザにより、改善のための推奨事項が提供される可能性があります。

4. 標準でない初期化パラメータの使用

   このようなパラメータは、不適切なアドバイスや不正確な前提に基づいて実装されていることがあります。ほとんどのデータベースでは、基本パラメータ・セットのみを使用して、許容できるパフォーマンスを提供します。特に、ラッチに対するSPIN_COUNTに関連するパラメータとマニュアルに記載されていないオプティマイザ機能は、多くの問題の原因となり、このためにかなりの調査が必要になることがあります。

   また、初期化パラメータ・ファイルに設定したオプティマイザ用パラメータが、実証済の最適実行計画をオーバーライドしてしまう可能性があります。このため、スキーマ、スキーマ情報およびオプティマイザの設定は、グループとして管理して一貫性のあるパフォーマンスを維持します。

関連項目:

- 初期化パラメータおよびデータベース作成の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

5. 誤ったデータベースI/Oの取得

   多くのサイトが、使用可能ディスク上にデータベースを適切にレイアウトしていません。また、ディスクをI/O帯域幅ではなくディスク領域によって構成し、ディスク数を誤って指定しているサイトもあります。

6. オンラインREDOログの設定問題

   多くのサイトが、非常に少ないオンラインREDOログ・ファイルおよび小さいファイルを使用して運用されています。REDOログ・ファイルのサイズが小さいと、システム・チェックポイントがバッファ・キャッシュとI/Oシステムに高い負荷をかけ続けることになります。REDOログ・ファイルの数が少なすぎると、アーカイブが間に合わないので、データベースはアーカイバが追い付くまで待機します。

7. 空きリスト、空きリスト・グループ、トランザクション・スロット(INITRANS)またはロールバック・セグメントの不足による、バッファ・キャッシュ内でのデータ・ブロックのシリアライズ

   この問題は、INSERT操作が多いアプリケーション、ブロック・サイズを8KB以上に上げたアプリケーション、またはアクティブ・ユーザーの数が多くロールバック・セグメントの数がほとんどないアプリケーションで特によく見受けられます。自動セグメント領域管理(ASSM)と自動UNDO管理を使用して、この問題を解決します。

8. 時間のかかる全表スキャン

   大量の全表スキャンまたは対話型のオンライン操作での全表スキャンで時間がかかる場合は、トランザクション設計が不適切であるか、索引が欠落しているか、SQL最適化が不十分です。時間のかかる表スキャンは、本質的にI/O集中型で、スケーラブルではありません。

9. 大量の再帰的(SYS)SQL

   SYSによって実行される大量の再帰的SQLは、エクステント割当てなどの領域管理操作が発生していることを示します。これはスケーラブルではなく、ユーザーのレスポンス時間に影響を与えます。ローカル管理表領域を使用して、エクステント割当てによる再帰的SQLを減らしてください。別のユーザーIDで実行される再帰的SQLは、SQLおよびPL/SQLである可能性が高く、問題ではありません。

10. デプロイメントおよび移行時のエラー

    アプリケーションが必要以上にリソースを使用する場合、その原因の多くは、表を所有するスキーマが開発環境または古い実装から正しく移行されていないことにあります。この例としては、索引の欠落や不正確な統計があります。このようなエラーは、不適切な実行計画や、ユーザー対話パフォーマンスの低下につながります。パフォーマンスがわかっているアプリケーションを移行するときは、DBMS_STATSパッケージを使用してスキーマ統計をエクスポートし、プラン・スタビリティを維持します。

    ADDMでは、この種のエラーが直接検出されることはありませんが、それによる高負荷SQLは明らかになります。

パフォーマンスの緊急の問題に対処する方法

この項では、パフォーマンスに関する緊急事態に対処する方法について説明します。アプリケーションのパフォーマンスを確立および改善する方法論がすでに存在すると推定されます。しかし、緊急時には、システム・コンポーネントの変化によって、信頼性の高い予測可能なシステムが予測不可能でユーザー・リクエストを満たせないシステムに変化します。

このような場合、パフォーマンス・エンジニアは、何が変化したかをすばやく判断し、適切な処理を行って、通常のサービスをできるだけ早く再開する必要があります。多くの場合、緊急の処理が必要であり、パフォーマンス改善方法を厳密に実行することは現実的ではありません。

パフォーマンス・エンジニアは、パフォーマンスの問題にただちに対処した後、十分なデバッグ情報を収集して、その問題をより明らかに把握するか、それができない場合は、少なくとも同じ問題が再発しないようにする必要があります。

緊急のパフォーマンスの問題をデバッグする方法は、このマニュアルの前の章で説明したパフォーマンス改善方法と同じです。ただし、急を要するという問題の性質上、様々な段階で簡便法が使用されます。デバッグ・プロセスの進行中に見つかった事実を詳細にノートしノートにとっておくことが、後で行う分析と修正作業の正当化には不可欠です。これは、医師が患者の容態を克明にノートにとって、将来の参照資料として役立てるのと同じです。

パフォーマンスの緊急の問題に対処する方法のステップ

パフォーマンスの緊急の問題に対処する方法は、次のとおりです。

1. パフォーマンスの問題を調査し、問題の症状を収集します。このプロセスには、次の作業が含まれます。
    - システムのパフォーマンスが低下した状況について、ユーザー・フィードバックを収集します。問題がスループットかレスポンス時間かを調べます。
    - 「パフォーマンスが正常だった時点から何が変化しましたか」という質問をユーザーにします。この質問に対する答が、問題の手がかりになることがあります。ただし、状況が悪化する中で先入観のない答を得るのは難しいことがあります。問題の前と後で取得された参照データ(収集済の統計やログ・ファイル)を見つけるようにします。
    - 自動チューニング機能を使用して問題を診断および監視します。また、上位のSQLおよびセッションの識別には、Enterprise Manager Cloud Control (Cloud Control)のパフォーマンス機能を使用できます。
2. アプリケーション・システムの全コンポーネントのハードウェア使用が健全であることをチェックします。CPU使用率が最も高いコンポーネントをチェックし、システムの全コンポーネントについて、ディスクとメモリーの使用およびネットワーク・パフォーマンスをチェックします。このプロセスによって、問題の原因となっている層をすばやく特定できます。問題がアプリケーション内で発生している場合は、分析の重点をアプリケーションのデバッグへ移します。それ以外の場合は、データベース・サーバーの分析に進みます。
3. データベース・サーバーがCPU上で制約を受けているかどうか、待機イベントで待機し続けているのかを判断します。
   データベース・サーバーがCPU上で制約を受けている場合は、次の点を調べます。
    - オペレーティング・システム・レベルで大量のCPUおよびデータベースを消費しているセッション
      (V$SESS_TIME_MODELでデータベースのCPU使用率を調べます。)
    - データベース・レベルで多数のバッファ読取りを実行しているセッションまたは文(V$SESSTATおよびV$SQLSTATSを調べます。)
    - 最適ではないSQLの実行の原因となった実行計画の変更(特定が困難な場合もあります。)
    - 初期化パラメータの誤設定
    - コードの変更または全コンポーネントのアップグレードによるアルゴリズムの問題

   データベース・セッションがイベントを待機している場合は、V$SESSION_WAITにリストされている待機イベントに従って、シリアライズの原因を判断します。V$ACTIVE_SESSION_HISTORYビューには、サンプリングされたセッション・アクティビティの履歴が含まれており、問題が終了してシステムが通常の操作に戻った後も診断に使用できます。ライブラリ・キャッ

シュに大量の競合がある場合は、データベースにログオンしたりSQLを送信するのが不可能なことがあります。このような場合は、履歴データを使用して、そのラッチで競合が突然発生した原因を判断します。ほとんどがI/O待機の場合は、V$ACTIVE_SESSION_HISTORYを調べて、すべての入出力を実行するセッションで実行中のSQLを判断します。

4. 緊急時の処理を適用して、システムを安定化します。これには、アプリケーションの一部をオフラインにする処理や、システムに適用できるワークロードを制限する処理が含まれます。また、システムの再起動や処理中のジョブの停止も含まれることがあります。これらの処理は、当然、サービス・レベルに影響を与えることになります。
5. システムが安定していることを確認します。システムを変更したり制限した場合は、システムが安定したことを確認し、データベースに関する参照統計情報を収集します。次に、このマニュアルの前の章で説明した厳密なパフォーマンス改善方法に従って、すべての機能とユーザーをシステムに戻します。このプロセスを完了するには、アプリケーションの大幅な再設計が必要になることがあります。

4 パフォーマンスを考慮したデータベースの構成

この章では、パフォーマンスを考慮してデータベースを構成するための、オラクル社の方法論の概要を説明しています。パフォーマンスの修正はOracle Databaseに対して継続的に実行できますが、データベースの初期構成を適切に行うことにより、多大な利益を得ることができます。

この章の構成は、次のとおりです。

- 初期インスタンス構成のパフォーマンスの考慮事項
- 最適なパフォーマンスを得る表の作成とメンテナンス
- 共有サーバーのパフォーマンスの考慮事項
- 事前起動プロセスによるクライアント接続のパフォーマンスの向上

初期インスタンス構成のパフォーマンスの考慮事項

データベースに重要なパフォーマンス面の影響を与えるデータベース・インスタンスの初期構成オプションは次のとおりです。

- 初期化パラメータ
- UNDO領域
- REDOログ・ファイル
- 表領域

ノート:

Database Configuration Assistant (DBCA)を使用してデータベースを作成する場合、提供されるシード・データベースには必要な基本的初期化パラメータが組み込まれており、このドキュメントで説明するパフォーマンス推奨事項を満たしています。

関連項目:

- Database Configuration Assistantを使用したデータベースの作成方法を学習するには、『Oracle Database管理者ガイド』を参照してください。
- SQL文を使用したデータベースの作成方法を学習するには、『Oracle Database管理者ガイド』を参照してください。

初期化パラメータ

実行中のOracleデータベース・インスタンスは、初期化パラメータ・ファイルで設定される初期化パラメータを使用して構成されます。これらのパラメータは、パフォーマンスへの影響を含め、実行中のインスタンスの動作に影響を与えます。一般的に、関連する設定がほとんどない非常に単純な初期化ファイルで、ほとんどの状況に対応できます。次の表に示す少数のパラメータを除き、初期化ファイルはパフォーマンス・チューニングを行う最初の場所ではありません。

次の表に、最小初期化ファイルに必要なパラメータを示します。これらのパラメータは必須ですが、パフォーマンスに影響を与えません。

表4-1 パフォーマンスに影響を与えない必要な初期化パラメータ

パラメータ	説明
DB_NAME	データベースの名前。これは、環境変数 ORACLE_SID に一致する必要があります。
DB_DOMAIN	インターネット・ドット表記法による、データベースの場所。
OPEN_CURSORS	1 セッション当たりのカーソル(アクティブな SQL 文)の最大数に対する制限。設定はアプリケーションにより異なり、推奨値は 500 です。
CONTROL_FILES	異なるディスク・ドライブ上に少なくとも 2 つのファイルを含めるように設定して、制御ファイルの消失による障害を防ぎます。
DB_FILES	データベースに割り当てることのできるファイルの最大数に設定します。

次の表に、パフォーマンスの考慮事項として設定する、最も重要なパラメータを示します。

表4-2 パフォーマンスに影響を与える重要な初期化パラメータ

パラメータ	説明
COMPATIBLE	互換性を維持しておく Oracle データベースのリリースを指定します。このパラメータを使用すると、新しい機能を自分の環境でテストせずに、ただちに本番システムで新しいリリースで改善された機能を利用できるようになります。アプリケーションが Oracle Database の特定のリリース用に設計されていて、実際にはそれより後のリリースをインストールする場合は、このパラメータをアプリケーション設計時のリリースに設定してください。
DB_BLOCK_SIZE	データベース・ファイルに格納され、SGA にキャッシュされる Oracle データベース・ブロックのサイズを設定します。値の範囲はオペレーティング・システムにより異なりますが、一般的にトランザクション処理システムの場合は 8192 で、データベース・ウェアハウス・システムの場合はさらに大きい値となります。
SGA_TARGET	すべての SGA コンポーネントのサイズ合計を指定します。SGA_TARGET が指定されている場合、バッファ・キャッシュ(DB_CACHE_SIZE)、Java プール(JAVA_POOL_SIZE)、ラージ・プール(LARGE_POOL_SIZE)および共有プール(SHARED_POOL_SIZE)のメモリー・プールは、自動的にサイズ設定されます。
PGA_AGGREGATE_TARGET	インスタンスに添付されたすべてのサーバー・プロセスに利用できるターゲット集計 PGA

パラメータ	説明
	メモリーを指定します。
PROCESSES	そのインスタンスで起動できるプロセスの最大数を設定します。このパラメータは、他のパラメータ値の多くがこのパラメータから導出されるため、設定する最も重要なプライマリ・パラメータとなります。
SESSIONS	これは、デフォルトで PROCESSES の値から設定されます。ただし、共有サーバーを使用する場合は、導出された値では不十分である可能性があります。
UNDO_MANAGEMENT	データベースで使用する UNDO 領域管理モードを指定します。デフォルトは AUTO です。指定しない場合、AUTO が使用されます。
UNDO_TABLESPACE	インスタンス起動時に使用される UNDO 表領域を指定します。

関連項目:

これらの初期化パラメータの詳細は、次のガイドを参照してください。

- 『Oracle Database管理者ガイド』
- Oracle Databaseリファレンス

UNDO領域

データベースは、読取り一貫性、リカバリおよびロールバック文で使用するデータを格納するために、UNDO領域を使用します。このデータは1つ以上のUNDO表領域に格納されます。Database Configuration Assistant(DBCA)を使用してデータベースを作成すると、UNDO表領域が自動的に作成されます。UNDO表領域を手動で作成するには、CREATE DATABASE文にUNDO TABLESPACE句を追加します。

UNDOデータを自動的に管理するため、Oracle Databaseでは、UNDOセグメントを透過的に作成および管理する自動UNDO管理を使用します。自動UNDO管理を有効にするには、UNDO_MANAGEMENT初期化パラメータをAUTO(デフォルト設定)に設定します。指定しない場合、UNDO_MANAGEMENT初期化パラメータでは、AUTO設定が使用されます。自動UNDO管理の使用をお薦めするのは、データベース管理が大幅に簡素化され、UNDO(ロールバック)セグメントを手動でチューニングする必要がないためです。ロールバック・セグメントを使用する手動UNDO管理は、下位互換性のためにサポートされています。

V$UNDOSTATビューには、UNDO領域を監視およびチューニングするための統計が含まれています。このビューを使用して、現行のワークロードに必要なUNDO領域の大きさをより的確に見積ることができます。この情報を使用して、UNDO使用量を簡単にチューニングすることもできます。V$ROLLSTATビューには、UNDO表領域内のUNDOセグメントの動作に関する情報が含まれています。

関連項目:

- UNDO管理アドバイザについて学習するには、『Oracle Database 2日でデータベース管理者』と、Oracle Enterprise Manager Cloud Control (Cloud Control)のオンライン・ヘルプを参照してください
- 自動UNDO管理を使用したUNDO領域の管理の詳細は、『Oracle Database管理者ガイド』を参照してください。
- V$ROLLSTATビューの詳細は、Oracle Databaseリファレンスを参照してください。
- V$UNDOSTATビューの詳細は、Oracle Databaseリファレンスを参照してください。

REDOログ・ファイル

REDOログ・ファイルのサイズはパフォーマンスに影響を与えます。これは、データベースのライター・プロセスおよびアーカイバ・プロセスの動作がREDOログのサイズによって変わるためです。一般に、REDOログ・ファイルが大きければ、パフォーマンスは向上します。ログ・ファイルのサイズが小さいと、チェックポイント・アクティビティが増加し、パフォーマンスが低下します。

REDOログ・ファイルのサイズはLGWRのパフォーマンスに影響を与えませんが、DBWRおよびチェックポイントの動作に影響を与える可能性があります。 チェックポイントの頻度には、ログ・ファイルのサイズやFAST_START_MTTR_TARGET初期化パラメータの設定など、複数の要因が影響します。インスタンスのリカバリ時間を制限するためにFAST_START_MTTR_TARGETパラメータが設定されている場合、Oracle Databaseは自動的にチェックポイントを必要に応じて頻繁に試みます。この場合は、ログ・ファイルが小さすぎることによるチェックポイントの追加発生を防ぐために、ログ・ファイルを十分なサイズに設定する必要があります。最適なサイズは、V$INSTANCE_RECOVERYビューからOPTIMAL_LOGFILE_SIZE列を問い合せることで取得できます。また、Oracle Enterprise Manager Cloud Control (Cloud Control)の「REDOログ・グループ」ページでも、サイズ設定のアドバイスを取得できます。

REDOログ・ファイルに特定のサイズを推奨できない場合もありますが、100MBから数GBの範囲内にあるREDOログ・ファイルが妥当であると考えられます。システムが生成するREDOの量に従って、オンラインREDOログ・ファイルをサイズ設定します。およその目安として、多くても20分ごとに1回ログ・ファイルを切り替えます。

関連項目:

REDOログの管理の詳細は、『Oracle Database管理者ガイド』を参照してください。

表領域

Database Configuration Assistant(DBCA)を使用してデータベースを作成する場合、シード・データベースには必要な表領域すべてが自動的に組み込まれます。DBCAを使用しないように選択した場合は、データベースの作成後に追加表領域を作成する必要があります。

すべてのデータベースには、SYSTEMおよびSYSAUX表領域に加えて複数の表領域が必要です。これらの追加表領域は次のとおりです。

- ソートなどの操作に使用される一時表領域
- 読取り一貫性、リカバリおよびUNDO文に関する情報を格納するためのUNDO表領域

- アプリケーションで使用する少なくとも1つの表領域(ほとんどの場合、アプリケーションには複数の表領域が必要です)

データ・ファイルが多数ある非常に大規模な表領域の場合は、複数のALTER TABLESPACE ... ADD DATAFILE文を同時に実行できます。表領域の作成時、表領域を構成するデータファイルは特殊な空のブロック・イメージで初期化されます。一時ファイルは初期化されません。

Oracle Databaseでは、この初期化により、すべてのデータファイル全体に書込みが行えるようになりますが、この処理をシリアルに実行すると時間がかかります。したがって、複数のCREATE TABLESPACE文を同時に実行して、表領域の作成を高速化します。永続的な表の場合、表領域作成時にローカルまたはグローバルのいずれかのエクステント管理を選択するかで、パフォーマンスに大きな影響を与える可能性があります。読取り操作と比べて、普通または大規模な、挿入、変更または削除操作を行う永続的な表領域の場合は、ローカル・エクステント管理を選択します。

永続表領域 - 自動セグメント領域管理

永続的な表領域では、自動セグメント領域管理を使用することをお薦めします。ビットマップ表領域と呼ばれることも多い永続表領域は、ビットマップ・セグメント領域管理によってローカルに管理される表領域です。

関連項目:

- 空き領域管理の詳細は、『Oracle Database概要』を参照してください。
- 表領域の自動セグメント領域管理の作成および使用の詳細は、『Oracle Database管理者ガイド』を参照してください。

一時表領域

一時表領域を正しく構成すると、ディスク・ソートのパフォーマンスの最適化に役立ちます。一時表領域はディクショナリ管理を行うこともローカル管理を行うこともできます。UNIFORMエクステント・サイズが1MBの、ローカル管理の一時表領域を使用することをお薦めします。

一時表領域アクティビティを監視して、一時セグメントに割り当てるエクステントの数をチェックする必要があります。たとえば、多数のユーザーが一時表を同時に使用しているような状況で、アプリケーションで一時表が頻繁に使用される場合は、1回の使用につき1つ以上のエクステントが必要になるため、エクステント・サイズを256KBなどの小さい値に設定できます。一時表領域はすべて均一サイズのローカル管理エクステントで作成されるため、EXTENT MANAGEMENT LOCAL句は一時表領域ではオプションです。SIZEのデフォルトは1MBです。

関連項目:

- 一時表領域の管理の詳細は、『Oracle Database管理者ガイド』を参照してください。
- 一時表領域の詳細は、『Oracle Database概要』を参照してください。
- TEMPORARY句を含むCREATE文およびALTER TABLESPACE文の使用の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

最適なパフォーマンスを得る表の作成とメンテナンス

アプリケーションをインストールする際、最初のステップで、必要なすべての表および索引を作成します。表のようなセグメントを作成すると、データ用の領域がデータベースで割り当てられます。後続のデータベース操作によってデータ容量が増大し、割り当てられた領域を上回るようになると、Oracle Databaseはそのセグメントを拡張します。

表および索引を作成する際には、次の点に注意してください。

- 表領域の自動セグメント領域管理の指定

  この方法では、最高のパフォーマンスが得られるよう、Oracle Databaseで自動的にセグメント領域が管理されます。

- 記憶領域オプションの慎重な設定

  アプリケーションでは、表または索引の用途によって記憶領域オプションを慎重に設定する必要があります。この設定には、PCTFREEの値の設定が含まれます。自動セグメント領域管理を使用すると、PCTUSEDを指定する必要がありません。

ノート:

空きリストの使用はお薦めしません。自動セグメント領域管理を使用するには、ローカル管理表領域を作成し、セグメント領域管理の句を AUTO に設定します。

表の圧縮

ヒープ構成表は、どのような種類のアプリケーションにも透過的な圧縮形式で格納できます。データベース・ブロック内の圧縮データは自己完結型です。つまり、ブロック内の未圧縮データの再作成に必要なすべての情報がそのブロック内にあります。バッファ・キャッシュでも、ブロックは圧縮されています。表圧縮により、ディスク記憶域のみでなく、メモリー使用量、特にバッファ・キャッシュ要件も削減されます。表へのアクセスに必要なI/O操作の量が削減され、バッファ・キャッシュ・ヒットの確率が高まることで、パフォーマンスの向上が実現します。

Oracle Databaseの拡張圧縮オプションを使用すると、データ・ウェアハウスおよびOLTPアプリケーションを含め、どのタイプのアプリケーションもワークロードのパフォーマンスが向上し、データベースに必要なディスク領域を削減します。拡張圧縮機能は、構造化データ、非構造化データ、バックアップ・データおよびネットワーク・データを含め、すべてのタイプのデータに使用できます。

関連項目:

表の圧縮の詳細は、『Oracle Database管理者ガイド』を参照してください

圧縮係数の見積り

表圧縮は、個々のブロック内の列値の繰返しを解消することによって機能します。ブロック内のすべての行と列の重複値は、ブロックの先頭にある、そのブロックのシンボル表と呼ばれるものに一度格納されます。こうした値の出現箇所はすべて、シンボル表への簡単な参照で置き換えられます。繰返しの値が多いブロックほど圧縮率が高くなります。

大規模な表を圧縮する前に、予測される圧縮係数を見積る必要があります。圧縮係数は、圧縮されていない形式で情報を

格納するために必要なブロック数を、圧縮記憶域に必要なブロック数で除算した値として定義されます。圧縮される表を代表する少数のデータ・ブロックをサンプリングし、圧縮されていない場合と圧縮された場合の各ブロックの平均レコード数を比較することで、圧縮係数を見積ることができます。これまでの経験では、およそ1000のデータ・ブロックを使用することで、きわめて正確な圧縮係数の見積りが可能であることがわかっています。サンプリングのブロック数が多いほど、見積り結果が正確になる点に注意してください。

チューニングによる圧縮率向上

Oracle Databaseでは、多くの場合、特別なチューニングなしで、優れた圧縮係数を達成しています。DBAまたはアプリケーション開発者は、圧縮が行われる際にレコードを再構成することによって、圧縮係数のチューニングを試行できます。チューニングを行うと、圧縮係数が若干改善されるケースもあれば大きく改善されるケースもあります。

圧縮係数を高めるには、データ・ブロック内の値の繰返しの可能性を高くする必要があります。達成できる圧縮係数は、特定の列または列のペアのカーディナリティ(列値の繰返しの可能性を表す)およびこれらの列の行の平均長さによって異なります。表圧縮では、単一列の重複値が圧縮されるだけでなく、可能な場合は、複数列値のペアの使用が試行されます。データ配分について熟知していない場合、最適な順序の予測は非常に困難です。

属性クラスタ化表の使用

属性クラスタ化表は、ユーザー指定のクラスタリング・ディレクティブに基づいて近接度の近いデータをディスク上に格納する、ヒープ構成表です。表内に格納されるデータの順序付けが特定の列に基づくか、複数列のI/O削減を許容する特殊なアルゴリズムに基づくかはディレクティブによって判断されます。属性クラスタリングは一括挿入操作(INSERT/*+APPEND*/またはALTER TABLE ... MOVE PARTITIONコマンドなど)のみに使用可能で、従来のDMLについては無視されます。

ゾーン・マップとともに物理I/Oを削減することで、属性クラスタ化表の使用は、表スキャンのI/Oコストを大幅に削減できます。さらに、同じ値がディスク上で近い場所にあるほどデータはより簡単に圧縮できるため、データ圧縮も改善できます。

関連項目:

- 属性クラスタ化表の詳細は、『Oracle Database概要』を参照してください
- 属性クラスタ化表の使用の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください

未使用領域の解放

一般に、時間経過とともにセグメント領域が断片化されたり、更新および削除操作の結果としてセグメントに多数の空き領域が生じます。このようにして粗密に移入されたオブジェクトは、問合せおよびDML操作中にパフォーマンスを低下させる可能性があります。オブジェクトに解放可能な領域がある場合は、セグメントを圧縮して縮小するか、またはセグメントの終わりにある未使用領域を割当て解除できます。

Oracle Databaseにはセグメント・アドバイザが用意されており、オブジェクト内の領域の断片化レベルに基づいて、オブジェクトに解放可能な領域があるかどうかのアドバイスを提供します。

関連項目:

セグメント・アドバイザおよびスキーマ・オブジェクトの領域の管理の詳細は、Oracle Database管理者ガイドを参照してください

データの索引付け

最も効率的に索引を作成するタイミングは、データのロード後です。この方法では、領域管理が簡単になり、行が挿入されるたびに索引がメンテナンスされることもありません。この技術は、SQL*Loaderにより自動的に使用されますが、他の方法で初期データをロードする場合は、手動で索引を作成する必要があります。また、CREATE INDEX文のPARALLEL句を使用して、索引作成をパラレルで実行することもできます。ただし、SQL*Loaderでは索引作成をパラレル化できないため、データのロード後に索引を手動でパラレルに作成する必要があります。

関連項目:

SQL*Loaderの詳細は、『Oracle Databaseユーティリティ』を参照してください。

ソート用データのメモリーの指定

データを含む表での索引の作成中に、データをソートする必要があります。このソートは、すべての使用可能なメモリーをソートに使用する場合に最も高速に行われます。PGA_AGGREGATE_TARGET初期化パラメータを設定して、SQL作業領域の自動サイズ指定を有効にすることをお薦めします。

関連項目:

- PGAメモリー管理の詳細は、「プログラム・グローバル領域のチューニング」を参照してください
- PGA_AGGREGATE_TARGET初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

共有サーバーのパフォーマンスの考慮事項

共有サーバーを使用すると、プロセス数と、データベース・ホストでのメモリー使用量が削減されます。共有サーバーは、多数のOLTPユーザーが断続的なトランザクションを実行するデータベースに有効です。

また、データベースへの単位時間当たりの接続数が高いシステムでは、一般的に専用サーバーよりも共有サーバーを使用する方が適しています。共有サーバーでは、接続リクエストを受信する際に、ディスパッチャを使用して同時接続リクエストを処理できます。しかし、専用サーバーの場合は、接続固有の専用サーバーが、接続リクエストごとに順次初期化されます。

共有サーバー・アーキテクチャを使用すると、特定のデータベース機能のパフォーマンスが向上しますが、パフォーマンスがいくらか低下する機能もあります。たとえば、パラレル実行がアクティブな場合、セッションが別の共有サーバーに移行できないことがあります。

クライアントからのリクエストが処理された後でもセッションを移行できない場合があります。これは、すべてのユーザー情報がUGAに格納されているとは限らないためです。サーバーがクライアントからのリクエストを処理するとしても、UGAに格納されていなかっ

たユーザー状態にはアクセスできません。この状況を回避するために、個々の共有サーバーは、しばしばユーザー・セッションにバインドされた状態のままである必要があります。

関連項目:

- 共有サーバーの管理方法を学習するには、『Oracle Database管理者ガイド』を参照してください。
- 共有サーバー用のディスパッチャの構成方法を学習するには、『Oracle Database Net Services管理者ガイド』を参照してください。

ある種の機能を使用する場合、あるサーバーがセッションに長期間バインドされる可能性があるため、追加の共有サーバーを構成する必要が生じる場合があります。

この項では、Oracle Databaseのアーキテクチャで使用するプロセスの競合を低減する方法について説明します。

- ディスパッチャ固有のビューを使用する競合の識別および低減
- 共有サーバーの競合の識別

ディスパッチャ固有のビューを使用する競合の識別および低減

次のビューによってディスパッチャのパフォーマンス統計が提供されます。

- V$DISPATCHER: ディスパッチャ・プロセスの一般的な情報
- V$DISPATCHER_RATE: ディスパッチャ・プロセスの統計情報

V$DISPATCHER_RATEビューには、いくつかのカテゴリについてディスパッチャ統計の現在、平均および最大の値が含まれています。接頭辞CUR_が付いている統計は、現在のサンプルの統計です。接頭辞AVG_が付いている統計は、収集期間の開始後の統計の平均値です。接頭辞MAX_が付いている統計は、統計収集の開始後のこれらのカテゴリでの最大値です。

ディスパッチャのパフォーマンスを調べるには、V$DISPATCHER_RATEビューを問い合せて、最大値と現在の値を比較します。現在のシステム・スループットが適切なレスポンス時間を提供し、このビューの現在の値が平均値に近くかつ最大値を下回る場合、共有サーバー環境は最適にチューニングされていると考えられます。

現在の値と平均値が最大値を大きく下回る場合は、ディスパッチャ数の削減を検討してください。反対に、現在の値と平均値が最大値に近い場合は、ディスパッチャを追加する場合があります。システム使用率が低い期間と高い期間の両方でV$DISPATCHER_RATE統計を調べてみることを一般規則としてお薦めします。共有サーバーの負荷パターンを識別した後で、それに従ってパラメータを調整します。

必要に応じて、システムのストレス・テストを実行し、定期的にV$DISPATCHER_RATE統計をポーリングすることによってワークロードをシミュレートすることもできます。これらの統計の正しい解釈方法は、プラットフォームごとに異なります。アプリケーションの種類が変わると、V$DISPATCHER_RATEに記録される統計値も大幅に異なる可能性があります。

関連項目:

- V$DISPATCHERおよびV$DISPATCHER_RATEビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

ディスパッチャ・プロセスの競合の低減

競合を低減するには、次の点を考慮してください。

- ディスパッチャ・プロセスの追加

  ディスパッチャ・プロセスの総数は、MAX_DISPATCHERS初期化パラメータの値で制限されます。ディスパッチャ・プロセスを追加する前に、この値を増やす必要がある場合があります。

- 接続プーリングの有効化

  システム負荷が増加し、ディスパッチャ・スループットが最大限になった場合は、すぐにディスパッチャを追加することが必ずしも適切とはかぎりません。そのかわりに、接続プーリングによってより多くのユーザーをサポートできるようにディスパッチャを構成することを検討してください。

- セッション多重化の有効化

  多重化は、Connection Managerプロセスで、複数ユーザーから個別のディスパッチャへのネットワーク・セッションを確立およびメンテナンスする場合に使用されます。たとえば、いくつかのユーザー・プロセスがConnection Managerプロセスからの1つの接続を経由して1つのディスパッチャに接続できます。ディスパッチャ・プロセスが最大限に使用されるため、セッションの多重化は有効です。多重化は、ディスパッチャ間のデータベース・リンク・セッションを多重化する場合にも役立ちます。

関連項目:

- ディスパッチャ・プロセスの構成方法を学習するには、『Oracle Database管理者ガイド』を参照してください
- 接続ポーリングの構成方法を学習するには、『Oracle Database Net Services管理者ガイド』を参照してください。
- DISPATCHERSおよびMAX_DISPATCHERS初期化パラメータについて学習するには、『Oracle Databaseリファレンス』を参照してください

共有サーバーの競合の識別

リクエスト・キューにおいて待機時間が一貫して増加する場合、それは共有サーバーの競合を示します。待機時間のデータを調べるには、動的パフォーマンス・ビューV$QUEUEを使用します。このビューには、共有サーバーのリクエスト・キューのアクティビティを示す統計が含まれます。デフォルトでは、SYSユーザーと、SELECT ANY TABLEシステム権限を持っている他のユーザー(SYSTEMなど)のみがこのビューを使用できます。表4-3は、リクエストの待機時間とキュー内のリクエスト数を示す列のリストです。

表4-3 V$QUEUE内の待機時間およびリクエストの列

列	説明
WAIT	キューに存在していた全リクエストについての待機時間の合計(100 分の 1 秒単位)
TOTALQ	キューに存在していたリクエストの総数

アプリケーションの実行時に次のSQL文を発行して、この統計を数回監視してください。

```
SELECT DECODE(TOTALQ, 0, 'No Requests',
```

```
   WAIT/TOTALQ || ' HUNDREDTHS OF SECONDS') "AVERAGE WAIT TIME PER REQUESTS"
  FROM V$QUEUE
 WHERE TYPE = 'COMMON';
```

この問合せは、次のような計算結果を戻します。

```
AVERAGE WAIT TIME PER REQUEST
-----------------------------
.090909 HUNDREDTHS OF SECONDS
```

この結果から、リクエストは処理される前にキューにおいて平均0.09待機したことがわかります(単位は100分の1秒)。

また、次の問合せを発行することによって、現在実行中の共有サーバーの数が判断できます。

```
SELECT COUNT(*) "Shared Server Processes"
  FROM V$SHARED_SERVER
 WHERE STATUS != 'QUIT';
```

この問合せの結果を次に示します。

```
Shared Server Processes
-----------------------
10
```

共有サーバーでのリソース競合を検出した場合は、まず、共有プールとラージ・プールを調査し、これがメモリー競合でないことを確認します。パフォーマンスが改善されない場合は、共有サーバー・プロセス競合を低減するためにさらにリソースを作成してください。その場合は、オプションのサーバー・プロセス初期化パラメータを変更します。

- MAX_DISPATCHERS
- MAX_SHARED_SERVERS
- DISPATCHERS
- SHARED_SERVERS

関連項目:

共有サーバー・プロセスの初期化パラメータの設定方法を学習するには、『Oracle Database管理者ガイド』を参照してください。

事前起動プロセスによるクライアント接続のパフォーマンスの向上

Oracle Databaseでは、専用ブローカ接続モードが有効化されたとき、またはスレッド化実行モードが有効化されたときに、サーバー・プロセスのプールが事前起動されます。この場合、クライアントがデータベース接続をリクエストしたときは常に、プロセス・プールから既存のサーバー・プロセスへの専用の接続が取得されるため、クライアント接続の効率が向上します。

V$PROCESS_POOLビューにこれらのサーバー・プロセス・プールの情報が表示され、DBMS_PROCESSパッケージを使用してこれらのプールを管理できます。

関連項目:

- Oracle Databaseでの事前起動プロセスの管理の詳細は、Oracle Database管理者ガイドを参照してください
- DBMS_PROCESSパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照してください

第II部 データベース・パフォーマンスの診断およびチューニング

この部は、次の章で構成されています。

5 データベース・パフォーマンスの計測

この章では、データベース統計を使用してOracle Databaseのパフォーマンスを計測する方法を説明します。

この章のトピックは、次のとおりです:

- データベース統計について
- データベース統計の解釈

データベース統計について

データベース統計により、データベースへの負荷のタイプや、データベースで使用されるリソースに関する情報が提供されます。データベース・パフォーマンスを効果的に計測するには、統計が使用可能であることが必要です。

Oracle Databaseはシステム、セッション、セグメント、サービスおよび個別のSQL文の累積統計のタイプを生成します。統計の累積値には、通常、動的パフォーマンス・ビュー(V$ビュー)を使用してアクセスします。これらの範囲でのデータベース・パフォーマンスを分析する場合は、対象となる期間における統計の変化(デルタ値)を調べます。特に、期間開始時における統計の累積値と終了時における累積値の違いに注意してください。

この項では、Oracle Databaseのパフォーマンス計測に使用される、より重要なデータベース統計のいくつかを説明します。

- 時間モデル統計
- アクティブ・セッション履歴の統計
- 待機イベント統計
- セッションおよびシステム統計

関連項目:

オプティマイザ統計の詳細は、Oracle Database SQLチューニング・ガイドを参照してください

時間モデル統計

時間モデル統計では、ログオン操作や解析など、データベースで実行される特定アクションの影響を数値化して把握するために時間が使用されます。最も重要な時間モデル統計はデータベース時間(DB時間)です。この統計は、フォアグラウンド・セッションのデータベース・コールで経過した合計時間を表し、またインスタンスのワークロードの合計の指標にもなります。DB時間は、インスタンスの開始時から累計的に計測され、アイドル待機イベントを待機していないすべてのフォアグラウンド・セッション(アイドル状態でないユーザー・セッション)のCPU時間および待機時間を集計することで計算されます。

ノート:

DB 時間はアイドル状態でないすべてのユーザー・フォアグラウンド・セッションの時間を合計して計算されるため、DB 時間がインスタンス起動後の実際の経過時間を超える可能性があります。たとえば、30 分実行されているインスタ

ンスに 4 つのアクティブ・ユーザー・セッションがあれば、その累積 DBtime は約 120 分になります。

Oracleデータベースをチューニングする場合は、コンポーネントごとに固有の統計セットがあります。システム全体を調べるには、共通の比較尺度が必要です。このため、Oracle Databaseの多くのアドバイザおよびレポートでは、統計が時間単位で記述されます。

最終的には、Oracleデータベースのチューニングは、データベースでアクションの実行にかかる時間を短縮すること、または単にDB時間を短縮することを目的としています。時間モデル統計には、V$SESS_TIME_MODELおよびV$SYS_TIME_MODELビューからアクセスできます。

関連項目:

V$SESS_TIME_MODELおよびV$SYS_TIME_MODELビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

アクティブ・セッション履歴の統計

データベースに接続中で、Idle待機クラスに属さないイベントを待機中のセッションは、アクティブ・セッションとみなされます。Oracle Databaseではアクティブ・セッションが毎秒サンプリングされ、サンプル・データが共有グローバル領域(SGA)の循環バッファに格納されます。

サンプリングされたセッション・アクティビティには、V$ACTIVE_SESSION_HISTORYビューを使用してアクセスできます。各セッションのサンプルには一連の行と、V$ACTIVE_SESSION_HISTORYビューが含まれ、最後のセッション・サンプルの行から順番に、各アクティブ・セッションに対してサンプルごとに1行が戻されます。アクティブ・セッションのサンプルはSGAの循環バッファに格納されるため、システム・アクティビティが多いほど、格納できるセッション・アクティビティの秒数は短くなります。これは、V$ビューに表示されるセッション・サンプルの期間が、データベース・アクティビティのレベルに完全に依存することを意味します。大規模なシステム・アクティビティ中は、このV$ビューの内容が極端に大きくなる可能性があるため、ディスクにはセッション・サンプルの一部のみが書き込まれます。

アクティブ・セッションのみを取得することで、システム上で許可されるセッション数ではなく、実行される作業に直接関連するサイズで管理可能なデータ・セットが取得されます。アクティブ・セッション履歴(ASH)を使用すると、V$ACTIVE_SESSION_HISTORYビューの現行データとDBA_HIST_ACTIVE_SESS_HISTORYビューの履歴データの両方を検査して詳細な分析を実行でき、通常、ワークロードを再現して追加のパフォーマンス情報をトレースする必要がありません。ASHには、取得されたSQL文ごとの実行計画情報も含まれます。この情報を使用することで、SQLの経過時間に多くの影響を与えているSQL実行部分を特定できます。ASHに存在するデータは、次のように、取得する各種ディメンションでロール・アップできます。

- SQL文のSQL識別子
- SQL文の実行に使用されるSQL計画のSQL計画識別子およびハッシュ値
- SQL実行計画情報
- オブジェクト数、ファイル数およびブロック数
- 待機イベント識別子およびパラメータ

- セッション識別子およびセッション・シリアル番号
- モジュールおよびアクション名
- セッションのクライアント識別子
- サービス・ハッシュ識別子
- コンシューマ・グループ識別子

指定した期間のこの情報は、ASHレポートに収集できます。

アクティブ・セッション履歴は、Active Data Guardフィジカル・スタンバイ・インスタンスおよびOracle Automatic Storage Management(Oracle ASM)インスタンスでも使用できます。これらのインスタンスでは、現在のセッションのアクティビティが収集されてV$ACTIVE_SESSION_HISTORYビューに表示されますが、ディスクへの書込みは行われません。

関連項目:

- ASHレポートの詳細は、「サンプル・データの分析」を参照してください
- Active Data Guardのフィジカル・スタンバイ・データベースの詳細は、『Oracle Data Guard概要および管理』を参照してください
- Oracle ASMインスタンスの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

待機イベント統計

待機イベントは、処理を継続する前にイベントが完了するまで待機する必要があることを示すために、サーバー・プロセスまたはスレッドによって増やされる統計です。待機イベント・データにより、ラッチ、バッファおよびI/Oの競合など、パフォーマンスへ悪影響を与える可能性のある症状が明らかになります。

待機イベントに関する高水準の分析を容易にするために、Oracle Databaseでは、イベントが次のクラスにグループ化されます。

- 管理
- アプリケーション
- クラスタ
- コミット
- 同時実行性
- 構成
- アイドル
- ネットワーク
- その他
- スケジューラ

- システムI/O
- ユーザーI/O

待機クラスは、一般に特定待機イベントで問題を解決するために適用される共通の解決策に基づきます。たとえば、排他TXロックは通常はアプリケーション・レベルの問題であり、HWロックは通常は構成の問題です。次のリストに、一部の待機クラスでの一般的な待機イベントの例を示します。

- Application: 行レベル・ロックまたは明示的ロック・コマンドが原因のロック待機。
- Commit: コミット後のREDOログ書込み確認の待機。
- Idle: SQL*Net message from clientなど、セッションが非アクティブであることを示す待機イベント
- Network: ネットワーク上でのデータ送信の待機。
- User I/O: ディスクからのブロック読取りの待機。

データベース・インスタンスの待機イベント統計には、バックグラウンド・プロセスとフォアグラウンド・プロセスの両方の統計が含まれます。通常、チューニングは、フォアグラウンド・アクティビティが中心となるため、データベース・インスタンス・アクティビティ全体は、チューニングに役立つように、関連するV$ビュー内でフォアグラウンドとバックグラウンドの統計に分類されます。

V$SYSTEM_EVENTビューには、データベース・インスタンスのフォアグラウンド・アクティビティの待機イベント統計と、データベース・インスタンスの待機イベント統計が表示されます。V$SYSTEM_WAIT_CLASSビューには、待機クラスへの集計後に、これらのフォアグラウンドおよびインスタンス・レベルの待機イベント統計が表示されます。V$SESSION_EVENTおよびV$SESSION_WAIT_CLASSには、セッション・レベルでの待機イベント統計と待機クラス統計が表示されます。

関連項目:

待機イベントの詳細は、『Oracle Databaseリファレンス』を参照してください

セッションおよびシステム統計

V$SYSSTATおよびV$SESSTATビューを使用すると、システム・レベルとセッション・レベルの多数の累積データベース統計にアクセスできます。

関連項目:

V$SYSSTATおよびV$SESSTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください

データベース統計の解釈

最初にパフォーマンス・データを調査するときは、データベース統計を調べることによって、データについての可能性のある解釈を考察できます。正しい解釈であることを確認するために、別のデータでクロスチェックし、統計またはイベントが本当に関連しているかどうかを確定してください。フォアグラウンド・アクティビティはチューニング可能であるため、バックグラウンド・アクティビティの統計を分析する前に、フォアグラウンド・アクティビティの統計を先に分析することをお薦めします。

次の各項では、データベース・パフォーマンスを計測するために、様々なタイプのデータベース統計を解釈するヒントを説明します。

- ヒット率の使用
- 時間統計のある待機イベントの使用
- 時間統計のない待機イベントの使用
- アイドル待機イベントの使用
- データベース統計とその他の要因の比較
- 計算済統計の使用

ヒット率の使用

チューニングを行う場合は、問題の有無を判断するために、比率を計算することが一般的です。そのような率には、バッファ・キャッシュ・ヒット率、ソフト解析率およびラッチ・ヒット率があります。これらの率を、パフォーマンス・ボトルネックがあるかどうかを厳密に判断する識別子として使用しないでください。かわりに、インジケータとして使用します。パフォーマンス・ボトルネックがあるかどうかを識別するには、他の関連するパフォーマンス・データを調べます。バッファ・キャッシュ・ヒット率の計算方法の詳細は、「バッファ・キャッシュ・ヒット率の計算」を参照してください。

時間統計のある待機イベントの使用

インスタンス・レベルでTIMED_STATISTICSをTRUEに設定すると、使用可能な待機カウントの他にイベントの待機時間も収集されます。このデータは、イベントの合計待機時間とデータ収集間の総経過時間を比較する場合に役立ちます。たとえば、待機イベントが2時間の期間のうちのわずか30秒であれば、このイベントが待機時間別に順序付けされるときに最高ランクの待機イベントであったとしても、このイベントを調べることで改善されるパフォーマンスはごくわずかです。ただし、イベントが45分間のうちの30分であれば、そのイベントは調べる価値があります。待機イベントの詳細は、「待機イベント統計」を参照してください。

ノート:

初期化パラメータ STATISTICS_LEVEL が TYPICAL または ALL に設定されている場合、データベースの時間統計は自動的に収集されます。STATISTICS_LEVEL を BASIC に設定した場合、時間統計の収集を有効にするには、TIMED_STATISTICS を TRUE に設定する必要があります。STATISTICS_LEVEL を BASIC に設定すると、多くの自動機能が無効になることに注意してください。この設定はお薦めしません。

DB_CACHE_ADVICE、TIMED_STATISTICS または TIMED_OS_STATISTICS を、初期化パラメータ・ファイルか、ALTER_SYSTEM または ALTER SESSION を使用して明示的に設定した場合、その値は STATISTICS_LEVEL から導出された値を上書きします。

関連項目:

STATISTICS_LEVEL初期化パラメータについては、『Oracle Databaseリファレンス』を参照してください。

時間統計のない待機イベントの使用

TIMED_STATISTICSがFALSEに設定されている場合、イベントの待機に費やされた時間の長さは使用できません。したがって、

各イベントを待機した回数で待機イベントを順序付けることのみ可能です。最大待機回数を持つイベントが潜在的なボトルネックを示すことがありますが、主要なボトルネックとは言えません。この状況は、イベントを何度も待機する場合に発生する可能性がありますが、そのイベントを待機する合計時間は短時間です。逆に、待機回数が少ないイベントは、待機時間が合計待機時間の大きな割合を占めている場合には、大きなボトルネックであることがあります。比較のために使用する待機時間がなければ、調査する価値がある待機イベントかどうかの判断は困難です。

アイドル待機イベントの使用

Oracle Databaseでは、いくつかの待機イベントを使用して、Oracleサーバー・プロセスがアイドル状態であるかどうかを示します。一般に、これらのイベントはパフォーマンスの問題を調べるときは有効でなく、待機イベントを調べるときは無視する必要があります。

データベース統計とその他の要因の比較

統計を検証する場合、統計に価値があるかどうかに影響を与える他の要因を考慮することが重要です。そのような要因には、ユーザー負荷やハードウェア能力があります。45分間のうちの30分間が待機時間であったイベントでも、システム上に2000人のユーザーがいて、ホスト・ハードウェアが64ノード・コンピュータであった場合はパフォーマンスの問題とはみなされないことがあります。

計算済統計の使用

計算済統計(割合、トランザクションについて正規化された統計、比率など)を解釈する場合は、計算済統計を実際の統計カウントで検証することが重要です。通常、統計カウントが少ないと異常な比率は軽減して考慮されるため、この比較により、導出された割合が実際に重要かどうかを確認できます。たとえば、最初の調査で、ソフト解析率50%は一般に潜在的なチューニング領域を示します。ただし、データ収集期間にハード解析が1つとソフト解析が1つのみあった場合、パフォーマンスに影響していないことを統計カウントが示していても、ソフト解析率は50%になります。この場合、生の統計カウントが低いため、比率は重要ではありません。

6 データベース統計の収集

この章では、Oracle Databaseのデータベース統計を収集する方法を説明しており、内容は次のとおりです。

- データベース統計の収集について
- 自動ワークロード・リポジトリの管理
- 自動ワークロード・リポジトリ・レポートの生成
- パフォーマンス・ハブ・アクティブ・レポートの生成

データベース統計の収集について

Oracle Databaseは、すべてのレベル(セッション・レベルを除く)でほとんどの統計の累積値とデルタ値を、自動ワークロード・リポジトリ(AWR)に自動的に存続させます。このプロセスは通常の期間に繰り返され、結果はAWRスナップショットに取得されます。スナップショットで取得したデルタ値は、時間ごとの各統計に対する変更を示します。

統計的ベースラインは、通常、システムが最適なレベルで正常に動作している期間中に取得された統計的割合の集合です。ベースラインで取得された統計を、パフォーマンスが低下している期間に取得された統計と比較してパフォーマンスの問題を診断するには、統計的ベースラインを使用します。これにより、大幅に値が大きくなっていて問題の原因だと思われる特定の統計を識別できます。AWRは、AWRスナップショットのペアまたは範囲をベースラインとして指定して保存できるようにすることで、ベースライン・データの取得をサポートします。

一般的にメトリックは、累積的な統計における変化の割合です。この割合は、時間、トランザクション、データベース・コールなど、様々な単位に対して測定できます。たとえば、1秒当たりのデータベース・コールの数はメトリックです。メトリック値は、一部のV$ビューに表示され、このビューに表示される値は、きわめて短い時間間隔(通常は60秒)での平均値です。最新のメトリック値の履歴は、V$ビューを介して使用でき、データの一部はAWRスナップショットでも存続されます。

次の各項では、より効果的なデータベース統計の収集を可能にするOracle Databaseの様々な機能について説明します。

- 自動ワークロード・リポジトリ
- スナップショット
- ベースライン
- 領域使用量
- 適応しきい値

ノート:

- プラガブル・データベース(PDB)とともに AWR 機能を使用している場合には、データ可視性および権限の要件が異なります。AWR 機能などの管理機能がマルチテナント・コンテナ・データベース(CDB)で機能する仕組みの詳細は、Oracle Multitenant 管理者ガイドを参照してください。
- この章に記載されている AWR 機能を使用するには、Oracle Diagnostic Pack のライセンスが必要で

す。

自動ワークロード・リポジトリ

AWRは、問題の検出および自己チューニングを目的として、パフォーマンス統計を収集、処理およびメンテナンスします。この収集されたデータは、メモリーとデータベースの両方に格納され、レポートおよびビューに表示されます。

AWRにより収集され処理される統計は、次のとおりです。

- データベース・セグメントのアクセス統計と使用統計を決定するオブジェクト統計。
- アクティビティの時間使用に基づく時間モデル統計。V$SYS_TIME_MODELおよびV$SESS_TIME_MODELビューに表示されます。
- V$SYSSTATおよびV$SESSTATビューで収集されるシステム統計とセッション統計の一部。
- システム上で最大負荷を生成しているSQL文。経過時間やCPUタイムなどの基準に基づきます。
- アクティブ・セッション履歴(ASH)統計は、最新のセッション・アクティビティの履歴を表します

関連項目:

- 様々なタイプのデータベース統計の詳細は、「データベース統計について」を参照してください
- V$SYS_TIME_MODEL、V$SESS_TIME_MODEL、V$SYSSTATおよびV$SESSTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください

スナップショット

スナップショットは、自動データベース診断モニター(ADDM)によりパフォーマンスの比較に使用される、特定の期間の履歴データのセットです。デフォルトでは、1時間ごとにパフォーマンス・データのスナップショットが自動的に生成され、その統計はAWRに8日間保存されます。スナップショットの作成や、スナップショット保存期間の変更を手動で行うこともできますが、通常は必要ありません。

AWRは、スナップショット間の違いを比較し、システム負荷への影響に基づいて収集するSQL文を判別します。これにより、期間中に収集する必要のあるSQL文の数が減少します。スナップショットが作成されると、スナップショットに取得されたデータがADDMにより分析され、パフォーマンス分析が実行されます。

関連項目:

スナップショットの管理の詳細は、「スナップショットの管理」を参照してください

ベースライン

ベースラインは、パフォーマンス上の問題が発生したときに、他のスナップショットと比較するために保持されている特定期間のスナップショットのセットです。ベースラインに含まれるスナップショットはAWRの自動消去プロセスから除外され、無期限に保持されます。

使用可能なベースラインには、次の複数のタイプがあります。

- 固定ベースライン
- 変動ウィンドウ・ベースライン
- ベースライン・テンプレート

固定ベースライン

固定ベースラインは、過去の指定時における、固定した連続的な期間に対応します。ベースラインは最適なレベルでのシステム動作を表す必要があるため、固定ベースラインを作成する前に、ベースラインとして選択する期間を慎重に検討してください。将来的に、このベースラインを、パフォーマンスの低下期間中に取得された別のベースラインまたはスナップショットと比較して、時間経過に沿ってパフォーマンスの低下を分析できます。

関連項目:

固定ベースラインの管理の詳細は、「ベースラインの管理」を参照してください

変動ウィンドウ・ベースライン

変動ウィンドウ・ベースラインは、AWR保存期間内に存在するすべてのAWRデータに対応します。これは、AWRの全保存期間におけるAWRデータを使用してメトリックしきい値を計算できるため、適応しきい値を使用する際に役立ちます。

Oracle Databaseでは、システム定義の変動ウィンドウ・ベースラインが自動的に保持されます。システム定義による変動ウィンドウ・ベースラインのデフォルトのウィンドウ・サイズは、現在のAWR保存期間であり、デフォルトでは8日です。適応しきい値を使用する予定の場合は、しきい値を正確に計算するため、より長い変動ウィンドウ(30日など)を使用することを検討してください。変動ウィンドウ・ベースラインのサイズを変更するには、変動ウィンドウの日数を、AWR保存期間の日数以下の値に変更します。したがって、変動ウィンドウのサイズを拡張するには、先にAWR保存期間を適切に拡張しておく必要があります。

関連項目:

変動ウィンドウ・ベースラインのサイズ変更の詳細は、「デフォルトの変動ウィンドウ・ベースラインのサイズ変更」を参照してください

ベースライン・テンプレート

ベースライン・テンプレートを使用すると、将来の連続的な期間におけるベースラインを作成できます。ベースライン・テンプレートには、次の2つのタイプがあります。

- 単一ベースライン・テンプレート
- 繰返しベースライン・テンプレート

関連項目:

ベースライン・テンプレートの管理の詳細は、「ベースライン・テンプレートの管理」を参照してください

単一ベースライン・テンプレート

単一ベースライン・テンプレートを使用すると、将来の単一の連続的な期間におけるベースラインを作成できます。これは、将来に取得する期間が前もってわかっている場合に便利です。たとえば、次の週末にスケジュールされているシステム・テストの最中にAWRデータを取得することが可能です。この場合、単一ベースライン・テンプレートを作成することで、テストが実行されたときに自動的にその期間を取得できます。

繰返しベースライン・テンプレート

繰返しベースライン・テンプレートを使用すると、時間スケジュールの繰返しに基づいてベースラインを作成および削除できます。このテンプレートは、Oracle Databaseにより継続して連続的な期間を自動的に取得する場合に便利です。たとえば、1か月間にわたり毎週月曜日の朝にAWRデータを取得する必要があるとします。この場合、繰返しベースライン・テンプレートを作成することで、毎週月曜日という繰返しスケジュールに基づいてベースラインを自動的に作成し、1か月といった指定の期限の経過後に古いベースラインを自動的に削除できます。

領域使用量

AWRによって使用される領域は、次のように複数の要因によって決まります。

- 任意の時点におけるデータベースのアクティブ・セッション数
- スナップショット間隔

  スナップショット間隔により、スナップショットの収集頻度が決定されます。スナップショット間隔が短いほど頻度が高くなり、AWRにより収集されるデータ量が増大します。
- 履歴データの保存期間

  保存期間により、このデータが消去前に保持されている期間が決定されます。保存期間が長いほど、AWRによって消費される領域が増加します。

デフォルトでは、Oracle Databaseにより、1時間に1回スナップショットが取得され、データベースに8日間保持されます。これらのデフォルト設定では、同時アクティブ・セッション数が平均10の標準的なシステムの場合、AWRデータ用に約200から300MBの領域が必要になる可能性があります。

AWRの領域消費量を減らすには、スナップショット間隔を長くして保存期間を短縮します。保存期間を短縮する場合は、Oracle Databaseの複数の自己管理機能が正常に機能するためにAWRデータに依存することに注意してください。十分なデータがないと、次のようなコンポーネントと機能の妥当性および正確さに影響する可能性があります。

- 自動データベース診断モニター(ADDM)
- SQLチューニング・アドバイザ
- UNDOアドバイザ
- セグメント・アドバイザ

可能な場合は、少なくとも1つのワークロード・サイクル全体で収集できるように、AWR保存期間を十分な長さに設定することをお薦めします。システムのワークロード・サイクルが平日にOLTPワークロード、週末にバッチ・ジョブというような週単位になっている場合、デフォルトのAWR保存期間である8日を変更する必要はありません。ただし、月末の帳簿締め処理時にシステムの月次ピーク負荷が発生する場合は、保存期間を1か月に設定する操作が必要になることがあります。

例外的な状況では、スナップショット間隔を0に設定して、自動スナップショット収集を無効にすることができます。この場合、ワークロードおよび統計データの自動収集は停止され、Oracle Databaseの自己管理機能の大部分が動作しなくなります。また、スナップショットを手動で作成することもできません。このため、自動スナップショット収集を無効にしないことをお薦めします。

ノート:

Oracle Database は、デフォルトでは SYSAUX 表領域を使用して AWR データを格納します。Oracle Database 19c 以降では、SYSAUX 表領域のオーバーロードを回避するために、AWR データを格納する他の表領域を指定できます。

関連項目:

スナップショット間隔および保存期間のデフォルト値を変更する方法の詳細は、「スナップショット設定の変更」を参照してください

適応しきい値

適応しきい値を使用すると、管理オーバーヘッドを最小限に抑えながら、パフォーマンスの問題を監視および検出できます。適応しきい値は、変動ウィンドウ・ベースラインで取得されるメトリック値から導出された統計を使用して、システム・メトリックの警告しきい値とクリティカル・アラートしきい値を自動的に設定します。これらのしきい値の統計は週に1回再計算され、時間の経過とともに変化するシステムのパフォーマンスに応じて新しいしきい値になります。また、適応しきい値では、定期的なワークロードのパターンに基づいて1日または週の異なる時間帯の様々なしきい値を計算できます。

たとえば、多くのデータベースでは、日中のオンライン・トランザクション処理(OLTP)と夜間のバッチ処理をサポートしています。トランザクションごとのレスポンス時間のパフォーマンス・メトリックは、日中のOLTPパフォーマンスの低下の検出に役立ちます。ただし、有用なOLTPのしきい値は、長時間実行トランザクションが一般的なバッチ・ワークロードに対しては、通常は低すぎます。そのため、OLTPに適切なしきい値では、バッチ処理時に不適切なパフォーマンス・アラートが頻繁にトリガーされることがあります。適応しきい値は、このようなワークロードのパターンを検出し、日中や夜間用の様々なしきい値を自動的に設定できます。

適応しきい値には、次の2種類があります。

- 最大パーセントのしきい値
- 重大レベルのしきい値

最大パーセントのしきい値

最大パーセントのしきい値は、変動ウィンドウ・ベースライン内のデータに関して観察された最大値の割合の倍数として計算されます。

最大パーセントのしきい値は、システムをピーク・ワークロードに対応するようサイジングし、現在のワークロード・ボリュームが以前の高い値に近づいているか、その値を超えているときにアラートを生成する必要がある場合に特に役立ちます。未知だが明確な制限値を持つメトリックは、このような設定の第一候補になります。たとえば、1秒当たりの生成REDOのメトリックは、通常、最大パーセントのしきい値に対する適切な候補になります。

重大レベルのしきい値

重大レベルのしきい値は、変動ウィンドウ・ベースライン内のデータに基づくしきい値を超える値が観察されることがどの程度異常かを表す統計の百分位数に設定されます。

重大レベルのしきい値は、システムが正常に機能しているときに統計的に安定した動作を示すメトリックでは非常に役立ちますが、パフォーマンスが低下している場合は大きく変化する可能性があります。たとえば、トランザクションごとのレスポンス時間のメトリックは、適切に調整されたOLTPシステムの場合は安定していますが、パフォーマンスの問題が発生する場合は広い範囲で変動する可能性があります。重大レベルのしきい値は、異常なメトリックの値と異常なシステムのパフォーマンスの両方が発生する場合にアラートを生成することを意図しています。

重大レベルのしきい値は、次のいずれかのレベルに設定できます。

- 高(.95)

  100の観測のうち5つのみがこの値を超えると予測されます。
- 非常に高い(.99)

  100の観測のうち1つのみがこの値を超えると予測されます。
- 重大(.999)

  1,000の観測のうち1つのみがこの値を超えると予測されます。
- 極度(.9999)

  10,000の観測のうち1つのみがこの値を超えると予測されます。

重大レベルのしきい値を指定した場合、Oracle Databaseでは内部計算を実行してしきい値を設定します。ベースラインのデータを使用してより重大度のレベルが高いしきい値を設定できず、重大レベルのしきい値が設定されない場合があります。

重大(.999)または極度(.9999)の重大レベルのしきい値を指定し、予測どおりにアラートを受信しない場合、このしきい値を非常に高い(.99)や高(.95)などの低い値に設定できます。または、最大パーセントのしきい値をかわりに使用することを検討してください。しきい値を変更して、受信するアラートが多すぎることがわかった場合は、アラートがトリガーされる状況の発生回数を増やしてください。

ノート:

ベースライン・メトリックを管理するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。ベースライン・メトリックの適応しきい値を作成するには、Oracle Database 2日でパフォーマンス・チューニング・ガイドの説明に従って Cloud Control を使用します。

関連項目:

- 変動ウィンドウ・ベースラインの詳細は、「変動ウィンドウ・ベースライン」を参照してください
- ベースライン・メトリックの管理の詳細は、「ベースラインの管理」を参照してください

自動ワークロード・リポジトリの管理

この項では、Oracle DatabaseのAWR機能の管理方法を説明しており、内容は次のとおりです。

- 自動ワークロード・リポジトリの有効化
- スナップショットの管理
- ベースラインの管理
- ベースライン・テンプレートの管理
- 別のシステムへの自動ワークロード・リポジトリ・データの転送
- 自動ワークロード・リポジトリ・ビューの使用
- マルチテナント環境内の自動ワークロード・リポジトリの管理
- Active Data Guardスタンバイ・データベースでの自動ワークロード・リポジトリの管理

関連項目:

AWRの説明は、「自動ワークロード・リポジトリ」を参照してください

自動ワークロード・リポジトリの有効化

AWRを使用したデータベース統計の収集はデフォルトで有効になっており、STATISTICS_LEVEL初期化パラメータで制御されます。

AWRによる統計収集を有効化するには:

- STATISTICS_LEVELパラメータをTYPICALまたはALLに設定します。

  このパラメータのデフォルト設定はTYPICALです。

STATISTICS_LEVELをBASICに設定すると、AWRを含む多くのOracle Databaseの機能が無効になるため、この設定はお薦めしません。STATISTICS_LEVELをBASICに設定しても、DBMS_WORKLOAD_REPOSITORYパッケージを使用すればAWR統計を手動で取得できます。ただし、セグメント統計やメモリー・アドバイザ情報など、多くのシステム統計のインメモリー収集が無効になるため、これらのスナップショットで取得された統計は不完全になる可能性があります。

関連項目:

STATISTICS_LEVEL初期化パラメータについては、『Oracle Databaseリファレンス』を参照してください。

スナップショットの管理

デフォルトでは、1時間ごとにスナップショットが生成され、その統計はワークロード・リポジトリに8日間保存されます。必要な場合には、スナップショットを手動で作成または削除し、スナップショット設定を変更できます。

この項では、スナップショットの管理方法を説明しており、内容は次のとおりです。

- スナップショットを管理するためのユーザー・インタフェース
- スナップショットの作成
- スナップショットの削除
- スナップショット設定の変更

関連項目:

スナップショットの詳細は、「スナップショット」を参照してください

スナップショットを管理するためのユーザー・インタフェース

スナップショットを管理するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してスナップショットを管理してください。

Cloud Controlを使用できない場合、スナップショットは、コマンドライン・インタフェースでDBMS_WORKLOAD_REPOSITORYパッケージを使用して管理します。DBAロールは、DBMS_WORKLOAD_REPOSITORYプロシージャの起動に必要です。

関連項目:

DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

スナップショットの作成

デフォルトでは、Oracle Databaseにより、1時間ごとにスナップショットが自動的に生成されます。ただし、自動生成されるスナップショットとは異なる時点の統計を取得するために、手動でスナップショットを作成することが必要な場合があります。

コマンドライン・インタフェースを使用したスナップショットの作成

手動でスナップショットを作成するには、CREATE_SNAPSHOTプロシージャを使用します。次の例に、CREATE_SNAPSHOTプロシージャ・コールを示します。

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT ();
END;
/
```

この例では、スナップショットはローカル・データベース・インスタンスにただちに作成されます。既存のスナップショットの情報を表示するには、DBA_HIST_SNAPSHOTビューを使用します。

ノート:

CREATE_SNAPSHOT プロシージャの flush_level パラメータの値に TYPICAL または ALL を指定できます。フラッ

シュ・レベルのデフォルト値は TYPICAL です。

フラッシュ・レベルは、取得する AWR 統計の範囲と深度を示します。すべての AWR 統計を取得する場合は、フラッシュ・レベルを ALL に設定します。パフォーマンス上の理由で、SQL 統計、セグメント統計、ファイルおよび表領域統計などの一部の AWR 統計をスキップする場合は、フラッシュ・レベルを TYPICAL に設定します。

関連項目:

- DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください
- DBA_HIST_SNAPSHOTビューの詳細は、『Oracle Databaseリファレンス』を参照してください

スナップショットの削除

デフォルトでは、Oracle Databaseにより、8日を超えてAWRに格納されているスナップショットが自動的に消去されます。ただし、領域を解放するために、一定範囲のスナップショットを手動で削除することが必要な場合があります。

コマンドライン・インタフェースを使用したスナップショットの削除

一定範囲のスナップショットを手動で削除するには、DROP_SNAPSHOT_RANGEプロシージャを使用します。次の例に、DROP_SNAPSHOT_RANGEプロシージャ・コールを示します。

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_SNAPSHOT_RANGE (low_snap_id  => 22,
                                                high_snap_id => 32,
                                                dbid         => 3310949047);
END;
/
```

この例では、スナップショットIDが22から32の範囲のスナップショットが、データベース識別子が3310949047のデータベース・インスタンスからただちに削除されます。このスナップショット範囲から取得された任意のASHデータも消去されます。

ヒント:

削除するスナップショットを決定するには、DBA_HIST_SNAPSHOT ビューを使用して、既存のスナップショットを確認します。

関連項目:

- DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください
- DBA_HIST_SNAPSHOTビューの詳細は、『Oracle Databaseリファレンス』を参照してください

スナップショット設定の変更

スナップショットの生成のためにフラッシュする間隔、保存期間および上位SQLの数を調整できますが、これはOracle Database診断ツールの精度に影響する可能性があることに注意してください。

コマンドライン・インタフェースを使用したスナップショット設定の変更

スナップショット設定は、DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGSプロシージャの次のパラメータを使用して変更できます。

パラメータ	説明
INTERVAL	この設定は、データベースがスナップショットを自動的に生成する頻度に影響します。
RETENTION	この設定は、データベースがスナップショットを AWR に保存する期間に影響します。
TOPNSQL	この設定は、SQL 基準(経過時間、CPU 時間、解析コール、共有可能メモリーおよびバージョン・カウント)ごとにフラッシュする上位 SQL の数に影響します。 この設定は、統計レベルまたはフラッシュ・レベルによる影響を受けず、AWR SQL 収集のシステム・デフォルト動作より優先されます。この設定値を MAXIMUM に設定して、共有 SQL 領域内の完全な SQL セットを取得することは可能ですが、この値(または非常に大きい数値)に設定すると、収集および保存するデータが増加するため、領域およびパフォーマンスの問題が生じる可能性があります。

次の例は、DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGSプロシージャを使用してスナップショット設定を変更する方法を示しています。

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS( retention  => 43200,
                                                     interval   => 30,
                                                     topnsql    => 100,
                                                     dbid       => 3310949047);
END;
/
```

この例では、データベース識別子が3310949047のデータベースのスナップショット設定が、次のように変更されます。

- 保存期間は43200分(30日)に指定されます。
- 各スナップショットの間の間隔は30分に指定されます。
- SQL基準ごとにフラッシュする上位SQLの数は100に指定されます。

データベースの現在のスナップショット設定に関する情報を取得するには、次の例に示すようにDBA_HIST_WR_CONTROLビューを使用します。

```
SQL> select snap_interval, retention from DBA_HIST_WR_CONTROL;
SNAP_INTERVAL  RETENTION
---------------------------------------
+00000 01:00:00.0 +00008 00:00:00.0
```

snap_intervalおよびretentionの値が、次の形式で表示されます。

```
+[days] [hours]:[minutes]:[seconds].[nanoseconds]
```

関連項目:

- DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGSプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。
- DBA_HIST_WR_CONTROLビューの詳細は、『Oracle Databaseリファレンス』を参照してください

ベースラインの管理

Oracle Databaseでは、デフォルトで、システム定義の変動ウィンドウ・ベースラインが自動的に保持されます。必要な場合には、ベースラインを手動で作成、削除または名前変更し、ベースラインしきい値の設定を表示できます。また、変動ウィンドウ・ベースラインのウィンドウ・サイズを手動で変更できます。

この項では、ベースラインの管理方法を説明しており、内容は次のとおりです。

- ベースラインを管理するためのユーザー・インタフェース
- ベースラインの作成
- ベースラインの削除
- ベースラインの名前変更
- ベースライン・メトリックの表示
- デフォルトの変動ウィンドウ・ベースラインのサイズの変更

関連項目:

ベースラインの詳細は、「ベースライン」を参照してください

ベースラインを管理するためのユーザー・インタフェース

ベースラインを管理するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してベースラインを管理してください。

Cloud Controlを使用できない場合、ベースラインは、コマンドライン・インタフェースでDBMS_WORKLOAD_REPOSITORYパッケージを使用して管理します。DBAロールは、DBMS_WORKLOAD_REPOSITORYプロシージャの起動に必要です。

関連項目:

- Cloud Controlを使用したベースラインの管理については、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください
- DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタ

イプ・リファレンス』を参照してください

ベースラインの作成

Oracle Databaseでは、デフォルトで、システム定義の変動ウィンドウ・ベースラインが自動的に保持されます。ただし、パフォーマンスが低下している期間に取得された別のベースラインまたはスナップショットと比較できるように、最適なレベルで運用されているシステムを表す固定ベースラインを手動で作成することが必要な場合があります。

コマンドライン・インタフェースを使用してベースラインを作成するには、次の例に示すように、CREATE_BASELINEプロシージャを使用します。

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE (start_snap_id => 270,
                                              end_snap_id   => 280,
                                              baseline_name => 'peak baseline',
                                              dbid          => 3310949047,
                                              expiration    => 30);
END;
/
```

この例では、データベース識別子が3310949047のデータベース・インスタンスに、次の設定でベースラインが作成されます。

- 開始スナップショットの順序番号は270です。
- 終了スナップショットの順序番号は280です。
- ベースライン名はpeak baselineです。
- ベースラインの有効期限は30日です。

ベースラインが作成されると、Oracle Databaseにより、一意のIDが新規ベースラインに自動的に割り当てられます。

ヒント:

ベースラインに含めるスナップショットの範囲を決定するには、DBA_HIST_SNAPSHOT ビューを使用して、既存のスナップショットを確認します。

関連項目:

- DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください
- DBA_HIST_SNAPSHOTビューの詳細は、『Oracle Databaseリファレンス』を参照してください

ベースラインの削除

ディスク領域を節約するには、使用されなくなったベースラインを定期的に削除することを検討してください。ベースラインに関連付けられたスナップショットは、ベースラインが明示的に削除されるか、ベースラインが期限切れになるまで無期限に保持されます。

コマンドライン・インタフェースを使用してベースラインを削除するには、次の例に示すように、DROP_BASELINEプロシージャを使用

します。

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE (baseline_name => 'peak baseline',
                                          cascade       => FALSE,
                                          dbid          => 3310949047);
END;
/
```

この例では、ベースラインpeak baselineが、データベース識別子が3310949047のデータベース・インスタンスから削除され、関連付けられたスナップショットが保存されます。

ヒント:

削除するベースラインを決定するには、DBA_HIST_BASELINE ビューを使用して、既存のベースラインを確認します。

ヒント:

関連付けられているスナップショットをベースラインとともに削除するには、cascade パラメータを TRUE に設定します。

関連項目:

DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

ベースラインの名前変更

コマンドライン・インタフェースを使用してベースラインの名前を変更するには、RENAME_BASELINEプロシージャを使用します。次の例に、RENAME_BASELINEプロシージャ・コールを示します。

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.RENAME_BASELINE (old_baseline_name => 'peak baseline',
                                             new_baseline_name => 'peak mondays',
                                             dbid              => 3310949047);
END;
/
```

この例では、データベース識別子が3310949047のデータベース・インスタンスに存在するベースラインの名前が、peak baselineからpeak mondaysに変更されます。

関連項目:

DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リ

ファレンス』を参照してください

ベースライン・メトリックの表示

適応しきい値とともに使用する場合、ベースラインには、データベースでメトリックしきい値を計算する際に使用できるAWRデータが含まれます。

コマンドライン・インタフェースを使用して、ベースライン期間におけるメトリック値のサマリー統計を表示するには、SELECT_BASELINE_METRICSファンクションを使用します。

```
DBMS_WORKLOAD_REPOSITORY.SELECT_BASELINE_METRICS (baseline_name IN VARCHAR2,
                                                  dbid          IN NUMBER DEFAULT NULL,
                                                  instance_num  IN NUMBER DEFAULT NULL)
   RETURN awr_baseline_metric_type_table PIPELINED;
```

関連項目:

- ベースライン・メトリックしきい値の詳細は、「適応しきい値」を参照してください
- DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

デフォルトの変動ウィンドウ・ベースラインのサイズ変更

Oracle Databaseでは、デフォルトで、システム定義の変動ウィンドウ・ベースラインが自動的に保持されます。システム定義による変動ウィンドウ・ベースラインのデフォルトのウィンドウ・サイズは、現在のAWR保存期間であり、デフォルトでは8日です。適応しきい値のしきい値をより正確に計算するためにサイズを大きくするなど、特定の状況では、デフォルトの変動ウィンドウ・ベースラインのウィンドウ・サイズを変更することが必要な場合があります。

コマンドライン・インタフェースを使用して、デフォルトの変動ウィンドウ・ベースラインのウィンドウ・サイズを変更するには、次の例に示すように、MODIFY_BASELINE_WINDOW_SIZEプロシージャを使用します。

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.MODIFY_BASELINE_WINDOW_SIZE (window_size => 30,
                                                          dbid        => 3310949047);
END;
/
```

この例では、データベース識別子が3310949047のデータベース・インスタンスのデフォルトの変動ウィンドウ・ベースラインが30日にサイズ変更されます。

ノート:

ウィンドウ・サイズは、AWR 保存設定の値以下の値に設定する必要があります。現在の AWR 保存期間を超えるウィンドウ・サイズを設定するには、スナップショット設定の変更の手順に従ってあらかじめ retention パラメータの値を増やしておく必要があります。

関連項目:

- 変動ウィンドウ・ベースラインの詳細は、「変動ウィンドウ・ベースライン」を参照してください
- DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

ベースライン・テンプレートの管理

ベースライン・テンプレートを使用すると、将来の指定された期間を取得するベースラインを自動的に作成できます。この項では、ベースライン・テンプレートの管理方法を説明しており、内容は次のとおりです。

- ベースライン・テンプレートを管理するためのユーザー・インタフェース
- 単一ベースライン・テンプレートの作成
- 繰返しベースライン・テンプレートの作成
- ベースライン・テンプレートの削除

関連項目:

ベースライン・テンプレートの詳細は、「ベースライン・テンプレート」を参照してください

ベースライン・テンプレートを管理するためのユーザー・インタフェース

ベースライン・テンプレートを管理するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してベースライン・テンプレートを管理します。

Cloud Controlを使用できない場合、ベースライン・テンプレートは、コマンドライン・インタフェースでDBMS_WORKLOAD_REPOSITORYパッケージを使用して管理します。DBAロールは、DBMS_WORKLOAD_REPOSITORYプロシージャの起動に必要です。

関連項目:

Cloud Controlを使用したベースライン・テンプレートの管理については、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

単一ベースライン・テンプレートの作成

単一ベースライン・テンプレートを使用して、将来の単一の固定期間におけるベースラインを作成できます。たとえば、単一ベースライン・テンプレートを作成し、2012年4月2日の午後5:00から午後8:00までに取得されるベースラインを生成できます。

コマンドライン・インタフェースを使用して単一ベースライン・テンプレートを作成するには、次の例に示すように、CREATE_BASELINE_TEMPLATEプロシージャを使用します。

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE (start_time    => '2012-04-02 17:00:00 PST',
                                                       end_time      => '2012-04-02 20:00:00 PST',
```

```
                                                    baseline_name => 'baseline_120402',
                                                    template_name => 'template_120402',
                                                    expiration    => 30,
                                                    dbid          => 3310949047);
END;
/
```

この例では、3310949047というデータベースIDを持つデータベース上に、2012年4月2日の午後5:00から午後8:00までの期間におけるベースラインbaseline_120402を生成するtemplate_120402というベースライン・テンプレートが作成されます。このベースラインは、30日後に期限切れとなります。

関連項目:

DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

繰返しベースライン・テンプレートの作成

繰返しベースライン・テンプレートを使用すると、将来の一定期間において特定の時間間隔で繰り返されるベースラインを自動的に作成できます。たとえば、繰返しベースライン・テンプレートを作成し、2012年の毎週月曜日の午後5:00から午後8:00までに繰り返されるベースラインを生成できます。

コマンドラインを使用して繰返しベースライン・テンプレートを作成するには、次の例に示すように、CREATE_BASELINE_TEMPLATEプロシージャを使用します。

```
BEGIN
    DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE_TEMPLATE (day_of_week => 'monday',
                                                    hour_in_day => 17,
                                                    duration => 3, expiration => 30,
                                                    start_time => '2012-04-02 17:00:00 PST',
                                                    end_time => '2012-12-31 20:00:00 PST',
                                                    baseline_name_prefix =>
'baseline_2012_mondays_',
                                                    template_name => 'template_2012_mondays',
                                                    dbid => 3310949047);
END;
/
```

この例では、3310949047というデータベースIDを持つデータベース上に、2012年4月2日の午後5:00から2012年12月31日の午後8:00までの期間において毎週月曜日の午後5:00から午後8:00までベースラインを生成するtemplate_2012_mondaysというベースライン・テンプレートが作成されます。各ベースラインは、baseline_2012_mondays_という接頭辞付きの名前で作成され、30日後に期限切れとなります。

関連項目:

DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

ベースライン・テンプレートの削除

すでに使用されていないベースライン・テンプレートは、ディスク領域を節約するために定期的に削除できます。

コマンドラインを使用してベースライン・テンプレートを削除するには、次の例に示すように、DROP_BASELINE_TEMPLATEプロシージャを使用します。

```
BEGIN
  DBMS_WORKLOAD_REPOSITORY.DROP_BASELINE_TEMPLATE (template_name => 'template_2012_mondays',
                                                   dbid          => 3310949047);
END;
/
```

この例では、template_2012_mondaysというベースライン・テンプレートが、データベース識別子が3310949047のデータベース・インスタンスから削除されます。

ヒント:

削除するベースライン・テンプレートを決定するには、DBA_HIST_BASELINE_TEMPLATE ビューを使用して、既存のベースライン・テンプレートを確認します

関連項目:

DBMS_WORKLOAD_REPOSITORYパッケージについては、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

別のシステムへの自動ワークロード・リポジトリ・データの転送

Oracle Databaseでは、システム間でAWRデータを転送できます。これは、別のシステムを使用してAWRデータの分析を実行し、本番システムのパフォーマンス分析によるオーバーヘッドを削減する場合に有効です。

システム間でAWRデータを転送するには、最初にソース・システムのデータベースからAWRデータをエクスポートし、次にそのデータをターゲット・システムのデータベースにインポートします。

この項では、次の項目について説明します。

- AWRデータのエクスポート
- AWRデータのインポート

AWRデータのエクスポート

awrextr.sqlスクリプトは、一定範囲のスナップショットに対応するAWRデータをデータベースからデータ・ポンプ・エクスポート・ファイルにエクスポートします。作成されたら、エクスポートしたAWRデータをインポートできる別のデータベースに、このダンプ・ファイルを転送できます。awrextr.sqlスクリプトを実行するには、SYSユーザーとしてデータベースに接続する必要があります。

AWRデータをエクスポートするには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrextr.sql
```

AWRスキーマのデータベースのリストが表示されます。

2. AWRデータをエクスポートするデータベースを指定します。

```
Enter value for db_id: 1377863381
```

この例では、1377863381というデータベース識別子を持つデータベースが指定されます。

3. すべてのスナップショットIDを表示する日数を指定します。

```
Enter value for num_days: 2
```

この例では、最近2日間に取得されたすべてのスナップショットが表示されます。

4. 最初と最後のスナップショットIDを指定して、AWRデータをエクスポートするスナップショットの範囲を定義します。

```
Enter value for begin_snap: 30
Enter value for end_snap: 40
```

この例では、スナップショットID 30が最初のスナップショットとして指定され、スナップショットID 40が最後のスナップショットとして指定されます。

ディレクトリ・オブジェクトのリストが表示されます。

5. エクスポート・ダンプ・ファイルを保存するディレクトリを示すディレクトリ・オブジェクトを指定します。

```
Enter value for directory_name: DATA_PUMP_DIR
```

この例では、ディレクトリORACLE_HOME/rdbms/logを指すディレクトリ・オブジェクトDATA_PUMP_DIRが指定されています。ORACLE_HOMEは/u01/app/oracle/product/database_release_number/dbhome_1です。

6. エクスポート・ダンプ・ファイルの名前をファイル拡張子なしで指定します。デフォルトでは、ファイル拡張子.dmpが使用されます。

```
Enter value for file_name: awrdata_30_40
```

この例では、awrdata_30_40.dmpという名前のエクスポート・ダンプ・ファイルが、ディレクトリ・オブジェクトDATA_PUMP_DIRで指定されたディレクトリに作成されます。

```
Dump file set for SYS.SYS_EXPORT_TABLE_01 is:
/u01/app/oracle/product/database_release_number/dbhome_1/rdbms/log/awrdata_30_40.dmp
Job "SYS"."SYS_EXPORT_TABLE_01" successfully completed at 08:58:20
```

エクスポートする必要のあるAWRデータの分量によっては、AWRエクスポート操作の完了までにしばらく時間がかかる場合があります。ダンプ・ファイルが作成されたら、データ・ポンプを使用してそのファイルを別のシステムに転送できます。

関連項目:

データ・ポンプの使用については、『Oracle Databaseユーティリティ』を参照してください。

AWRデータのインポート

エクスポート・ダンプ・ファイルをターゲット・システムに転送したら、awrload.sqlスクリプトを使用してエクスポートされたAWRデータをインポートします。awrload.sqlスクリプトにより、スナップショット・データをデータ・ポンプ・ファイルからデータベースに転送するためのステージング・スキーマが作成されます。次に、データはステージング・スキーマから適切なAWR表に転送されます。
awrload.sqlスクリプトを実行するには、SYSユーザーとしてデータベースに接続する必要があります。

AWRデータをインポートするには:

1. SQLプロンプトで次のように入力します。

   ```
   @$ORACLE_HOME/rdbms/admin/awrload.sql
   ```

   ディレクトリ・オブジェクトのリストが表示されます。

2. エクスポート・ダンプ・ファイルが含まれるディレクトリを示すディレクトリ・オブジェクトを指定します。

   ```
   Enter value for directory_name: DATA_PUMP_DIR
   ```

   この例では、エクスポート・ダンプ・ファイルが存在するディレクトリを指すディレクトリ・オブジェクトDATA_PUMP_DIRが指定されています。

3. エクスポート・ダンプ・ファイルの名前をファイル拡張子なしで指定します。デフォルトでは、ファイル拡張子.dmpが使用されます。

   ```
   Enter value for file_name: awrdata_30_40
   ```

   この例では、awrdata_30_40.dmpという名前のエクスポート・ダンプ・ファイルが選択されています。

4. AWRデータをインポートするステージング・スキーマの名前を指定します。

   ```
   Enter value for schema_name: AWR_STAGE
   ```

   この例では、AWR_STAGEという名前のステージング・スキーマが作成されます。

5. ステージング・スキーマのデフォルト表領域を指定します。

   ```
   Enter value for default_tablespace: SYSAUX
   ```

   この例では、SYSAUX表領域が指定されています。

6. ステージング・スキーマの一時表領域を指定します。

   ```
   Enter value for temporary_tablespace: TEMP
   ```

   この例では、TEMP表領域が指定されています。

7. 最初にAWRデータがAWR_STAGEスキーマにインポートされ、次にSYSスキーマのAWR表に転送されます。

   ```
   Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/CONSTRAINT
   Completed 113 CONSTRAINT objects in 11 seconds
   Processing object type TABLE_EXPORT/TABLE/CONSTRAINT/REF_CONSTRAINT
   Completed 1 REF_CONSTRAINT objects in 1 seconds
   Job "SYS"."SYS_IMPORT_FULL_03" successfully completed at 09:29:30
   ... Dropping AWR_STAGE user
   End of AWR Load
   ```

   インポートする必要のあるAWRデータの分量によっては、AWRインポート操作の完了までにしばらく時間がかかる場合

があります。AWRデータのインポート後、ステージング・スキーマは自動的に削除されます。

自動ワークロード・リポジトリ・ビューの使用

通常は、Oracle Enterprise Manager Cloud Control (Cloud Control)またはAWRレポートを使用してAWRデータを表示します。ただし、AWRに格納されている履歴データは、DBA_HISTビューを使用して表示できます。

ノート:

マルチテナント環境では、これらの DBA_HIST ビューを、AWR_ROOT ビューおよび AWR_PDB ビューと、それぞれ CDB レベルおよび PDB レベルで交換することもできます。たとえば、PDB レベルでのアクティブ・セッション履歴に関する AWR データの取得には AWR_PDB_ACTIVE_SESS_HISTORY ビューを使用でき、これは、非マルチテナント環境の独立したデータベースでの DBA_HIST_ACTIVE_SESS_HISTORY ビューと同等です。PDB レベルのスナップショットが収集されていない場合、AWR_PDB ビューには AWR データが表示されません。

表6-1 DBA_HISTビュー

DBA_HISTビュー	説明
DBA_HIST_ACTIVE_SESS_HISTORY	最新のシステム・アクティビティに関するインメモリー・アクティブ・セッション履歴の内容の履歴が表示されます。
DBA_HIST_BASELINE	システム上に収集されたベースラインに関する情報(各ベースラインの時間範囲やベースライン・タイプなど)が表示されます。
DBA_HIST_BASELINE_DETAILS	特定のベースラインに関する詳細情報が表示されます。
DBA_HIST_BASELINE_TEMPLATE	ベースラインを生成するためにシステムによって使用されるベースライン・テンプレートに関する情報が表示されます。
DBA_HIST_CON_SYS_TIME_MODEL	OLAP の時刻に関連する統計など、システムの時間モデル統計の履歴が表示されます。
DBA_HIST_CON_SYSMETRIC_HIST	システム・メトリック値に関する履歴情報が表示されます。
DBA_HIST_CON_SYSMETRIC_SUMM	長期(60 秒)グループのシステム・メトリックのすべてのメトリック値について、統計サマリーの履歴が表示されます。
DBA_HIST_CON_SYSSTAT	OLAP カーネル統計などのシステム統計の履歴が表示されます。
DBA_HIST_CON_SYSTEM_EVENT	イベントの待機の合計に関する履歴情報が表示されます。

DBA_HISTビュー	説明
DBA_HIST_DATABASE_INSTANCE	データベース環境に関する情報が表示されます。
DBA_HIST_DB_CACHE_ADVICE	各行に対応するキャッシュ・サイズの物理読取りの数の履歴予測が表示されます。
DBA_HIST_DISPATCHER	スナップショット時の各ディスパッチャ・プロセスの履歴情報が表示されます。
DBA_HIST_DYN_REMASTER_STATS	動的リマスタリング・プロセスに関する統計情報が表示されます。
DBA_HIST_IOSTAT_DETAIL	ファイル・タイプと機能別に集計された履歴 I/O 統計が表示されます。
DBA_HIST_RSRC_PDB_METRIC	プラガブル・データベース(PDB)のリソース・マネージャ・メトリックに関する、過去 1 時間の履歴情報が表示されます。
DBA_HIST_RSRC_METRIC	コンシューマ・グループのリソース・マネージャ・メトリックに関する、過去 1 時間の履歴情報が表示されます。
DBA_HIST_SHARED_SERVER_SUMMARY	共有サーバー・アクティビティ、一般的なキュー、およびディスパッチャ・キューなど、共有サーバーの履歴情報が表示されます。
DBA_HIST_SNAPSHOT	システム内のスナップショットの情報が表示されます。
DBA_HIST_SQL_PLAN	SQL 実行計画が表示されます。
DBA_HIST_WR_CONTROL	AWR 制御用の設定が表示されます。
DBA_HIST_WR_SETTINGS	AWR の設定とメタデータが表示されます。
DBA_HIST_PROCESS_WAITTIME	プロセス・タイプの CPU および待機時間が表示されます。

関連項目:

DBA_HISTビューの詳細は、Oracle Databaseリファレンスを参照してください

マルチテナント環境内の自動ワークロード・リポジトリの管理

集中化された自動ワークロード・リポジトリ(AWR)に、マルチテナント環境内のCDBおよびPDBに関連するパフォーマンス・データが格納されます。

CDBおよび個々のPDBでは、AWRデータを格納、表示および管理できます。CDBレベルまたはPDBレベルでAWRのスナップショットを取得できます。

この項では、次の項目について説明します。

- マルチテナント環境でのAWRデータのカテゴリ化
- マルチテナント環境でのAWRデータの格納および取得
- マルチテナント環境のAWRデータの表示

マルチテナント環境でのAWRデータのカテゴリ化

マルチテナント環境では、AWRデータが様々なカテゴリに分類されます。

カテゴリは次のとおりです。

- 一般AWRデータ

  このデータはセキュリティには関係がありません。CDB内のすべてのテナントで安全に共有できます。このデータは、すべてのPDBからアクセスでき、CDBとPDBの両方のレベルのスナップショットで取得されます。一般的なAWRデータの例としては、統計、ラッチおよびパラメータの名前などがあります。

- CDBのAWRデータ

  このカテゴリには、CDB内のすべてのテナントのデータが集計されます。このデータにはデータベース全体のステータスが含まれており、CDB管理者にのみ有益です。このデータはCDBレベルのスナップショットでのみ取得されます。

- 個々のPDBのAWRデータ

  このデータはCDB内の個々のPDBを記述しています。データベース全体のインスタンスに対する個々のPDBの貢献度を表すコンテナ固有データが表示されます。そのため、このデータはCDBとPDB両方の管理者に役立ちます。このデータは、CDBとPDBの両方のレベルのスナップショットで取得されます。

マルチテナント環境でのAWRデータの格納および取得

この項では、マルチテナント環境でのAWRデータのエクスポートおよびインポートと、スナップショットの管理プロセスについて説明します。

スナップショットの管理

Oracle Database 12cリリース2 (12.2)以降では、CDBレベル(CDBルート)とPDBレベル(個別PDB)でAWRスナップショットを取得できます。デフォルトでは、CDBレベルのスナップショット・データはCDBルートのSYSAUX表領域に格納され、PDBレベルのスナップショット・データはPDBのSYSAUX表領域に格納されます。

CDBレベルのスナップショットには、ASH、SQL統計およびファイル統計などのPDB統計とCDB統計に関する情報が含まれます。CDB管理者は、AWRデータ保存期間の設定、スナップショット・スケジュールの設定、手動スナップショットの取得、CDBルートのスナップショット・データのパージなどの、CDB固有の管理操作を実行できます。

PDBレベルのスナップショットには、PDB統計と、PDBに関連するパフォーマンス上の問題を診断するために役立つ一部のグローバル統計も含まれます。PDB管理者は、AWRデータ保存期間の設定、スナップショット・スケジュールの設定、手動スナップショットの取得、PDBのスナップショット・データのパージなどの、PDB固有の管理操作を実行できます。

スナップショットの作成やスナップショットのパージなどの、CDBレベルおよびPDBレベルのスナップショット操作は、自動モードまたは手動モードのいずれでも実行できます。

自動スナップショット操作はスケジュールされ、特定の時間に自動的に実行されます。AWR_PDB_AUTOFLUSH_ENABLED初期化パラメータによって、CDB内のすべてのPDBまたはCDB内の個別のPDBに対して、自動スナップショットを有効化または無効化するように指定できます。自動スナップショット操作は、CDBに対してデフォルトで有効化されますが、PDBに対してデフォルトで無効化されます。PDBに対して自動スナップショットを有効化するには、PDB管理者が対象のPDBに接続し、AWR_PDB_AUTOFLUSH_ENABLEDパラメータの値をtrueに設定し、スナップショット生成間隔を0より大きい値に設定する必要があります。

関連項目:

AWR_PDB_AUTOFLUSH_ENABLED初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください

手動スナップショット操作はユーザーによって明示的に開始されます。自動スナップショットと手動スナップショットは同じAWR情報を取得します。PDBの場合、通常は手動スナップショットを使用することをお薦めします。PDBに対する自動スナップショットは、パフォーマンス上の理由で、選択的にのみ有効化することをお薦めします。

スナップショットを管理するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。Cloud Controlが使用可能でない場合は、DBMS_WORKLOAD_REPOSITORYパッケージに含まれるプロシージャを使用してスナップショットを管理します。DBMS_WORKLOAD_REPOSITORYパッケージに含まれるプロシージャを使用するには、Oracle DBAロールが必要です。CDBルートおよびPDBのスナップショットを作成、削除および変更するSQLプロシージャは、非CDB用のそれらのプロシージャと同じです。これらのSQLプロシージャは、プロシージャ・コールでターゲット・データベースの情報が指定されない場合、デフォルトでローカル・データベース上でその操作を実行します。

ノート:

- PDB レベルのスナップショットには一意のスナップショット ID があり、CDB レベルのスナップショットに関連していません。
- CDB 内の PDB の接続および切断操作によって、PDB に格納されている AWR データが影響を受けることはありません。
- CDB 管理者は、PDB に次の SQL 文を実行することによって、PDB ロックダウン・プロファイルを使用して PDB の AWR 機能を無効化できます。

```
SQL> alter lockdown profile profile_name disable feature=('AWR_ACCESS');
```

PDB で AWR 機能を無効化すると、その PDB に対するスナップショット操作を実行できなくなります。

次の SQL 文を PDB で実行すると、その PDB の AWR 機能を再度有効化できます。

```
SQL> alter lockdown profile profile_name enable feature=('AWR_ACCESS');
```

- スナップショット・データは、デフォルトでは CDB および PDB の SYSAUX 表領域に格納されます。Oracle Database 19c 以降では、スナップショット設定を変更することで、CDB および PDB のスナップショット・データを格納する他の表領域を指定できます。

関連項目:

- スナップショットの作成
- スナップショットの削除
- スナップショット設定の変更
- PDBロックダウン・プロファイルの詳細は、Oracle Databaseセキュリティ・ガイドを参照してください

AWRデータのエクスポートおよびインポート

マルチテナント環境のCDBルートおよびPDBのAWRデータをエクスポートおよびインポートするプロセスは、非CDBのAWRデータをエクスポートおよびインポートするプロセスに似ています。

関連項目:

- Oracle DatabaseからのAWRデータのエクスポートの詳細は、AWRデータのエクスポートを参照してください
- Oracle DatabaseへのAWRデータのインポートの詳細は、AWRデータのインポートを参照してください

マルチテナント環境のAWRデータの表示

Oracle Databaseの様々なレポートおよびビューを使用して、マルチテナント環境のAWRデータを表示できます。

AWRレポート

AWRレポートを生成するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してAWRレポートを生成してください。

関連項目:

Cloud Controlを使用したAWRレポートの生成の詳細は、Oracle Database 2日でパフォーマンス・チューニング・ガイドを参照してください

Cloud Controlを使用できない場合は、次のようにSQLスクリプトを実行して、AWRレポートを生成します。これらのスクリプトを実行するには、DBAロールが必要です。

- マルチテナント環境全体のグローバル・システム・データ統計を示す、CDB固有のAWRレポートをCDBルートから生成できます。ローカル・データベースのAWRレポートの生成の項で説明するSQLスクリプトを使用して、このAWRレポートを作成できます。
- PDBに関連する統計を示す、PDB固有のAWRレポートをPDBから生成できます。ローカル・データベースのAWRレポートの生成の項で説明するSQLスクリプトを使用して、このAWRレポートを作成できます。
- 特定のPDBに関連する統計を示す、PDB固有のAWRレポートをCDBルートから生成できます。特定データベースのAWRレポートの生成の項で説明するSQLスクリプトを使用して、このAWRレポートを作成できます。

AWRビュー

次の表に、マルチテナント環境のCDBルートおよび個別のPDBに格納されているAWRデータにアクセスするためのOracle Databaseのビューをリストします。

関連項目:

これらのAWRの詳細は、自動ワークロード・リポジトリ・ビューの使用を参照してください

表6-2 マルチテナント環境のAWRデータにアクセスするためのビュー

ビュー	説明
DBA_HIST ビュー	● DBA_HIST ビューには、CDB ルートにのみ存在する AWR データが表示されます。 ● CDB ルートから DBA_HIST ビューにアクセスした場合は、CDB ルートに格納されているすべての AWR データが表示されます。 ● PDB から DBA_HIST ビューにアクセスした場合は、その PDB に固有の、CDB ルートの AWR データのサブセットが表示されます。
DBA_HIST_CON ビュー	● DBA_HIST_CON ビューは DBA_HIST ビューに似ていますが、各コンテナに関するより詳細な情報が提供されるため、DBA_HIST ビューより多くのデータが含まれています。 ● DBA_HIST_CON ビューには、CDB ルートにのみ存在する AWR データが表示されます。 ● CDB ルートから DBA_HIST_CON ビューにアクセスした場合は、CDB ルートに格納されているすべての AWR データが表示されます。

ビュー	説明
	● PDB から DBA_HIST_CON ビューにアクセスした場合は、その PDB に固有の、CDB ルートの AWR データのサブセットが表示されます。
AWR_ROOT ビュー	● AWR_ROOT ビューは、Oracle Database 12c リリース 2 (12.2)以降の、マルチテナント環境でのみ使用できます。 ● AWR_ROOT ビューは DBA_HIST ビューと同等です。 ● AWR_ROOT ビューには、CDB ルートにのみ存在する AWR データが表示されます。 ● CDB ルートから AWR_ROOT ビューにアクセスした場合は、CDB ルートに格納されているすべての AWR データが表示されます。 ● PDB から AWR_ROOT ビューにアクセスした場合は、その PDB に固有の、CDB ルートの AWR データのサブセットが表示されます。
AWR_PDB ビュー	● AWR_PDB ビューは、Oracle Database 12c リリース 2 (12.2)以降で使用できます。 ● AWR_PDB ビューには、CDB ルートまたは PDB に存在するローカル AWR データが表示されます。 ● CDB ルートから AWR_PDB ビューにアクセスした場合は、CDB ルートに格納されている AWR データが表示されます。 ● PDB から AWR_PDB ビューにアクセスした場合は、その PDB に格納されている AWR データが表示されます。
CDB_HIST ビュー	● CDB_HIST ビューには、PDB に格納されている AWR データが表示されます。 ● CDB ルートから CDB_HIST ビューにアクセスした場合は、すべての PDB に格納されている AWR データの和集合が表示されます。

ビュー	説明
	● PDB から CDB_HIST ビューにアクセスした場合は、その PDB に格納されている AWR データが表示されます。

Active Data Guardスタンバイ・データベースでの自動ワークロード・リポジトリの管理

Oracle Database 12cリリース2 (12.2)からは、Active Data Guard (ADG)スタンバイ・データベースの自動ワークロード・リポジトリ(AWR)データを取得できるようになりました。この機能を使用すると、ADGスタンバイ・データベースのパフォーマンス関連の問題を分析できます。

ADGスタンバイ・データベースのAWRスナップショットは、リモート・スナップショットと呼ばれます。宛先と呼ばれるデータベース・ノードには、ソースと呼ばれるリモートADGスタンバイ・データベース・ノードから収集されたスナップショットが格納されます。

ADGプライマリ・データベースまたは非ADGデータベースのいずれも、宛先になることができます。宛先がADGプライマリ・データベースである場合、これはソース・データベースでもあり、そのスナップショットはローカル・スナップショットです。

ソースは、一意の名前または宛先にとって既知のソース名によって識別されます。

宛先ノードまたはソース・ノードには、その構成中に名前を割り当てることができます。そうしない場合、初期化パラメータDB_UNIQUE_NAMEの値がノードの名前として割り当てられます。

各ソースには、2つのデータベース・リンク(宛先からソースへのデータベース・リンクとソースから宛先へのデータベース・リンク)が存在する必要があります。これらのデータベース・リンクは、ADGデプロイメントの間に各ソースに構成されます。フェイルオーバー、スイッチオーバー、ホストの追加と削除などの特定のADGイベントの後に、データベース・アプリケーションが引き続き正しく動作するように、これらのデータベース・リンクを手動で再構成する必要があります。

リモート・スナップショットは、スケジュールされた時間間隔で自動的に、または手動で取得できます。リモート・スナップショットは、常に宛先ノードで開始されます。宛先でスナップショットの作成プロセスが開始されると、ソースはデータベース・リンクを使用して、自身のスナップショット・データを宛先にプッシュします。宛先に格納されているスナップショット・データまたはAWRデータには、AWRレポート、Oracle Databaseのインポートおよびエクスポート機能、ユーザー定義の問合せを使用してアクセスできます。自動データベース診断モニター(ADDM)アプリケーションは、AWRデータを使用して、データベースのパフォーマンスに関連した問題を分析します。

宛先データベースの役割

宛先データベースでは次のタスクが管理されます。

- ソースの登録
- 各ソースへの一意の識別子の割当て
- 宛先とソースの間のデータベース・リンクの作成
- ソースの自動スナップショットのスケジュールおよび開始
- ソースのスナップショットの調整による宛先ワークロードの管理

- 各ソースのスナップショット設定の管理
- 新しく生成されたスナップショットへの識別子の割当て
- AWR表のパーティション化
- ローカルAWRへのパフォーマンス・データの格納
- 宛先およびソースのAWRデータのパージ

ソース・データベースの役割

ソース・データベースでは次のタスクが管理されます。

- ローカルAWRへの自身のパフォーマンス・データの格納
- 宛先への自身のAWRデータの送信
- 宛先からのサービス・リクエストへの応答
- 宛先からのAWRデータの抽出

ADGスタンバイ・データベースでのAWR管理の主なステップ

ADGスタンバイ・データベースでAWRを管理するための主なステップは、次のとおりです。

1. リモート管理フレームワーク(RMF)の構成
2. Active Data Guardスタンバイ・データベースのスナップショットの管理
3. Active Data Guardスタンバイ・データベースのAWRデータの表示

ノート:

ADG 環境の AWR の構成を開始する前に、すべての ADG スタンバイ・データベースのデータベース・リンクが、ADG のデプロイ時にすでに構成されていることを確認してください。

リモート管理フレームワーク(RMF)の構成

リモート管理フレームワーク(RMF)は、Oracle Databaseのパフォーマンス統計(AWRデータ)を取得するためのアーキテクチャです。

ノート:

ADG スタンバイ・データベースおよびスタンドアロン・データベースに対してのみ、RMF を使用できます。

RMFトポロジは、すべての参加データベース・ノードとそのメタデータおよび接続情報で構成される、集中型アーキテクチャです。RMFトポロジには宛先と呼ばれるデータベース・ノードが1つ存在し、ソースと呼ばれるデータベース・ノードから収集されたパフォーマンス・データ(AWRデータ)を格納および管理する役割を持っています。宛先候補は、元の宛先が使用できないかダウングレードされた場合に、元の宛先を置き換えるように構成できるソースです。トポロジには、1つ以上の宛先候補と、宛先を1つのみ指定できます。

トポロジ内の各データベース・ノードには、一意の名前が割り当てられている必要があります。これは、ノードの構成時にプロシージャDBMS_UMF.configure_node()を使用して実行します。このプロシージャでノードの名前を指定しない場合は、初期化パラメータDB_UNIQUE_NAMEの値をノードの名前として使用します。

トポロジ内のデータベース・ノードは、データベース・リンクを使用して相互に通信します。宛先からソース、およびソースから宛先へのデータベース・リンクは、ADGのデプロイ時にADGスタンバイ・データベースごとに作成する必要があります。

サービスは、トポロジ上で実行されるアプリケーションです。たとえば、トポロジ上で実行されるAWRサービスによって、そのトポロジ内のすべてのデータベース・ノードに対するリモートAWRスナップショットが有効化されます。

RMF APIは、RMFトポロジを構成するために使用できるPL/SQLプロシージャおよび関数です。RMF APIは、PL/SQLパッケージDBMS_UMFで宣言されています。

ノート:

- SYS$UMF ユーザーは、システム・レベルの RMF ビューおよび表にアクセスするためのすべての権限を持つ、デフォルトのデータベース・ユーザーです。RMF の AWR に関連するすべての操作は、SYS$UMF ユーザーのみが実行できます。SYS$UMF ユーザーはデフォルトでロックされており、RMF トポロジをデプロイする前にロックを解除する必要があります。
- RMF トポロジでデータベース・リンクを作成する際には、SYS$UMF ユーザーのパスワードを指定する必要があります。SYS$UMF ユーザーのパスワードを変更した場合は、RMF トポロジ内のすべてのデータベース・リンクを再作成する必要があります。

関連項目:

DBMS_UMFパッケージの詳細は、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください

RMFトポロジの設定

Oracle Databaseのパフォーマンス統計を収集するように、RMFトポロジを設定する必要があります。

RMFトポロジを設定するための前提条件は、次のとおりです。

- RMFトポロジに登録するすべてのデータベース・ノードについて、宛先からソース、およびソースから宛先へのデータベース・リンクを作成する必要があります。この設定は、ADGのデプロイ時に行います。

RMFトポロジを設定するためのステップは、次のとおりです。

1. トポロジに追加するデータベース・ノードを構成します。
2. トポロジを作成します。
3. トポロジにデータベース・ノードを登録します。
4. (オプション)トポロジ内のノード間のデータベース・リンクを登録します。この構成は、宛先が使用できなくなった場合に、宛先候補がデータベース・リンクを使用してトポロジ内の残りのノードに接続するために必要です。

RMFトポロジの設定の例

この例では、T、S0およびS1の3つのデータベース・ノードをトポロジTopology_1に追加します。ノードTは宛先ノードであり、S0およびS1のノードはソース・ノードです。ノードS1は宛先候補であり、元の宛先Tが使用できない場合は、ノードS1が新しい宛先になります。AWRサービスは、トポロジ内のすべてのソースに対して有効化されています。

ADGのデプロイメント中に、次のデータベース・リンクがすでに作成されているとします。

- DBLINK_T_to_S0: TからS0へのデータベース・リンク。
- DBLINK_T_to_S1: TからS1へのデータベース・リンク。
- DBLINK_S0_to_T: S0からTへのデータベース・リンク。
- DBLINK_S0_to_S1: S0からS1へのデータベース・リンク。
- DBLINK_S1_to_T: S1からTへのデータベース・リンク。
- DBLINK_S1_to_S0: S1からS0へのデータベース・リンク。

RMFトポロジを設定するためのサンプル・コードを次に示します。

```
/* Configure the nodes T, S0, and S1 by executing these procedures */
/* Execute this procedure on node T */
SQL> exec DBMS_UMF.configure_node ('T');
/* Execute this procedure on node S0 */
SQL> exec DBMS_UMF.configure_node ('S0', 'DBLINK_S0_to_T');
/* Execute this procedure on node S1 */
SQL> exec DBMS_UMF.configure_node ('S1', 'DBLINK_S1_to_T');
/* Execute all the following procedures on the destination node T */
/* Create the topology 'Topology_1' */
SQL> exec DBMS_UMF.create_topology ('Topology_1');
/* Register the node S0 with the topology 'Topology_1' */
SQL> exec DBMS_UMF.register_node ('Topology_1',
                                  'S0',
                                  'DBLINK_T_to_S0',
                                  'DBLINK_S0_to_T',
                                  'TRUE'  /* Set it as a source */,
                                  'FALSE' /* Set it as not a candidate destination */);
/* Register the node S1 with the topology 'Topology_1' */
SQL> exec DBMS_UMF.register_node ('Topology_1',
                                  'S1',
                                  'DBLINK_T_to_S1',
                                  'DBLINK_S1_to_T',
                                  'TRUE'  /* Set it as a source */,
                                  'TRUE'  /* Set it as a candidate destination */);
/* Register the database links between the nodes S0 and S1 in the topology 'Topology_1'.
 * When destination T is unavailable at the time of failover, the source S0 can connect
 * to the candidate destination S1 using this database link.
 */
SQL> exec DBMS_UMF.create_link ('Topology_1',
                                'S0',
                                'S1',
                                'DBLINK_S0_to_S1',
                                'DBLINK_S1_to_S0');
/* Enable the AWR service on the node S0 in the topology 'Topology_1' */
SQL> exec DBMS_WORKLOAD_REPOSITORY.register_remote_database(node_name=>'S0');
/* Enable the AWR service on the node S1 in the topology 'Topology_1' */
```

```
SQL> exec DBMS_WORKLOAD_REPOSITORY.register_remote_database(node_name=>'S1');
```

ノート:

次のプロシージャを使用して、ノードに対する AWR サービスを無効化できます。

```
SQL> exec DBMS_WORKLOAD_REPOSITORY.unregister_remote_database(node_name)
```

ADGロール・トランジションの管理

ADGプライマリまたは元の宛先が失敗(フェイルオーバーイベント)したとき、または、メンテナンス・フェーズ(スイッチオーバー・イベント)中にADGスタンバイ・データベースまたは宛先候補がADGプライマリのロールを引き継いだときに、ADGロール・トランジションが発生します。

ロールを変更する前、つまり、フェイルオーバーまたはスイッチオーバー・イベントが発生したために宛先候補を新しい宛先へと変更する前に、次の構成ステップを実行することをお薦めします。

1. ソースと宛先候補の間にデータベース・リンクを作成します。各ソースに次のプロシージャを実行して、すべてのソースにこれが構成されている必要があります。

```
SQL> EXEC DBMS_UMF.CREATE_LINK (topology name,
                                 source name,
                                 candidate destination name,
                                 source to candidate destination database link,
                                 candidate destination to source database link);
```

ノート:

ロールの変更時に予期しない問題が発生することを避けるために、トポロジ内のすべてのノード間にデータベース・リンクを作成しておくことをお薦めします。

2. 宛先候補のAWRスナップショットを取得します。

ノート:

ロールの変更後に宛先候補の AWR レポートを生成するには、ロールの変更前に、宛先候補のスナップショットを少なくとも 1 つ取得します。

3. 宛先候補とすべてのソースを再起動します。

後続の構成ステップを完了したら、宛先候補で次のプロシージャを実行して、ロールを変更できます。

```
SQL> EXEC DBMS_UMF.SWITCH_DESTINATION(topology name, force_switch=>FALSE);
```

ノート:

ロールの遷移期間にはソースのスナップショットを取得しないことをお薦めします。DBMS_UMF.SWITCH_DESTINATION プロシージャを実行して、ロール変更プロセスが完了したら、ソースのスナップショットを取得できます。ロールの変更後にソースの AWR レポートを生成する場合は、ロールの変更後に取得されたスナップショットのみを選択する必要があります。

登録されたRMFトポロジの詳細の取得

次に説明するRMFビューには、マルチテナント環境に登録されているすべてのRMFトポロジの構成情報が表示されます。

表6-3 RMFビュー

RMFビュー	説明
DBA_UMF_TOPOLOGY	マルチテナント環境に登録されたすべてのトポロジが表示されます。各トポロジには、トポロジ名、宛先 ID、トポロジの状態があります。RMF を有効化するには、少なくとも 1 つのトポロジの状態が ACTIVE である必要があります。
DBA_UMF_REGISTRATION	マルチテナント環境内のすべてのトポロジの、登録されたすべてのノードが表示されます。
DBA_UMF_LINK	マルチテナント環境内のすべてのトポロジの、登録されたすべてのデータベース・リンクが表示されます。
DBA_UMF_SERVICE	マルチテナント環境内のすべてのトポロジの、登録されたすべてのサービスが表示されます。

関連項目:

これらのRMFビューの詳細は、『Oracle Databaseリファレンス』を参照してください

Active Data Guardスタンバイ・データベースのスナップショットの管理

ADGスタンバイ・データベースのAWRスナップショットは、リモート・スナップショットと呼ばれます。ローカルAWRスナップショットと同じように、リモートAWRスナップショットは、スケジュールされた時間間隔で自動的に、または手動で生成できます。リモート・スナップショットの生成には、プッシュ・オン・デマンド・メカニズムが使用されます。ここでは、宛先がスナップショットの生成プロセスを開始し、データベース・リンクを介してスナップショット・データを宛先にプッシュするように、ソースに対して指示します。宛先は、各ソースに対して構成されているスナップショットの時間間隔に基づいて、自動スナップショットを定期的に開始します。

ノート:

宛先は、個々のソースに設定されているスナップショット・データまたは AWR データの保持設定に基づいて、期限切れのリモート・スナップショットをパージします。ローカルで生成されたスナップショットのパージは、スケジュールされた通

常のパージ・プロセスで発生します。デフォルトでは、Oracle Database により、8 日を超えて AWR に格納されているスナップショットが自動的に消去されます。リモート・スナップショットの AWR 表のパーティション化は、ローカル・スナップショットと同様に実行されます。

リモート・スナップショットの作成、変更および削除

リモート・スナップショットを作成、変更および削除するAPIは、ローカル・スナップショットに対してそれらを実行するAPIと同じです。

ノート:

リモート・スナップショットを作成するために、DBMS_WORKLOAD_REPOSITORY.CREATE_REMOTE_SNAPSHOT API を使用することもできます。この API は、ローカル・スナップショットを作成する API DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT と同様に動作しますが、RMF トポロジ名のパラメータを追加で取得します。

関連項目:

- スナップショットの作成
- スナップショット設定の変更
- スナップショットの削除
- DBMS_WORKLOAD_REPOSITORY.CREATE_REMOTE_SNAPSHOT APIの構文は、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください。

リモート・スナップショットのベースラインの管理

リモート・スナップショットのベースラインを管理するAPIは、ローカル・スナップショットに対してそれを実行するAPIと同じです。

関連項目:

ベースラインの管理

リモート・スナップショットのエクスポートとインポート

ノート:

ADG スタンバイ・データベース(つまり、ソース・データベース)上で、リモート・スナップショットに関連した AWR のエクスポートおよびインポート・スクリプトを実行することはできません。これらのスクリプトは、常に宛先データベースで実行します。

リモート・スナップショットのAWRデータをエクスポートおよびインポートするプロセスは、ローカル・スナップショットに対してそれを実行するプロセスと同じです。Oracle Database 12cリリース2 (12.2)以降では、AWRデータのエクスポートおよびインポート・

スクリプトであるawrextr.sqlおよびawrload.sqlは、特定のソースで作成されたスナップショットを識別するために、ソース名識別子を使用します。ソース名は、エクスポート操作中にダンプ・ファイルに格納され、インポート操作中にデフォルト・ソース名として使用されます。

関連項目:

ローカル・スナップショットのAWRデータのエクスポートおよびインポートの詳細は、「別のシステムへの自動ワークロード・リポジトリ・データの転送」を参照してください。

awrextr.sqlスクリプトを使用したリモート・スナップショットのエクスポート

リモート・スナップショットをエクスポートするプロセスは、awrextr.sqlスクリプトを使用したローカル・スナップショットのエクスポート(「AWRデータのエクスポート」の項を参照)と似ていますが、次の点が異なります。

- エクスポート・ログ・ファイルのデフォルトのディレクトリはダンプ・ファイルと同じですが、エクスポート・ログ・ファイルには他の任意のディレクトリを指定できます。
- エクスポートするダンプ・ファイルの名前には、.dmp接尾辞を指定できます。
- エクスポート・スクリプトによって、エクスポートする「マップ済データベースID」値のプロンプトが表示される前に、AWR表のSOURCE_DBIDとSOURCE_NAMEの列値が表示されます。

awrload.sqlスクリプトを使用したリモート・スナップショットのインポート

リモート・スナップショットをインポートするプロセスは、awrload.sqlスクリプトを使用したローカル・スナップショットのインポート(「AWRデータのインポート」の項を参照)と似ていますが、次の点が異なります。

- インポート・ログ・ファイルのデフォルトのディレクトリはダンプ・ファイルと同じですが、インポート・ログ・ファイルには他の任意のディレクトリを指定できます。これは特に、ダンプ・ファイルが読取り専用ディレクトリに配置されている場合に便利です。
- インポートするダンプ・ファイルの名前には、.dmp接尾辞を指定できます。
- インポート・スクリプトは、ダンプ・ファイルにあるSOURCE_DBIDおよびSOURCE_NAMEの列値を使用して、AWRのスナップショットを格納するために使用する正しい「マップ済データベースID」を判断します。

ノート:

スナップショットのインポート操作は、スナップショットのダンプが生成された Oracle Database のバージョンの影響を受けません。

Active Data Guardスタンバイ・データベースのAWRデータの表示

Oracle提供のAWRビューおよびAWRレポートを使用して、ADGスタンバイ・データベースに格納されたAWRデータを表示できます。

AWRビューを使用したAWRデータの表示

「自動ワークロード・リポジトリ・ビューの使用」の項で説明しているとおり、DBA_HISTビューを使用して、AWRに格納された履歴

データを表示できます。

ノート:

Oracle Database 12c リリース 2 (12.2)以降では、ビューDBA_HIST_DATABASE_INSTANCE に、ADG スタンバイ・データベースの AWR をサポートするための列 DB_UNIQUE_NAME が含まれています。列 DB_UNIQUE_NAME には、宛先に既知であるソースの一意識別子が格納されます。

AWRレポートを使用したAWRデータの表示

AWRレポートを使用して、ADGスタンバイ・データベースに関連したパフォーマンス統計を表示できます。AWRレポートを生成するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してAWRレポートを生成してください。Cloud Controlを使用できない場合は、Oracle提供のSQLスクリプトを使用して、AWRレポートを生成します。これらのスクリプトを実行するには、DBAロールが必要です。

ソース名とマップ済データベースIDのペアを使用して、特定のソースのAWRデータを問い合せることができます。マップ済データベースIDは、AWRでデータベース・インスタンスを識別するために使用され、AWR表のDBID列に格納されている、データベース識別子(DBID)に似ています。ADGスタンバイ・データベースのAWR DBID値は、次のように導出されます。

- 宛先の場合、AWR DBID値はV$DATABASE.CON_DBIDの値です。
- ソースの場合、AWR DBID値は、DBMS_UMF.GET_NODE_ID_LOCAL()の値またはDBA_UMF_REGISTRATIONビューのNODE_ID列の値です。

スナップショットIDはソース間で一意ではないため、特定のソースのスナップショットは、スナップショットIDとマップ済データベースIDのペアで識別されます。

関連項目:

Oracle提供のSQLスクリプトを使用したAWRレポートの生成の詳細は、特定データベースのAWRレポートの生成を参照してください。

自動ワークロード・リポジトリ・レポートの生成

AWRレポートは、2つのスナップショット(または2つの時点)の間に取得されたデータを示します。AWRレポートは複数のセクションに分割されています。レポートの内容には、選択した範囲のスナップショットに関するシステムのワークロード・プロファイルが含まれます。HTMLレポートには、セクション間ですばやくナビゲートできるようにリンクが組み込まれています。

ノート:

指定した範囲のスナップショットにワークロード・アクティビティのないデータベースに対してレポートを実行すると、一部のレポート統計について 0 より小さいか 100 を超えるパーセンテージが計算される場合があります。この結果は、その統計に意味のある値がないことを意味します。

この項では、AWRレポートの生成方法を説明しており、内容は次のとおりです。

- AWRレポートを生成するためのユーザー・インタフェース
- コマンドライン・インタフェースを使用したAWRレポートの生成

AWRレポートを生成するためのユーザー・インタフェース

AWRレポートを生成するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してAWRレポートを生成してください。

Cloud Controlを使用できない場合は、SQLスクリプトを実行してAWRレポートを生成します。これらのスクリプトを実行するには、DBAロールが必要です。

関連項目:

Cloud Controlを使用したAWRレポートの生成の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

コマンドライン・インタフェースを使用したAWRレポートの生成

この項では、コマンドライン・インタフェースでSQLスクリプトを実行し、AWRレポートを生成する方法を説明します。これらのスクリプトを実行するには、DBAロールが必要です。必要なAWRレポートを生成するための詳細なステップは、次の表で適切なタスクのリンクをクリックしてください。

表6-4 AWRレポートを生成するためのSQLスクリプト

タスク	SQLスクリプト	説明
ローカル・データベースの AWR レポートの生成	awrrpt.sql	ローカル・データベース・インスタンスの一定範囲のスナップショット ID の統計を表示する AWR レポートを HTML またはテキスト形式で生成します。
特定データベースの AWR レポートの生成	awrrpti.sql	特定のデータベース・インスタンスの一定範囲のスナップショット ID の統計を表示する AWR レポートを HTML またはテキスト形式で生成します。
Oracle RAC におけるローカル・データベースの AWR レポートの生成	awrgrpt.sql	Oracle RAC 環境におけるローカル・データベース・インスタンスの一定範囲のスナップショット ID の統計を表示する AWR レポートを HTML またはテキスト形式で生成します。

タスク	SQLスクリプト	説明
Oracle RAC における特定データベースの AWR レポートの生成	awrgrpti.sql	Oracle RAC 環境における特定のデータベース・インスタンスの一定範囲のスナップショット ID の統計を表示する AWR レポートを HTML またはテキスト形式で生成します。
ローカル・データベースにおける SQL 文の AWR レポートの生成	awrsqrpt.sql	ローカル・データベース・インスタンスの一定範囲のスナップショット ID について、特定の SQL 文の統計を表示する AWR レポートを HTML またはテキスト形式で生成します。
特定データベースにおける SQL 文の AWR レポートの生成	awrsqrpi.sql	特定のデータベース・インスタンスの一定範囲のスナップショット ID について、特定の SQL 文の統計を表示する AWR レポートを HTML またはテキスト形式で生成します。

ローカル・データベースのAWRレポートの生成

awrrpt.sql SQLスクリプトでは、一定範囲のスナップショットIDの統計を表示する、HTMLまたはテキストのレポートが生成されます。

コマンドライン・インタフェースを使用して、ローカル・データベース・インスタンスにAWRレポートを生成するには:

1. SQLプロンプトで次のように入力します。

   ```
   @$ORACLE_HOME/rdbms/admin/awrrpt.sql
   ```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

   ```
   Enter value for report_type: text
   ```

   この例では、テキスト形式のレポートが選択されます。

3. スナップショットIDをリストする日数を指定します。

   ```
   Enter value for num_days: 2
   ```

   指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

4. ワークロード・リポジトリ・レポートの最初と最後のスナップショットIDを指定します。

   ```
   Enter value for begin_snap: 150
   Enter value for end_snap: 160
   ```

   この例では、スナップショットIDが150のスナップショットが最初のスナップショットとして選択され、スナップショットIDが

160のスナップショットが最後のスナップショットとして選択されます。

5. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrrpt_1_150_160
```

この例では、デフォルト名が使用され、awrrpt_1_150_160というAWRレポートが生成されます。

特定データベースのAWRレポートの生成

awrrpti.sql SQLスクリプトでは、特定のデータベース・インスタンスを使用して、一定範囲のスナップショットIDの統計を表示する、HTMLまたはテキストのレポートが生成されます。このスクリプトでは、AWRレポートの生成に使用されるデータベース識別子およびインスタンスを指定できます。

コマンドライン・インタフェースを使用して、特定のデータベース・インスタンスにAWRレポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrrpti.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: text
```

この例では、テキスト形式のレポートが選択されます。

使用可能なデータベース識別子およびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
```

3. 次のようにデータベース識別子(dbid)およびインスタンス番号(inst_num)の値を入力します。

```
Enter value for dbid: 3309173529
Using 3309173529 for database Id
Enter value for inst_num: 1
```

ノート:

ADG スタンバイ・データベースの場合、dbid の値は次のように決定されます。

- 宛先ノードには、v$database.con_dbid の値を使用します。
- ソース・ノードには、dbms_umf.get_node_id_local () の値を使用します。

4. スナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 2
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

5. ワークロード・リポジトリ・レポートの最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 150
Enter value for end_snap: 160
```

この例では、スナップショットIDが150のスナップショットが最初のスナップショットとして選択され、スナップショットIDが160のスナップショットが最後のスナップショットとして選択されます。

6. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrrpt_1_150_160
```

この例では、デフォルト名が使用され、3309173529というデータベースIDを持つデータベース・インスタンスでawrrpt_1_150_160というAWRレポートが生成されます。

Oracle RACにおけるローカル・データベースのAWRレポートの生成

awrgrpt.sql SQLスクリプトでは、Oracle Real Application Clusters (Oracle RAC)環境における現在のデータベース・インスタンスを使用して、一定範囲のスナップショットIDの統計を表示する、HTMLまたはテキストのレポートが生成されます。

ノート:

HTML のレポートは、テキストのレポートよりも読みやすいため、Oracle RAC 環境では、テキストではなく、HTML レポートを生成することをお薦めします。

コマンドライン・インタフェースを使用して、ローカル・データベース・インスタンスにOracle RAC AWRレポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrgrpt.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

3. スナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 2
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最終日に取得されたスナップショットが表示されます。

4. ワークロード・リポジトリ・レポートの最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 150
Enter value for end_snap: 160
```

この例では、スナップショットIDが150のスナップショットが最初のスナップショットとして選択され、スナップショットIDが

160のスナップショットが最後のスナップショットとして選択されます。

5. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrrpt_rac_150_160.html
```

この例では、デフォルト名が使用され、awrrpt_rac_150_160.htmlというAWRレポートが生成されます。

Oracle RACにおける特定データベースのAWRレポートの生成

awrgrpti.sql SQLスクリプトでは、Oracle RAC環境で実行されている特定のデータベース・インスタンスを使用して、一定範囲のスナップショットIDの統計を表示する、HTMLまたはテキストのレポートが生成されます。このスクリプトでは、AWRレポートの生成に使用されるデータベース識別子およびデータベース・インスタンスのカンマ区切りのリストを指定できます。

ノート:

HTML のレポートは、テキストのレポートよりも読みやすいため、Oracle RAC 環境では、テキストではなく、HTML レポートを生成することをお薦めします。

コマンドライン・インタフェースを使用して、特定のデータベース・インスタンスにOracle RAC AWRレポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrgrpti.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

使用可能なデータベース識別子およびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
 3309173529        2 TINT251      tint252      samp252
```

3. データベース識別子(dbid)の値を入力します。

```
Enter value for dbid: 3309173529
Using 3309173529 for database Id
```

4. レポートに含めるOracle RACインスタンスのインスタンス番号(instance_numbers_or_all)の値を入力します。

```
Enter value for instance_numbers_or_all: 1,2
```

5. スナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 2
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

6. ワークロード・リポジトリ・レポートの最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 150
Enter value for end_snap: 160
```

この例では、スナップショットIDが150のスナップショットが最初のスナップショットとして選択され、スナップショットIDが160のスナップショットが最後のスナップショットとして選択されます。

7. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrrpt_rac_150_160.html
```

この例では、デフォルト名が使用され、3309173529というデータベースIDを持つデータベース・インスタンスでawrrpt_rac_150_160.htmlというAWRレポートが生成されます。

ローカル・データベースにおけるSQL文のAWRレポートの生成

awrsqrpt.sql SQLスクリプトでは、一定範囲のスナップショットIDにおける特定のSQL文の統計を表示する、HTMLまたはテキストのレポートが生成されます。SQL文のパフォーマンスを検査またはデバッグする場合にこのレポートを実行します。

コマンドライン・インタフェースを使用して、ローカル・データベース・インスタンスにSQL文のAWRレポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrsqrpt.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

3. スナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 1
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、前日に取得されたスナップショットが表示されます。

4. ワークロード・リポジトリ・レポートの最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 146
Enter value for end_snap: 147
```

この例では、スナップショットIDが146のスナップショットが最初のスナップショットとして選択され、スナップショットIDが147のスナップショットが最後のスナップショットとして選択されます。

5. 特定のSQL文のSQL IDを指定して統計を表示します。

```
Enter value for sql_id: 2b064ybzkwf1y
```

この例では、2b064ybzkwf1yというSQL IDを持つSQL文が選択されます。

6. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrrpt_1_146_147.html
```

この例では、デフォルト名が使用され、awrrpt_1_146_147というAWRレポートが生成されます。

特定データベースにおけるSQL文のAWRレポートの生成

awrsqrpi.sql SQLスクリプトでは、特定のデータベース・インスタンスを使用して、一定範囲のスナップショットIDにおける特定のSQL文の統計を表示する、HTMLまたはテキストのレポートが生成されます。このスクリプトでは、AWRレポートの生成対象となるデータベース識別子およびインスタンスを指定できます。このレポートは、特定のデータベースおよびインスタンスにおけるSQL文のパフォーマンスを調査またはデバッグする場合に実行します。

コマンドライン・インタフェースを使用して、特定のデータベース・インスタンスにSQL文のAWRレポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrsqrpi.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

使用可能なデータベース識別子およびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
```

3. 次のようにデータベース識別子(dbid)およびインスタンス番号(inst_num)の値を入力します。

```
Enter value for dbid: 3309173529
Using 3309173529 for database Id
Enter value for inst_num: 1
Using 1 for instance number
```

4. スナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 1
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、前日に取得されたスナップショットが表示されます。

5. ワークロード・リポジトリ・レポートの最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 146
Enter value for end_snap: 147
```

この例では、スナップショットIDが146のスナップショットが最初のスナップショットとして選択され、スナップショットIDが147のスナップショットが最後のスナップショットとして選択されます。

6. 特定のSQL文のSQL IDを指定して統計を表示します。

```
Enter value for sql_id: 2b064ybzkwf1y
```

この例では、2b064ybzkwf1yというSQL IDを持つSQL文が選択されます。

7. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrrpt_1_146_147.html
```

この例では、デフォルト名が使用され、3309173529というデータベースIDを持つデータベース・インスタンスでawrrpt_1_146_147というAWRレポートが生成されます。

パフォーマンス・ハブ・アクティブ・レポートの生成

EM Expressのパフォーマンス・ハブ機能では、指定した期間のすべてのパフォーマンス・データの統合ビューを示すアクティブ・レポートが提供されます。レポートは完全にインタラクティブです。この内容はHTMLファイルに保存され、このファイルはWebブラウザを使用してオフラインでアクセスできます。

関連項目:

EM Expressのパフォーマンス・ハブ機能の詳細は、『Oracle Database 2日でデータベース管理者』を参照してください

この項では、パフォーマンス・ハブ・アクティブ・レポートの生成方法を説明しており、内容は次のとおりです。

- パフォーマンス・ハブ・アクティブ・レポートの概要
- パフォーマンス・ハブ・アクティブ・レポートを生成するためのコマンドライン・ユーザー・インタフェース
- SQLスクリプトを使用したパフォーマンス・ハブ・アクティブ・レポートの生成

パフォーマンス・ハブ・アクティブ・レポートの概要

パフォーマンス・ハブ・アクティブ・レポートを使用すると、指定した期間に使用可能なすべてのパフォーマンス・データが表示されます。期間に対してリアルタイム・データまたは履歴データのいずれを選択するかによって、パフォーマンス・ハブで使用可能なタブが異なります。リアルタイム・データを選択した場合、最後の時間のリアルタイム・データが表示されるため、より詳細なデータが表示されます。履歴データを選択した場合は、より詳細なデータが表示されますが、データ・ポイントは選択した期間の自動ワークロード・リポジトリ(AWR)の間隔に平均化されます。

この項では、パフォーマンス・ハブ・アクティブ・レポートについて説明しており、内容は次のとおりです。

- パフォーマンス・ハブ・アクティブ・レポートのタブについて
- パフォーマンス・ハブ・アクティブ・レポートのタイプについて

パフォーマンス・ハブ・アクティブ・レポートのタブについて

パフォーマンス・ハブ・アクティブ・レポートには、様々なパフォーマンス領域に分類されたパフォーマンス・データを表示してナビゲートできるインタラクティブ・タブが含まれます。

パフォーマンス・ハブ・アクティブ・レポートに含まれるタブには次のものがあります。

- サマリー

  「サマリー」タブにはリソース使用量、平均アクティブ・セッションおよびロード・プロファイル情報などのシステム・パフォーマンスの概要が示されます。このタブはリアルタイム・データ、ならびに履歴データに使用できます。

- アクティビティ

  「アクティビティ」タブにはASH分析が表示されます。このタブはリアルタイム・データ、ならびに履歴データに使用できます。

- ワークロード

  「ワークロード」タブには、コール率、ログオン率および上位SQLなどのワークロード・プロファイルに関するメトリック情報が表示されます。このタブはリアルタイム・データ、ならびに履歴データに使用できます。

- RAC

  「RAC」タブには、受け取ったグローバル・キャッシュ・ブロック数および平均ブロック待機時間などのOracle RAC固有のメトリックが表示されます。このタブはOracle RAC環境でのみ使用できます。このタブはリアルタイム・データ、ならびに履歴データに使用できます。

- 監視されたSQL

  「監視されたSQL」タブには、監視対象のSQL文に関する情報が表示されます。このタブはリアルタイム・データ、ならびに履歴データに使用できます。

- ADDM

  「ADDM」タブには、ADDM分析タスクとリアルタイムADDM分析レポートの情報が表示されます。このタブはリアルタイム・データ、ならびに履歴データに使用できます。

- 最新のADDM結果

  「最新のADDM結果」タブには、過去5分間のシステム・パフォーマンスのリアルタイム分析が表示されます。このタブは、パフォーマンス・ハブ・アクティブ・レポートの指定期間が過去の時間内にある場合のみ使用できます。このタブはリアルタイム・データのみに使用できます。

- データベース時間

  「データベース時間」タブには、様々なメトリックのカテゴリ別の待機イベントが表示されます。このタブは履歴データのみに使用できます。

- リソース

  「リソース」タブには、オペレーティング・システムおよびI/Oの使用状況統計が表示されます。このタブは履歴データのみに使用できます。

- システム統計

  「システム統計」タブにはデータベースおよびシステム統計が表示されます。このタブは履歴データのみに使用できます。

パフォーマンス・ハブ・アクティブ・レポートのタイプについて

レポート・タイプを選択することで、パフォーマンス・ハブ・アクティブ・レポートの各タブ内に表示される詳細のレベルを選択できます。

パフォーマンス・ハブ・アクティブ・レポートに使用可能なレポート・タイプには次のものがあります。

- 基本

  すべてのタブの基本情報のみがレポートに保存されます。

- 標準

  基本レポート・タイプに保存された情報以外に、「監視されたSQL」タブに含まれた上位SQL文のSQL監視情報とADDMレポートがレポートに保存されます。

- すべて

  標準レポート・タイプに保存された情報以外に、「監視されたSQL」タブに含まれた全SQL文のSQL監視情報と、すべてのタブに対するすべての詳細レポートがレポートに保存されます。

パフォーマンス・ハブ・アクティブ・レポートを生成するためのコマンドライン・ユーザー・インタフェース

次のいずれかの方法でコマンドライン・インタフェースを使用して、パフォーマンス・ハブ・アクティブ・レポートを生成できます。

- 「SQLスクリプトを使用したパフォーマンス・ハブ・アクティブ・レポートの生成」の説明に従った、SQLスクリプトの使用。
- 『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』の説明に従った、DBMS_PERFパッケージの使用。

SQLスクリプトを使用したパフォーマンス・ハブ・アクティブ・レポートの生成

この項では、コマンドライン・インタフェースでperfhubrpt.sql SQLスクリプトを実行して、パフォーマンス・ハブ・アクティブ・レポートを生成する方法を説明します。このスクリプトの実行にはDBAロールが必要です。

パフォーマンス・ハブ・アクティブ・レポートを生成するには:

1. SQLプロンプトで次のように入力します。

   ```
   @$ORACLE_HOME/rdbms/admin/perfhubrpt.sql
   ```

2. 目的のレポート・タイプを指定します。

   ```
   Please enter report type: typical
   ```

   使用可能なレポート・タイプの詳細は、「パフォーマンス・ハブ・アクティブ・レポートのタイプについて」を参照してください。

3. 使用するデータベースのデータベース識別子の値を入力します。

   ```
   Please enter database ID: 3309173529
   ```

   ローカル・データベースを使用する場合は、null値(デフォルト値)を入力します。ローカル・データベース以外のデータベースのデータベース識別子を指定する場合、パフォーマンス・ハブ・アクティブ・レポートはインポートされたAWRデータから生成されます。

4. 使用するデータベース・インスタンスのインスタンス番号の値を入力します。

   ```
   Please enter instance number: all instances
   ```

   すべてのインスタンスを指定する場合は、all instances(デフォルト値)と入力します。

5. 終了時間および開始時間をdd:mm:yyyy hh:mi:ssの形式で指定して、目的の期間を入力します。

   ```
   Please enter end time in format of dd:mm:yyyy hh24:mi:ss: 03:04:2014 17:00:00
   ```

```
Please enter start time in format of dd:mm:yyyy hh24:mi:ss: 03:04:2014 16:00:00
```

6. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name: my_perfhub_report.html
```

この例では、2014年4月3日の午後4時から午後5時までの指定期間について、データベースID値3309173529のすべてのデータベース・インスタンスでmy_perfhub_reportというパフォーマンス・ハブ・アクティブ・レポートが生成されます。

7 自動パフォーマンス診断

この章では、Oracle Databaseの自動パフォーマンス診断およびチューニング機能について説明します。

この章のトピックは、次のとおりです:

- 自動データベース診断モニターの概要
- ADDMの設定
- ADDMを使用したデータベース・パフォーマンスの問題の診断
- ADDMビュー

関連項目:

Oracle Enterprise Manager Cloud Control (Cloud Control)で、自動データベース診断モニターを使用してデータベースの診断およびチューニングを行う方法の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

自動データベース診断モニターの概要

自動ワークロード・リポジトリ(AWR)には、Oracleデータベースのパフォーマンス関連統計が格納されます。自動データベース診断モニター(ADDM)は、AWRデータの定期的な分析、パフォーマンスに関する問題の根本原因の特定、問題を修正するための推奨事項の提示、問題のないシステム領域の特定を行う診断ツールです。AWRはパフォーマンスに関する履歴データのリポジトリであるため、ADDMでは、イベントの後にパフォーマンスの問題を分析でき、時間とリソースをかけて問題を再現する必要がありません。

多くの場合、データベース管理者はパフォーマンスの問題に関して通知を受け取った時点で、まずADDM出力を調べる必要があります。ADDMには次の利点があります。

- デフォルトで1時間ごとの自動パフォーマンス診断レポート
- 数十人のチューニング専門家に基づく問題の診断
- 問題の影響と推奨事項による利点の時間ベースの数量化
- 症状ではなく根本的な原因の特定
- 問題の根本的な原因の処理に関する推奨事項
- システムのうち正常な領域の特定
- 診断処理中のシステムに対する最小限のオーバーヘッド

チューニングは反復的なプロセスであり、ある問題を修復したことが原因でシステムの他の部分がボトルネックになる場合があります。ADDM分析による利点があるとしても、システム・パフォーマンスが許容レベルに達するまでには複数のチューニング・サイクルを必要とする場合があります。ADDMによる利点は本番システムに適用されるのみでなく、開発およびテスト・システムでもパフォーマンスの問題を早期に警告できます。

この項では、次の項目について説明します。

- ADDM分析
- Oracle Real Application ClustersでのADDMの使用方法
- マルチテナント環境でのADDMの使用
- リアルタイムADDM分析
- ADDMの分析結果
- ADDMの分析結果の確認：例

ノート:

プラガブル・データベース(PDB)とともに ADDM 機能を使用している場合には、データ可視性および権限の要件が異なります。ADDM 機能などの管理機能がマルチテナント・コンテナ・データベース(CDB)で機能する仕組みの詳細は、Oracle Multitenant 管理者ガイドを参照してください。

ADDM分析

ADDM分析は、AWRスナップショットのペアと、同じデータベースのインスタンスのセットに対して実行できます。AWRスナップショットのペアにより分析の期間が定義され、インスタンスのセットにより分析のターゲットが定義されます。

Oracle Real Application Clusters (Oracle RAC)を使用している場合、ADDMには次の3つの分析モードがあります。

- データベース

  データベース・モードでは、ADDMによりデータベースのすべてのインスタンスが分析されます。

- インスタンス

  インスタンス・モードでは、ADDMによりデータベースの特定のインスタンスが分析されます。

- 部分

  部分モードでは、ADDMによりすべてのデータベース・インスタンスのサブセットが分析されます。

Oracle RACを使用しない場合は、存在するデータベース・インスタンスが1つのみであるため、ADDMはインスタンス・モードでのみ動作します。

ADDM分析はAWRスナップショットが作成されるたびに実行され、結果がデータベースに保存されます。ADDMにより分析される期間は、最新の2つのスナップショットにより定義されます(デフォルトでは最後の1時間)。ADDMでは、インスタンス・モードで指定されたインスタンスが常に分析されます。Oracle RAC以外の環境または単一インスタンス環境の場合、インスタンス・モードで実行される分析は、データベース全体の分析と同じになります。Oracle RACを使用している場合、「Oracle Real Application ClustersでのADDMの使用方法」に記載されているとおり、ADDMではデータベース・モードでもデータベース全体を分析します。

ADDMで分析が完了すると、Cloud Control、DBMS_ADDMパッケージ・サブプログラム、またはDBA_ADDM_*ビューとDBA_ADVISOR_*ビューのいずれかを使用してADDM結果を表示できます。

ADDM分析は、トップダウン方式で実行され、最初に症状を識別し、次にパフォーマンスの問題の根本的な原因に到達するまで精査します。分析の目標は、DB timeと呼ばれる1つのスループット・メトリックを減少させることです。DB timeは、データベース・パフォーマンスの基本的な測定基準で、データベースがユーザー・リクエストの処理に費やした累積時間を表します。これには、アイドル状態でないすべてのユーザー・フォアグラウンド・セッションの待機時間とCPU時間が含まれます。DB timeは、V$SESS_TIME_MODELビューおよびV$SYS_TIME_MODELビューに表示されます。

DB timeの短縮により、データベースで同じリソースを使用してサポートできるユーザー・リクエストの数が増加し、スループットが改善されます。ADDMによりレポートされる問題は、それに関係するDB timeの量でソートされます。DB timeの大部分に関係しないシステム領域は、正常領域としてレポートされます。

ADDMでは、次のようなタイプの問題が考慮されます。

- CPUのボトルネック - システムCPUがOracle Databaseまたは他のアプリケーションによりバインドされているかどうか。
- 過小なメモリー構造 - SGA、PGA、バッファ・キャッシュなど、Oracle Databaseメモリー構造が十分なサイズに設定されているかどうか。
- I/O能力の問題 - I/Oサブシステムが予想どおりに動作しているかどうか。
- 高負荷のSQL文: システム・リソースを過剰に使用しているSQL文があるかどうか。
- 高負荷のPL/SQL実行およびコンパイルと、高負荷のJavaの使用
- Oracle RAC固有の問題 - グローバル・キャッシュのホット・ブロックおよびオブジェクトの問題、相互接続遅延の問題があるかどうか。
- アプリケーションによるOracle Databaseの不適切な使用 - 不適切な接続管理、過剰な解析またはアプリケーション・レベルのロック競合による問題があるかどうか。
- データベース構成の問題 - 不適切なログ・ファイル・サイズの設定、アーカイブの問題、過剰なチェックポイントまたは不適切なパラメータ設定を示す兆候があるかどうか。
- 同時実行性の問題 - バッファ・ビジーの問題があるかどうか。
- 様々な問題領域のホット・オブジェクトおよび上位SQL。

ノート:

これは、ADDM の分析で考慮されるすべての問題タイプの包括的なリストではありません。

ADDMでは、システムの正常領域もドキュメント化されます。たとえば、システムのパフォーマンスにほとんど影響しない待機イベント・クラスが識別され、早期段階でチューニング考慮事項から削除されます。これにより、システム全体のパフォーマンスに影響しない項目に費やされる時間および労力が節約されます。

関連項目:

- V$SESS_TIME_MODELおよびV$SYS_TIME_MODELビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

- 時間モデル統計およびDB timeの詳細は、「時間モデル統計」を参照してください
- サーバー・プロセスの詳細は、『Oracle Database概要』を参照してください。

Oracle Real Application ClustersでのADDMの使用方法

Oracle RACを使用する場合は、データベース分析モードでADDMを実行して、すべてのデータベース・インスタンスのスループット・パフォーマンスを分析します。データベース・モードでは、ADDMは、DB時間をすべてのデータベース・インスタンスのデータベース時間の合計とみなします。データベース分析モードを使用すると、データベース全体にとって重要なすべての検出結果が単一のレポートに表示されます(各インスタンスの個別レポートは表示されません)。

データベース・モードのレポートには、データベース・リソース(I/Oやインターコネクトなど)に関する検出結果が含まれます。レポートには、データベース全体にとって重要な場合は、様々なインスタンスからの検出結果も集計されます。たとえば、単一のインスタンスに対するCPU負荷が高く、データベース全体に影響する場合、データベース・モード分析に、問題の原因である特定のインスタンスを示す検出結果が表示されます。

関連項目:

Oracle RACでのADDMの使用の詳細は、Oracle Real Application Clusters管理およびデプロイメント・ガイドを参照してください

マルチテナント環境でのADDMの使用

Oracle Database 12c以降、ADDMはマルチテナント・コンテナ・データベース(CDB)のルート・コンテナでデフォルトで有効になっています。Oracle Database 19c以降では、プラガブル・データベース(PDB)でもADDMを使用できます。

CDBでは、ADDMは非CDBでの動作と同様に機能します。つまり、ADDM分析はAWRスナップショットがCDBルートまたはPDBで取得されるたびに実行され、ADDM結果はスナップショットが取得されたのと同じデータベース・システムに格納されます。ADDMにより分析される期間は、最新の2つのスナップショットにより定義されます(デフォルトでは最後の1時間)。

ADDMで分析が完了したら、次のいずれかの方法を使用してADDM結果を表示できます。

- Enterprise Manager Cloud Control (Cloud Control)の使用
- DBA_ADDM_*ビューおよびDBA_ADVISOR_*ビューの使用

ノート:

- CDB ルートでは ADDM はデフォルトで有効化されています。
- PDB では自動 AWR スナップショットがデフォルトで無効になっているため、デフォルトでは ADDM は PDB で機能しません。PDB で ADDM を使用するには、PDB で自動 AWR スナップショットを有効にする必要があります。
- 現在のコンテナが CDB ルートであるユーザーは、CDB 全体の ADDM 結果を表示できます。ADDM 結果には、複数の PDB に関する情報が含まれる場合があります。PDB が切断されていると、PDB に関連する ADDM 結果は含まれません。現在のコンテナが PDB である場合、CDB ルートに格納されている

ADDM 結果を表示することはできません。

- PDB に対する ADDM 結果では、PDB 固有の結果および推奨事項のみが提供されます。現在のコンテナが PDB であるユーザーは、現在の PDB のみの ADDM 結果を表示できます。ADDM 結果には、CDB 全体に該当する診断結果(バッファ・キャッシュ・サイズに関連する I/O の問題など)は含まれません。
- PDB で AWR スナップショットを有効にしても、CDB ルートの ADDM レポートは変更されません
- PDB の AWR データは CDB ルートからアクセスできません

PDBレベルのADDMの制限事項

非CDBとは異なり、ADDMはPDBの次の問題を報告しません。これらの問題はCDB全体に適用され、個々のPDBに適用されないためです。

- 次の原因によるI/Oの問題:
  - バッファ・キャッシュのサイズが小さい
  - ストリーム・プールのサイズが小さい
  - 一時的な書込みが多すぎる
  - チェックポイントの書込みが多すぎる
  - UNDOの書込みが多すぎる
  - PQチェックポイントの書込みが多すぎる
  - 切捨ての書込みが多すぎる
  - 表領域DDLのチェックポイントが多すぎる
  - I/O容量の制限
- 次の原因によるSQLハード解析の問題:
  - カーソルのエージング
  - メモリー不足により失敗した解析
- SGAサイズの設定の問題

ADDMではPDBの次の問題もレポートされません。これらの問題はPDBレベルで解決できないためです。

- ネットワークの待機時間、輻輳、競合、損失ブロックなどのクラスタのメッセージ機能関連の問題
- ログ・ファイル・スイッチがアーカイブおよびチェックポイント未完了を待機している
- 空きバッファの待機が多すぎる
- ログ・バッファの待機の競合
- CPUボトルネックによる待機
- オペレーティング・システムのVMページング
- セッション・スロット待機イベント
- CPU Quantum待機イベント
- RMAN関連の待機イベント(PQキュー待機イベント、PGA制限待機イベント、I/Oキュー待機イベントなど)

関連項目:

- PDBでADDMを有効にする方法の詳細は、プラガブル・データベースでのADDMの有効化を参照してください

- ADDMビューのDBA_ADDM_*およびDBA_ADVISOR_*の詳細は、ADDMビューを参照してください
- DBMS_ADDMパッケージのサブプログラムを使用してOracle DatabaseでADDMを実行する方法の詳細は、「ADDMを使用したデータベース・パフォーマンスの問題の診断」を参照してください。
- Cloud Controlを使用してOracle DatabaseでADDMを実行する方法の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください。

プラガブル・データベースでのADDMの有効化

PDBでは自動AWRスナップショットがデフォルトで無効になっているため、デフォルトではADDMはプラガブル・データベース(PDB)で機能しません。PDBでADDMを使用するには、AWR_PDB_AUTOFLUSH_ENABLED初期化パラメータをTRUEに設定し、AWRスナップショット間隔を0より大きく設定して、PDBの自動AWRスナップショットを有効にする必要があります。

PDBでADDMを有効にするには:

1. 次のコマンドを使用して、PDBでAWR_PDB_AUTOFLUSH_ENABLED初期化パラメータをTRUEに設定します。

```
SQL> ALTER SYSTEM SET AWR_PDB_AUTOFLUSH_ENABLED=TRUE;
```

2. 次の例に示すようなコマンドを使用して、PDBでAWRスナップショット間隔を0より大きい値に設定します。

```
SQL> EXEC dbms_workload_repository.modify_snapshot_settings(interval=>60);
```

関連項目:

- AWR_PDB_AUTOFLUSH_ENABLED初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください
- DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGSプロシージャの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください。

リアルタイムADDM分析

Oracle Enterprise Manager Cloud Control (Cloud Control) 12cで導入されたリアルタイムADDMを使用すると、これまではデータベースの再起動が必要であった、応答しないデータベースやハングしているデータベースの問題を分析および解決できます。リアルタイムADDMは、事前定義された一連の基準を調査し、データベースの現在の問題を分析します。問題の分析後は、リアルタイムADDMにより、デッドロック、ハング、共有プールの競合やその他の例外的な状況などの特定された問題を、データベースを再起動することなく解決できます。

この項では、リアルタイムADDMについて説明しており、内容は次のとおりです。

- リアルタイムADDM接続モード
- リアルタイムADDMのトリガー
- リアルタイムADDMのトリガー・コントロール

関連項目:

Cloud ControlでリアルタイムADDMを使用する方法の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガ

イド』を参照してください

リアルタイムADDMの接続モード

Cloud Controlを使用してデータベースに接続する際、リアルタイムADDMではデータベースの状態に応じて、2つの異なるタイプの接続モードが使用されます。

- 通常接続

  このモードでは、リアルタイムADDMにより、データベースへの通常のJDBC接続が実行されます。このモードは、一部の接続が利用できる場合に、広範囲にわたるパフォーマンス分析を行うことを目的としています。

- 診断接続

  このモードでは、リアルタイムADDMにより、データベースへのラッチレス接続が実行されます。このモードは、通常のJDBC接続を実行できない極端なハング状況で使用します。

リアルタイムADDMのトリガー

Oracle Database 12c以降、リアルタイムADDMにより、データベースの一時的なパフォーマンス問題の検出が可能になりました。これを実行するために、リアルタイムADDMが3秒ごとに自動的に実行され、インメモリー・データを使用して、データベース内のパフォーマンス・スパイクが診断されます。

次のステップで説明するように、パフォーマンスの問題が検出されると、リアルタイムADDMにより分析が自動的にトリガーされます。

1. 管理性モニター・プロセス(MMON)により、ロックやラッチを発生させることなく、パフォーマンス統計を取得するためのアクションが3秒ごとに実行されます。
2. MMONプロセスは統計を確認し、表7-1にリストされている問題が検出された場合には、リアルタイムADDM分析をトリガーします。
3. MMONスレーブ・プロセスによりレポートが作成され、AWRに格納されます。

   レポートのメタデータを表示するには、DBA_HIST_REPORTSビューを使用します。

表7-1に、リアルタイムADDM分析がトリガーされる問題と条件をリストします。

表7-1 リアルタイムADDMがトリガーされる問題および条件

問題	条件
高負荷	平均のアクティブ・セッション数が、CPU コア数の 3 倍を超えている場合
I/O バウンド	アクティブ・セッションに対する I/O の影響が単一ブロック読取りのパフォーマンスに基づいている場合
CPU バウンド	アクティブ・セッションが合計負荷の 10%を超えていて、CPU 使用率が 50%を超えている場合

問題	条件
メモリーの過剰割当て	メモリー割当てが物理メモリーの 95%を超えている場合
インターコネクト・バウンド	単一ブロックのインターコネクト転送時間に基づく場合
セッション制限	セッション制限が 100%に近い場合
プロセス制限	プロセス制限が 100%に近い場合
ハング・セッション	ハング・セッションが合計セッションの 10%を超える場合
デッドロックの検出	デッドロックが検出された場合

関連項目:

DBA_HIST_REPORTSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

リアルタイムADDMのトリガー・コントロール

自動トリガーが大量のシステム・リソースを消費したり、システムに負荷をかけたりしないように、リアルタイムADDMには、次のコントロールが用意されています。

- レポート間の期間

  自動トリガーによって、過去5分の間にリアルタイムADDMレポートが作成されている場合、新しいレポートは生成されません。

- Oracle RACコントロール

  自動トリガーはデータベース・インスタンスに対してローカルです。ロックが必要であり、実際にレポートが生成される前にMMONスレーブ・プロセスによって問合せが実行されるため、Oracle RACの場合、任意の時点でリアルタイムADDMレポートを作成できるのは1つのデータベース・インスタンスのみです。

- 繰返しトリガー

  任意の問題に対する自動トリガーは、過去45分以内に起きた同じ問題のトリガーに関する前回のレポートと比較して、100%以上大きい影響があることが必要です。たとえば、8セッションの影響があるアクティブ・セッションに対してレポートがトリガーされた場合、その後45分以内に別のレポートをトリガーするには、少なくとも16のアクティブ・セッションが存在する必要があります。この場合、データベースでレポートされる問題は、時間の経過とともにより重大になっています。一方、同じレポートが45分ごとに1回生成されている場合、データベースには一定の影響がある問題が継続的に起きていることになります。

- 新しく識別された問題

  (過去45分間には検出されていなかった)新しい問題が検出された場合は、新しいレポートが生成されます。たとえば、レポートが8つのアクティブ・セッションに対してトリガーされ、新しいデッドロックの問題が検出された場合、新しいアクティ

ブ・セッションの負荷に関係なく、新しいレポートが生成されます。

ADDM分析の結果

ADDMでは、問題の診断に加えて可能な解決策も推奨されます。ADDMの分析結果は、一連の検出結果として表示されます。ADDMの分析結果の例は、例7-1を参照してください。ADDMの検出結果は、それぞれ次のタイプのいずれかに属します。

- 問題の検出結果では、データベース・パフォーマンスの問題の根本的な原因が記述されます。
- 症状の検出結果には、1つ以上の問題の検出につながる情報が含まれます。
- 情報の検出結果は、データベースのパフォーマンスの理解に役立つが、パフォーマンスの問題には関連しない情報(データベースの正常領域や、自動データベース・メンテナンスのアクティビティなど)をレポートする際に使用されます。
- 警告の検出結果には、ADDM分析の完全性や正確性に影響する可能性のある問題に関する情報(AWRの欠落データなど)が含まれます。

問題の検出結果はそれぞれ、影響によって数量化されます。この影響は、その検出結果のパフォーマンスの問題に起因するDB timeの割合を見積ったものです。問題の検出結果を推奨事項のリストに関連付けて、パフォーマンスの問題による影響を軽減できます。推奨事項のタイプは、次のとおりです。

- ハードウェア変更: CPUの追加またはI/Oサブシステム構成の変更
- データベース構成: 初期化パラメータ設定の変更
- スキーマ変更: 表または索引のハッシュ・パーティション化、あるいは自動セグメント領域管理(ASSM)の使用
- アプリケーション変更: 順序のキャッシュ・オプションの使用またはバインド変数の使用
- その他のアドバイザの使用: 高負荷SQLに対するSQLチューニング・アドバイザの実行、またはホット・オブジェクトに対するセグメント・アドバイザの実行

推奨事項のリストには、同じ問題に関する様々な代替解決策が含まれる場合があります。特定の問題を解決するために推奨事項をすべて適用する必要はありません。推奨事項にはそれぞれメリットがあります。このメリットは、その推奨事項を実装した場合に節約できるDB timeの割合を見積ったものです。推奨は、アクションおよび論理で構成されます。見積られたメリットを得るためには、推奨事項のアクションをすべて適用する必要があります。理論的根拠は、一連のアクションの推奨理由を説明し、提案された推奨事項の実装に関する追加情報を提供するために使用されます。

ADDMの分析結果の確認: 例

例7-1に示すADDMレポートの次のセクションを検討します。

例7-1 ADDMレポートの例

```
FINDING 1: 31% impact (7798 seconds)
-----------------------------------
SQL statements were not shared due to the usage of literals. This resulted in
additional hard parses which were consuming significant database time.
RECOMMENDATION 1: Application Analysis, 31% benefit (7798 seconds)
  ACTION: Investigate application logic for possible use of bind variables
    instead of literals. Alternatively, you may set the parameter
    "cursor_sharing" to "force".
  RATIONALE: SQL statements with PLAN_HASH_VALUE 3106087033 were found to be
    using literals. Look in V$SQL for examples of such SQL statements.
```

例7-1では、検出結果は特定の根本的な原因を示しています。SQL文のリテラルの使用は、分析期間中のDB time合計の約31%に影響があると見積られています。

この検出結果には、アクション1つおよび理論的根拠1つで構成される推奨事項が関連付けられています。アクションでは検出された問題の解決策が指定され、その最大メリットは分析期間中のDB timeの31%以内と見積られます。メリットは、検出結果の影響の割合ではなく、DB time合計の割合として表示されることに注意してください。理論的根拠は、リテラルを使用していてこのパフォーマンスの問題を引き起こした可能性のあるSQL文を追跡するための追加情報を提供します。データベース管理者は、問題の原因と思われるSQL文について指定の計画ハッシュ値を使用すると、少数のサンプル文をすばやく検査できます。

特定の問題に複数の原因がある場合は、ADDMにより複数の問題と症状の検出結果がレポートされることがあります。この場合、複数の検出結果による影響には、DB timeが同じ割合で含まれる可能性があります。検出結果のパフォーマンスの問題はオーバーラップしている場合があるため、検出結果の影響を合算すると、DB timeの100%を超える場合があります。たとえば、システムで多数の読取りが実行される場合、ADDMでは、I/OアクティビティによるDB timeの50%に関係するSQL文が1つの検出結果としてレポートされ、DB timeの75%に関係するサイズ不足のバッファ・キャッシュが別の検出結果としてレポートされる場合があります。

問題の検出結果に複数の推奨事項が関連付けられている場合は、それぞれに問題の代替解決策が含まれていることがあります。この場合、推奨事項によるメリットの合計は検出結果の影響よりも大きいことがあります。

該当する場合は、ADDMアクションで複数の解決策が推奨され、管理者はその中から選択できます。この例では、最も有効な解決策はバインド変数を使用することです。ただし、通常、アプリケーションを変更することは困難です。CURSOR_SHARING初期化パラメータの値を変更する方が実装が容易で、大幅な改善が可能です。

ADDMの設定

自動データベース診断モニターはデフォルトで有効になり、CONTROL_MANAGEMENT_PACK_ACCESSおよびSTATISTICS_LEVEL初期化パラメータにより制御されます。

自動データベース診断モニターを有効にするには、CONTROL_MANAGEMENT_PACK_ACCESSパラメータをDIAGNOSTICまたはDIAGNOSTIC+TUNINGに設定する必要があります。デフォルト設定はDIAGNOSTIC+TUNINGです。CONTROL_MANAGEMENT_PACK_ACCESSをNONEに設定すると、ADDMが無効になります。

自動データベース診断モニターを使用可能にするには、STATISTICS_LEVELパラメータをTYPICALまたはALLに設定する必要があります。デフォルトの設定はTYPICALです。STATISTICS_LEVELをBASICに設定すると、ADDMを含む多くのOracle Databaseの機能が無効化されるため、この方法はお薦めしません。

関連項目:

CONTROL_MANAGEMENT_PACK_ACCESSおよびSTATISTICS_LEVELの各初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください

I/OパフォーマンスのADDM分析は、単一の引数DBIO_EXPECTEDによって異なり、I/Oサブシステムの予測されるパフォーマンスを示しています。DBIO_EXPECTEDの値は、1つのデータベース・ブロックの読取りにかかる平均時間(マイクロ秒単位)です。

Oracle Databaseではデフォルト値の10ミリ秒が使用されます。この値は、ほとんどの最新のハード・ドライブに適しています。非常に古いハードウェアや超高速のRAMディスクなど、ハードウェアが大きく異なっている場合は、他の値を使用することを検討してください。

DBIO_EXPECTEDパラメータの適切な設定を決定するには:

1. ハードウェアについて、1データベース・ブロックを読み取るための平均所要時間を測定します。

   この測定の対象はランダムI/Oであり、標準ハード・ドライブを使用している場合はシーク時間が含まれることに注意してください。ハード・ドライブの標準の値は5000マイクロ秒から20000マイクロ秒です。

2. 後続のすべてのADDM実行の値を一度設定します。

   たとえば、測定値が8000マイクロ秒の場合は、次のコマンドをSYSユーザーとして実行する必要があります。

```
EXECUTE DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER(
                        'ADDM', 'DBIO_EXPECTED', 8000);
```

ADDMを使用したデータベース・パフォーマンスの問題の診断

データベース・パフォーマンスの問題を診断するには、AWRスナップショットが取得されるたびに自動的に作成されるADDMの分析結果を最初に確認する必要があります。異なる分析が必要な場合(分析期間の延長、異なるDBIO_EXPECTED設定の使用、分析モードの変更など)、この項の手順に従って手動でADDMを実行できます。

ADDMでは、2つのAWRスナップショットが消去されずにAWRに保存されているかぎり、その(同じデータベース上の)2つのスナップショットを分析できます。ADDMで分析できるのは、開始スナップショットの前に開始され、終了スナップショットまで実行が継続していたインスタンスのみです。また、ADDMでは、AWRスナップショットの生成時に重大なエラーが発生したインスタンスは分析できません。このような場合、ADDMでは、重大な問題の発生していないインスタンスの最大のサブセットが分析されます。

診断モニターのプライマリ・インタフェースは、Cloud Controlです。可能なかぎり、ADDMは、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』の説明に従ってCloud Controlで実行する必要があります。Cloud Controlを使用できない場合は、DBMS_ADDMパッケージを使用してADDMを実行します。DBMS_ADDM APIを実行するには、ユーザーにADVISOR権限が付与されている必要があります。

この項では、次の項目について説明します。

- データベース・モードでのADDMの実行
- インスタンス・モードでのADDMの実行
- 部分モードでのADDMの実行
- ADDMレポートの表示

関連項目:

DBMS_ADDMパッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

データベース・モードでのADDMの実行

Oracle RAC構成の場合、データベース・モードでADDMを実行して、データベースのすべてのインスタンスを分析できます。単一インスタンス構成の場合も、データベース・モードでADDMを実行できますが、この場合、ADDMはインスタンス・モードと同じように動作します。

データベース・モードでADDMを実行するには、DBMS_ADDM.ANALYZE_DBプロシージャを次のように使用します。

```
BEGIN
DBMS_ADDM.ANALYZE_DB (
   task_name            IN OUT VARCHAR2,
   begin_snapshot       IN     NUMBER,
   end_snapshot         IN     NUMBER,
   db_id                IN     NUMBER := NULL);
END;
/
```

task_nameパラメータでは、作成する分析タスクの名前を指定します。begin_snapshotパラメータでは、分析期間の開始スナップショットのスナップショット番号を指定します。end_snapshotパラメータでは、分析期間の終了スナップショットのスナップショット番号を指定します。db_idパラメータでは、分析するデータベースのデータベース識別子を指定します。指定しない場合、このパラメータは、現在接続しているデータベースのデータベース識別子にデフォルト設定されます。

次の例では、データベース分析モードでADDMタスクを作成して実行し、スナップショット137および145で定義される期間のデータベース全体のパフォーマンスを診断します。

```
VAR tname VARCHAR2(30);
BEGIN
  :tname := 'ADDM for 7PM to 9PM';
  DBMS_ADDM.ANALYZE_DB(:tname, 137, 145);
END;
/
```

インスタンス・モードでのADDMの実行

データベースの特定のインスタンスを分析するには、インスタンス・モードでADDMを実行します。インスタンス・モードでADDMを実行するには、DBMS_ADDM.ANALYZE_INSTプロシージャを次のように使用します。

```
BEGIN
DBMS_ADDM.ANALYZE_INST (
   task_name            IN OUT VARCHAR2,
   begin_snapshot       IN     NUMBER,
   end_snapshot         IN     NUMBER,
   instance_number      IN     NUMBER := NULL,
   db_id                IN     NUMBER := NULL);
END;
/
```

task_nameパラメータでは、作成する分析タスクの名前を指定します。begin_snapshotパラメータでは、分析期間の開始スナップショットのスナップショット番号を指定します。end_snapshotパラメータでは、分析期間の終了スナップショットのスナップショット番号を指定します。instance_numberパラメータでは、分析するインスタンスのインスタンス番号を指定します。指定しない場合、このパラメータは、現在接続しているインスタンスのインスタンス番号にデフォルト設定されます。db_idパラメータでは、

分析するデータベースのデータベース識別子を指定します。指定しない場合、このパラメータは、現在接続しているデータベースのデータベース識別子にデフォルト設定されます。

次の例では、インスタンス分析モードでADDMタスクを作成して実行し、スナップショット137および145で定義される期間のインスタンス番号1のパフォーマンスを診断します。

```
VAR tname VARCHAR2(30);
BEGIN
  :tname := 'my ADDM for 7PM to 9PM';
  DBMS_ADDM.ANALYZE_INST(:tname, 137, 145, 1);
END;
/
```

部分モードでのADDMの実行

すべてのデータベース・インスタンスのサブセットを分析するには、部分モードでADDMを実行します。部分モードでADDMを実行するには、DBMS_ADDM.ANALYZE_PARTIALプロシージャを次のように使用します。

```
BEGIN
DBMS_ADDM.ANALYZE_PARTIAL (
   task_name           IN OUT VARCHAR2,
   instance_numbers    IN     VARCHAR2,
   begin_snapshot      IN     NUMBER,
   end_snapshot        IN     NUMBER,
   db_id               IN     NUMBER := NULL);
END;
/
```

task_nameパラメータでは、作成する分析タスクの名前を指定します。instance_numbersパラメータでは、分析するインスタンスのインスタンス番号のカンマ区切りリストを指定します。begin_snapshotパラメータでは、分析期間の開始スナップショットのスナップショット番号を指定します。end_snapshotパラメータでは、分析期間の終了スナップショットのスナップショット番号を指定します。db_idパラメータでは、分析するデータベースのデータベース識別子を指定します。指定しない場合、このパラメータは、現在接続しているデータベースのデータベース識別子にデフォルト設定されます。

次の例では、部分分析モードでADDMタスクを作成して実行し、スナップショット137および145で定義される期間のインスタンス番号1、2、4のパフォーマンスを診断します。

```
VAR tname VARCHAR2(30);
BEGIN
  :tname := 'my ADDM for 7PM to 9PM';
  DBMS_ADDM.ANALYZE_PARTIAL(:tname, '1,2,4', 137, 145);
END;
/
```

ADDMレポートの表示

実行したADDMタスクのテキスト・レポートを表示するには、DBMS_ADDM.GET_REPORTファンクションを次のように使用します。

```
DBMS_ADDM.GET_REPORT (
   task_name           IN VARCHAR2
  RETURN CLOB);
```

次の例では、tname変数によるタスク名で指定されたADDMタスクのテキスト・レポートを表示します。

```
SET LONG 1000000 PAGESIZE 0;
SELECT DBMS_ADDM.GET_REPORT(:tname) FROM DUAL;
```

レポートの戻り値のタイプはCLOBであり、行サイズ(80)に合うように書式化されます。ADDMレポートでADDMの分析結果を確認する方法の詳細は、「ADDMの分析結果」を参照してください。

ADDMのビュー

通常、Cloud ControlまたはDBMS_ADDMパッケージのサブプログラムを使用してADDM分析を表示する必要があります。

ただし、DBA_ADDM_*ビューおよびDBA_ADVISOR_*ビューを使用してADDM情報を取得することもできます。このグループのビューには次のようなものがあります。

- DBA_ADVISOR_FINDINGS

  このビューには、すべてのアドバイザで取得されたすべての検出結果が表示されます。各検出結果は、関連する結果ID、名前およびタイプとともに表示されます。複数の実行に関連するタスクの場合、それぞれの検出結果に関連する各タスク実行の名前もリストされます。

- DBA_ADDM_FINDINGS

  このビューには、関連するDBA_ADVISOR_FINDINGSビューに表示される検出結果のサブセットが含まれます。このビューには、すべてのアドバイザで取得されたADDMの検出結果のみが表示されます。ADDMの各検出結果は、関連する結果ID、名前およびタイプとともに表示されます。

- DBA_ADVISOR_FINDING_NAMES

  このビューには、アドバイザ・フレームワークに登録されているすべての検出結果名がリストされます。

- DBA_ADVISOR_RECOMMENDATIONS

  このビューには、完了した診断タスクの結果と、実行ごとに識別された問題に対する推奨事項が表示されます。推奨事項は、RANK列の順序どおりに参照する必要があります。この順序が推奨事項における問題の重要性を伝えているためです。BENEFIT列には、推奨事項を実行した場合に予想される、システムに対するメリットが示されます。複数の実行に関連するタスクの場合、それぞれのアドバイザ・タスクに関連する各タスク実行の名前もリストされます。

- DBA_ADVISOR_TASKS

  このビューでは、タスクID、タスク名およびタスク作成日時など、既存のタスクに関する基本情報が示されます。複数の実行に関連するタスクの場合、それぞれのアドバイザ・タスクに関連する最後または現在の実行の名前とタイプもリストされます。

関連項目:

- DBA_ADDM_*ビューおよびDBA_ADVISOR_*ビューの詳細は、Oracle Databaseリファレンスを参照してください
- ADDMレポートの表示

8 一定期間にわたるデータベース・パフォーマンスの比較

この章では、自動ワークロード・リポジトリ(AWR)の期間比較レポートを使用して、一定期間にわたるデータベース・パフォーマンスを比較する方法を説明しており、内容は次のとおりです。

- 自動ワークロード・リポジトリの期間比較レポートについて
- 自動ワークロード・リポジトリ期間比較レポートの生成
- 自動ワークロード・リポジトリの期間比較レポートの解釈

自動ワークロード・リポジトリの期間比較レポートについて

データベースのパフォーマンスの低下は、それまで最適に稼働していたデータベースが、長い期間をかけてユーザーが気付くところまで徐々にパフォーマンスが低下した場合に発生します。AWRの期間比較レポートによって、一定期間のデータベース・パフォーマンスを比較できます。

AWRレポートは、2つのスナップショット(または2つの時点)の間の期間のAWRデータを示します。一方、AWRの期間比較レポートは、2つの期間の差(つまり、4つのスナップショットに相当する2つのAWRレポート)を示します。AWRの期間比較レポートを使用すると、2つの期間で異なる詳細なパフォーマンス属性および構成設定を識別できます。

たとえば、午後10:00から午前0:00のメンテナンス・ウィンドウ中に毎日実行されるバッチ・ワークロードのパフォーマンスが低下しており、現在では完了が午前2:00になっているとします。パフォーマンスが良好であった日の午後10:00から午前0:00までの期間のAWRの期間比較レポートと、パフォーマンスが低下していた日の午前10:00から午前2:00までの期間の別のレポートを生成できます。これらのレポートを比較すると、構成の設定、ワークロード・プロファイルおよびこれらの2つの期間で異なる統計を識別できます。それらの相違点に基づいて、パフォーマンス低下の原因をより簡単に診断できます。

このレポートでは、各期間にデータベースで費やされた時間によって統計が正規化され、期間ごとの差異が大きい順に統計データが表示されるため、AWRの期間比較レポートには、異なる2つの期間を選択することができます。

> ノート:
>
> プラガブル・データベース(PDB)とともに AWR 機能を使用している場合には、データ可視性および権限の要件が異なります。AWR 機能などの管理機能がマルチテナント・コンテナ・データベース(CDB)で機能する仕組みの詳細は、Oracle Multitenant 管理者ガイドを参照してください。

関連項目:

- AWRの詳細は、「自動ワークロード・リポジトリ」を参照してください
- AWRレポートの詳細は、「自動ワークロード・リポジトリ・レポートの生成」を参照してください

自動ワークロード・リポジトリ期間比較レポートの生成

一定期間にわたりデータベースのパフォーマンスが低下する場合は、AWRの期間比較レポートを使用すると、2つの期間を比較して、パフォーマンス低下の原因の診断に役立つ主な違いを特定できます。

AWRの期間比較レポートは複数のセクションに分かれています。HTMLレポートには、セクション間ですばやくナビゲートできるようにリンクが組み込まれています。レポートの内容には、選択した範囲のスナップショットに関するシステムのワークロード・プロファイルが含まれます。

- AWRの期間比較レポートを生成するためのユーザー・インタフェース
- コマンドライン・インタフェースを使用したAWRの期間比較レポートの生成

AWRの期間比較レポートを生成するためのユーザー・インタフェース

AWRの期間比較レポートを生成するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してAWRの期間比較レポートを生成してください。

Cloud Controlを使用できない場合は、SQLスクリプトを実行してAWRの期間比較レポートを生成します。これらのスクリプトを実行するには、DBAロールが必要です。

関連項目:

Cloud Controlを使用したAWRの期間比較レポートの生成の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

コマンドライン・インタフェースを使用したAWRの期間比較レポートの生成

このトピックでは、コマンドライン・インタフェースでSQLスクリプトを実行し、AWRの期間比較レポートを生成する方法を説明します。

- ローカル・データベースのAWRの期間比較レポートの生成
- 特定データベースのAWRの期間比較レポートの生成
- ローカル・データベースのOracle RAC AWRの期間比較レポートの生成
- 特定データベースのOracle RAC AWRの期間比較レポートの生成

ローカル・データベースのAWRの期間比較レポートの生成

awrddrpt.sql SQLスクリプトでは、ローカル・データベース・インスタンスにおける選択された2つの期間の詳細なパフォーマンス属性および構成設定を比較する、HTMLまたはテキストのレポートが生成されます。

コマンドライン・インタフェースを使用して、ローカル・データベース・インスタンスにAWRの期間比較レポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrddrpt.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

3. 第1期間のスナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 2
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

4. 第1期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 102
Enter value for end_snap: 103
```

この例では、第1期間において、スナップショットIDが102のスナップショットが最初のスナップショットとして選択され、スナップショットIDが103のスナップショットが最後のスナップショットとして選択されます。

5. 第2期間のスナップショットIDをリストする日数を指定します。

```
Enter value for num_days2: 1
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、前日に取得されたスナップショットが表示されます。

6. 第2期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap2: 126
Enter value for end_snap2: 127
```

この例では、第2期間において、スナップショットIDが126のスナップショットが最初のスナップショットとして選択され、スナップショットIDが127のスナップショットが最後のスナップショットとして選択されます。

7. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrdiff_1_102_1_126.txt
```

この例では、デフォルト名が使用され、awrdiff_1_102_126というAWRレポートが生成されます。

特定データベースのAWRの期間比較レポートの生成

awrddrpi.sql SQLスクリプトでは、特定のデータベースおよびインスタンスにおける選択された2つの期間の詳細なパフォーマンス属性および構成設定を比較する、HTMLまたはテキストのレポートが生成されます。このスクリプトでは、AWRの期間比較レポートの生成に使用されるデータベース識別子およびインスタンスを指定できます。

コマンドライン・インタフェースを使用して、特定のデータベース・インスタンスにAWRの期間比較レポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrddrpi.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: text
```

この例では、テキスト形式のレポートが選択されます。

3. 使用可能なデータベース識別子およびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
```

次のように、第1期間に対するデータベース識別子(dbid)およびインスタンス番号(inst_num)の値を入力します。

```
Enter value for dbid: 3309173529
Using 3309173529 for Database Id for the first pair of snapshots
Enter value for inst_num: 1
Using 1 for Instance Number for the first pair of snapshots
```

4. 第1期間のスナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 2
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

5. 第1期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 102
Enter value for end_snap: 103
```

この例では、第1期間において、スナップショットIDが102のスナップショットが最初のスナップショットとして選択され、スナップショットIDが103のスナップショットが最後のスナップショットとして選択されます。

6. 次のように、第2期間に対するデータベース識別子(dbid)およびインスタンス番号(inst_num)の値を入力します。

```
Enter value for dbid2: 3309173529
Using 3309173529 for Database Id for the second pair of snapshots
Enter value for inst_num2: 1
Using 1 for Instance Number for the second pair of snapshots
```

7. 第2期間のスナップショットIDをリストする日数を指定します。

```
Enter value for num_days2: 1
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、前日に取得されたスナップショットが表示されます。

8. 第2期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap2: 126
Enter value for end_snap2: 127
```

この例では、第2期間において、スナップショットIDが126のスナップショットが最初のスナップショットとして選択され、スナップショットIDが127のスナップショットが最後のスナップショットとして選択されます。

9. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrdiff_1_102_1_126.txt
```

この例では、デフォルト名が使用され、3309173529というデータベースIDを持つデータベース・インスタンスでawrdiff_1_102_126というAWRレポートが生成されます。

ローカル・データベースのOracle RAC AWRの期間比較レポートの生成

awrgdrpt.sql SQLスクリプトでは、現在のデータベース識別子およびOracle Real Application Clusters (Oracle RAC)環境のすべての使用可能なデータベース・インスタンスを使用して、選択された2つの期間の詳細なパフォーマンス属性および構成設定を比較する、HTMLまたはテキストのレポートが生成されます。

ノート:

HTML のレポートは、テキストのレポートよりも読みやすいため、Oracle RAC 環境では、テキストではなく、HTML レポートを生成してください。

コマンドライン・インタフェースを使用して、ローカル・データベース・インスタンスにOracle RAC AWRの期間比較レポートを生成するには:

1. SQLプロンプトで次のように入力します。

   ```
   @$ORACLE_HOME/rdbms/admin/awrgdrpt.sql
   ```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

   ```
   Enter value for report_type: html
   ```

   この例では、HTML形式のレポートが選択されます。

3. 第1期間のスナップショットIDをリストする日数を指定します。

   ```
   Enter value for num_days: 2
   ```

   指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

4. 第1期間の最初と最後のスナップショットIDを指定します。

   ```
   Enter value for begin_snap: 102
   Enter value for end_snap: 103
   ```

   この例では、第1期間において、スナップショットIDが102のスナップショットが最初のスナップショットとして選択され、スナップショットIDが103のスナップショットが最後のスナップショットとして選択されます。

5. 第2期間のスナップショットIDをリストする日数を指定します。

   ```
   Enter value for num_days2: 1
   ```

   指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、前日に取得されたスナップショットが表示されます。

6. 第2期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap2: 126
Enter value for end_snap2: 127
```

この例では、第2期間において、スナップショットIDが126のスナップショットが最初のスナップショットとして選択され、スナップショットIDが127のスナップショットが最後のスナップショットとして選択されます。

7. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name awrracdiff_1st_1_2nd_1.html
```

この例では、デフォルト名が使用され、awrrac_1st_1_2nd_1.htmlというAWRレポートが生成されます。

特定データベースのOracle RAC AWRの期間比較レポートの生成

awrgdrpi.sql SQLスクリプトでは、Oracle RAC環境の特定のデータベースおよびインスタンスを使用して、選択された2つの期間の詳細なパフォーマンス属性および構成設定を比較する、HTMLまたはテキストのレポートが生成されます。このスクリプトでは、AWRの期間比較レポートの生成に使用されるデータベース識別子およびデータベース・インスタンスのカンマ区切りのリストを指定できます。

ノート:

HTML のレポートは、テキストのレポートよりも読みやすいため、Oracle RAC 環境では、テキストではなく、常に HTML レポートを生成してください。

コマンドライン・インタフェースを使用して、特定データベースにOracle RAC AWRの期間比較レポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/awrgdrpi.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

3. 使用可能なデータベース識別子およびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
 3309173529        2 TINT251      tint252      samp252
 3309173529        3 TINT251      tint253      samp253
 3309173529        4 TINT251      tint254      samp254
```

次のように、第1期間のデータベース識別子(dbid)およびインスタンス番号(instance_numbers_or_all)の値を入力します。

```
Enter value for dbid: 3309173529
Using 3309173529 for Database Id for the first pair of snapshots
```

```
Enter value for inst_num: 1,2
Using instances 1 for the first pair of snapshots
```

4. 第1期間のスナップショットIDをリストする日数を指定します。

```
Enter value for num_days: 2
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、最近2日間に取得されたスナップショットが表示されます。

5. 第1期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap: 102
Enter value for end_snap: 103
```

この例では、第1期間において、スナップショットIDが102のスナップショットが最初のスナップショットとして選択され、スナップショットIDが103のスナップショットが最後のスナップショットとして選択されます。

6. 使用可能なデータベース識別子およびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
 3309173529        2 TINT251      tint252      samp252
 3309173529        3 TINT251      tint253      samp253
 3309173529        4 TINT251      tint254      samp254
INSTNUM1
--------------------------------------------------
1,2
```

次のように、第2期間のデータベース識別子(dbid2)およびインスタンス番号(instance_numbers_or_all2)の値を入力します。

```
Enter value for dbid2: 3309173529
Using 3309173529 for Database Id for the second pair of snapshots
Enter value for instance_numbers_or_all2: 3,4
```

7. 第2期間のスナップショットIDをリストする日数を指定します。

```
Enter value for num_days2: 1
```

指定した時間範囲に対応する既存のスナップショットのリストが表示されます。この例では、前日に取得されたスナップショットが表示されます。

8. 第2期間の最初と最後のスナップショットIDを指定します。

```
Enter value for begin_snap2: 126
Enter value for end_snap2: 127
```

この例では、第2期間において、スナップショットIDが126のスナップショットが最初のスナップショットとして選択され、スナップショットIDが127のスナップショットが最後のスナップショットとして選択されます。

9. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
```

```
Using the report name awrracdiff_1st_1_2nd_1.html
```

この例では、デフォルト名が使用され、awrrac_1st_1_2nd_1.htmlというAWRレポートが生成されます。

自動ワークロード・リポジトリの期間比較レポートの解釈

比較する期間のAWRの期間比較レポートが生成されたら、その内容を確認し、一定期間にわたるパフォーマンス低下の可能性のある原因を特定します。

AWRの期間比較レポートのコンテンツは次の項に分割して示しています。

- AWR期間の比較レポートのサマリー
- AWR期間の比較レポートの詳細
- AWR期間の比較レポートの補足情報

AWRの期間比較レポートのサマリー

レポート・サマリーはAWRの期間比較レポートの一部で、スナップショット・セットおよびレポートで使用されるワークロードに関する情報を示します。

レポート・サマリーは次のセクションで構成されています。

- スナップショット・セット
- ホスト構成の比較
- システム構成の比較
- ロード・プロファイル
- 上位5位の時間消費イベント

スナップショット・セット

スナップショット・セットに関するセクションに、インスタンス、ホストおよびスナップショットの詳細などのこのレポートに使用されたスナップショット・セットに関連する情報が表示されます。

ホスト構成の比較

「ホスト構成の比較」セクションでは2つのスナップショット・セットで使用したホスト構成を比較します。たとえば、前述のレポートでは物理メモリーとCPUの数を比較します。構成の差は、「%Diff」列に割合として数値化されます。

システム構成の比較

「システム構成の比較」セクションでは、2つのスナップショット・セットで使用したデータベース構成を比較します。たとえば、レポートでは、システム・グローバル領域(SGA)とログ・バッファのサイズを比較します。構成の差は、「%Diff」列に割合として数値化されます。

ロード・プロファイル(Load Profile)

「ロード・プロファイル」セクションでは、2つのスナップショット・セットで使用されたワークロードが比較されます。ワークロードの差は、

「%Diff」列に割合として数値化されます。

上位5位の時間消費イベント

上位5位の時間消費イベント・セクションに、各スナップショット・セットで総データベース時間(DB時間)の最も高い割合を消費する5つの時間イベントまたは操作が表示されます。

AWRの期間比較レポートの詳細

「詳細」セクションはAWRの期間比較レポートのレポート・サマリーに続いて表示され、レポートで使用したスナップショット・セットおよびワークロードの詳細情報が表示されます。

レポートの詳細は次のセクションで構成されています。

- 時間モデル統計
- オペレーティング・システム統計
- 待機イベント
- サービス統計
- SQLの統計
- インスタンス・アクティビティ統計
- I/O統計
- アドバイザの統計
- 待機統計
- UNDOセグメントのサマリー
- ラッチ統計
- セグメント統計
- インメモリー・セグメント統計
- ディクショナリ・キャッシュ統計
- ライブラリ・キャッシュ統計
- メモリー統計
- ストリームの統計

時間モデル統計

「時間モデル統計」セクションでは、2つのスナップショット・セットの時間モデル統計を比較します。時間モデル統計は2つのスナップショット・セットの操作の特定のタイプに費やされるDB時間の合計の差に基づいて並び替えられ、降順でリストされます。このセクション上部の時間モデル統計には2つのスナップショット・セット間で最も大きな差、および時間の経過によるパフォーマンスの低下の原因になっていた可能性のある関連操作が含まれます。

関連項目:

時間モデル統計の詳細は、「時間モデル統計」を参照してください

オペレーティング・システム統計

「Operating System Statistics」セクションでは、2つのスナップショット・セットでオペレーティング・システム統計を比較します。このセクションでは、比較するそれぞれの2つの期間におけるオペレーティング・システムの全体の状態が表示されます。

待機イベント

「待機イベント」セクションでは、2つのスナップショット・セットの待機イベントを比較します。

最初のセクションではユーザーI/OおよびシステムI/Oなどの待機イベントのクラスをリストします。このクラスはDB時間の割合の列の絶対値に応じてリストされます。

2番目のセクションでは、待機イベントをリストします。待機イベントは2つのスナップショット・セット間の待機イベントに費やされる総DB時間に基づいて並べ替えられ、降順にリストされます。このセクション上部の待機イベントには2つのスナップショット・セット間で最も大きな差が含まれ、時間の経過によるパフォーマンスの低下の原因である可能性があります。

関連項目:

待機イベントおよび待機クラスの詳細は、「待機イベント統計」を参照してください

サービス統計

「Service Statistics」セクションは2つのスナップショット・セットのサービスを比較します。サービスは2つのスナップショット・セット間で特定のサービスに費やされるDB時間の合計の差に基づいて並び替えられ、降順でリストされます。

SQLの統計

「SQL Statistics」セクションでは、2つのスナップショット・セットの上位SQL文が比較されます。SQL文はあらゆる比較方法に基づいて並べられますが、すべての場合で2つのスナップショット・セット間で最も差が大きい上位10セグメントが表示されます。

このセクションに記載されているSQL文は、時間の経過によるパフォーマンスの低下の原因になる可能性があり、次のカテゴリに基づいて並べられています。

- 上位10 SQLの実行時間ごとの比較
- 上位10 SQLのCPU時間ごとの比較
- 上位10 SQLのバッファ読取りごとの比較
- 上位10 SQLの物理読取りごとの比較
- 上位10 SQLの実行ごとの比較
- 上位10 SQLの解析コールごとの比較
- SQLテキストの完全なリスト(Complete List of SQL Text)

上位10 SQLの実行時間ごとの比較

このサブセクションのSQL文は、2つのスナップショット・セット間で、SQL文の処理に使用される合計DB時間の差に基づいて、降順でリストされます。

このサブセクションのSQL文は、ある期間においてDB時間の高い割合を占めていましたが、他の期間ではそのようなことはなく、パフォーマンス低下の原因となっている高負荷SQL文である可能性が高いため、調査が必要です。レポートのSQLテキストの完全なリスト・サブセクションのSQL文を確認して、必要に応じてチューニングを行います。

関連項目:

SQL文のチューニングの詳細は、Oracle Database SQLチューニング・ガイドを参照してください

上位10 SQLのCPU時間ごとの比較

このサブセクションのSQL文は、2つのスナップショット・セット間で、SQL文の処理に使用されるCPU時間の差に基づいて、降順でリストされます。

上位10 SQLのバッファ読取りごとの比較

このサブセクションのSQL文は、2つのスナップショット・セット間で、SQL文を処理するときに行われるバッファ・キャッシュ読取りまたはバッファ読取りの合計回数の差に基づいて、降順でリストされます。

上位10 SQLの物理読取りごとの比較

このサブセクションのSQL文は、2つのスナップショット・セット間で、SQL文を処理する際に行われる物理読取り回数の差に基づいて、降順でリストされます。

上位10 SQLの実行ごとの比較

このサブセクションのSQL文は、2つのスナップショット・セット間で、SQL文を処理するときの(DB時間での) 1秒間の実行回数の差に基づいて、降順でリストされます。

上位10 SQLの解析コールごとの比較

このサブセクションのSQL文は、2つのスナップショット・セット間で、SQL文を処理する際に行われる解析の合計数の差に基づいて、降順でリストされます。解析はSQL文の処理に含まれます。

アプリケーションでSQL文が発行されるとき、アプリケーションによりOracle Databaseに解析応答が実行されます。解析応答を実行するとデータベースのパフォーマンスに大きく影響する可能性があるため、可能なかぎり最小限にする必要があります。

関連項目:

索引の詳細は、Oracle Database概要を参照

SQLテキストの完全なリスト(Complete List of SQL Text)

このサブセクションでは「SQLの統計」セクションにリストされたすべてのSQL文のSQLテキストを表示します。

インスタンス・アクティビティ統計

インスタンス・アクティビティの統計セクションでは、2つのスナップショット・セット間でインスタンス・アクティビティ統計値が比較されます。各統計には、DB時間、経過時間およびトランザクションごとに測定された差で統計の値が表示されます。

インスタンス・アクティビティ統計は次のサブセクションに分類されます。

- 主要なインスタンス・アクティビティの統計
- 他のインスタンス・アクティビティの統計

主要なインスタンス・アクティビティの統計

このサブセクションには、2つのスナップショット・セット間の主要なインスタンス・アクティビティ統計値の差が表示されます。

他のインスタンス・アクティビティの統計

このサブセクションには、2つのスナップショット・セット間におけるその他すべての統計のインスタンス・アクティビティの差が表示されます。

I/O統計

「I/O統計」セクションでは、2つのスナップショット・セット間の表領域およびデータベースで実行されたI/O操作が比較されます。2つのスナップショット・セット間のI/O操作の大幅な増加は時間の経過によるパフォーマンスの低下の原因である可能性があります。

各表領域またはデータベース・ファイルでは読取り、書込みおよびバッファ・キャッシュ待機(バッファ読取り)の数の差が割合として定量化されます。データベース・ファイルはあらゆる比較方法に基づいてリストされますが、すべての場合で2つのスナップショット・セット間で最も差が大きい上位10個のデータベース・ファイルが表示されます。

I/O統計は次のカテゴリに分類されます。

- 表領域I/O統計
- 上位10ファイルのI/Oごとの比較
- 上位10ファイルの読取り時間ごとの比較
- 上位10ファイルのバッファ待機ごとの比較

表領域I/O統計

このサブセクションに表示される表領域は、2つのスナップショット・セット間において、表領域で実行された正規化されたI/Oの数の差に基づいて、降順でリストされます。正規化されたI/Oは1秒当たりの読取り回数および書込み回数の平均の合計です。

上位10ファイルのI/Oごとの比較

このサブセクションに表示されるデータベース・ファイルは、2つのスナップショット・セット間において、データベース・ファイルで実行された正規化されたI/Oの数の差に基づいて、降順でリストされます。正規化されたI/Oは1秒当たりの読取り回数および書込み回数の平均の合計です。

上位10ファイルの読取り時間ごとの比較

このサブセクションに表示されるデータベース・ファイルは、2つのスナップショット間で、データベース・ファイルからのデータの読取りに

使用したDB時間の割合の差に基づいて、降順でリストされます。

上位10ファイルのバッファ待機ごとの比較

このサブセクションに表示されるデータベース・ファイルは、2つのスナップショット・セット間において、データベース・ファイルで実行されるバッファ待機(バッファ・キャッシュの空きバッファ参照時に発生する待機)の数の差に基づいて、降順でリストされます。

アドバイザの統計

アドバイザの統計のセクションでは、2つのスナップショット・セット間でプログラム・グローバル領域(PGA)メモリー統計が比較され、次のカテゴリに分割されます。

- PGA集計のサマリー
- PGA集計ターゲットの統計

PGA集計のサマリー

このサブセクションでは、2つのスナップショット・セット間でPGAキャッシュ・ヒット率が比較されます。

PGA集計ターゲットの統計

このサブセクションでは、2つのスナップショット・セット間で自動PGAメモリー管理に関連する主要統計が比較されます。

待機統計

待機統計のセクションでは、バッファ待機統計が比較され、2つのスナップショット・セット間でエンキューされます。

待機統計は次のカテゴリに分類されます。

- バッファ待機統計
- エンキュー・アクティビティ

バッファ待機統計

このサブセクションでは、2つのスナップショット・セット間でバッファ待機が比較されます。バッファ待機は、バッファ・キャッシュの空きバッファ参照中に発生します。

エンキュー・アクティビティ

このサブセクションでは、2つのスナップショット・セット間でエンキュー・アクティビティが比較されます。エンキューは、データベース・リソースへのアクセスをシリアライズした共有メモリー構造(またはロック)で、セッションまたはトランザクションに関連付けられます。

関連項目:

エンキューの詳細は、Oracle Databaseリファレンスを参照

UNDOセグメントのサマリー

UNDOセグメントのサマリーのセクションでは、2つの期間のUNDOセグメントの使用を比較します。グラフは2つの期間のUNDOブロックの数、それらのブロックを使用するトランザクションの数および最大長の問合せを比較します。STO/OOS列では、古すぎ

るスナップショットの数および不足領域の件数を示します。

ラッチ統計

ラッチ統計セクションでは、2つのスナップショット・セット間でラッチのスリープの合計回数が降順に比較されます。

ラッチは、単純で低レベルなシリアライズ・メカニズムで、SGAの共有データ構造を保護します。たとえば、ラッチにより、データベースに現在アクセスしているユーザーのリストと、バッファ・キャッシュのブロックを記述するデータ構造が保護されます。ラッチは、これらの構造の1つを操作または検索するときに、サーバー・プロセスまたはバックグラウンド・プロセスによって非常に短い時間のみ取得されます。ラッチの実装(特にプロセスがラッチを待機するかどうか、およびプロセスがラッチを待機する時間)はオペレーティング・システムに依存します。

セグメント統計

「セグメント統計」セクションでは、2つのスナップショット・セット間のセグメントまたはデータベース・オブジェクト(表や索引など)が比較されます。セグメントはあらゆる比較方法に基づいて並べられますが、どの場合でも、2つのスナップショット・セット間で最も差が大きい上位5セグメントが表示されます。

ここに表示されるセグメントは、時間の経過によるパフォーマンスの低下の原因になる可能性があり、次のカテゴリに基づいて並べられています。

- 上位5セグメントの論理読取りごとの比較
- 上位5セグメントの物理読取りごとの比較
- 上位5セグメントの行ロック待機ごとの比較
- 上位5セグメントのITL待機ごとの比較
- 上位5セグメントのバッファ・ビジー待機ごとの比較

上位5セグメントの論理読取りごとの比較

このサブセクションに表示されるセグメントは、2つのスナップショット・セット間において、セグメントで実行された論理読取り回数(ディスクまたはメモリーからの読取り回数の合計)の差に基づいて、降順にリストされます。

非常に高い割合の論理読取りがデータベース・オブジェクトで行われている場合は、関連付けられているSQL文を調査し、索引またはマテリアライズド・ビューを使用して、データベース・オブジェクトへのデータ・アクセスのチューニングが必要かどうかを判断してください。

関連項目:

データ・アクセス・パスの最適化の詳細は、Oracle Database SQLチューニング・ガイドを参照してください

上位5セグメントの物理読取りごとの比較

このサブセクションに表示されるセグメントは、2つのスナップショット・セット間において、セグメントで実行された物理読取り(ディスク読取りなど)の回数の差に基づいて、降順にリストされます。

上位5セグメントの行ロック待機ごとの比較

このサブセクションに表示されるセグメントは、2つのスナップショット・セット間のセグメントに対する行ロック待機数の差に基づいて、降順にリストされます。

行レベルのロックの使用の主な目的は、2つのトランザクションが同一の行を変更しないようにするためです。トランザクションが行を変更する必要がある場合は、行ロックが取得されます。

関連項目:

行ロックの詳細は、Oracle Database概要を参照

上位5セグメントのITL待機ごとの比較

このサブセクションに表示されるセグメントは、2つのスナップショット・セット間において、セグメントの関連トランザクション・リスト(ITL)待機数の差に基づいて、降順でリストされます。

上位5セグメントのバッファ・ビジー待機ごとの比較

このサブセクションに表示されるセグメントは、2つのスナップショット・セット間のセグメントに対するバッファ・ビジー待機数の差に基づいて、降順でリストされます。

インメモリー・セグメント統計

インメモリー・セグメント統計セクションでは、2つのスナップショット・セット間でインメモリー・セグメント統計を比較し、スキャン、データベース・ブロック変更、CU移入アクティビティおよびCU再移入アクティビティの数に基づいて上位インメモリー・セグメントをリストします。これらの統計で、インメモリー・セグメントがユーザー・ワークロードによってどのように利用されているかを把握できます。インメモリー・セグメント統計セクションは、Oracle Databaseにインメモリー・アクティビティがある場合のみ、AWRの期間比較レポートに表示されます。

ディクショナリ・キャッシュ統計

「ディクショナリ・キャッシュ統計」セクションでは、ディクショナリ・キャッシュで実行されたGETリクエストの数を、降順に表示された2つのスナップショット・セット間で比較されます。DB時間および経過時間の両方で、GETリクエストの数ごとに秒単位で測定されます。

ディクショナリ・キャッシュは、データベース、構造およびそのユーザーの情報を格納するSGAの一部です。また、ディクショナリ・キャッシュは、SQL文の解析中にOracle Databaseにアクセスされたスキーマ・オブジェクトの説明情報(メタデータ)が格納されます。

関連項目:

ディクショナリ・キャッシュの詳細は、「データ・ディクショナリ・キャッシュの概念」を参照してください

ライブラリ・キャッシュ統計

ライブラリ・キャッシュ統計のセクションでは、2つのスナップショット・セット間でライブラリ・キャッシュ上で実行されたGETリクエストの回数が降順に比較されます。DB時間および経過時間の両方で、GETリクエストの数ごとに秒単位で測定されます。

ライブラリ・キャッシュは、表情報、オブジェクト定義、SQL文およびPL/SQLプログラムを格納するSGAの一部です。

関連項目:

ライブラリ・キャッシュの詳細は、「ライブラリ・キャッシュの概念」を参照してください

メモリー統計

「メモリー統計」セクションでは、2つのスナップショット・セット間でプロセスおよびSGAメモリー統計が比較され、次のカテゴリに分類されます。

- プロセス・メモリーのサマリー
- SGAメモリー・サマリー
- SGAブレークダウン差異

プロセス・メモリーのサマリー

このサブセクションでは、2つの期間におけるプロセスのメモリー使用の概要をまとめます。プロセスのカテゴリには、SQL、PL/SQLおよびその他が含まれます。

SGAメモリー・サマリー

このサブセクションでは、2つのスナップショット・セットのSGAメモリー構成をまとめます。

SGAブレークダウン差異

このサブセクションでは、2つのスナップショット・セット間で各サブコンポーネントのSGAメモリー使用率が比較されます。差は、2つのスナップショット・セット間のメモリー使用率の開始時と終了時の変化に基づいて測定されます。

アドバンスト・キューイングの統計

アドバンスト・キューイングの統計セクションでは、CPU時間、I/O時間およびその他の統計が比較されます。

AWRの期間比較レポートの補足情報

補足情報は、AWRの期間比較レポートの最後にあり、有用ですがスナップショット・セットおよびレポートで使用したワークロードには必須ではない情報を提供します。

補足情報は次のセクションで構成されています。

- init.oraパラメータ
- SQLテキストの完全なリスト(Complete List of SQL Text)

init.oraパラメータ

「init.oraパラメータ」セクションには、最初のスナップショット・セットの初期化パラメータ値がリストされています。2つのスナップショット・セット間で初期化パラメータの値が変更されている場合は、2番目のスナップショット・セットに変更された値がリストされます。

SQLテキストの完全なリスト(Complete List of SQL Text)

SQLテキストの完全なリスト・セクションには、ワークロードに含まれる各文がSQL IDごとにリストされ、SQL文のテキストが表示されます。

9 サンプル・データの分析

この章では、サンプル・データを使用して、Oracle Databaseの一時的なパフォーマンスの問題を特定する方法を説明しており、内容は次のとおりです。

- アクティブ・セッション履歴について
- アクティブ・セッション履歴レポートの生成
- アクティブ・セッション履歴レポートの結果の解釈

アクティブ・セッション履歴について

アクティブ・セッション履歴(ASH)は診断ツールで、Oracleデータベースのすべてのアクティブ・セッションに関する情報を記録します。

短時間のパフォーマンスの問題は、一時的なものであるため、自動データベース診断モニター(ADDM)分析では把握できない場合があります。ASH診断ツールは、アクティブ・セッションを毎秒サンプリングし、サンプル・データを共有グローバル領域(SGA)の循環バッファに格納することで、一時的なパフォーマンスの問題を捕捉します。データベースに接続中で、Idle待機クラスに属さないイベントを待機中のセッションは、アクティブ・セッションとみなされます。アクティブ・セッションのみを取得することで、システム上で許可されるセッション数ではなく、実行される作業に直接関連するサイズで管理可能なデータ・セットが表示されます。

ASHを使用すると、V$ACTIVE_SESSION_HISTORYビューで、サンプル・セッション・アクティビティを検査して詳細な分析を実行できます。ASHに存在するデータは、指定した期間に取得する各種ディメンションでロール・アップされ、ASHレポートに収集されます。

ノート:

ADDM では、DB 時間への影響の観点から、分析期間中に最も重要なパフォーマンス上の問題をレポートしようと試みます。パフォーマンス問題が ADDM で取得されるかどうかは、AWR のスナップショット間隔と比較する期間によって決まります。

スナップショット間で相当な時間続くパフォーマンスの問題の場合、ADDM により取得されます。たとえば、スナップショット間隔が 1 時間に設定されている場合、30 分間続くパフォーマンスの問題は、この期間がスナップショット間隔で相当な時間を占め、ADDM により取得される可能性が高いため、一時的なパフォーマンスの問題と考えるのは不適切です。

特定の問題の継続期間がごく短い場合、その重大度は、分析期間にわたり平均化されるか、他のパフォーマンス問題によって最小化されるため、問題が ADDM の検出結果に表れない場合があります。スナップショット間隔が 1 時間に設定されている同じ例を使用した場合、継続期間がわずか 2 分のパフォーマンスの問題は、その期間がスナップショット間隔に占める割合が小さいため ADDM の検出結果に現れにくく、一時的なパフォーマンスの問題と言えます。

関連項目:

- ASHの詳細は、「アクティブ・セッション履歴の統計」を参照してください
- マルチテナントのコンテナ・データベースにおける、ASHなどの管理機能の仕組みの詳細は、Oracle Multitenant管理者ガイドを参照してください。

アクティブ・セッション履歴レポートの生成

ASHレポートを使用すると、次の分析の実行が可能になります。

- 通常数分間で収まる一時的なパフォーマンスの問題
- 時間、セッション、モジュール、アクション、またはSQL識別子など、様々なディメンションやその組合せによる、指定範囲または特定のターゲットのパフォーマンス分析

ASHレポートは複数のセクションに分かれています。HTMLレポートには、セクション間ですばやくナビゲートできるようにリンクが組み込まれています。レポートの内容には、指定された期間の、ブロッカIDおよび待機中IDとその関連トランザクション識別子、およびSQL文の識別に使用されたASH情報が含まれています。

この項では、ASHレポートの生成方法を説明しており、内容は次のとおりです。

- ASHレポートを生成するためのユーザー・インタフェース
- コマンドライン・インタフェースを使用したASHレポートの生成

ASHレポートを生成するためのユーザー・インタフェース

ASHレポートを生成するためのプライマリ・インタフェースは、Oracle Enterprise Manager Cloud Control (Cloud Control)です。可能な場合は、Cloud Controlを使用してASHレポートを生成してください。

Cloud Controlを使用できない場合は、SQLスクリプトを実行してASHレポートを生成します。これらのスクリプトを実行するには、DBAロールが必要です。

関連項目:

Cloud Controlを使用したASHレポートの生成の詳細は、『Oracle Database 2日でパフォーマンス・チューニング・ガイド』を参照してください

コマンドライン・インタフェースを使用したASHレポートの生成

この項では、コマンドライン・インタフェースでSQLスクリプトを実行し、ASHレポートを生成する方法を説明します。

この項では、次の項目について説明します。

- ローカル・データベース・インスタンスのASHレポートの生成
- 特定のデータベース・インスタンスのASHレポートの生成
- Oracle RAC ASHレポートの生成

ローカル・データベース・インスタンスのASHレポートの生成

ashrpt.sql SQLスクリプトでは、ローカル・データベース・インスタンスの指定された期間のASH情報を表示する、HTMLまたはテキストのレポートが生成されます。

コマンドライン・インタフェースを使用して、ローカル・データベース・インスタンスにASHレポートを生成するには:

1. SQLプロンプトで次のように入力します。

   ```
   @$ORACLE_HOME/rdbms/admin/ashrpt.sql
   ```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

   ```
   Enter value for report_type: text
   ```

   この例では、テキスト形式のレポートが選択されます。

3. システム日付より前の開始時間を分単位で指定します。

   ```
   Enter value for begin_time: -10
   ```

   この例では、現在の時刻より10分前が選択されます。

4. 開始時間からASH情報を取得する期間を分単位で指定します。

   ```
   Enter value for duration:
   ```

   この例では、システム日付から開始時間をマイナスしたデフォルトの期間が使用されます。

5. レポート名を入力するか、デフォルトのレポート名を受け入れます。

   ```
   Enter value for report_name:
   Using the report name ashrpt_1_0310_0131.txt
   ```

   この例では、デフォルト名が使用され、ashrpt_1_0310_0131というASHレポートが生成されます。このレポートでは、現在の時刻の10分前から開始され、現在の時刻で終了するASH情報が収集されます。

特定のデータベース・インスタンスのASHレポートの生成

ashrpti.sql SQLスクリプトでは、指定されたデータベースおよびインスタンスの指定された期間のASH情報を表示する、HTMLまたはテキストのレポートが生成されます。このスクリプトでは、ASHレポートの生成に使用されるデータベースおよびインスタンスを指定できます。

コマンドライン・インタフェースを使用して、特定のデータベース・インスタンスにASHレポートを生成するには:

1. SQLプロンプトで次のように入力します。

   ```
   @$ORACLE_HOME/rdbms/admin/ashrpti.sql
   ```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

   ```
   Enter value for report_type: html
   ```

   この例では、HTML形式のレポートが選択されます。

   使用可能なデータベースIDおよびインスタンス番号のリストが表示されます。

   ```
   Instances in this Workload Repository schema
   ```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   DB Id    Inst Num DB Name      Instance     Host
------------ -------- ------------ ------------ ------------
3309173529         1 MAIN         main         examp1690
3309173529         1 TINT251      tint251      samp251
```

3. 次のようにデータベース識別子(dbid)およびインスタンス番号(inst_num)の値を入力します。

```
Enter value for dbid: 3309173529
Using 3309173529 for database id
Enter value for inst_num: 1
```

4. フィジカル・スタンバイ・インスタンスのASHレポートを生成するには、スタンバイ・データベースを読取り専用でオープンする必要があります。ディスク上のASHデータはプライマリ・データベースでのアクティビティを表し、メモリー内のASHデータはスタンバイ・データベースでのアクティビティを表します。

ノート:

このステップは、Active Data Guard フィジカル・スタンバイ・インスタンスの ASH レポートを生成する場合にのみ該当します。そうでない場合は、このステップをスキップしてください。

レポートの生成に、プライマリ・データベースまたはスタンバイ・データベースのどちらのサンプリング・データを使用するかを指定します。

```
You are running ASH report on a Standby database.
To generate the report over data sampled on the Primary database, enter 'P'.
Defaults to 'S' - data sampled in the Standby database.
Enter value for stdbyflag:
Using Primary (P) or Standby (S): S
```

この例では、デフォルト値のスタンバイ(S)が選択されています。

5. システム日付より前の開始時間を分単位で指定します。

```
Enter value for begin_time: -10
```

この例では、現在の時刻より10分前が選択されます。

6. 開始時間からASH情報を取得する期間を分単位で指定します。

```
Enter value for duration:
```

この例では、システム日付から開始時間をマイナスしたデフォルトの期間が使用されます。

7. レポートの「アクティビティの経過(Activity Over Time)」セクションで使用する時間帯(秒数)を指定します。

```
Enter value for slot_width:
```

この例では、デフォルト値を使用します。「アクティビティの経過(Activity Over Time)」セクションおよび時間帯の指定方法の詳細は、「アクティビティの経過(Activity Over Time)」を参照してください。

8. 後続のプロンプトの手順に従って、次のレポート・ターゲットの値を入力します。
   - target_session_id
   - target_sql_id

- target_wait_class
- target_service_hash
- target_module_name
- target_action_name
- target_client_id
- target_plsql_entry

9. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name ashrpt_1_0310_0131.txt
```

この例では、デフォルト名が使用され、ashrpt_1_0310_0131というASHレポートが生成されます。このレポートでは、3309173529というデータベースIDを持つデータベース・インスタンスにおいて、現在の時刻の10分前から開始され、現在の時刻で終了するASH情報が収集されます。

Oracle RAC ASHレポートの生成

ashrpti.sql SQLスクリプトは、Oracle Real Application Clusters (Oracle RAC)環境の指定されたデータベースおよびインスタンスにおける指定された期間のASH情報を表示する、HTMLまたはテキストのレポートを生成します。ディスクに書き込まれるASHデータのみがレポートの生成に使用されます。このレポートでは、DBA_HIST_ACTIVE_SESS_HISTORY表にある最後の10分間でのASHサンプルのみが使用されます。

Oracle RAC ASHレポートを生成するには:

1. SQLプロンプトで次のように入力します。

```
@$ORACLE_HOME/rdbms/admin/ashrpti.sql
```

2. レポートの形式としてHTMLまたはテキストのいずれかを指定します。

```
Enter value for report_type: html
```

この例では、HTML形式のレポートが選択されます。

使用可能なデータベースIDおよびインスタンス番号のリストが表示されます。

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ ------------
 3309173529        1 MAIN         main         examp1690
 3309173529        1 TINT251      tint251      samp251
 3309173529        2 TINT251      tint252      samp252
 3309173529        3 TINT251      tint253      samp253
 3309173529        4 TINT251      tint254      samp254
```

3. 次のようにデータベース識別子(dbid)およびインスタンス番号(inst_num)の値を入力します。

```
Enter value for dbid: 3309173529
Using database id: 3309173529
Enter instance numbers. Enter 'ALL' for all instances in an Oracle
RAC cluster or explicitly specify list of instances (e.g., 1,2,3).
Defaults to current instance.
```

```
Enter value for inst_num: ALL
Using instance number(s): ALL
```

4. システム日付より前の開始時間を分単位で指定します。

```
Enter value for begin_time: -1:10
```

この例では、現在の時刻より1時間10分前が選択されます。

5. 開始時間からASH情報を取得する期間を分単位で指定します。

```
Enter value for duration: 10
```

この例では、期間は10分間に設定されます。

6. レポートの「アクティビティの経過(Activity Over Time)」セクションで使用する時間帯(秒数)を指定します。

```
Enter value for slot_width:
```

この例では、デフォルト値を使用します。「アクティビティの経過(Activity Over Time)」セクションおよび時間帯の指定方法の詳細は、「アクティビティの経過(Activity Over Time)」を参照してください。

7. 後続のプロンプトの手順に従って、次のレポート・ターゲットの値を入力します。

- target_session_id
- target_sql_id
- target_wait_class
- target_service_hash
- target_module_name
- target_action_name
- target_client_id
- target_plsql_entry

8. レポート名を入力するか、デフォルトのレポート名を受け入れます。

```
Enter value for report_name:
Using the report name ashrpt_rac_0310_0131.txt
```

この例では、デフォルト名が使用され、ashrpt_rac_0310_0131というASHレポートが生成されます。このレポートでは、3309173529というデータベースIDを持つデータベースに所属するすべてのインスタンスにおいて、現在の時刻の1時間10分前から開始され、現在の時刻の1時間前に終了するASH情報が収集されます。

アクティブ・セッション履歴レポートの結果の解釈

ASHレポートを生成したら、その内容を確認して一時的なパフォーマンス問題の可能性のある原因を特定します。

ASHレポートの内容は、次のセクションに分けられます。

- 上位イベント(Top Events)
- ロード・プロファイル(Load Profile)
- 上位SQL(Top SQL)

- 上位PL/SQL(Top PL/SQL)
- 上位Java(Top Java)
- 上位セッション(Top Sessions)
- 上位オブジェクト/ファイル/ラッチ(Top Objects/Files/Latches)
- アクティビティの経過(Activity Over Time)

関連項目:

Oracle Real Application Clusters(Oracle RAC)に固有のASHレポートのセクションの詳細は、『Oracle Real Application Clusters管理およびデプロイメント・ガイド』を参照してください。

上位イベント(Top Events)

「上位イベント(Top Events)」セクションは、ユーザー、バックグラウンドおよび優先順位で分類されたサンプル・セッション・アクティビティの上位待機イベントを示します。このセクションの情報を使用して、一時的なパフォーマンス問題の原因となっている可能性のある待機イベントを特定します。

「上位イベント(Top Events)」セクションには、次のサブセクションが含まれます。

- 上位ユーザー・イベント(Top User Events)
- 上位バックグラウンド・イベント(Top Background Events)
- 上位イベントP1/P2/P3(Top Event P1/P2/P3)

上位ユーザー・イベント(Top User Events)

「Top User Events」サブセクションはサンプリングされたセッション・アクティビティの最も高い割合を占める、ユーザー・プロセスから上位の待機イベントをリストします。

上位バックグラウンド・イベント(Top Background Events)

「Top Background Events」サブセクションはサンプリングされたセッション・アクティビティの最も高い割合を占めるバックグラウンドからの上位の待機イベントをリストします。

上位イベントP1/P2/P3

「上位イベントP1/P2/P3(Top Event P1/P2/P3)」サブセクションでは、サンプリングされたセッション・アクティビティの最も高い割合を占める、上位の待機イベントの待機イベント・パラメータ値がリストされ、総待機時間の割合(%イベント)の順に表示されます。各待機イベントにおいて、「P1値、P2値、P3値(P1 Value, P2 Value, P3 Value)」列の値は、「パラメータ1(Parameter 1)」、「パラメータ2(Parameter 2)」、「パラメータ3(Parameter 3)」の各列に表示される待機イベント・パラメータに対応します。

ロード・プロファイル(Load Profile)

「ロード・プロファイル(Load Profile)」セクションは、サンプル・セッション・アクティビティで分析された負荷を示します。このセクショ

ンの情報を使用して、一時的なパフォーマンス問題の原因となっている可能性のあるサービス、クライアントまたはSQLコマンド・タイプを特定します。

「ロード・プロファイル(Load Profile)」セクションには、次のサブセクションが含まれます。

- 上位サービス/モジュール(Top Service/Modules)
- 上位クライアントID(Top Client IDs)
- 上位SQLコマンド・タイプ(Top SQL Command Types)
- 上位実行フェーズ(Top Phases of Execution)

上位サービス/モジュール(Top Service/Module)

「上位サービス/モジュール」セクションには、サンプリングされたセッション・アクティビティの最も高い割合を占めるサービスおよびモジュールがリストされます。

上位クライアントID (Top Client IDs)

「Top Client IDs」サブセクションに、クライアントIDに基づいてサンプリングされたセッション・アクティビティの最も高い割合を占めるクライアントがリストされます。クライアントIDはデータベース・セッションのアプリケーション固有の識別子です。

上位SQLのコマンド・タイプ

上位SQLのコマンド・タイプ・サブセクションには、サンプリングされたセッション・アクティビティの最も高い割合を占めるSQLコマンド・タイプ(SELECTまたはUPDATEコマンド)がリストされます。

上位実行フェーズ

上位実行フェーズ・サブセクションには、サンプリングされたセッション・アクティビティの最も高い割合を占めている実行のフェーズ(SQL、PL/SQLおよびJavaのコンパイルや実行など)がリストされます。

上位SQL

「上位SQL」セクションではサンプリングされたセッション・アクティビティの上位SQL文を説明します。この情報を使用して、一時的なパフォーマンス問題の原因となっている可能性のある高負荷のSQL文を特定します。

「上位SQL(Top SQL)」セクションには、次のサブセクションが含まれます。

- 上位SQLと上位イベント(Top SQL with Top Events)
- 上位SQLと上位行ソース(Top SQL with Top Row Sources)
- リテラルを使用する上位SQL(Top SQL Using Literals)
- 上位解析モジュール/アクション(Top Parsing Module/Action)
- SQLテキストの完全なリスト(Complete List of SQL Text)

上位SQLと上位イベント(Top SQL with Top Events)

「上位SQLと上位イベント(Top SQL with Top Events)」サブセクションは、サンプル・セッション・アクティビティで最高の割合

を占めるSQL文と、それらのSQL文に対して発生した上位待機イベントを示します。「実行のサンプリング数」列に、特定のSQL文ごとに実行のサンプリング数が表示されます。

上位SQLと上位行ソース(Top SQL with Top Row Sources)

「上位SQLと上位行ソース(Top SQL with Top Row Sources)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるSQL文と、それらの詳細な実行計画情報を示します。この情報を使用することで、SQLの経過時間に多くの影響を与えているSQL実行部分を特定できます。

リテラルを使用する上位SQL(Top SQL Using Literals)

「リテラルを使用する上位SQL(Top SQL Using Literals)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるリテラルを使用するSQL文を示します。このレポートにリストされた文を確認して、リテラルをバインド変数に置換できるかどうかを決定する必要があります。

上位解析モジュール/アクション(Top Parsing Module/Action)

「上位解析モジュール/アクション(Top Parsing Module/Action)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるSQL文解析時のモジュールおよびアクションを示します。

SQLテキストの完全なリスト(Complete List of SQL Text)

SQLテキストの完全なリスト・サブセクションには、「上位SQL」セクションに表示されたSQL文のテキスト全体が表示されます。

上位PL/SQL (Top PL/SQL)

「上位PL/SQL(Top PL/SQL)」セクションは、サンプル・セッション・アクティビティで最高の割合を占めるPL/SQLプロシージャを示します。

「PL/SQLエントリ・サブプログラム(PL/SQL Entry Subprogram)」列は、PL/SQLに対するアプリケーションの最上位レベルのエントリ・ポイントを示します。「PL/SQL現行サブプログラム(PL/SQL Current Subprogram)」列は、サンプリング時点で実行されていたPL/SQLサブプログラムを示します。この列の値がSQLの場合、「%現行(% Current)」列に、このサブプログラムのSQL実行にかかった時間の割合が示されます。

上位Java

「上位Java(Top Java)」セクションは、サンプル・セッション・アクティビティの上位Javaプログラムを示します。

上位セッション

「上位セッション(Top Sessions)」セクションは、特定の待機イベントを待機していたセッションを示します。この情報を使用して、一時的なパフォーマンス問題の原因となっている可能性のある、サンプリングされたセッション・アクティビティの最も高い割合を占めるセッションを特定します。

「上位セッション(Top Sessions)」セクションには、次のサブセクションが含まれます。

- 上位セッション

- 上位のブロック・セッション(Top Blocking Sessions)
- PQを実行している上位セッション(Top Sessions Running PQs)

上位セッション

「上位セッション(Top Sessions)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占める特定の待機イベントを待機していたセッションを示します。

上位のブロック・セッション(Top Blocking Sessions)

「上位のブロック・セッション(Top Blocking Sessions)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるブロック・セッションを示します。

PQを実行している上位セッション(Top Sessions Running PQs)

「PQを実行している上位セッション(Top Sessions Running PQs)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占める、特定の待機イベントを待機していたパラレル問合せ(PQ)を実行中のセッションを示します。

上位オブジェクト/ファイル/ラッチ(Top Objects/Files/Latches)

「上位オブジェクト/ファイル/ラッチ(Top Objects/Files/Latches)」セクションは、最もよく使用されるデータベース・リソースに関する追加情報を示し、次のサブセクションを含みます。

- 上位DBオブジェクト(Top DB Objects)
- 上位DBファイル(Top DB Files)
- 上位ラッチ(Top Latches)

上位DBオブジェクト(Top DB Objects)

「上位DBオブジェクト(Top DB Objects)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるデータベース・オブジェクト(表や索引など)を示します。

上位DBファイル(Top DB Files)

「上位DBファイル(Top DB Files)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるデータベース・ファイルを示します。

上位ラッチ(Top Latches)

「上位ラッチ(Top Latches)」サブセクションは、サンプル・セッション・アクティビティで最高の割合を占めるラッチを示します。

ラッチは、単純で低レベルなシリアライズ・メカニズムで、システム・グローバル領域(SGA)の共有データ構造の保護に使用されます。たとえば、ラッチにより、データベースに現在アクセスしているユーザーのリストと、バッファ・キャッシュのブロックを記述するデータ構造が保護されます。サーバーまたはバックグラウンド・プロセスは、これらの構造の操作または調査時に、ごく短時間ラッチを取得します。ラッチの実装は、特に、プロセスがラッチを待機するかどうかとその待機時間に関して、オペレーティング・システムに依存します。

アクティビティの経過(Activity Over Time)

「アクティビティの経過(Activity Over Time)」セクションは、ASHレポートの中で最も有益な情報の含まれるセクションの1つです。このセクションには、分析期間中のアクティビティおよびワークロード・プロファイルに関する詳細な情報が含まれるため、長い期間分析する場合に特に役立ちます。

「アクティビティの経過(Activity Over Time)」セクションは、10個の時間帯に分けられます。各時間帯のサイズは、分析期間の継続時間に基づいて変化します。通常、最初と最後の時間帯は、余り部分になります。その内側の時間帯は、すべて同じサイズで、相互に比較できます。たとえば、分析期間が10分間続くと、すべての時間帯がそれぞれ1分となります。ただし、分析期間が9分30秒続くと、両端の時間帯はそれぞれ15秒となり、内側の時間帯はそれぞれ1分となります。

表9-1に示すとおり、各時間帯にはその特定の時間帯に関する情報が含まれます。

表9-1 アクティビティの経過

列	説明
時間帯(期間) (Slot Time (Duration))	時間帯の継続時間
時間帯の数(Slot Count)	時間帯のサンプル・セッションの数
イベント(Event)	時間帯の上位 3 つの待機イベント
イベント数(Event Count)	待機イベントを待機している ASH サンプルの数
イベントの割合(% Event)	分析期間全体で待機イベントを待機している ASH サンプルの割合

内側の時間帯を比較する場合、「イベント数(Event Count)」列と「時間帯の数(Slot Count)」列のスパイクを識別して、スキュー解析を実行します。「イベント数(Event Count)」列のスパイクは、特定のイベントを待機しているサンプル・セッションの数の増大を示します。「時間帯の数(Slot Count)」列のスパイクは、アクティブ・セッションの増大(ASHデータはアクティブ・セッションからのみサンプリングされるため)と、データベース・ワークロードの相対的な増大を示します。通常、アクティブ・セッションのサンプルの数と、待機イベントに関連付けられたセッションの数が増えた場合、スロットは一時的なパフォーマンス問題の原因となる可能性があります。

ユーザー定義の時間帯サイズでASHレポートを生成するには、「特定のデータベース・インスタンスのASHレポートの生成」の手順に従ってashrpti.sqlスクリプトを実行します。

10 パフォーマンス・ビューを使用したインスタンスのチューニング

データベースの初期構成後は、インスタンスの定期的な監視およびチューニングがパフォーマンスの潜在的なボトルネックを解消するために重要になります。この章では、OracleのV$パフォーマンス・ビューを使用したチューニング・プロセスについて説明します。

この章の構成は、次のとおりです。

- インスタンスのチューニング・ステップ
- Oracle Database統計の解釈
- 待機イベント統計
- インスタンス・リカバリのパフォーマンスのチューニング: ファスト・スタート・フォルト・リカバリ

インスタンスのチューニング・ステップ

次に、インスタンスのチューニング用のOracleパフォーマンス・メソッドの主なステップを示します。

1. 問題の定義

   パフォーマンス問題の範囲についてユーザーから候補フィードバックを取得します。

2. ホスト・システムの検査およびOracle Database統計の調査
   - オペレーティング・システム、データベースおよびアプリケーション統計一式を取得後に、パフォーマンスの問題の徴候を探すためにデータを調べます。
   - 一般的なパフォーマンス・エラーのリストを検討して、収集されたデータが問題に影響を与えていることを示しているかどうかを確認します。
   - 収集されたパフォーマンス・データを使用して、何がシステムで起こっているかを示す概念モデルを構築します。
3. 変更の実装および測定

   行う変更および変更を実装した場合に予測される結果を提示します。次に、変更を実装してアプリケーション・パフォーマンスを測定します。

4. ステップ1で定義したパフォーマンスの目的が達成されたかどうかを判断します。達成されていない場合は、パフォーマンスの目標が達成されるまでステップ2と3を繰り返します。

この章の残りの部分では、Oracle Databaseの動的パフォーマンス・ビューを使用したインスタンスのチューニングについて説明します。ただし、機能リストの拡張により、統計の収集、監視およびチューニングには、自動ワークロード・リポジトリ(AWR)および自動データベース診断モニター(ADDM)を使用することをお薦めします。

ノート:

AWR および ADDM 機能がない場合、Statspack を使用して Oracle データベース・インスタンスの統計情報を収集できます。

問題の定義

ソリューションの実装を試みる前に、チューニング調査の目的と問題の性質をよく理解しておくことが不可欠です。これについて理解していないと、事実上、効果的な変更は実装できません。この段階で収集されたデータを使用して、行う次のステップおよび調査する事象を簡単に決定できます。

次のデータを収集します。

1. パフォーマンスの目的を識別します。

   許容できるパフォーマンスの測定尺度は何ですか。1時間、または1秒間当たり何件のトランザクションで、レスポンス時間が必要なパフォーマンス・レベルを満たしますか。

2. 問題の範囲を識別します。

   スローダウンで何が影響を受けますか。たとえば、インスタンス全体は低速ですか。それは、特定のアプリケーション、プログラム、特定の操作またはシングル・ユーザーですか。

3. 問題が発生したときの時間帯を識別します。

   その問題はピーク時間のみ明白ですか。パフォーマンスはその日の経過に伴って低下しますか。スローダウンは徐々に(月または週の単位で)発生しましたか、または突然発生しましたか。

4. スローダウンを検証します。

   これは、問題の範囲の識別に役立ち、問題の修復のために実装された変更により実際に改善されたかどうかを判断するときの比較結果の測定基準としての役割を果します。一貫して再生可能なレスポンス時間またはジョブ実行時間の測定値を検索します。プログラムの動作が正常だったときよりタイミングがどのくらい悪化していますか。

5. 変更を識別します。

   パフォーマンスが許容可能になった後に変化した内容を識別します。これにより、潜在的な原因を素早くつきとめることができます。たとえば、オペレーティング・システムのソフトウェア、ハードウェア、アプリケーション・ソフトウェアまたはOracle Databaseリリースのアップグレードを実行しましたか。さらに多くのデータがシステムにロードされたか、データ・ボリュームまたはユーザー人口が増加しましたか。

このフェーズの終わりまでに、症状についてよく理解しておく必要があります。症状をプログラムまたはプログラム・セットにローカルなものとして識別できる場合、その問題はインスタンス全体のパフォーマンスの問題とは異なる方法で処理されます。

ホスト・システムの検査

データベース・サーバーに対する負荷とデータベース・インスタンスを調べてください。オペレーティング・システム、I/Oサブシステムおよびネットワーク統計を検討してください。これらの領域を調べると、調査する価値のあるものは何かが容易にわかります。多層のシステムでは、アプリケーション・サーバーの中間層ホストも調べてください。

ホスト・ハードウェアを調べると、システム内のボトルネックがよくわかります。この調査によって、相互参照および今後の診断に役立つOracle Databaseパフォーマンス・データを判断できます。

調べるデータには、次のものがあります。

- CPU使用率
- I/Oの問題の識別

- ネットワークの問題の識別

CPU使用率

アイドル状態のCPUが大量にある場合、I/O、アプリケーションまたはデータベースのボトルネックが存在する可能性があります。I/O待機はアイドル状態のCPUとしてみなす必要があります。

CPU使用率が高い場合は、CPUが効果的に使用されているかどうかを判断してください。CPU使用率の大部分は、CPU使用率の高い少数のプログラムによるものですか、または均等に分散されたワークロードでCPUが消費されていますか。

CPUが使用頻度の高い少量のプログラムで使用されている場合は、プログラムを調べて原因を判断してください。一部のプロセスのみが1つのCPUの能力全体を使用しているかどうかを確認してください。プロセスによっては、この情報はCPUまたはプロセスによりワークロードがバインドされていることを示している場合があり、プロセス・アクティビティを分割またはパラレル化することで解決できます。

Oracle以外のプロセス

プログラムがOracleのプログラムではない場合は、それらのプログラムがそのような量のCPUを必要としているかどうかを識別してください。必要としている場合は、プログラムの実行をピーク以外の時間に遅らせることができるかどうかを判断します。これらのCPU集中型プロセスを識別することで、I/O、ネットワークおよびページングなどのどのアクティビティがリソースを消費しているかを特定し、データベース・ワークロードとの関係を確認できます。

Oracleプロセス

少数のOracleプロセスによって多くのCPUリソースが使用されている場合は、SQL_TRACEとTKPROFを使用してSQL文またはPL/SQL文を特定して、特定の問合せまたはPL/SQLプログラム・ユニットをチューニングできるかどうかを確認します。たとえば、SELECT文の実行によりキャッシュ内の多数のデータ読取り(論理読取り)が発生する場合、さらなるSQLの最適化によってCPUの集中的な使用を回避できます。

Oracle Database CPU統計

Oracle Database CPU統計は、V$ビューで取得できます。

- V$SYSSTATは、すべてのセッションのOracle Database CPU使用率を示します。CPU used by this session統計は、すべてのセッションで使用されているCPUの集計を示します。parse time cpu統計は、解析に使用された合計CPU時間を示します。
- V$SESSTATは、各セッションのOracle Database CPU使用率を示します。このビューを使用して、CPUを最も多く使用している特定のセッションを調べます。
- Oracle Database Resource Managerを実行している場合、各コンシューマ・グループのCPU使用率の統計情報がV$RSRC_CONSUMER_GROUPに表示されます。

CPU統計の解釈

CPU時間と実時間の違いを認識しておくことは重要です。8個のCPUの場合、実時間で1分間に使用可能なCPU時間は8分間です。この時間は、WindowsおよびUNIXではユーザー時間またはシステム時間です(Windowsでは特権モード)。したがって、システム上のすべてのプロセス(スレッド)で使用される平均CPU時間は、1分の実時間間隔当たり1分を超える可能性があります。

どの特定の時点においても、システム上でOracle Databaseが使用した時間を識別できます。使用可能な時間が8分間で、Oracle Databaseがそのうちの4分間を使用した場合、総CPU時間の50%がOracleによって使用されたことがわかります。ユーザーのプロセスがその時間を消費していない場合は、他のプロセスが消費しています。CPU時間を使用しているプロセスを識別し、原因を解明し、それらのプロセスのチューニングを試行してください。

CPU使用率が多数のOracleサーバー・プロセスに均一に分散している場合は、V$SYS_TIME_MODELビューを調べると、最長時間が消費されているプロセスを正確に把握できます。

関連項目:

様々な待機イベントと考えられるその原因の詳細は、「表10-1」を参照してください

I/Oの問題の識別

過度にアクティブなI/Oシステムは、2より大きいディスク・キューの長さ、すなわち、20から30ミリ秒を超えるディスク・サービス時間でわかります。I/Oシステムが過度にアクティブである場合、さらに多くのディスク間にI/Oを分散させることで利益を得られる潜在的なホット・スポットの有無をチェックします。また、これらのリソースを使用して、プログラムのリソース要件を少なくして負荷を減らせるかどうかも識別します。I/O問題の原因がOracle Databaseである場合、I/Oチューニングを開始できます。Oracle Databaseが使用可能なI/Oリソースを消費していない場合、I/Oを消費しているプロセスを識別します。プロセスがI/Oを消費している原因を究明して、プロセスをチューニングします。

I/O問題は、Oracle DatabaseのV$ビューおよびオペレーティング・システムのモニタリング・ツールによって識別できます。次の項では、I/O問題の識別について説明します。

- V$ビューを使用したI/Oの問題の識別
- オペレーティング・システムのモニタリング・ツールを使用したI/Oの問題の識別

V$ビューを使用したI/Oの問題の識別

V$SYSTEM_EVENTのOracle待機イベント・データをチェックして、トップの待機イベントがI/O関連かどうかを確認します。I/O関連イベントには、db file sequential read、db file scattered read、db file single write、db file parallel writeおよびlog file parallel writeがあります。これらはいずれも、データファイルおよびログ・ファイルに対して実行されたI/Oに対応するイベントです。これらの待機イベントが高い平均時間に該当する場合は、I/Oの競合を調べる必要があります。

自動ワークロード・リポジトリ・レポート内のI/Oセクションでホスト I/Oシステム・データを相互参照し、ホット・データファイルおよび表領域を識別します。さらに、オペレーティング・システムから報告されたI/O時間と、Oracle Databaseから報告された時間とを比較して、それらに一貫性があるかどうかを確認します。

I/Oの問題によって、I/Oに関連しない待機イベントが明らかになる場合もあります。たとえば、バッファ・キャッシュ内で空きバッファを検出できない場合や、ディスクへのログ書込みが完了するまでの待機時間が長い場合も、I/O問題の症状を示す場合があります。I/Oシステムを再構成する必要があるかどうかを調べる前に、I/Oシステム上の負荷を減らせるかどうかを判断します。

Oracle Databaseを原因とするI/O負荷を減らすには、次のビューを使用して、データベースによるすべてのI/Oコールに対して収集されたI/O統計を調査します。

- V$IOSTAT_CONSUMER_GROUP

  V$IOSTAT_CONSUMER_GROUPビューには、コンシューマ・グループのI/O統計が取得されます。Oracle Database Resource Managerが有効な場合、現在有効なリソース・プランに含まれるすべてのコンシューマ・グループのI/O統計が取得されます。

- V$IOSTAT_FILE

  V$IOSTAT_FILEビューには、現在アクセスされているか、または過去にアクセスされたデータベース・ファイルのI/O統計が取得されます。SMALL_SYNC_READ_LATENCY列には、単一ブロック同期読取り(ミリ秒単位)の待機時間が表示されますが、これはクライアントが次の操作に移行する前に待機する必要のある時間を表します。これにより、現在の負荷に基づいたストレージ・サブシステムのレスポンス性が定義されます。重要なデータファイルに対する待機時間が長い場合、それらのファイルを再配置してサービス時間を短縮することを検討してください。待機時間の統計を計算するには、timed_statisticsがTRUEに設定されている必要があります。

- V$IOSTAT_FUNCTION

  V$IOSTAT_FUNCTIONビューには、データベース機能(LGWRやDBWRなど)のI/O統計が取得されます。

  I/Oは、異なる機能を持つ様々なOracleプロセスによって発行されます。上位のデータベース機能は、V$IOSTAT_FUNCTIONビューに分類されます。I/O機能間の競合がある場合、そのI/OはよりFUNCTION_IDのバケットに配置されます。たとえば、XDBがバッファ・キャッシュからI/Oを発行する場合、そのI/OはXDB I/Oに分類されますが、これはXDB I/OのFUNCTION_ID値の方が小さいためです。未分類の機能は、Othersバケットに配置されます。次のようにV$IOSTAT_FUNCTIONビューを問い合せると、FUNCTION_IDの階層を表示できます。

```
select FUNCTION_ID, FUNCTION_NAME
from v$iostat_function
order by FUNCTION_ID;
FUNCTION_ID FUNCTION_NAME
----------- ------------------
          0 RMAN
          1 DBWR
          2 LGWR
          3 ARCH
          4 XDB
          5 Streams AQ
          6 Data Pump
          7 Recovery
          8 Buffer Cache Reads
          9 Direct Reads
         10 Direct Writes
         11 Others
```

これらのV$IOSTATビューには、単一ブロックと複数ブロックの読取り/書込み操作のI/O統計が含まれます。単一ブロック操作は、128KB以下の小規模なI/Oです。複数ブロック操作は、128KBを超える大規模なI/Oです。これらの操作ごとに、次の統計が収集されます。

- 識別子
- 合計待機時間(ミリ秒)
- 実行された待機操作数(コンシューマ・グループおよび機能)
- 操作ごとのリクエスト数
- 単一および複数ブロックの読取りバイト数

- 単一および複数ブロックの書込みバイト数

また、V$SQLAREAビューの問合せか、自動ワークロード・リポジトリ・レポートの「読取りによるSQLのソート」セクションの確認によって、多数の物理読取りを実行するSQL文についても調べる必要があります。これらの文を調べて、I/Oの回数を減らすようにこれらのSQL文をチューニングする方法を調べます。

関連項目:

V$IOSTAT_CONSUMER_GROUP、V$IOSTAT_FUNCTION、V$IOSTAT_FILEおよびV$SQLAREAビューの詳細は、『Oracle Databaseリファレンス』を参照してください

オペレーティング・システムのモニタリング・ツールを使用したI/Oの問題の識別

オペレーティング・システムのモニタリング・ツールを使用して、システム全体で実行されているプロセスを判別し、すべてのファイルに対するディスク・アクセスを監視してください。データファイルとREDOログ・ファイルを保持しているディスクは、Oracle Databaseに関連しないファイルも保持している可能性があります。データベース・ファイルを含むディスクに対する過度のアクセスを減らしてください。データベース以外のファイルへのアクセスは、V$ビューではなく、オペレーティング・システムの機能でのみ監視できます。

多数のUNIXシステム上のsar -d(またはiostat)やWindowsシステム上の管理パフォーマンス・モニタリング・ツールなどのユーティリティは、システム全体のI/O統計を調べます。

関連項目:

プラットフォームで使用可能なツールのオペレーティング・システムのマニュアル

ネットワークの問題の識別

オペレーティング・システムのユーティリティを使用して、ネットワーク・ラウンドトリップのping時間と衝突数を調べます。ネットワークでレスポンス時間の大幅な遅延が発生している場合は、考えられる原因を調べてください。

データベース・ファイルへのリモート・アクセスを原因とするネットワークI/Oを識別するには、V$IOSTAT_NETWORKビューを調査します。このビューには、リモート・データベース・インスタンスのファイルへのアクセスに基づく次のようなネットワークI/O統計が含まれます。

- ネットワークI/Oを開始したデータベース・クライアント(RMANやPLSQLなど)
- 発行された読取りおよび書込み操作の数
- 読取りおよび書込みKB数
- 読取り操作の合計待機時間(ミリ秒単位)
- 書込み操作の合計待機時間(ミリ秒単位)

ネットワーク問題の原因の識別後にネットワーク負荷を減らすには、大きいデータ転送をピーク時間外へスケジュールするか、リモート・ホストに対してリクエスト当たり1回ずつ(またはそれ以上)アクセスせずに、リクエストをバッチ処理するようにアプリケーションをコーディングします。

Oracle Database統計の調査

Oracle Database統計を確認し、それをオペレーティング・システムの統計と照合して、一貫性のある問題の診断を行います。オペレーティング・システムの統計では、チューニングの出発点となる最適な情報を取得できます。ただし、Oracleデータベース・インスタンスをチューニングすることが目的の場合は、修正処置を行う前に、Oracle Database統計を調べ、データベースの観点からリソースのボトルネックを特定します。

この項では、次の項目について説明します。

- 統計収集のレベルの設定
- 待機イベント
- 待機イベント統計を含む動的パフォーマンス・ビュー
- システム統計
- セグメント・レベルの統計

関連項目:

Oracle Database統計の解釈

統計収集のレベルの設定

Oracle Databaseには、データベースのすべての主要な統計収集またはアドバイザを制御する初期化パラメータSTATISTICS_LEVELがあります。このパラメータは、データベースの統計収集レベルを設定します。

STATISTICS_LEVELの設定に応じて、次のように一定のアドバイザまたは統計が収集されます。

- BASIC: アドバイザも統計も収集されません。監視機能および多数の自動機能が無効です。重要なOracle Database機能が無効になるため、この設定は使用しないことをお薦めします。
- TYPICAL: これはデフォルト値であり、すべての主要統計が収集され、データベース全体のパフォーマンスが最高になります。ほとんどの環境では、この設定で十分です。
- ALL: TYPICAL設定を使用して収集されるすべてのアドバイザと統計に加えて、オペレーティング・システム時間統計および行ソース実行統計が含まれます。

関連項目:

- STATISTICS_LEVEL初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください
- V$STATISTICS_LEVELビューの詳細は、Oracle Databaseリファレンスを参照してください。このビューには、STATISTICS_LEVEL初期化パラメータで制御される統計またはアドバイザのステータスがリストされます。

待機イベント

待機イベントは、処理を継続する前にイベントが完了するまで待機する必要があることを示すために、サーバー・プロセスまたはスレッドによって増やされる統計です。待機イベント・データは、ラッチの競合、バッファの競合、I/Oの競合などのパフォーマンスに影響を与えると思われる様々な問題の症状を表します。ただし、これらの問題は実際の原因でなく、問題の症状にすぎないこ

とに注意してください。

待機イベントは、クラス別にグループ化されています。待機イベント・クラスには、Administrative、Application、Cluster、Commit、Concurrency、Configuration、Idle、Network、Other、Scheduler、System I/O、およびUser I/Oがあります。

サーバー・プロセスには、次のような待機があります。

- バッファやラッチなどのリソースが使用可能になるのを待機します。
- I/Oなどのアクションが完了するのを待ちます。
- 実行する追加作業(クライアントが次に実行するSQL文を提供するまで待機する場合など)。サーバー・プロセスが追加作業の待機中であることを識別するイベントのことをアイドル・イベントと呼びます。

待機イベント統計には、イベントを待機した回数や、イベントが完了するまでの待機時間があります。TIMED_STATISTICS初期化パラメータをtrueに設定すると、各リソースを待機した時間も表示されます。

ユーザー・レスポンス時間をできるだけ少なくするには、イベントが完了するまでサーバー・プロセスが待機する時間を減らします。すべての待機イベントが同じ待機時間を持っているとはかぎりません。したがって、発生回数の多い待機イベントより、最大の合計時間を持つイベントを調べるほうが重要です。通常は、少なくともパフォーマンスの監視中にTIMED_STATISTICS動的パラメータをtrueに設定することが最善です。

関連項目:

- 待機イベント統計
- 時間統計のある待機イベントの使用
- Oracle Database待機イベントの詳細は、『Oracle Databaseリファレンス』を参照してください

待機イベント統計を含む動的パフォーマンス・ビュー

待機イベント統計について、これらの動的パフォーマンス・ビューの問合せを行うことができます。

- V$ACTIVE_SESSION_HISTORY

  V$ACTIVE_SESSION_HISTORYビューには、1秒ごとにサンプリングされたアクティブなデータベース・セッションのアクティビティが表示されます。

- V$SESS_TIME_MODELおよびV$SYS_TIME_MODEL

  V$SESS_TIME_MODELビューおよびV$SYS_TIME_MODELビューには、データベース・コールの所要時間の合計であるDB timeなど、時間モデル統計が含まれます。

- V$SESSION_WAIT

  V$SESSION_WAITビューには、各セッションの現在または最後の待機に関する情報(待機ID、クラス、時間など)が表示されます。

- V$SESSION

  V$SESSIONビューには、各現行セッションに関する情報が表示されます。また、このビューには、V$SESSION_WAITビューと同じ待機統計が含まれています。該当する場合、このビューには、セッションが現在待機中のオブジェクトの詳細情報

(オブジェクト番号、ブロック番号、ファイル番号、行番号など)、現在の待機の原因となったブロック・セッション(ブロック・セッションID、ステータス、タイプなど)、および待機時間も含まれます。

- V$SESSION_EVENT

  V$SESSION_EVENTビューは、セッションが開始した後に待機したすべてのイベントのサマリーを示します。

- V$SESSION_WAIT_CLASS

  V$SESSION_WAIT_CLASSビューは、待機数および各セッションの待機イベントの各クラスで消費される時間を示します。

- V$SESSION_WAIT_HISTORY

  V$SESSION_WAIT_HISTORYビューには、各アクティブ・セッションの最新10件の待機イベントに関する情報(イベント・タイプや待機時間など)が表示されます。

- V$SYSTEM_EVENT

  V$SYSTEM_EVENTビューは、インスタンス起動後のインスタンスの、全イベント待機のサマリーを示します。

- V$EVENT_HISTOGRAM

  V$EVENT_HISTOGRAMビューには、待機数、最大待機時間およびイベントごとの合計待機時間を示すヒストグラムが表示されます。

- V$FILE_HISTOGRAM

  V$FILE_HISTOGRAMビューには、ファイルごとに1ブロック読取り中の待機回数を示すヒストグラムが表示されます。

- V$SYSTEM_WAIT_CLASS

  V$SYSTEM_WAIT_CLASSビューは、待機数に対するインスタンス全体の総時間および待機イベントの各クラスで消費される時間を示します。

- V$TEMP_HISTOGRAM

  V$TEMP_HISTOGRAMビューには、一時ファイルごとに1ブロック読取り中の待機回数を示すヒストグラムが表示されます。

パフォーマンス・チューニングを実行するときに、待機イベントと関連するタイミング・データを調査します。最大時間がリストされるイベントは、多くの場合、パフォーマンス・ボトルネックを顕著に示しています。たとえば、V$SYSTEM_EVENTを参照することで、多くのbuffer busy waitsが発生していると気づくことがあります。おそらく、多数のプロセスが同じブロックに挿入しようとするときに、各プロセスが他のプロセスの挿入を待機してからでないと挿入できないことが原因です。問題となっているオブジェクトに自動セグメント領域管理またはパーティション化を使用することで解決する可能性があります。

関連項目:

- V$SESSION_WAIT、V$SESSION_EVENTおよびV$SYSTEM_EVENTビューの違いについては、「待機イベント統計」を参照してください
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

システム統計

システム統計は通常、パフォーマンスの問題の原因をさらに示すものを見つけるために、待機イベント・データとともに使用されます。

たとえば、最大の待機イベント(待機時間の点で)がbuffer busy waitsイベントであることをV$SYSTEM_EVENTが示している場

合、V$WAITSTATビューで使用できる特定のバッファ待機統計を調べて、どのブロック・タイプが最大の待機カウントと最大の待機時間を持っているかを識別します。

ブロック・タイプを識別した後、問題の発生中にV$SESSIONをリアルタイムで調べるか、問題の発生後にV$ACTIVE_SESSION_HISTORYビューおよびDBA_HIST_ACTIVE_SESS_HISTORYビューを調べ、表示されたオブジェクト番号を使用して競合対象のオブジェクトを識別します。このデータの組合せは、適切な修正アクションを示しています。

統計は、多数のV$ビューで使用できます。システム統計を含むV$ビューをいくつか次にあげます。

V$ACTIVE_SESSION_HISTORY

このビューには、1秒ごとにサンプリングされたアクティブなデータベース・セッションのアクティビティが表示されます。

V$SYSSTAT

ロールバック、論理I/O、物理I/O、解析データをはじめとするOracle Databaseの様々な部分の全般的な統計が含まれます。バッファ・キャッシュ・ヒット率などの比率を計算するには、V$SYSSTATからのデータを使用します。

V$FILESTAT

これには、ファイル当たりのI/O回数や平均読取り時間など、ファイルごとの詳細なファイルI/O統計が含まれています。

V$ROLLSTAT

これには、詳細なロールバック・セグメントおよびUNDOセグメント統計が含まれています。

V$ENQUEUE_STAT

これには、エンキューが要求された回数やエンキューを待機した回数、待機時間など、各エンキューの詳細なエンキュー統計が含まれています。

V$LATCH

これには、各ラッチが要求された回数やラッチを待機した回数など、各ラッチの詳細なラッチ使用統計が含まれています。

関連項目:

動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

セグメント・レベルの統計

個別のセグメントに関連するパフォーマンス問題に焦点をあてるのに役立つ、セグメント・レベルの統計を収集できます。セグメント・レベルの統計を収集して表示することは、インスタンスで競合度の高い表あるいは索引を効果的に識別するための優れた方法です。

パフォーマンスの問題を識別するために待機イベントおよびシステム統計を表示した後で、セグメント・レベルの統計を使用して問題の原因となっている特定の表または索引を検索できます。バッファ・ビジー待機が、大半の待機時間の原因になっていることをV$SYSTEM_EVENTが示している例を考えます。バッファ・ビジー待機の原因になっているトップ・セグメントを

V$SEGMENT_STATISTICSから選択できます。これにより、それらのセグメントの問題の解決に集中できます。

セグメント・レベルの統計は、次の動的ビューを使用して問い合せます。

- V$SEGSTAT_NAME: : このビューには収集するセグメント統計と、各種統計(たとえばサンプル統計など)のプロパティがリストされます。
- V$SEGSTAT: これは非常に効率的で、リアルタイム監視が可能なビューであり、統計値、統計名およびその他の基本情報が表示されます。
- V$SEGMENT_STATISTICS: ユーザーが扱いやすい統計値のビューです。V$SEGSTATのすべての列の他、ここにはセグメント所有者や表領域名などの情報があります。統計の理解は容易になりますが、コストがより高くなります。

関連項目:

動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

変更の実装および測定

チューニングを実施した後、多くの場合、問題を軽減できると思われる2つまたは3つの変更を識別できます。どの変更が最高の利益を提供するかを識別するには、一度に1回の変更を実装することをお薦めします。変更の効果は、問題定義段階でみられたベースライン・データ測定と対照して測定する必要があります。

一般に、パフォーマンスの問題を持つ大半のサイトでは、一度に重複した変更を実装するので、どの変更が利益を実現したかを識別できません。これはすぐに問題になることはありませんが、どの変更が最も効果をあげ、どのような作業を優先する必要があるかを知ることは不可能なので、今後同様の問題が発生した場合に大きな障害になります。

個別に変更を実装できない場合は、異なる変更の効果の測定を試みてください。たとえば、変更された問合せのパフォーマンスを向上するために新しい索引を作成する効果とは別に、REDOの生成を最適化するために初期化変更を行う効果を測定します。SQLがチューニングされ、オペレーティング・システムのディスク・レイアウトが変更され、初期化パラメータも同時に変更されている場合は、オペレーティング・システムをアップグレードすることの利益は測定できません。

パフォーマンス・チューニングは反復操作です。インスタンス全体のパフォーマンスの問題を解決する特効薬的な対策が見つかることはほとんどありません。ほとんどの場合、あるボトルネックを解決しても別の(ときにはさらに悪い)問題が発生するため、優れたパフォーマンスにはパフォーマンス・チューニング段階を反復する必要があります。

いつチューニングを停止するかを知ることも重要です。パフォーマンスの最も優れた測定は、統計が理想的な値にどの程度近いかではなく、ユーザーの理解力です。

Oracle Database統計の解釈

インスタンスにパフォーマンスの問題があった時間範囲の統計を収集します。比較のためのベースライン・データをすでに収集してある場合は、問題のワークロードを最も代表するベースラインからのデータと、現行のデータを比較できます。

2つのレポートを比較する場合、それらのレポートが、システムを比較できるようなワークロードか確認してください。

負荷の検査

待機イベントは通常、最初に検査するデータです。ただし、ベースライン・レポートがある場合は、負荷が変化したかどうかをチェックします。ベースラインがあるかどうかにかかわらず、リソースの使用率が高いかどうかを確認すると便利です。

検査する負荷に関連する統計には、redo size、session logical reads、db block changes、physical reads、physical read total bytes、physical writes、physical write total bytes、parse count (total)、parse count (hard)およびuser callsがあります。このデータは、V$SYSSTATから問合せが行われます。秒ごとおよびトランザクションごとに、このデータを正規化することが最も有効です。また、physical read total bytesおよびphysical write total bytesの合計を使用して、1秒当たりの合計I/O負荷(MB)を調べることも有益です。合計値には、Recovery Manager(RMAN)バックアップおよびリカバリとOracle Databaseバックグラウンド・プロセスによる、バッファ・キャッシュ、REDOログ、アーカイブ・ログでのI/Oが含まれます。

AWRレポートの「ロード・プロファイル」セクションを参照してください。データは、トランザクションおよび秒ごとに正規化されています。

負荷の変更

秒ごとの負荷プロファイル統計は、スループットの変化(すなわち、インスタンスの作業実行量が毎秒ごとに増えているかどうか)を示します。トランザクションごとの統計は、アプリケーション特性の変化をベースライン・レポートからの対応する統計と比較することで識別します。

高いアクティビティ率

アクティビティ率が非常に高いかどうかを識別するには、秒ごとに正規化した統計を調べます。サイトごとにしきい値が異なり、アプリケーションの特性、CPUの個数と速度、オペレーティング・システム、I/OシステムおよびOracle Databaseのリリースに左右されるため、高い値に関する包括的な推奨事項を示すのは困難です。

次に、いくつかの一般化された例を示します(許容値は各サイトで異なります)。

- 秒当たり100を超えるハード解析率は、非常に大量なハード解析がシステム上にあることを示します。高いハード解析率は重大なパフォーマンスの問題を発生させるので、調査する必要があります。通常は、高いハード解析率に共有プール上のラッチの競合とライブラリ・キャッシュ・ラッチが伴います。
- ライブラリ・キャッシュおよび共有プール・ラッチ・イベント(latch: library cache、latch: library cache pin、latch: library cache lockおよびlatch: shared pool)の待機時間の合計が、V$SYSSTATに表示される統計のDB timeに比べて大きいかどうかを調べます。大きい場合は、AWRレポートのSQL ordered by Parse Callsセクションを調べます。
- 高いソフト解析率は、秒当たり300以上の率になる可能性があります。不必要なソフト解析もアプリケーションのスケーラビリティを制限します。最適な方法として、SQL文をセッション当たり1回ソフト解析し、何回も実行します。

待機イベント統計を使用したボトルネックへのドリルダウン

Oracleプロセスは待機状態になると、一連の事前定義済待機イベントのいずれかを使用して待機状態を記録します。これらの待機イベントは、待機クラス別にグループ化されます。Idle待機クラスには、実行する作業がなく、さらに作業が実行されるのを待っている場合にプロセスが待機するイベントすべてがグループ化されます。アイドル状態でないイベントはリソースあるいはアクションが完了するまでの非生産的な待機時間を示します。

ノート:

すべての症状が待機イベントによって証明されるわけではありません。チェックできる統計は、「追加された統計情報」を参照してください。

待機イベント・データを使用する最も効率的な方法は、待機時間別にイベントを順序付けすることです。この方法は、TIMED_STATISTICSがtrueに設定されているときのみ可能です。設定しない場合は、待機イベントを待機数別に順位付けします。これは、一般的に問題を最もよく表す順序付けではありません。

どこで時間が消費されているかが判明してから、次のステップを実行してください。

1. V$SYSTEM_EVENTのデータ収集を調べます。対象のイベントは、待機時間別に順位付けする必要があります。

   待機時間の最も大きいパーセンテージを持つ待機イベントを識別します。待機時間のパーセンテージを決定するには、アイドル・イベント(Null event、SQL*Net message from client、SQL*Net message to client、SQL*Net more data to clientなど)を除くすべての待機イベントの合計待機時間を加算します。各イベントの待機時間をすべてのイベントの総待機時間で除算し、5つの最も重要なイベントの相対的なパーセンテージを計算します。

   別の方法として、自動ワークロード・リポジトリ・レポートの先頭の「トップ5待機イベント」の項を参照してください。この項では、待機イベント(アイドル・イベントを除く)を自動的に順序付けし、相対的なパーセンテージを計算します。

```
Top 5 Timed Events
~~~~~~~~~~~~~~~~~~
                                                     % Total
Event                                 Waits    Time (s) Call Time
------------------------------------- ------------ ----------- ---------
CPU time                                             559     88.80
log file parallel write                 2,181         28      4.42
SQL*Net more data from client         516,611         27      4.24
db file parallel write                 13,383         13      2.04
db file sequential read                   563          2       .27
```

   状況によっては、同程度のパーセンテージを持つイベントがいくつか存在する場合があります。このため、すべてのイベントが同じタイプのリソース・リクエスト(たとえば、すべてがI/O関連イベント)に関連している場合に、追加の証拠を提供できます。

2. これらのイベントの待機数と平均待機時間を見てください。たとえば、I/O関連イベントの場合、平均時間がI/Oシステムが低速であるかどうかを識別するのに役立つ場合もあります。次に、AWRレポートの「待機イベント」セクションから引用した、このデータの例を示します。

```
                                                        Avg
                                          Total Wait    wait     Waits
Event                            Waits    Time (s)      (ms)      /txn
-------------------------------- -------- ---------- ------ ----------
log file parallel write             2,181          0         28     13       41.2
SQL*Net more data from clie       516,611          0         27      0   9,747.4
db file parallel write             13,383          0         13      1     252.5
```

3. トップの待機イベントは、次に調査する場所を識別します。表10-1に、一般的な待機イベントを示します。高負荷SQLを調べることもお薦めします。

4. 待機イベントで指示される関連データを調べて、このデータから得られる他の情報を確認します。この情報が待機イベ

ント・データとの一貫性を持っているかどうかを判断します。ほとんどの場合、パフォーマンス・ボトルネックの潜在的な原因に関する理論の展開を開始するためのデータは十分にあります。

5. この理論が有効かどうかを確認するには、使用可能な他の統計で調べたデータをクロスチェックして一貫性のあることを確認します。適切な統計は問題により異なりますが、通常はV$SYSSTATやオペレーティング・システム統計などにある、ロード・プロファイル関連のデータが含まれています。他のデータとのクロスチェックを行って、展開中の理論を肯定または否定します。

関連項目:

- アイドル待機イベントのリストは、「アイドル待機イベント」を参照してください
- 待機イベントの詳細は、『Oracle Databaseリファレンス』を参照してください

待機イベントおよび潜在的な原因の表

表10-1に、待機イベントと考えられる原因との関連付けの他、次に検討するのに最も有益と思われるOracleデータの概要を示します。

表10-1 待機イベントおよび潜在的な原因

待機イベント	一般的な領域	考えられる原因	検索/調査
buffer busy waits	バッファ・キャッシュ、DBWR	バッファ・タイプによって異なります。たとえば、索引ブロックの待機は、昇順に基づく主キーが原因である場合があります。	問題が発生している間に V$SESSION を調べ、競合したブロックのタイプを判別します。
free buffer waits	バッファ・キャッシュ、DBWR、I/O	低速な DBWR(おそらく I/O に起因) 小さすぎるキャッシュ	オペレーティング・システム統計を使用して書込み時間を調べます。キャッシュが小さすぎることの証拠があるかどうかについてバッファ・キャッシュ統計をチェックします。
db file scattered read	I/O、SQL 文のチューニング	チューニングが適切ではない SQL 低速な I/O システム	V$SQLAREA を調べて、多数のディスク読取りを実行する SQL 文があるかどうかを確認します。I/O システムと V$FILESTAT をクロスチェックして、読取り時間に問題がないかをチェックします。
db file 順次読取り	I/O、SQL 文のチューニング	チューニングが適切ではない SQL 低速な I/O システム	V$SQLAREA を調べて、多数のディスク読取りを実行する SQL 文があるかどうかを確認します。I/O システムと V$FILESTAT をクロスチェックして、読取り時間に問題がないかをチェックします。

待機イベント	一般的な領域	考えられる原因	検索/調査
enqueue 待機(enq:で始まる待機)	ロック	エンキューのタイプにより異なる	V$ENQUEUE_STAT を参照します。
ライブラリ・キャッシュ・ラッチ待機: library cache、library cache pin および library cache lock	ラッチの競合	SQL の解析または共有	V$SQLAREA を調べて、比較的多数の解析コールまたは多数の子カーソルを使用する SQL 文があるかどうかを確認します(VERSION_COUNT 列)。V$SYSSTAT の解析統計と毎秒の対応する割合を調べます。
log buffer space	ログ・バッファの I/O	小さいログ・バッファ 低速な I/O システム	V$SYSSTAT の統計 redo buffer allocation retries をチェックします。メモリーの構成の章の、ログ・バッファの構成の項をチェックしてください。オンライン REDO ログを格納するディスクをチェックして、リソースの競合の有無をチェックします。
log file sync	I/O、コミット過剰	オンライン・ログを格納する低速なディスク バッチされないコミット	オンライン REDO ログを格納するディスクをチェックして、リソースの競合の有無をチェックします。V$SYSSTAT から毎秒のトランザクション数(コミット数+ロールバック数)をチェックします。

関連項目:

- 表10-1に示した各イベントの詳細およびクロスチェックするその他の情報は、「待機イベント統計」を参照してください
- 動的パフォーマンス・ビューの詳細は、『Oracle Databaseリファレンス』を参照してください。
- buffer busy waits (34405.1)およびfree buffer waits (62172.1)に関するMy Oracle Supportの通知。これらの通知および関連通知には、My Oracle SupportのWebサイトで「busy buffer waits」と「free buffer waits」を検索してアクセスすることも可能です。

追加された統計情報

対応する待機イベントのない一部の統計にも、パフォーマンスの問題を示す情報が含まれる場合があります。

REDOログ領域リクエスト統計

V$SYSSTAT統計のredo log space requestsは、サーバー・プロセスが、REDOログ・バッファの領域ではなく、オンラインREDOログの領域を待機した回数を示します。この統計および待機イベントは、LGWRではなく、チェックポイント、DBWRまたはアーカイバ・アクティビティのチューニングが必要であることを示しています。ログ・バッファの容量を増加しても役に立ちません。

読取り一貫性

システムは、一貫したビューを維持するために、ブロックの変更内容のロールバックに長時間を費やすことがあります。次のシナリオ

を参考にしてください。

- 多数の小さいトランザクションがあり、変化が起こっている同じ表のバックグラウンド内でアクティブな長時間実行問合せが動作している場合、表の一貫読取りイメージを取得するために、問合せはこれらの変化を頻繁にロールバックする必要がある場合があります。次のV$SYSSTAT統計を比較して、変化が発生しているかどうかを判断します。
  - consistentchanges: 統計は、ブロックの読取り一貫性を実施するために、データベース・ブロックにロールバック・エントリが適用される回数を示します。多数のconsistent changesを生成するワークロードは、多数のリソースを消費する可能性があります。
  - consistent gets: 統計は、一貫性モードでの論理読取り数をカウントします。
- 大きいロールバック・セグメントがほとんどない場合、システムはどのシステム変更番号(SCN)のトランザクションがコミットされたかを正確に知るために、遅延ブロックのクリーンアウト時に長時間かけてトランザクション表をロールバックする可能性があります。Oracle Databaseでトランザクションをコミットしたとき、変更されたブロックすべてがコミットSCNで即時に更新されるとはかぎりません。この場合は、ブロックの読取り時または更新時に必要に応じて更新されます。これを遅延ブロック・クリーンアウトと呼びます。

  次のV$SYSSTAT統計の比率は、1に近い値であることが必要です。

```
ratio = transaction tables consistent reads - undo records applied /
        transaction tables consistent read rollbacks
```

  解決策は、自動UNDO管理を使用することをお薦めします。

- ロールバック・セグメントが十分にない場合、ロールバック・セグメント(ヘッダーまたはブロック)の競合が発生します。この問題は、次のようにすると明らかになります。
  - WAITS数をV$ROLLSTAT内のGETS数と比較する方法。GETSに対するWAITSの比率は小さい値である必要があります。
  - V$WAITSTATを調べて、CLASS 'undo header'のバッファに対して多数のWAITSがあるかどうかを確認する方法。

  解決策は、自動UNDO管理を使用することをお薦めします。

継続行による表フェッチ

V$SYSSTAT内のtable fetch continued row統計数をチェックして、移行行または連鎖行を検出できます。少数の連鎖行(1%以下)は、システム・パフォーマンスに影響を与える可能性はほとんどありません。ただし、連鎖行のパーセンテージが大きいと、パフォーマンスに影響を与える可能性があります。

ブロック・サイズより大きい行の連鎖は避けられません。そのようなデータについては、ブロック・サイズのより大きい表領域の使用を考慮してください。

ただし、小さい行の場合は、適切な領域パラメータとアプリケーション設計を使用することで連鎖を回避できます。たとえば、キー値が入力され、かつその他のほとんどの列がNULLである行を挿入した後に、実際のデータで更新しないでください。その場合は初めからデータが入力された行を挿入します。

UPDATE文によって行のデータ量が増加し、行がそのデータ・ブロックに収まらなくなった場合、Oracle Databaseは行全体を保持するのに十分な空き領域を持つ別のブロックを見つけようとします。そのようなブロックが利用可能であれば、Oracle Databaseは新しいブロックへ行全体を移動します。この操作は、行の移行と呼ばれています。行が大きすぎて利用可能なブロックに収まらない場合、データベースはその行を複数の断片に分割し、各断片を別々のブロックに格納します。この操作は、行

の連鎖と呼ばれています。データベースでは、行の挿入時にも行連鎖が発生する可能性があります。

移行と連鎖は、特に次の場合のパフォーマンスに影響があります。

- 移行と連鎖の原因となるUPDATE文のパフォーマンスはよくありません。
- 移行行または連鎖行が追加入出力を実行するため、これらの行を選択する問合せをします。

サンプルの出力表CHAINED_ROWSの定義が、配布媒体上の使用可能なSQLスクリプトに収録されています。このスクリプトの一般的な名前はUTLCHN1.SQLですが、正確な名前と位置は使用しているプラットフォームによって異なります。出力表の列名、データ型およびサイズは、CHAINED_ROWS表と同じである必要があります。

移行行を回避するには、PCTFREEを増やします。ブロック内に使用可能な空き領域を多く残しておくと、行の拡張に対処できます。削除割合が高い表と索引を再編成すなわち再作成することもできます。頻繁に行が削除される表の場合は、データ・ブロックに部分的に空き領域が生じることがあります。行を挿入し後から拡張する場合、行の削除されたブロックにその行が挿入されることがありますが、拡張の余地はありません。表を再編成すると主な空き領域を完全に空のブロックにできます。

ノート:

PCTUSED は、PCTFREE の反対の意味ではありません。

関連項目:

- PCTUSEDの詳細は、『Oracle Database概要』を参照してください
- 表の再編成方法を学習するには、『Oracle Database管理者ガイド』を参照してください。

解析関連の統計

アプリケーションの解析が長くなるほど、競合の可能性が高くなり、システムの待機時間が長くなります。parse time CPU(CPU解析時間)がCPU時間の大半を占める場合、文の実行ではなく解析に時間が消費されています。この場合には、アプリケーションはリテラルSQLを使用しているためにSQLを共有できないか、または共有プールの構成が適切でないことがあります。

統計により、Oracleが解析に費やした時間の長さを識別することができます。V$SYSSTATから解析関連の統計の問合せを行います。次に例を示します。

```
SELECT NAME, VALUE
  FROM V$SYSSTAT
 WHERE NAME IN (  'parse time cpu', 'parse time elapsed',
                  'parse count (hard)', 'CPU used by this session' );
```

様々な比率を計算することで、解析に問題があるかどうかを判断できます。

- parse time CPU / parse time elapsed

  この比率は、解析に費やされる時間のうちのどのくらいが、ラッチなどのリソースに対する待機ではなく、解析操作自体によるものかを示します。比率1は良好であり、経過時間が競合率の高いリソースを待機するために費やされなかったことを示します。

- parse time CPU / CPU used by this session

  この比率は、Oracleサーバー・プロセスで使用されるCPU全体のうちのどのくらいが解析関係の操作で費やされたかを示します。0(ゼロ)に近い値ほど良好であり、CPUの大部分が解析に費やされないことを示します。

待機イベント統計

V$SESSION、V$SESSION_WAIT、V$SESSION_HISTORY、V$SESSION_EVENTおよびV$SYSTEM_EVENTの各ビューは、どのようなリソースを待機したかに関する情報を表示し、構成パラメータTIMED_STATISTICSがtrueに設定されている場合は、各リソースを待機した時間に関する情報も表示されます。

関連項目:

- STATISTICS_LEVELの設定の詳細は、統計収集のレベルの設定を参照してください
- 待機イベント統計を含むV$ビューの詳細は、Oracle Databaseリファレンスを参照してください

パフォーマンス・チューニングを実行するときに、待機イベントと関連するタイミング・データを調査します。最大時間がリストされるイベントは、多くの場合、パフォーマンス・ボトルネックを顕著に示しています。

次の各ビューには、同じデータの関連する(ただし、異なる)ビューが含まれています。

- V$SESSIONは、各現行セッションのセッション情報をリストします。このビューには、現在待機されているイベントか、各セッションで最後に待機されたイベントがリストされます。また、このビューには、ブロック・セッション、待機状態および待機時間に関する情報も含まれます。
- V$SESSION_WAITは、現在の状態ビューです。このビューには、現在待機されているイベントか、各セッションで最後に待機されたイベント、待機状態および待機時間がリストされます。
- V$SESSION_WAIT_HISTORYは、各現行セッションの最新10件の待機イベントと、関連する待機時間をリストします。
- V$SESSION_EVENTは、各セッションで待機されるイベントの累積履歴をリストします。セッションが終了した後、そのセッションに対する待機イベント統計はこのビューから削除されます。
- V$SYSTEM_EVENTは、インスタンスの起動以降にインスタンス全体により待機されるイベントと回数(すなわち、ロールアップされた、すべてのセッション待機イベント・データ)をリストします。

V$SESSION_WAITは現在の状態ビューであるため、V$SESSION_EVENTまたはV$SYSTEM_EVENTよりも細かい単位の情報も含まれています。このビューには、P1、P2およびP3の3つのパラメータ列があり、現行イベントの追加識別データが含まれます。

たとえば、V$SESSION_EVENTはセッション124(SID=124)がdb file scattered readで多く待機していたことを示すことはできますが、どのファイルとブロック番号かは示しません。ただし、V$SESSION_WAITはP1内のファイル番号、P2内に読み取られたブロック番号およびP3内に読み取られたブロック数を示します(P1とP2により待機イベントがどのセグメントに対して発生するかを判断できます)。

この項では、V$SESSION_WAITの使用例を中心に説明します。ただし、時間間隔のパフォーマンス・データの収集と、パフォーマンスおよび容量分析のためにこのデータを保存することをお薦めします。この形式のロールアップ・データの問合せは、AWRによりV$SYSTEM_EVENTビューで行います。

最も一般的に発生するイベントについては、この章で、大/小文字を区別するアルファベット順にリストして説明します。調べる対

象の他のイベント関連データも含まれています。各イベント名に使用する大/小文字区別は、V$SYSTEM_EVENTビューでの表示と同一です。

過去のリリースからの待機イベント統計の変更

Oracle Database 11g以降、Oracle Databaseでは待機イベントの待機カウントおよびタイムアウトの累計が過去のリリースとは異なります(V$SYSTEM_EVENTビューなど)。特定のタイプのリソース(エンキューなど)に対する継続的な待機は、内部で短時間の待機コールのセットに分割されます。Oracle Database 11gより前のリリースでは、個々の内部の待機コールが、別々の待機としてカウントされていました。Oracle Database 11gから、待機中にセッションで発生した内部タイムアウトの回数にかかわらず、1つのリソースに対する待機は、1つの待機として記録されます。

Oracle Databaseでは、この変更により、待機カウントをより忠実に表現し、リソースの合計待機時間を正確に表示できます。タイムアウトは、個々の内部の待機コールではなく、リソースの待機を参照するようになりました。この変更は、平均待機時間および最大待機時間にも影響を与えます。たとえば、トランザクション行をロックして表の1行を更新するために、ユーザー・セッションがエンキューを待機する必要があり、そのエンキューを取得するのに10秒かかった場合、Oracle Databaseでは、エンキューの待機を3秒間の待機コールに分割します。この例では、3秒間の待機コールが3回と、1秒間の待機コールが1回です。ただし、セッションの観点からは、エンキューの待機は1回のみです。

Oracle Database 11gより前のリリースでは、この待機シナリオは、V$SYSTEM_EVENTビューで次のように表現されます。

- TOTAL_WAITS: 待機回数4(3秒間の待機が3回、1秒間の待機が1回)
- TOTAL_TIMEOUTS: タイムアウト回数3(最初の3回の待機はタイムアウトになり、最後の待機中にエンキューを取得)
- TIME_WAITED: 10秒(4回の待機時間の合計)
- AVERAGE_WAIT: 2.5秒
- MAX_WAIT: 3秒

Oracle Database 11gから、この待機シナリオは次のように表現されます。

- TOTAL_WAITS: 待機回数1(10秒間の待機が1回)
- TOTAL_TIMEOUTS: タイムアウト回数0(リソースの待機中にエンキューを取得)
- TIME_WAITED: 10秒(リソースを待機した時間)
- AVERAGE_WAIT: 10秒
- MAX_WAIT: 10秒

この変更の影響を受ける一般的な待機イベントは次のとおりです。

- エンキュー待機(enq: name - reason待機など)
- ライブラリ・キャッシュ・ロック待機
- ライブラリ・キャッシュ・ピン待機
- 行キャッシュ・ロック待機

この変更の影響を受ける統計情報は次のとおりです。

- 待機回数

- 待機のタイムアウト回数
- 平均待機時間
- 最大待機時間

この変更の影響を受けるビューは次のとおりです。

- V$EVENT_HISTOGRAM
- V$EVENTMETRIC
- V$SERVICE_EVENT
- V$SERVICE_WAIT_CLASS
- V$SESSION_EVENT
- V$SESSION_WAIT
- V$SESSION_WAIT_CLASS
- V$SESSION_WAIT_HISTORY
- V$SYSTEM_EVENT
- V$SYSTEM_WAIT_CLASS
- V$WAITCLASSMETRIC
- V$WAITCLASSMETRIC_HISTORY

関連項目:

V$SYSTEM_EVENTビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

buffer busy waits

この待機は、複数のプロセスがバッファ・キャッシュ内のいくつかのバッファに同時にアクセスしようとしていることを示します。バッファのクラスごとに、待機統計についてV$WAITSTATを問い合せます。バッファ・ビジー待機を持つ一般的なバッファ・クラスには、data block、segment header、undo headerおよびundo blockがあります。

次のV$SESSION_WAITパラメータ列をチェックします。

- P1: ファイルID
- P2: ブロックID
- P3: クラスID

原因

考えられる原因を判別するには、最初にV$SESSIONを問い合せて、セッションがbuffer busy waitsを待機しているときのROW_WAIT_OBJ#の値を識別します。次に例を示します。

```
SELECT row_wait_obj#
  FROM V$SESSION
 WHERE EVENT = 'buffer busy waits';
```

競合対象のオブジェクトとオブジェクト型を識別するには、V$SESSIONから戻されるROW_WAIT_OBJ#の値を使用して

DBA_OBJECTSを問い合せます。次に例を示します。

```
SELECT owner, object_name, subobject_name, object_type
  FROM DBA_OBJECTS
 WHERE data_object_id = &row_wait_obj;
```

処置

必要な処置は、競合対象のクラスと実際のセグメントにより異なります。

セグメント・ヘッダー

競合がセグメント・ヘッダー上にある場合、これは最も起こりうる空きリストの競合です。

ローカルに管理されている表領域で自動セグメント領域管理を行えば、PCTUSED、FREELISTSおよびFREELIST GROUPSの各パラメータを指定する必要はありません。可能であれば、手動領域管理から自動セグメント領域管理(ASSM)に切り替えます。

ASSMを使用できない場合(たとえば、表領域でディクショナリ領域管理を使用しているため)、これに関連する情報は次のとおりです。

空きリストは、通常セグメントの様々なエクステント内に存在するブロックを含む空きデータ・ブロックのリストです。空きリストは、空き領域がPCTFREEに達していないブロック、または使用済領域がPCTUSEDを下回っているブロックで構成されます。FREELISTSパラメータでプロセスの空きリスト数を指定します。FREELISTSのデフォルト値は1です。最大値はデータ・ブロック・サイズによって決まります。

そのセグメントの空きリストに対する現在の設定を見つけるには、次を実行します。

```
SELECT SEGMENT_NAME, FREELISTS
  FROM DBA_SEGMENTS
 WHERE SEGMENT_NAME = segment name
   AND SEGMENT_TYPE = segment type;
```

空きリスト、または空きリスト数の増分を設定します。さらに空きリストを追加しても問題が軽減されない場合は、空きリスト・グループを使用します(このようにすると、単一のインスタンス内でも違いが出る可能性があります)。Oracle RACを使用する場合、各インスタンスに独自の空きリスト・グループがあることを確認してください。

関連項目:

自動セグメント領域管理、空きリスト、PCTFREEおよびPCTUSEDの詳細は、『Oracle Database概要』を参照してください。

データ・ブロック

表または索引(セグメント・ヘッダーではない)に対する競合がある場合、次のようにします。

- 昇順インデックスを調べます。これは、多数のプロセスによって同じ点に挿入される索引です。たとえば、キー値に順序番号ジェネレータを使用する索引をチェックしてください。
- ASSMの使用またはグローバル・ハッシュ・パーティション索引の使用を検討します。また、複数のプロセスによる同一ブロックへの挿入を回避するために空きリストの増加も検討してください。

UNDOヘッダー

ロールバック・セグメント・ヘッダーに対する競合の場合

- 自動UNDO管理を使用しない場合は、さらにロールバック・セグメントを追加してください。

UNDOブロック

ロールバック・セグメント・ブロックに対する競合の場合

- 自動UNDO管理を使用しない場合は、ロールバック・セグメント・サイズを大きくすることを検討してください。

db file scattered read

このイベントは、ユーザー・プロセスがSGAバッファ・キャッシュにバッファを読み取り、物理I/Oコールが戻るまで待機することを意味します。db file scattered readは、データを複数の不連続メモリー位置に読み取るために散布読取りを発行します。散布読取りは通常、マルチブロック読取りです。全体スキャンの他、(索引の)高速全スキャンでも行うことができます。

db file scattered read待機イベントは、全体スキャンが発生していることを識別します。バッファ・キャッシュへの全体スキャンを実行すると、読み取られたブロックは物理的に相互に接近していないメモリー位置に読み取られます。このような読取りが散布読取りコールと呼ばれるのは、ブロックがメモリー全体に分散されているからです。対応する待機イベントが「db file scattered read」と呼ばれるのは、このためです。バッファ・キャッシュへの全体スキャンのためのマルチブロック(最大でDB_FILE_MULTIBLOCK_READ_COUNTブロック)読取りは、db file scattered readに対する待機として現れます。

次のV$SESSION_WAITパラメータ列をチェックします。

- P1: 絶対ファイル番号
- P2: 読み取られるブロック
- P3: ブロック数(1より大きい値が必要)

処置

正常なシステムで、物理読取り待機はアイドル待機後の最大の待機になります。ただし、小さい索引付きアクセスを行う必要のある運用(OLTP)システム上に、直接読取り待機(パラレル問合せによる全表スキャンを意味する)またはdb file scattered read待機があるかどうかも確認してください。

システム上の過剰なI/O負荷を示す他の要素として、次のものがあります。

- 低いバッファ・キャッシュ・ヒット率
- ユーザーへのレスポンス時間を長くしている待機時間のほとんどを発生させている待機イベント

過剰なI/Oの管理

過剰なI/O待機には、様々な対処方法があります。有効性の順に示すと、これらの方法は次のとおりです。

- SQLチューニングによるI/Oアクティビティの削減。
- ワークロードの管理による、I/Oを実行する必要性の削減。
- DBMS_STATSパッケージを使用したシステム統計の収集。問合せオプティマイザによる、全体スキャンを使用する可能性のあるアクセス・パスのコストの正確な見積りが可能になります。

- 自動ストレージ管理の使用。
- さらに多くのディスクを追加することによる、各ディスクのI/O数の削減。
- 既存のディスク間へのI/Oの再分散によるI/Oホット・スポットの削減。

最初に行うことは、I/Oを削減するためのチャンスを見つけることです。これらのイベントを待機するセッションによって実行されるSQL文と、V$SQLAREAから多数の物理I/Oを実行する文を調査します。過剰なI/Oを実行し、実行計画にマイナスの影響を与える可能性のある要因として、次のものがあります。

- 不適切に最適化されたSQL
- 索引の欠落
- 表の高度な並列度(スキャン方向へオプティマイザを偏らせる)
- オプティマイザの正確な統計がないこと
- DB_FILE_MULTIBLOCK_READ_COUNT初期化パラメータの設定値が全体スキャンには大きすぎること

不十分なI/O分散

I/Oを削減する他、ディスク間のファイルのI/O分散も調べます。I/Oはディスク間に均等に分散されていますか、あるいは、いくつかのディスク上にホット・スポットがありますか。データベースのI/Oリクエストを満たすだけの十分な個数のディスクがありますか。

データベースによるI/O操作(読取りと書込み)の総数を調べ、それと使用したディスク数を比較します。必ず、LGWRプロセスとARCHプロセスのI/Oアクティビティを含めてください。

I/Oを待機しているセッションで実行されたSQL文の検索

次の問合せを使用して、ある時点でどのセッションがI/Oを待機しているかを判断します。

```
SELECT SQL_ADDRESS, SQL_HASH_VALUE
  FROM V$SESSION
 WHERE EVENT LIKE 'db file%read';
```

I/Oを必要とするオブジェクトの検索

考えられる原因を判別するには、最初にV$SESSIONを問い合せて、セッションがdb file scattered readを待機しているときのROW_WAIT_OBJ#の値を識別します。次に例を示します。

```
SELECT row_wait_obj#
  FROM V$SESSION
 WHERE EVENT = 'db file scattered read';
```

競合対象のオブジェクトとオブジェクト型を識別するには、V$SESSIONから戻されるROW_WAIT_OBJ#の値を使用してDBA_OBJECTSを問い合せます。次に例を示します。

```
SELECT owner, object_name, subobject_name, object_type
  FROM DBA_OBJECTS
 WHERE data_object_id = &row_wait_obj;
```

db file順次読取り

このイベントは、ユーザー・プロセスがSGA内のバッファ・キャッシュにバッファを読み取り、物理I/Oコールが戻るまで待機することを

意味します。順次読取りは、単一ブロック読取りです。

単一ブロックI/Oは通常、索引を使用した結果です。非常にまれなケースとして、エクステントの境界のため、またはバッファ・キャッシュ内にバッファが存在するため、全表スキャン・コールが単一ブロック・コールに切り捨てられることがあります。これらの待機もdb file sequential readとして現れます。

次のV$SESSION_WAITパラメータ列をチェックします。

- P1: 絶対ファイル番号
- P2: 読み取られるブロック
- P3: ブロック数(値は1)

関連項目:

過剰I/Oの管理、不適切なI/O分散、さらにI/Oを発生させるSQLおよびI/Oが実行されるセグメントの検索の詳細は、「db file scattered read」を参照してください。

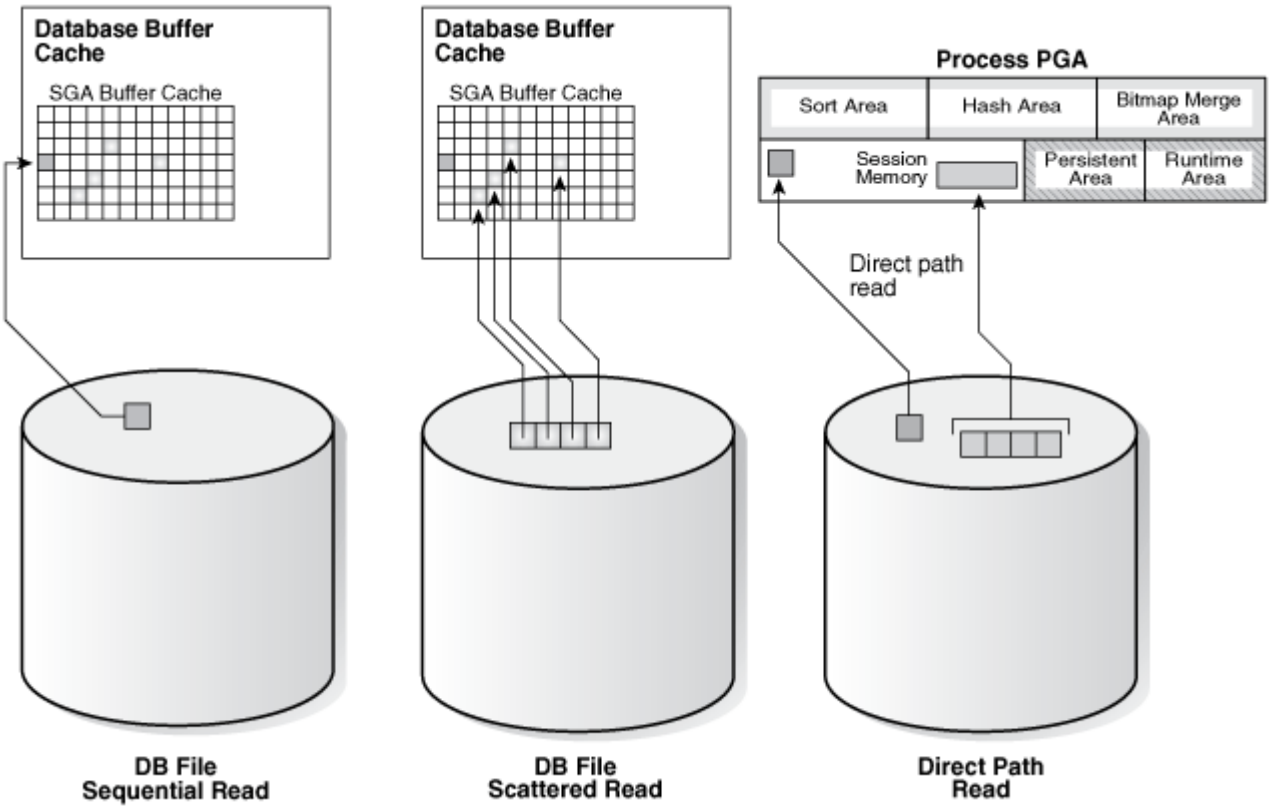
処置

正常なシステムで、物理読取り待機はアイドル待機後の最大の待機になります。ただし、パラレル問合せによる全表スキャンを多く実行する必要のある大規模なデータ・ウェアハウスに対するdb file sequential readsがあるかどうかも確認してください。

次の図は、これらの待機イベントの相違を示しています。

- db file sequential read(1つのSGAバッファに読み取られる単一ブロック)
- db file scattered read(多数の不連続SGAバッファに読み取られるマルチブロック)
- direct read(SGAをバイパスして、PGAに読み取られる単一またはマルチブロック)

図10-1 散布読取り、順次読取りおよびダイレクト・パス読取り



direct path readおよびdirect path read temp

SGAのバッファ・キャッシュではなく、ディスクからPGAに直接バッファの読取りを実行しているセッションは、このイベントで待機します。I/Oサブシステムが非同期I/Oをサポートしない場合、各待機は物理読取りリクエストに対応します。

I/Oサブシステムが非同期I/Oをサポートする場合、このプロセスでは読取りリクエストの発行を、PGAに存在するブロックの処理に重複させることができます。プロセスがディスクからまだ読み取られていないPGA内のブロックにアクセスしようとする場合、待機コールを発行し、このイベントの統計を更新します。したがって、待機数は必ずしも読取りリクエスト数と同じではありません(db file scattered readおよびdb file sequential readとは異なります)。

次のV$SESSION_WAITパラメータ列をチェックします。

- P1: 読取りコールのFile_id
- P2: 読取りコールのStart block_id
- P3: 読取りコールのブロック数

原因

これは次の状況で発生します。

- ソートが大きすぎてメモリーに収まらないため、ソート・データの一部がディスクに直接書き出される。そのデータは、直接読取りで後から再度読み取られます。
- データのスキャンにパラレル実行サーバーが使用される。
- サーバー・プロセスが、I/Oシステムがバッファを戻すより早くバッファを処理する。これは過負荷のI/Oシステムを示します。

処置

file_idにより、読取りがTEMP表領域内のオブジェクトのためのもの(ディスクへのソート)か、パラレル実行サーバーによる全表ス

キャンであるかが示されます。この待機は、大規模なデータ・ウェアハウス・サイトでは最大の待機です。ただし、ワークロードが意思決定支援システム(DSS)ワークロードでない場合は、この状況が発生した理由を調査してください。

ディスクへのソート

待機が発生しているセッションで、現在実行されているSQL文を調べて、ソートの原因を確認します。ソートを生成するSQL文を検索するために、V$TEMPSEG_USAGEを問い合せます。また、ソートのサイズを決定するセッションに関するV$SESSTATからの統計を問い合せます。SQL文をチューニングしてソートを削減できるかどうかを確認します。WORKAREA_SIZE_POLICYがMANUALである場合、システム(ソートが大きすぎない場合)または個々のプロセスのSORT_AREA_SIZEを大きくすることを検討します。WORKAREA_SIZE_POLICYがAUTOである場合、PGA_AGGREGATE_TARGETを大きくするかどうかを調べます。

全表スキャン

高い並列度で表を定義すると、パラレル実行サーバーによる全表スキャンを行うようにオプティマイザが偏る可能性があります。ダイレクト・パス読取りを使用して読み取るオブジェクトをチェックします。全表スキャンが有効なワークロード部分である場合は、I/Oサブシステムが並列度に対して適切かどうかを確認します。ディスクのストライプ化またはOracle Automatic Storage Management(Oracle ASM)を使用していない場合は、ディスクのストライプ化を使用することを検討してください。

ハッシュ領域サイズ

ハッシュ結合を呼び出す問合せ計画の場合、過剰なI/OはHASH_AREA_SIZEが小さすぎることから発生する可能性があります。WORKAREA_SIZE_POLICYがMANUALである場合、システムまたは個々のプロセスのHASH_AREA_SIZEを大きくすることを検討してください。WORKAREA_SIZE_POLICYがAUTOである場合、PGA_AGGREGATE_TARGETを大きくするかどうかを調べます。

関連項目:

- 「db file scattered read」の項の過剰なI/Oの管理を参照してください

direct path writeおよびdirect path write temp

プロセスがバッファをPGAから直接書き込む(バッファ・キャッシュからバッファを書き込むDBWRとは反対)場合、プロセスは書込みコールが完了するまでこのイベント上で待機します。ダイレクト・パス書込みを実行する可能性のある操作には、ディスクでのソート、パラレルDML操作、ダイレクト・パスINSERT、パラレルCREATE TABLE AS SELECT処理、およびいくつかのLOB操作があります。

ダイレクト・パス読取りと同様に、I/Oサブシステムが非同期書込みをサポートする場合、待機数は発行された書込みコール数と同じではありません。セッションがPGA内のすべてのバッファを処理し、I/Oリクエストが完了するまで処理を継続できない場合、セッションは待機します。

関連項目:

ダイレクト・パス・インサートの詳細は、『Oracle Database管理者ガイド』を参照してください。

次のV$SESSION_WAITパラメータ列をチェックします。

- P1: 書込みコールのFile_id
- P2: 書込みコールのStart block_id
- P3: 書込みコール内のブロック数

原因

これは次の状況で発生します。

- ソートが大きすぎ、メモリー内に収まらないため、ディスクに書き込まれる。
- オブジェクトを作成/移入するために、パラレルDMLが発行される。
- ダイレクト・パス・ロード

処置

大規模なソートは、「ディスクへのソート」を参照してください。

パラレルDMLについては、ディスク間のI/O分散をチェックし、I/Oサブシステムが並列度に対して適切に構成されているかどうかを確認してください。

enqueue (enq:)待機

エンキューは、データベース・リソースへのアクセスを調整するロックです。このイベントは、セッションが別のセッションで保持されているロックを待機していることを示します。

エンキュー名は待機イベント名の一部としてenq: enqueue_type - related_details形式で含まれています。次の関連TXタイプなど、同じエンキュー・タイプを異なる目的で保持できる場合があります。

- enq: TX - allocate ITL entry
- enq: TX - contention
- enq: TX - index contention
- enq: TX - row lock contention

V$EVENT_NAMEビューには、すべてのenq:待機イベントのリストが表示されます。

次のV$SESSION_WAITパラメータ列で追加情報を確認できます。

- P1: ロックのTYPE(または名前)およびMODE
- P2: ロックのリソース識別子ID1
- P3: ロックのリソース識別子ID2

関連項目:

Oracle Databaseエンキューの詳細は、『Oracle Databaseリファレンス』を参照してください

ロックおよびロック・ホルダーの検索

ロックを保持するセッションを見つけるには、V$LOCKの問合せを行います。イベント・エンキューを待機するすべてのセッションでは、REQUEST <> 0を持つV$LOCK内に行があります。次の2つの問合せのうちの1つを使用して、ロックを保持しているセッションと

ロックを待機しているセッションを検索します。

エンキュー待機がある場合は、次の文を使用することによりこれらを確認できます。

```
SELECT * FROM V$LOCK WHERE request > 0;
```

待機中のロックのホルダーおよびウェイタのみを表示するには、次の文を使用します。

```
SELECT DECODE(request,0,'Holder: ','Waiter: ') ||
          sid sess, id1, id2, lmode, request, type
   FROM V$LOCK
 WHERE (id1, id2, type) IN (SELECT id1, id2, type FROM V$LOCK WHERE request > 0)
   ORDER BY id1, request;
```

処置

適切な処置は、エンキューのタイプにより異なります。

競合されたエンキューがSTエンキューである場合、問題は動的な領域割当てにある可能性が最も高いと言えます。セグメントにそれ以上空き領域がない場合、Oracle Databaseはセグメントにエクステントを動的に割り当てます。このエンキューは、ディクショナリ管理表領域にのみ使用します。

このリソースに対する競合を解決するには:

- 一時(すなわち、ソート)表領域がTEMPFILESを使用するかどうかを確認します。使用しない場合は、TEMPFILESを使用するように切り替えます。
- 動的に拡張するセグメントを含む表領域がディクショナリ管理表領域の場合は、ローカル管理表領域を使用するように切り替えます。
- ローカル管理表領域に切り替えることができない場合は、拡張するオブジェクトの次のエクステント・サイズを、一定の領域割当てを回避できる十分な大きさに変更することによって、STエンキュー・リソースの使用量を減らすことができます。どのセグメントが常に拡張するかを判断するには、すべてのSEGMENT_NAMEについてDBA_SEGMENTSビューのEXTENTS列を監視します。
- ALTER TABLE ALLOCATE EXTENT SQL文でエクステントを割り当てるなどして、セグメント内の領域を事前に割り当てます。

関連項目:

- TEMPFILESおよびローカル管理表領域の詳細は、『Oracle Database管理者ガイド』を参照してください
- 領域使用量取得の詳細は、『Oracle Database管理者ガイド』を参照してください

HWエンキューは、セグメントの最高水位標を超える領域の割当てをシリアライズする場合に使用します。

- V$SESSION_WAIT.P2 / V$LOCK.ID1は表領域番号です。
- V$SESSION_WAIT.P3 / V$LOCK.ID2は、領域が割り当てられるオブジェクトのセグメント・ヘッダーの相対データ・ブロック・アドレス(DBA)です。

これがオブジェクトの競合点である場合、エクステントの手動割当てにより問題が解決されます。

TMロック上の待機の最も一般的な理由は、制約される列が索引付けされない場合の外部キー制約に関係している傾向があ

ります。この問題を回避するには、外部キー列を索引付けします。

これらのロックは、トランザクションがその最初の変更を開始するときに排他的に取得され、トランザクションがCOMMITまたはROLLBACKを行うまで保持されます。

- モード6のTXの待機は、あるセッションが別のセッションによって保持されている行レベル・ロックを待機しているときに発生します。これは、あるユーザーがある行を更新または削除し、別のセッションがその行を更新または削除する場合に発生します。このタイプのTXエンキュー待機は、待機イベントenq: TX - row lock contentionに対応します。

  解決方法は、ロックを保持している最初のセッションにCOMMITまたはROLLBACKを実行させることです。

- モード4のTXの待機は、セッションがブロック内でITL(必要なトランザクション・リスト)スロットを待機している場合に発生する可能性があります。これは、セッションがそのブロック内の行をロックしても、1つ以上の他のセッションの行が同じブロックでロックされ、そのブロック内に空きITLスロットがない場合に発生します。通常は、Oracle Databaseが別のITLスロットを動的に追加します。この操作は、ITLを追加するためのブロック内の空き領域が十分にない場合には実行できません。空き領域が十分にある場合、セッションはモード4でTXエンキューを持つスロットを待機します。このタイプのTXエンキュー待機は、enq: TX - allocate ITL entryに対応します。

  解決方法は、表のINITRANSまたはMAXTRANSを変更する(ALTER文を使用するか、それより高い値で表を再作成する)ことによって使用可能なITLの個数を増やすことです。

- モード4のTXの待機は、セッションがUNIQUE索引内の潜在的な重複のためにセッションが待機している場合にも発生します。2つのセッションが同じキー値を挿入しようとする場合、第2のセッションはORA-0001が発生したかどうかを確認するまで、待機する必要があります。このタイプのTXエンキュー待機は、待機イベントenq: TX - row lock contentionに対応します。

  解決方法は、ロックを保持している最初のセッションにCOMMITまたはROLLBACKを実行させることです。

- モード4のTXの待機は、共有ビットマップ・インデックス・フラグメントのためにセッションが待機している場合にも発生する可能性があります。ビットマップ索引では、キー値およびROWID範囲が索引付けされます。ビットマップ索引の各エントリには実際の表の行を数多く含めることができます。2つのセッションが同じビットマップ索引フラグメントでカバーされる行を更新する場合、第2のセッションは、モード4でTXロックを待機して、第1のトランザクションのCOMMITまたはROLLBACKを待機します。このタイプのTXエンキュー待機は、待機イベントenq: TX - row lock contentionに対応します。

- モード4でのTXの待機は、PREPAREDトランザクションを待機している場合にも発生する可能性があります。

- モード4のTXの待機は、索引に1行を挿入するトランザクションが、他のトランザクションによる索引ブロック分割の終了を待つ必要がある場合にも発生します。このタイプのTXエンキュー待機は、待機イベントenq: TX - index contentionに対応します。

関連項目:

参照整合性およびデータの明示的ロックの詳細は、『Oracle Database開発ガイド』を参照してください

Other待機クラスのイベント

このイベントは、「その他」の待機クラスに属し、通常はシステムで発生しないようにしてください。このイベントは、latch freeなど、「その他」の待機クラスのその他すべてのイベントの集計であり、V$SESSION_EVENTおよびV$SERVICE_EVENTビューでのみ使用されます。これらのビューでは、「その他」待機クラスのイベントは、セッションごとにはメンテナンスされません。かわりに、この1つ

のイベントにロール・アップされ、「その他」待機クラスのイベント統計をメンテナンスするためのメモリーを削減します。

free buffer waits

この待機イベントは、サーバー・プロセスが空きバッファを検索できず、使用済バッファを書き出すことによって空きバッファを作成するためにデータベース・ライターを転送したことを示します。使用済バッファは、内容が変更されたバッファです。使用済バッファは、DBWRがブロックをディスクに書き込み終えると、再利用するために解放されます。

原因

DBWRは、次の状況では、使用済バッファの書込みに対応できません。

- I/Oシステムが低速である。
- ラッチなど、空くまで待機しているリソースがある。
- バッファ・キャッシュが小さすぎるため、DBWRはサーバー・プロセスのバッファのクリーニングに大部分の時間を費やす。
- バッファ・キャッシュが大きすぎるため、1つのDBWRプロセスでは、リクエストの対応に十分なだけキャッシュ内のバッファを解放できない。

処置

このイベントが頻繁に発生する場合は、DBWRに対するセッション待機を調べて、DBWRを遅らせる原因があるかどうかを確認してください。

セッションが書込みを待機している場合は、書込みを遅らせている原因を解明し、修正してください。次の点をチェックします。

- V$FILESTATを調べて、書込みの大半が発生する場所を確認してください。
- I/Oシステムのホスト・オペレーティング・システム統計を調べてください。書込み時間は許容できるものですか。

I/Oが低速である場合は、次のようにしてください。

- さらに高速なI/O手段を使用して書込み時間を高速化することを検討してください。
- 多数のスピンドル(ディスク)とコントローラの間にI/Oアクティビティを拡散してください。

キャッシュが小さすぎるためにDBWRが非常にアクティブである可能性があります。バッファ・キャッシュ・ヒット率が低いかどうかを確認して、これが考えられる原因であるかどうかを調べます。また、V$DB_CACHE_ADVICEビューを使用して、それより大きいキャッシュ・サイズが有利かどうかを判断します。

キャッシュ・サイズが適切であり、I/Oが均等に分散されている場合は、非同期I/Oを使用するか、複数のデータベース・ライターを使用して、DBWRの動作を修正できます。

複数のデータベース・ライター(DBWR)・プロセスまたはI/Oスレーブの検討

複数のデータベース・ライター・プロセスを構成したり、I/Oスレーブを使用するのは、トランザクション・レートが高い場合や、バッファ・キャッシュ・サイズが大きすぎて単一のDBWnプロセスが負荷に耐えられない場合に役立ちます。

DB_WRITER_PROCESSES初期化パラメータを使用すると、複数のデータベース・ライター・プロセス(DBW0からDBW9までと、DBWaからDBWj)を構成できます。複数のDBWRプロセスを構成すると、書き込まれるバッファの識別に必要な作業が分散され、また、これらのプロセス間にI/O負荷もが分散されます。複数のデータベース・ライター・プロセスは、複数のCPUを持つシス

テム(CPU 8つにつき最低1つのデータベース・ライター)や、複数のプロセッサ・グループを持つシステム(最低でプロセス・グループと同数のデータベース・ライター)にお薦めします。

Oracle Databaseでは、CPU数またはプロセッサ・グループ数に基づいて、DB_WRITER_PROCESSESに適切なデフォルト設定を選択するか、ユーザー定義の設定を調整します。

複数のDBWRプロセスを使用することが実用的ではない場合、Oracle Databaseには複数のスレーブ・プロセス間にI/O負荷を分散する機能があります。DBWRプロセスは、ブロックを書き出すためにバッファ・キャッシュLRUリストをスキャンする唯一のプロセスです。ただし、これらのブロックのI/OはI/Oスレーブで実行されます。I/Oスレーブの個数は、DBWR_IO_SLAVESパラメータで決定されます。

DBWR_IO_SLAVESは、(CPUが1つの場合など)複数のDB_WRITER_PROCESSESを使用できない場合を想定しています。I/Oスレーブは、非同期I/Oが使用できないときにも役立ちます。これは、書き込まれるキャッシュ内のブロックの識別を継続するためにDBWRを解放することによって、複数のI/Oスレーブが非同期の非ブロック・リクエストをシミュレートするためです。一般的に、非同期I/Oがある場合、オペレーティング・システム・レベルの非同期I/Oをお薦めします。

DBWRのI/Oスレーブは、初回のI/Oリクエストが実行されるときに、データベースが開いた直後に割り当てられます。DBWRは、I/Oの実行とは別に、DBWR関連のすべての作業の実行を継続します。I/Oスレーブは単に、DBWRのためにI/Oを実行します。バッチの書込みはI/Oスレーブ間でパラレル化されます。

ノート:

DBWR_IO_SLAVES を実装するには、余分な共有メモリーを I/O バッファとリクエスト・キューに割り当てる必要があります。複数の DBWR プロセスを I/O スレーブと併用することはできません。I/O スレーブを構成すると、1 つの DBWR プロセスのみ起動するように設定されます。

複数のDBWRプロセスを構成すると、単一のDBWRプロセスでは必要なワークロードに耐えられない場合のパフォーマンスに役立ちます。ただし、複数のDBWRプロセスを構成する前に、非同期I/Oが使用でき、システム上に構成されるかどうかを確認します。システムが非同期I/Oをサポートしても現在使用されていない場合は、これにより問題が軽減されるかどうかを確認するために非同期I/Oを使用可能にします。システムが非同期I/Oをサポートしない場合、または非同期I/Oが構成され、DBWRボトルネックが依然として存在する場合は、複数のDBWRプロセスを構成します。

ノート:

プラットフォーム上で非同期 I/O を使用できない場合は、DISK_ASYNCH_IO 初期化パラメータを FALSE に設定して非同期 I/O を無効にできます。

複数のDBWRを使用すると、バッファの収集と書込みがパラレル化されます。そのため、複数のDBWnプロセスは同じ数のI/Oスレーブを使用する1つのDBWRプロセスよりもスループットを向上します。このため、複数のDBWRプロセスを優先して、I/Oスレーブの使用は非推奨になりました。I/Oスレーブは、複数のDBWRプロセスを構成できない場合のみ使用してください。

アイドル待機イベント

これらのイベントはIdle待機クラスに属し、作業がないためにサーバー・プロセスが待機中であることを示します。これは通常、ボトルネックが存在する場合にボトルネックがデータベース・リソースに対するものではないことを意味します。チューニングの際、アイドル・イベントの大部分を無視することが必要なのは、アイドル・イベントがパフォーマンス・ボトルネックの性質を示さないためです。アイドル・イベントの中には、ボトルネックでないものを示す際に役立つものがあります。このタイプのイベントの例として、最も一般的に発生するアイドル待機イベントであるSQL Net message from clientがあります。表10-2に、このアイドル・イベントと他のアイドル・イベント(およびそれらのカテゴリ)のリストを示します。

表10-2 アイドル待機イベント

待機名	バックグラウンド・プロセス・アイドル・イベント	ユーザー・プロセス・アイドル・イベント	パラレル問合せアイドル・イベント	共有サーバー・アイドル・イベント	Oracle Real Application Clustersアイドル・イベント
dispatcher timer	.	.	.	X	.
pipe get	.	X	.	.	.
pmon timer	X	.	.	.	.
PX Idle Wait	.	.	X	.	.
PX Deq Credit: need buffer	.	.	X	.	.
rdbms ipc message	X	.	.	.	.
shared server idle wait	.	.	.	X	.
smon timer	X	.	.	.	.
SQL*Net message from client	.	X	.	.	.

関連項目:

各アイドル待機イベントの説明は、『Oracle Databaseリファレンス』を参照してください。

ラッチ・イベント

ラッチは、メモリー構造を保護するためにOracle Databaseで使用される下位レベルの内部ロックです。ラッチ解放イベントは、サーバー・プロセスがラッチの取得を試み、最初の試行でラッチを取得できないときに更新されます。

通常は大きな競合を生成する一般的なラッチ用に、専用のラッチ関連待機イベントがあります。これらのイベントの場合は、latch: library cacheまたはlatch: cache buffers chainsのように、ラッチ名が待機イベント名に含まれます。このため、ラッチに関連するほとんどの競合の原因が特定のタイプのラッチであるかどうかをすばやく確認できます。他のすべてのラッチの待機は、汎用のlatch free待機イベントにグループ化されています。

関連項目:

ラッチおよび内部ロックの詳細は、『Oracle Database概要』を参照してください。

処置

このイベントは、ラッチ待機がシステム上の待機時間全体の重要な部分である場合か、または問題が発生している個々のユーザーに対してのみかを考慮してください。

- 関連するリソースのリソース使用量を調べます。たとえば、ライブラリ・キャッシュ・ラッチの競合が大きい場合は、ハードとソフトの解析率を調べます。
- ラッチ競合が発生しているセッションのSQL文を調べて、共通性があるかどうかを確認してください。

次のV$SESSION_WAITパラメータ列をチェックします。

- P1: ラッチのアドレス
- P2: ラッチ番号
- P3: ラッチの待機中にプロセスがスリープした回数

例: 現在待機されているラッチの検索

```
SELECT EVENT, SUM(P3) SLEEPS, SUM(SECONDS_IN_WAIT) SECONDS_IN_WAIT
  FROM V$SESSION_WAIT
 WHERE EVENT LIKE 'latch%'
  GROUP BY EVENT;
```

前述の問合せには、インスタンスまたは長期的なインスタンスのチューニングよりも、セッションのチューニングまたは短期的なインスタンスのチューニングに関して多くの情報が示されるという問題があります。

次の問合せは長期的なインスタンス・チューニングの詳細情報を提供し、ラッチの待機がデータベース全体の時間に占める割合が大きいかどうかを示します。

```
SELECT EVENT, TIME_WAITED_MICRO,
       ROUND(TIME_WAITED_MICRO*100/S.DBTIME,1) PCT_DB_TIME
  FROM V$SYSTEM_EVENT,
    (SELECT VALUE DBTIME FROM V$SYS_TIME_MODEL WHERE STAT_NAME = 'DB time') S
 WHERE EVENT LIKE 'latch%'
 ORDER BY PCT_DB_TIME ASC;
```

ラッチ待機固有でない、より汎用的な問合せは次のようになります。

```
SELECT EVENT, WAIT_CLASS,
      TIME_WAITED_MICRO,ROUND(TIME_WAITED_MICRO*100/S.DBTIME,1) PCT_DB_TIME
  FROM V$SYSTEM_EVENT E, V$EVENT_NAME N,
    (SELECT VALUE DBTIME FROM V$SYS_TIME_MODEL WHERE STAT_NAME = 'DB time') S
   WHERE E.EVENT_ID = N.EVENT_ID
    AND N.WAIT_CLASS NOT IN ('Idle', 'System I/O')
  ORDER BY PCT_DB_TIME ASC;
```

表10-3 ラッチ待機イベント

ラッチ	SGA領域	考えられる原因	検索対象
共有プール、ライブラリ・キャッシュ	共有プール	文の再利用不足 バインド変数を使用しない文 アプリケーション・カーソル・キャッシュのサイズが不十分 各実行後に明示的にクローズされたカーソル 頻繁なログインとログオフ 基礎的なオブジェクト構造の変更(たとえば、切捨て) 小さすぎる共有プール	次の項目が高いセッション(V$SESSTAT 内) ● CPU 解析時間 ● 所要解析時間 ● 解析数(ハード) / 実行数の比率 ● 解析数(合計) / 実行数の比率 次のカーソル(V$SQLAREA/V$SQLSTATS 内) ● PARSE_CALLS / EXECUTIONS の高い比率 ● EXECUTIONS = 1 WHERE 句のリテラルのみ異なる(すなわち、バインド変数を使用しない) ● 高い RELOADS ● 高い INVALIDATIONS ● 大きい(> 1MB)SHARABLE_M

ラッチ	SGA領域	考えられる原因	検索対象
			EM
キャッシュ・バッファ LRU 連鎖	バッファ・キャッシュ LRU リスト	過剰なバッファ・キャッシュ・スループット。たとえば、正しくない索引に繰り返しアクセスする非効率的なSQL(大きい索引レンジ・スキャン)、または多くの全表スキャンがあります。 実行中のワークロードに耐えられないDBWR。これにより、フォアグラウンド・プロセスが空きバッファを検索するためにラッチを保持する時間が長くなります。 小さすぎるキャッシュ	論理 I/O または物理 I/O が非常に多く、選択性のない索引が使用される文
キャッシュ・バッファ連鎖	バッファ・キャッシュ	ホット・ブロックと呼ばれる 1 つ(または少数)のブロックへのアクセスの繰返し	順序番号ジェネレータを使用せずに番号を生成するための、表の行を更新する順序番号生成コード 非常に多くのプロセスが、きわめて類似した述語を使用して選択性のない同一の索引をスキャンすることから発生する、索引リーフ・ブロックの競合 ホット・ブロックが属するセグメントの識別
行キャッシュ・オブジェクト			
共有プールとライブラリ・キャッシュ・ラッチの競合			

共有プールまたはライブラリ・キャッシュ・ラッチの競合の主な原因は解析です。不要な解析およびそのタイプは、いくつかの方法によって識別できます。

この方法では、リテラルがバインド変数と置換された場合に共有できる類似したSQL文を識別します。その概念は次のいずれかです。

- 実行を1つのみ持つSQL文を手動で検査して、SQL文が類似しているかどうかを確認します。

```
SELECT SQL_TEXT
  FROM V$SQLSTATS
 WHERE EXECUTIONS < 4
 ORDER BY SQL_TEXT;
```

- または、類似した文をグループ化することによって、このプロセスを自動化します。類似したSQL文のバイト数を見積り、そのバイト数によってSQL文をグループ化します。たとえば、次の例では、最初の60バイト以降のみが異なる文をグループ化します。

```
SELECT SUBSTR(SQL_TEXT, 1, 60), COUNT(*)
  FROM V$SQLSTATS
 WHERE EXECUTIONS < 4
 GROUP BY SUBSTR(SQL_TEXT, 1, 60)
 HAVING COUNT(*) > 1;
```

- 同じ実行計画を持つ個別のSQL文をレポートします。次の問合せでは、同じ実行計画を4回以上共有する個別のSQL文が選択されます。この種のSQL文では、バインド変数ではなくリテラルが使用されている可能性があります。

```
SELECT SQL_TEXT FROM V$SQLSTATS WHERE PLAN_HASH_VALUE IN
  (SELECT PLAN_HASH_VALUE
     FROM V$SQLSTATS
    GROUP BY PLAN_HASH_VALUE HAVING COUNT(*) > 4)
  ORDER BY PLAN_HASH_VALUE;
```

V$SQLSTATSビューをチェックします。次の問合せを入力します。

```
SELECT SQL_TEXT, PARSE_CALLS, EXECUTIONS
  FROM V$SQLSTATS
ORDER BY PARSE_CALLS;
```

PARSE_CALLSの値が指定の文のEXECUTIONS値に近い場合は、その文の再解析を継続できます。解析コールが多い文をチューニングします。

不要な解析コールが発生しているセッションを識別して、不要なコールを識別します。特定のバッチ・プログラムやある種のアプリケーションがほとんどの再解析を行っている場合があります。この目的を達成するには、次の問合せを実行します。

```
SELECT pa.SID, pa.VALUE "Hard Parses", ex.VALUE "Execute Count"
  FROM V$SESSTAT pa, V$SESSTAT ex
 WHERE pa.SID = ex.SID
   AND pa.STATISTIC#=(SELECT STATISTIC#
       FROM V$STATNAME WHERE NAME = 'parse count (hard)')
   AND ex.STATISTIC#=(SELECT STATISTIC#
       FROM V$STATNAME WHERE NAME = 'execute count')
   AND pa.VALUE > 0;
```

結果として、すべてのセッションのリストおよびセッションで実行された解析の量が得られます。セッション識別子(SID)ごとに、

V$SESSIONで、再解析の原因となっているプログラムの名前を検索します。

ノート:

この問合せではインスタンス起動後のすべての解析コールがカウントされるため、解析率の高いセッションを検索することをお薦めします。たとえば、50 日間の接続が高い解析値を示すとしても、2 番目の 10 分間の接続の方がより速い速度で解析される場合があります。

出力は、次のようなものです。

```
   SID  Hard Parses  Execute Count
------  -----------  -------------
     7            1             20
     8            3          12690
     6           26            325
    11           84           1619
```

cache buffers lru chainのラッチは、キャッシュ内のバッファのリストを保護します。リストからバッファの追加、移動または削除を行う場合、ラッチを取得する必要があります。

対称型マルチプロセッサ(SMP)システムでは、Oracle Databaseによって、LRUラッチの数がシステムのCPU数の2分の1に等しい値になるように自動的に設定されます。非SMPシステムでは、LRUラッチは1つあれば十分です。

LRUラッチの競合は、多数のCPUを搭載したSMPコンピュータでのパフォーマンスを低下させることがあります。LRUラッチの競合は、V$LATCH、V$SESSION_EVENTおよびV$SYSTEM_EVENTに問い合せることによって検出できます。競合を回避するには、アプリケーションのチューニング、DSSジョブのバッファ・キャッシュのバイパスまたはアプリケーションの再設計を検討してください。

cache buffers chainsのラッチは、バッファ・キャッシュでバッファ・リストを保護する場合に使用します。これらのラッチは、バッファの検索、追加、またはバッファ・キャッシュからの削除を行う際に使用されます。このラッチの競合は、競合度の高い(ホットな)ブロックが存在することを意味します。

頻繁にアクセスされるバッファ連鎖を識別して競合するブロックを識別するには、V$LATCH_CHILDRENビューを使用してcache buffers chainsのラッチのラッチ統計を調べます。他の子ラッチと比較して、多くのGETS、MISSESおよびSLEEPSを持つ特定のcache buffers chainsの子ラッチがある場合、これは子ラッチで競合します。

このラッチには、ADDR列で識別されるメモリー・アドレスがあります。このラッチで保護されているブロックを識別するには、X$BH表と結合されたADDR列の値を使用します。たとえば、頻繁に競合するラッチのアドレス(V$LATCH_CHILDREN.ADDR)を指定すると、ファイル番号とブロック番号の問合せが作成されます。

```
SELECT OBJ data_object_id, FILE#, DBABLK,CLASS, STATE, TCH
  FROM X$BH
 WHERE HLADDR = 'address of latch'
  ORDER BY TCH;
```

X$BH.TCHは、バッファのタッチ・カウントです。X$BH.TCHの値が高い場合は、ホット・ブロックであることを示します。

多くのブロックは、各ラッチにより保護されています。これらのバッファの1つは、おそらくホット・ブロックになります。TCH値の高いブロックは、潜在的なホット・ブロックです。この問合せを複数回実行し、出力に一貫して表示されるブロックを識別します。ホット・

ブロックを識別した後は、ファイル番号とブロック番号を使用してDBA_EXTENTSの問合せを行い、セグメントを識別します。

ホット・ブロックの識別後に、次の問合せを使用してそのホット・ブロックが属するセグメントを識別できます。

```
SELECT OBJECT_NAME, SUBOBJECT_NAME
  FROM DBA_OBJECTS
 WHERE DATA_OBJECT_ID = &obj;
```

この問合せでの&objは、X$BHへの前述の問合せにおけるOBJ列の値です。

row cache objectsのラッチは、データ・ディクショナリを保護します。

log file parallel write

このイベントには、ログ・バッファからREDOログ・ファイルへのREDOレコードの書込みが含まれます。

library cache pin

このイベントはライブラリ・キャッシュの同時実行性を管理します。オブジェクトを確保すると、ヒープがメモリーにロードされます。クライアントがオブジェクトを変更または検討するには、クライアントはロック後に確保を取得する必要があります。

library cache lock

このイベントは、ライブラリ・キャッシュの複数クライアント間の同時実行性を制御します。これによって、オブジェクト・ハンドルのロックが取得されるため、次の利点があります。

- クライアントは、別のクライアントが同じオブジェクトにアクセスしないようにできます。
- クライアントは、別のクライアントにオブジェクトの変更を許可しない依存関係を長期間維持

このロックの取得には、ライブラリ・キャッシュ内のオブジェクト位置を見つける働きもあります。

log buffer space

このイベントは、サーバー・プロセスがログ・バッファ内の空き領域を待機しているときに発生します。これは、LGWRがREDOを書き出すよりも、すべてのREDOが生成されるほうが速いためです。

処置

REDOログ・バッファ・サイズを修正します。ログ・バッファのサイズが適切な場合、オンラインREDOログが存在するディスクでI/O競合が発生しないことを確認します。log buffer space待機イベントは、REDOログが存在するディスク上のディスクI/O競合を示しているか、小さすぎるログ・バッファを示している可能性があります。REDOログを含むディスクのI/Oプロファイルをチェックして、I/Oシステムがボトルネックであるかどうかを調べます。I/Oシステムが問題ではない場合、REDOログ・バッファが小さすぎる可能性があります。このイベントが問題にならなくなるまで、REDOログ・バッファのサイズを大きくします。

log file switch

一般に発生する待機イベントは、次の2つです。

- Log file switch (archiving needed)

- Log file switch (checkpoint incomplete)

これらのイベントが発生すると、LGWRは次のオンラインREDOログ・ファイルへの切替えができません。すべてのコミット・リクエストは、このイベントを待機します。

処置

log file switch (archiving needed)イベントの場合、アーカイバがログをアーカイブできない理由を適時調べます。次の理由が考えられます。

- アーカイブ先に空き領域が不足している。
- アーカイバがREDOログを十分高速に読み取れない(LGWRとの競合)。
- アーカイバが十分高速に書込みを行えない(アーカイブ先での競合、またはARCHプロセスの数が不足している)。その他の可能性(ディスクが低速であったり、アーカイブ先がいっぱいなど)が除外された場合は、ARCnプロセス数の増加を検討してください。デフォルトは2です。
- 必須でリモートに転送されるアーカイブ・ログがある場合は、ネットワーク遅延によってこのプロセスが遅くなっていないか、またはエラーによって書込みが完了していないかをチェックしてください。

ボトルネックの性質により、I/Oを再分散したり、アーカイブ先に領域を追加して問題を軽減することが必要な場合があります。log file switch (checkpoint incomplete)イベントの場合、次を実行してください。

- DBWRが、I/Oシステムがオーバーロードしているか、低速のために遅くなっているのかどうかを調べます。DBWR書込み回数を調べ、I/Oシステムをチェックし、必要に応じてI/Oを分散させます。
- REDOログが少なすぎないか、または小さすぎないかを調べます。REDOログが少なすぎるか小さすぎる場合(たとえば、2 x 100KBのログ)、またDBWRがチェックポイントを完了する前に、すべてのログを巡回する十分なREDOが生成される場合、REDOログのサイズまたは個数、あるいはその両方を増やします。

log file sync

ユーザー・セッションがコミットする(またはロールバックする)場合、LGWRでセッションのREDO情報をREDOログ・ファイルにフラッシュする必要があります。COMMITまたはROLLBACKを実行するサーバー・プロセスは、REDOログへの書込みが完了するまで、このイベントで待機します。

処置

このイベントの待機がシステム上での長時間の待機か、レスポンス時間の問題が発生しているユーザーまたはシステム上のユーザーによる長時間の待機を構成している場合は、平均待機時間を調べます。

平均待機時間は短いが、待機数が多い場合、アプリケーションはCOMMITをバッチ処理するのではなく、すべてのINSERT後にコミットできます。アプリケーションは、行ごとではなく50行後にコミットして待機を減らすことができます。

平均待機時間が多い場合は、ログ・ライターに対するセッション待機を調べ、何を実行および待機するために多くの時間を費やしているかを調べます。待機が低速のI/Oによるものである場合は、次のことを試行してください。

- REDOログを含むディスク上の他のI/Oアクティビティを削減するか、専用ディスクを使用します。
- 異なるディスク上に交互のREDOログを設定して、ログ・ライターに対するアーカイバの影響をできるだけ少なくします。
- REDOログをさらに高速なディスクまたはさらに高速なI/Oサブシステム(たとえば、RAID 5からRAID 1への切替え)に

移動します。

- RAWデバイス(またはディスク・ベンダーが提供しているシミュレートされたRAWデバイス)を使用して書込みを高速化することを検討してください。
- アプリケーションのタイプにより異なりますが、1行ごとではなく、N行ごとにコミットして、COMMITをバッチ処理することで、ログ・ファイルの同期が少なくてすみます。

rdbms ipc reply

このイベントは、バックグラウンド・プロセスの1つからのメッセージを待機するために使用されます。

SQL*Netイベント

次のイベントは、データベース・プロセスがデータベース・リンクまたはクライアント・プロセスからの確認を待機していることを示します。

- SQL*Net break/reset to client
- SQL*Net break/reset to dblink
- SQL*Net message from client
- SQL*Net message from dblink
- SQL*Net message to client
- SQL*Net message to dblink
- SQL*Net more data from client
- SQL*Net more data from dblink
- SQL*Net more data to client
- SQL*Net more data to dblink

これらの待機がシステム上で、またはレスポンス時間の問題に対処しているユーザーに対して、待機時間の大部分を占めている場合、ネットワークまたは中間層がボトルネックになる可能性があります。

クライアント関連のイベントは、SQL*Net message from clientイベントで説明したとおりに診断する必要があります。dblink関連のイベントは、SQL*Net message from dblinkイベントで説明したとおりに診断する必要があります。

SQL*Net message from client

これはアイドル・イベントですが、このイベントを診断に使用できるときに何が問題でないかを説明することが重要です。このイベントは、サーバー・プロセスがクライアント・プロセスからの処理を待機していることを示します。ただし、このイベントが、長いレスポンス時間を経験しているユーザーの待機時間の大半を生成している状況がいくつかあります。その原因は、ネットワーク・ボトルネックまたはクライアント・プロセスでのリソース・ボトルネックのいずれかである可能性があります。

ネットワーク・ボトルネックは、アプリケーションによってサーバーとクライアントとの間で多量の通信が発生し、ネットワークの待機時間(ラウンドトリップの時間)が長い場合に発生する可能性があります。症状には次のものがあります。

- このイベントに対する多数の待機
- データベースとクライアント・プロセスは、時間のほとんどがアイドル状態(ネットワーク通信待ち状態)です。

ネットワーク・ボトルネックを軽減するには、次のことを試行します。

- アプリケーションをチューニングしてラウンドトリップを軽減します。
- 待機時間を削減するためのオプション(たとえば、VSATリンクとは反対の地上回線)を探索します。
- 通信量の大きいコンポーネントを待機時間の少ないリンクに移動するように、システム構成を変更します。

クライアント・プロセスがリソースの大半を使用している場合、データベース内で実行できることはありません。症状には次のものがあります。

- 待機数は多くなくても、待機時間は長い。
- クライアント・プロセスは、リソースを多く使用している。

場合によっては、クライアント・プロセスで使用されるCPUの量により、待機しているユーザーのトラッキングのための待機時間がわかります。この場合のクライアントという用語は、n層アーキテクチャにおける、データベース・プロセス(中間層、デスクトップ・クライアント)以外の任意のプロセスを指します。

SQL*Net message from dblink

このイベントは、セッションがリモート・ノードにメッセージを送り、データベース・リンクからのレスポンスを待機している状態であることを意味します。この時間は、次の理由で増える可能性があります。

- ネットワーク・ボトルネック

  詳細は、「SQL*Net message from client」を参照してください。

- リモート・ノードでSQLを実行するのに要する時間

  リモート・ノードで実行されているSQLを確認することが有益です。リモート・データベースにログインし、データベース・リンクで作成されたセッションを検索し、そのセッションで実行されるSQL文を調べます。

- ラウンドトリップ・メッセージの数

  セッションとリモート・ノード間の各メッセージにより、遅延時間が長くなり、処理オーバーヘッドが増加します。交換されるメッセージの数を減らすには、配列フェッチと配列挿入を使用します。

SQL*Net more data to client

サーバー・プロセスは、クライアントにさらに多くのデータまたはメッセージを送信します。クライアントに対する以前の操作も送信されました。

関連項目:

ネットワーク最適化の詳細は、『Oracle Database Net Services管理者ガイド』を参照してください。

インスタンス・リカバリのパフォーマンスのチューニング: ファスト・スタート・フォルト・リカバリ

この項では、インスタンス・リカバリと、クラッシュまたはインスタンス障害が発生した場合のOracleのファスト・スタート・フォルト・リカバリによる可用性の向上について説明します。また、クラッシュ・リカバリやインスタンス・リカバリの実行に必要な時間をチューニングするためのガイドラインも示します。

この項では、次の項目について説明します。

- インスタンス・リカバリについて
- キャッシュ・リカバリ時間の構成: FAST_START_MTTR_TARGET
- FAST_START_MTTR_TARGETのチューニングとMTTRアドバイザの使用

インスタンス・リカバリについて

インスタンス・リカバリおよびクラッシュ・リカバリとは、クラッシュやシステム障害後に、Oracleデータ・ブロックにREDOログ・レコードを自動的に適用することです。通常の操作中は、インスタンスが異常終了ではなく(SHUTDOWN IMMEDIATE文を使用する場合のように)正常に停止した場合、ディスクのデータファイルに書き込まれていないインメモリーの変更は、停止時に実行されるチェックポイントの一環としてディスクに書き込まれます。

ただし、シングル・インスタンス・データベースのクラッシュまたはOracle RAC構成のすべてのインスタンスのクラッシュが発生した場合、次の起動時にクラッシュ・リカバリが実行されます。Oracle RAC構成の1つ以上のインスタンスがクラッシュした場合は、残りのインスタンスによってインスタンス・リカバリが自動的に実行されます。インスタンス・リカバリとクラッシュ・リカバリのステップは、キャッシュ・リカバリ、トランザクション・リカバリの順に行われます。

キャッシュ・リカバリが完了するとすぐにデータベースを開くことができるため、可用性の向上にはキャッシュ・リカバリのパフォーマンスの向上が重要になります。

キャッシュ・リカバリ(ロールフォワード)

キャッシュ・リカバリ・ステップでは、REDOログ・ファイル内のコミット済およびコミットされていない変更がすべて、影響のあるデータ・ブロックに適用されます。キャッシュ・リカバリ処理に必要な作業量は、データベースの変更率(1秒当たりの更新トランザクション数)とチェックポイント間の時間に比例します。

トランザクション・リカバリ(ロールバック)

データベースの一貫性を保つには、クラッシュ時にコミットされなかった変更を元に戻す必要があります(ロールバック)。トランザクション・リカバリ・ステップでは、ロールバック・セグメントを適用してコミットされていない変更が元に戻されます。

チェックポイントとキャッシュ・リカバリ

Oracle Databaseでは、チェックポイントが定期的に記録されます。チェックポイントは最大のシステム変更番号(SCN)であるため、このSCN以下のデータ・ブロックはすべてデータファイルに書込み済であることがわかります。障害が発生した場合、チェックポイントよりも高いSCNでの変更を含むREDOレコードのみがリカバリ時に適用される必要があります。キャッシュ・リカバリ処理の時間は、チェックポイントのSCNよりも高いSCNでの変更を含むデータ・ブロックの数と、これらの変更を検出するために読取りが必要なログ・ブロックの数の2つの要因で決定されます。

チェックポイントによるパフォーマンスへの影響

チェックポイントの頻度が高くなると、そうでない場合よりも高い頻度で使用済バッファがデータファイルに書き込まれるため、インスタンス障害時のキャッシュ・リカバリ時間が短縮されます。チェックポイントの頻度が高い場合、REDOログの現在のチェックポイント位置とログの末尾までの間にあるREDOレコードを適用する際、相対的に少ないデータ・ブロックが処理されます。つまり、リカ

バリのキャッシュ・リカバリ・フェーズがかなり短くなります。

ただし、更新の多いシステムでは、チェックポイントの頻繁な発生により、実行時のパフォーマンスが低下する可能性があります。これはチェックポイントによってDBWnプロセスで書込みが実行されるためです。

ファスト・キャッシュ・リカバリのトレードオフ

キャッシュ・リカバリ時間を最短にするには、Oracle Databaseでのチェックポイントを頻繁にして、リカバリ時に適用されるREDOログ・レコードの数を最小限にします。ただし、更新の多いシステムでは、チェックポイントの頻繁な発生により、通常のデータベース操作のオーバーヘッドが増加します。

リカバリ時間の短縮よりも日常の操作効率の方が重要な場合は、チェックポイントによるデータファイルへの書込み頻度を減らします。これにより、操作効率は向上しますが、キャッシュ・リカバリ時間も増加します。

キャッシュ・リカバリ時間の構成: FAST_START_MTTR_TARGET

ファスト・スタート・フォルト・リカバリ機能により、キャッシュ・リカバリに必要な時間が短縮されます。また、使用済バッファの数、および最新のREDOレコードと最後のチェックポイントとの間に生成されるREDOレコードの数を制限して、リカバリの制御および予測を行うことができます。

ファスト・スタート・フォルト・リカバリの基本は、ファスト・スタート・チェックポイント・アーキテクチャです。バルク書込みを行う従来のイベント・ドリブン(ログ・スイッチ)・チェックポイントではなく、ファスト・スタート・チェックポイントは段階的に発生します。各DBWnプロセスは、定期的にバッファからディスクへの書込みを行い、チェックポイント位置を先に進めます。最も古い変更ブロックは、各書込みによってチェックポイントが先に進められるように最初に書き込まれます。ファスト・スタート・チェックポイントでは、従来のチェックポイントで発生するバルク書込みと、その結果発生するI/Oスパイクを回避できます。

ファスト・スタート・フォルト・リカバリ機能では、初期化パラメータFAST_START_MTTR_TARGETによって、インスタンス障害またはシステム障害からのリカバリ時間を簡単に構成できます。FAST_START_MTTR_TARGETは、平均リカバリ時間(MTTR)の目標値、つまりインスタンスを起動してキャッシュ・リカバリの実行にかかる時間(秒単位)の目標値を指定します。
FAST_START_MTTR_TARGETを設定すると、データベースはその目標値を達成するように増分チェックポイントの書込みを管理します。FAST_START_MTTR_TARGETに現実的な値を選択した場合、選択したおおよその平均秒数内でデータベースがリカバリされます。

ノート:

FAST_START_MTTR_TARGET を使用する場合は、初期化パラメータ FAST_START_IO_TARGET、LOG_CHECKPOINT_INTERVAL および LOG_CHECKPOINT_TIMEOUT を無効にするか削除する必要があります。これらのパラメータを設定すると、FAST_START_MTTR_TARGET を達成するためにキャッシュ・リカバリ時間の管理に使用するメカニズムが適切に機能しなくなります。

FAST_START_MTTR_TARGETの現実的な値

FAST_START_MTTR_TARGETの最大値は3600秒(1時間)です。3600を超える値を設定すると、Oracle Databaseでその値は3600に丸められます。

次に、FAST_START_MTTR_TARGETの値を設定する方法の例を示します。

```
SQL> ALTER SYSTEM SET FAST_START_MTTR_TARGET=30;
```

原則ではFAST_START_MTTR_TARGETの最小値は1秒です。ただし、FAST_START_MTTR_TARGETにそのような小さい値を設定できるからといって、その目標値を達成できるということにはなりません。データベースの起動時間などの要因によって、達成可能なMTTR目標値には現実的な下限があります。

FAST_START_MTTR_TARGETの現在の値を前提として、データベースが達成可能なMTTR目標値を有効MTTR目標値といいます。現在の有効MTTRを参照するには、V$INSTANCE_RECOVERYビューのTARGET_MTTR列を参照します。

使用するデータベースの現実的なMTTR目標値の範囲は、そのデータベースで達成可能な有効MTTR目標値の最小値から、最悪の場合(バッファ・キャッシュ全体が使用済の場合)に起動とキャッシュ・リカバリにかかる時間の最大値までです。「FAST_START_MTTR_TARGETの現実的な範囲の決定」に記載された達成可能なMTTR目標値の範囲を決定する手順は、FAST_START_MTTR_TARGET値のチューニング・プロセスの1ステップです。

ノート:

FAST_START_MTTR_TARGET を現実的な範囲外の値に設定するのは、通常、有益ではありません。FAST_START_MTTR_TARGET の値が現実的な範囲の下限よりも小さい場合、その効果は現実的な範囲の下限に設定した場合と同様になります。このような場合、有効な MTTR 目標値はシステムで達成できる最高の MTTR 目標値になりますが、チェックポイントが最高頻度で実行され、通常のデータベースのパフォーマンスに影響を与える可能性があります。FAST_START_MTTR_TARGET を現実的な範囲よりも長い時間に設定する場合、MTTR 目標値は最悪の場合と同じになります。

ランタイム・パフォーマンスを最適化するためのチェックポイント頻度の低減

チェックポイントの頻度を低くしてランタイム・パフォーマンスを最適化するには、次の手順を実行します。

- FAST_START_MTTR_TARGETの値を3600に設定します。これにより、ファスト・スタート・チェックポイントとファスト・スタート・フォルト・リカバリ機能が有効になりますが、実行時パフォーマンスへの影響は最小限に抑えられ、FAST_START_MTTR_TARGETによるパフォーマンスのチューニングが必要なくなります。
- システムが生成するREDOの量に従って、オンラインREDOログ・ファイルをサイズ設定します。ログの切替えは、最も頻繁に行う場合でも20分間隔とします。ログ・ファイルのサイズが小さすぎると、チェックポイント・アクティビティが増加し、パフォーマンスが低下します。また、すべてのREDOログ・ファイルのサイズが同じになるようにしてください。

関連項目:

チェックポイントの詳細は、Oracle Database概要を参照してください

V$INSTANCE_RECOVERYによるキャッシュ・リカバリの監視

V$INSTANCE_RECOVERYビューには、現在のリカバリ・パラメータの設定が表示されます。このビューの統計を使用して、チェックポイントに最大の影響を与えている要因を判断することもできます。

次の表に、キャッシュ・リカバリのパフォーマンスの予測を管理する際に役立つ列を示します。

表10-4 V$INSTANCE_RECOVERYの列

列	説明
TARGET_MTTR	有効な MTTR 目標値(秒単位)。 FAST_START_MTTR_TARGET を指定しない場合、このフィールドは 0 です。
ESTIMATED_MTTR	現在の使用済バッファ数およびログ・ブロック数に基づいた、現在の推定 MTTR(秒単位)。FAST_START_MTTR_TARGET を指定しているかどうかにかかわらず、このフィールドは常に計算されます。

使用するデータベースで実行中の監視の一環として、V$INSTANCE_RECOVERY. TARGET_MTTRとFAST_START_MTTR_TARGETを定期的に比較できます。この2つの値は、FAST_START_MTTR_TARGETの値が現実的な範囲内にある場合は通常同じになります。TARGET_MTTRがFAST_START_MTTR_TARGETよりも常に大きい場合は、FAST_START_MTTR_TARGETをTARGET_MTTR以上の値に設定します。TARGET_MTTRが常に小さい場合は、FAST_START_MTTR_TARGETをTARGET_MTTR以下の値に設定します。

関連項目:

V$INSTANCE_RECOVERYビューの詳細は、『Oracle Databaseリファレンス』を参照してください

FAST_START_MTTR_TARGETのチューニングとMTTRアドバイザの使用

使用するデータベースに適切なFAST_START_MTTR_TARGET値を判断するには、次の4つのステップを使用します。

- FAST_START_MTTR_TARGETの測定
- FAST_START_MTTR_TARGETの現実的な範囲の決定
- MTTRアドバイザによる様々な目標値の評価
- 最適なREDOログ・サイズの決定

FAST_START_MTTR_TARGETの測定

初期化パラメータFAST_START_MTTR_TARGETにより、REDOログの長さとデータ・キャッシュの使用済データ・バッファの数を制限するために、内部システム・トリガーの値が計算されます。この計算では、REDOブロックの読取りの概算時間、データ・ブロックの読取りと書込みの概算時間、およびシステムの一般的なワークロードの特性(変更ベクトル数に対応する使用済バッファ数など)が使用されます。

最初は、内部的なデフォルト値が計算に使用されます。これらのデフォルト値は、時間の経過とともに、システム操作時および実際のキャッシュ・リカバリ時にI/Oパフォーマンスに関して収集されたデータに置き換えられます。

FAST_START_MTTR_TARGET値を適切に測定するには、インスタンス・リカバリを複数回実行する必要があります。測定を開始

する前に、データベース・クラッシュとハードウェア・クラッシュのどちらに対してFAST_START_MTTR_TARGETを測定するかを決定してください。これは、データベース・ファイルがファイル・システムに格納される場合、またはI/Oサブシステムにメモリー・キャッシュがある場合の考慮事項です。ファイルがキャッシュされるかどうかによって、ディスクへの読取り時間と書込み時間に大幅な違いがあるためです。FAST_START_MTTR_TARGETの適切な値は、どのタイプのクラッシュがより迅速にリカバリする必要があるかによって異なります。

FAST_START_MTTR_TARGETを効率的に測定するには、システムの一般的なワークロードを長時間実行し、リカバリ時のREDOブロックの読取り時間およびデータ・ブロックの読取りまたは書込み時間が正確に記録されるようにインスタンス・リカバリを複数回実行します。

FAST_START_MTTR_TARGETの現実的な範囲の決定

測定後、テストを実行して、使用するデータベースのFAST_START_MTTR_TARGETの現実的な範囲を決定できます。

FAST_START_MTTR_TARGETの下限の決定: シナリオ

現実的な範囲の下限を決定するには、FAST_START_MTTR_TARGETを1に設定してデータベースを起動します。次に、V$INSTANCE_RECOVERY.TARGET_MTTRの値を確認して、この値をFAST_START_MTTR_TARGETの有効な下限として使用します。この下限の決定には、通常、キャッシュ・リカバリ時間ではなくデータベースの起動時間の方が主要な要因となります。

たとえば、次のようにFAST_START_MTTR_TARGETを1に設定します。

```
SQL> ALTER SYSTEM SET FAST_START_MTTR_TARGET=1;
```

次に、データベースを開いた直後に次の問合せを実行します。

```
SQL> SELECT TARGET_MTTR, ESTIMATED_MTTR
    FROM V$INSTANCE_RECOVERY;
```

次のような結果が返されます。

```
TARGET_MTTR ESTIMATED_MTTR
18          15
```

TARGET_MTTRの18秒という値はシステムで達成可能な最小MTTR目標値、つまりFAST_START_MTTR_TARGETの現実的な下限値です。この最小値は、データベースの平均起動時間に基づいて計算されます。

ESTIMATED_MTTRフィールドには、実行中のデータベースの現在の状態に基づいた、概算平均リカバリ時間が含まれます。データベースは開いた直後であり、システムに含まれるのは少数の使用済バッファであるため、この時点でインスタンスに障害が発生した場合でも、それほど多くのキャッシュ・リカバリは必要ありません。そのため、ESTIMATED_MTTRは、この時点ではTARGET_MTTRに指定可能な最小値よりも小さい値になる場合があります。

ESTIMATED_MTTRは、最近のデータベース・アクティビティによって短期的に影響を受ける場合があります。データベースで大量の更新アクティビティを一定期間行った直後にV$INSTANCE_RECOVERYに対して問合せを実行するとします。次のような結果が返されます。

```
TARGET_MTTR ESTIMATED_MTTR
18          30
```

有効MTTR目標値は18秒のままで、(その時点でクラッシュが発生した場合の)概算MTTRは30秒です。これは許容可能な結果です。つまり、一部のチェックポイントの書込みが完了していないため、バッファ・キャッシュには目標よりも多くの使用済バッファが含まれています。

ここで60秒待機してから、V$INSTANCE_RECOVERYに対して再度問合せを発行します。次のような結果が返されます。

```
TARGET_MTTR ESTIMATED_MTTR
18          25
```

この期間に一部の使用済バッファが書き込まれたため、今回の概算MTTRは25秒に減少しています。

FAST_START_MTTR_TARGETの上限の決定

現実的な範囲の上限を決定するには、FAST_START_MTTR_TARGETを3600に設定し、一般的なワークロードでデータベースを一定期間操作します。次にV$INSTANCE_RECOVERY.TARGET_MTTRの値を確認します。この値が、FAST_START_MTTR_TARGETの有効な上限です。

この手順は「FAST_START_MTTR_TARGETの下限の決定: シナリオ」と実質的に似ています。

FAST_START_MTTR_TARGETの初期値の選択

FAST_START_MTTR_TARGETパラメータの現実的な範囲を決定した後、このパラメータの初期値を選択します。データベースのパフォーマンスを懸念する場合は、現実的な範囲内で高い値を選択し、リカバリ時間の短縮が優先される場合は現実的な範囲内で低い値を選択します。当然のことながら、現実的な範囲が狭いほど、選択は容易になります。

たとえば、現実的な範囲が17から19秒の場合、単純に19を選択できます。これはリカバリ時間にはほとんど差がなく、同時にシステムのパフォーマンスに対するチェックポイントの影響が最小限に抑えられるためです。しかし、現実的な範囲が18から40秒の場合は、中間的な値として30を選択し、この値に従ってパラメータを設定できます。

```
SQL> ALTER SYSTEM SET FAST_START_MTTR_TARGET=30;
```

その後、MTTRアドバイザを使用して最適値を判断できます。

MTTRアドバイザによる様々な目標値の評価

FAST_START_MTTR_TARGETの初期値を選択した後、MTTRアドバイザを使用して、異なるFAST_START_MTTR_TARGETの設定がシステム・パフォーマンスに与える影響を、選択した設定と比較して評価できます。

MTTRアドバイザの有効化

MTTRアドバイザを有効にするには、2つの初期化パラメータSTATISTICS_LEVELおよびFAST_START_MTTR_TARGETを設定します。

STATISTICS_LEVELでは、MTTRアドバイザのみでなく、すべてのアドバイザの有効化を制御します。このパラメータがTYPICALまたはALLに設定されていることを確認します。次にFAST_START_MTTR_TARGETをゼロ以外の値に設定すると、MTTRアドバイザが有効になります。

MTTRアドバイザの使用

MTTRアドバイザを有効にした後、一般的なデータベース・ワークロードを一定期間実行します。MTTRアドバイザが有効な場

合、現在のFAST_START_MTTR_TARGET値、およびFAST_START_MTTR_TARGET値の有効範囲内の最大4つの異なるMTTR設定に基づいて、チェックポイント・キュー動作がシミュレートされます。(この場合、FAST_START_MTTR_TARGETの有効範囲が決定されてから、その範囲内の複数の値がテストされます。)

MTTRアドバイザの結果の表示: V$MTTR_TARGET_ADVICE

動的パフォーマンス・ビューV$MTTR_TARGET_ADVICEでは、MTTRアドバイザによって収集された統計またはアドバイスを表示できます。

V$MTTR_TARGET_ADVICEには、データベースに対するFAST_START_MTTR_TARGETの各設定の影響に関するアドバイスが移入されます。FAST_START_MTTR_TARGETの値ごとに、行にはFAST_START_MTTR_TARGETの値に対してテストに使用されたワークロード下で実行されるキャッシュ書込み数に関する詳細が表示されます。

具体的には、各行には、その行のFAST_START_MTTR_TARGET値でのキャッシュ書込み数、合計物理書込み数(直接書込みを含む)および合計I/O(読取りを含む)に関する情報が、合計操作数と、選択したFAST_START_MTTR_TARGET値での操作との比率の両方で表示されます。たとえば、比率が1.2の場合は、キャッシュ書込みが20%多いことを表しています。

様々なFAST_START_MTTR_TARGET設定がキャッシュの書込みアクティビティやその他のI/Oに与える影響を理解すると、リカバリやパフォーマンスの要件に最適なFAST_START_MTTR_TARGET値をより効率的に決定できます。

MTTRアドバイザが現在有効な場合は、V$MTTR_TARGET_ADVICEに、収集されたアドバイザ情報が表示されます。現在、MTTRアドバイザがOFFの場合、データベースを起動してからMTTRアドバイザが最後にONであったときに収集された情報が表示されます(情報が存在する場合)。最後にMTTRアドバイザを使用してからデータベースが再起動された場合、またはMTTRアドバイザが1回も使用されなかった場合、このビューには行は表示されません。

関連項目:

V$MTTR_TARGET_ADVICEビューの列の詳細は、Oracle Databaseリファレンスを参照してください

最適なREDOログ・サイズの決定

V$INSTANCE_RECOVERYビューのOPTIMAL_LOGFILE_SIZE列を使用して、オンラインREDOログのサイズを決定できます。このフィールドには、FAST_START_MTTR_TARGETの現在の設定に基づいて最適と判断されるREDOログ・ファイルのサイズ(MB単位)が表示されます。このフィールドに、最も小さいオンライン・ログのサイズよりも大きな値が常に表示される場合は、すべてのオンライン・ログをこのサイズ以上に構成する必要があります。

ただし、REDOログ・ファイルのサイズがMTTRに影響を与えることに注意してください。ログ・ファイルのサイズを、希望する最適な値に設定してMTTRアドバイザを再実行すると、選択した最適なFAST_START_MTTR_TARGET値をさらに適切に調整できる場合もあります。

第III部 データベース・メモリーのチューニング

この部は、次の章で構成されています。

11 データベース・メモリーの割当て

この章では、Oracle Databaseにおけるメモリー割当て、およびメモリー管理の様々な方法について説明します。

この章のトピックは、次のとおりです:

- データベース・メモリー・キャッシュと他のメモリー構造について
- データベース・メモリーの管理方法
- 自動メモリー管理の使用
- メモリー管理の監視

データベース・メモリー・キャッシュと他のメモリー構造について

Oracle Databaseでは、メモリー・キャッシュおよびディスクに情報を格納します。メモリー・アクセスは、ディスク・アクセスよりはるかに高速です。ディスク・アクセス(物理I/O)は、メモリー・アクセスに比べ、時間がかかります(通常は約10ミリ秒)。また、物理I/Oでは、デバイス・ドライバやオペレーティング・システムのイベント・スケジューラのパス長のために必要なCPUリソースも増加します。このため、頻繁にアクセスされるオブジェクトに対するデータ・リクエストは、ディスク・アクセスではなくメモリー・アクセスで実行するほうが効率的です。Oracle Databaseメモリー・キャッシュを適切にサイズ設定して効率的に使用すると、データベースのパフォーマンスが大幅に向上します。

パフォーマンスに影響するOracle Databaseの主なメモリー・キャッシュは、次のとおりです。

- データベース・バッファ・キャッシュ

  データベース・バッファ・キャッシュには、ディスクから読み取られたデータ・ブロックが格納されます。

- REDOログ・バッファ

  REDOログ・バッファには、バッファ・キャッシュのデータ・ブロックに行われた変更のREDOエントリが格納されます。

- 共有プール

  共有プールからは様々なタイプのデータがキャッシュされ、主に次のコンポーネントで構成されています。

  - ライブラリ・キャッシュ
  - データ・ディクショナリ・キャッシュ
  - サーバー結果キャッシュ

- ラージ・プール

  ラージ・プールでは、次のようなOracle Database機能に対して大量のメモリー割当てを行うことができます。

  - 共有サーバー・アーキテクチャ
  - パラレル問合せ
  - Recovery Manager (RMAN)

- Javaプール

  Javaプールには、セッションに特化したJavaコードおよびJava Virtual Machine (JVM)データが格納されます。

- Streamsプール

  Streamsプールは、Oracle Advanced Queuing (AQ)およびレプリケーション・プロセスのためのメモリーを提供します。

- プロセス・プライベート・メモリー

  プロセス・プライベート・メモリーには、ソートやハッシュ結合などの操作に使用されるメモリーが含まれます。

- インメモリー列ストア(IM列ストア)

  Oracle Database 12cリリース1 (12.1.0.2)以降、IM列ストアは、表およびパーティションのコピーを格納するオプションの静的SGAプールです。IM列ストアでは、データは特殊な列形式に格納され、これによってスキャン、結合および集計などの操作のパフォーマンスが向上します。

ノート:

IM 列ストアでは、バッファ・キャッシュは置換しませんが、両方のメモリー領域において同じデータを異なる形式で格納するための補足としての役割を果たします。

関連項目:

Oracle Databaseのメモリー・アーキテクチャの詳細は、『Oracle Database概要』を参照してください

データベース・メモリーの管理方法

メモリー管理の目標は、必要なデータがメモリー内にある可能性を高くしたり、必要なデータを取り出すプロセスをさらに効率的にし、できるだけ多くの物理I/Oオーバーヘッドを削減することです。この目標を達成するためには、Oracle Databaseメモリー・キャッシュを適切にサイズ設定して効率的に使用することが不可欠です。

Oracle Databaseでは、次のような方法でデータベース・メモリーを管理します。

- 自動メモリー管理
- 自動共有メモリー管理
- 手動共有メモリー管理
- 自動PGAメモリー管理
- 手動PGAメモリー管理

自動メモリー管理

自動メモリー管理により、Oracle Databaseでは、データベース・メモリーを自動的に管理およびチューニングできます。自動メモリー管理モードでは、共有グローバル領域(SGA)およびプログラム・グローバル領域(インスタンスPGA)のメモリー管理が、Oracle Databaseによって完全に行われます。最も自動化されているため、この方法を使用することをお薦めします。メモリー・プール・サイズを手動で設定する前に、自動メモリー管理を使用することを積極的に検討してください。

自動メモリー管理の使用方法の詳細は、「自動メモリー管理の使用」を参照してください。

自動共有メモリー管理

自動メモリー管理が無効化されている場合、Oracle Databaseでは、自動共有メモリー管理を使用してSGAメモリーを管理します。このモードでは、Oracle Databaseにより、合計SGAメモリーに設定されているターゲット・サイズに基づき、メモリーが自動的に個々のSGAコンポーネントに配分されます。

自動共有メモリー管理の使用方法の詳細は、「自動共有メモリー管理の使用」を参照してください。

手動共有メモリー管理

自動メモリー管理および自動共有メモリー管理の両方が無効化されている場合は、SGAの個々のメモリー・プールのサイズを設定して、手動でSGAメモリーを管理する必要があります。このモードでは、SGAメモリーの分散方法をユーザーが完全に制御できますが、SGAコンポーネントを継続的に手動でチューニングする必要があるため、最も手間がかかります。

手動共有メモリー管理の詳細は、「手動によるSGAコンポーネントのサイズ設定」を参照してください。

自動PGAメモリー管理

自動メモリー管理が無効化されている場合、Oracle Databaseでは、自動PGAメモリー管理を使用してPGAメモリーを管理します。このモードでは、Oracle Databaseにより、合計PGAメモリーに設定されているターゲット・サイズに基づき、メモリーがインスタンスPGAの作業領域に自動的に配分されます。

自動PGAメモリー管理の詳細は、「プログラム・グローバル領域のチューニング」を参照してください。

手動PGAメモリー管理

自動メモリー管理および自動PGAメモリー管理の両方が無効化されている場合は、各作業領域に割り当てられているPGAメモリーの一部を調整し、手動でPGAメモリーを管理する必要があります。ワークロードは常に変化するため、この方法は非常に困難になる可能性があり、薦めしません。Oracle Databaseでは、手動によるPGAメモリー管理がサポートされていますが、自動メモリー管理または自動PGAメモリー管理を使用することをお薦めします。

自動メモリー管理の使用

自動メモリー管理を使用するには、次の初期化パラメータを設定します。

- MEMORY_TARGET

  MEMORY_TARGET初期化パラメータには、ターゲット・メモリー・サイズを指定します。データベースは、SGAおよびインスタンスPGAの間で必要に応じてメモリーを再配分し、このパラメータに指定された値になるよう調整します。このパラメータは動的であるため、データベースを再起動することなく、いつでもこの値を変更できます。

- MEMORY_MAX_TARGET

  MEMORY_MAX_TARGET初期化パラメータには、最大メモリー・サイズを指定します。このパラメータに指定された値は、MEMORY_TARGET初期化パラメータに設定できる制限値の役割をはたします。このパラメータは静的であるため、インスタンスの起動後にこの値を変更することはできません。

MEMORY_TARGETパラメータのチューニングにアドバイスが必要な場合は、V$MEMORY_TARGET_ADVICEビューを使用してください。

関連項目:

自動メモリー管理の使用方法の詳細は、『Oracle Database管理者ガイド』を参照してください。

メモリー管理の監視

表11-1に、メモリー・サイズ変更操作に関する情報が表示されるビューを示します。

表11-1 メモリー管理ビュー

ビュー	説明
V$MEMORY_CURRENT_RESIZE_OPS	現在進行中のメモリー・サイズ変更操作(自動および手動の両方)に関する情報が表示されます。
V$MEMORY_DYNAMIC_COMPONENTS	動的にチューニングされたすべてのメモリー・コンポーネントの現在のサイズ(SGA およびインスタンス PGA の合計サイズなど)に関する情報が表示されます。
V$MEMORY_RESIZE_OPS	最新 800 件の実行済メモリー・サイズ変更操作(自動および手動の両方)に関する情報が表示されます。これには現在進行中の操作は含まれません。
V$MEMORY_TARGET_ADVICE	MEMORY_TARGET 初期化パラメータのチューニング・アドバイスが表示されます。

関連項目:

これらのビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

12 システム・グローバル領域のチューニング

この章では、システム・グローバル領域(SGA)のチューニング方法を説明します。自動メモリー管理を使用してシステムのデータベース・メモリーを管理している場合は、この章で説明されているように、SGAのチューニングは必要ありません。

この章のトピックは、次のとおりです:

- 自動共有メモリー管理の使用
- 手動によるSGAコンポーネントのサイズ設定
- 共有メモリー管理の監視
- インメモリー列ストアによる問合せのパフォーマンス向上
- Memoptimizeされた行ストアを使用した高パフォーマンス・データ・ストリーミングの有効化

自動共有メモリー管理の使用

自動共有メモリー管理を使用すると、SGAのメモリーが次のメモリー・プールに自動的に配分され、SGAの構成が簡略化されます。

- データベース・バッファ・キャッシュ(デフォルト・プール)
- 共有プール
- ラージ・プール
- Javaプール
- Streamsプール

自動共有メモリー管理は、SGA_TARGETパラメータで制御します。SGA_TARGETパラメータの値を変更すると、これらのメモリー・プールが自動的にサイズ変更されます。これらのメモリー・プールがゼロ以外の値に設定されている場合、自動共有メモリー管理では、これらの値が最低レベルとして使用されます。最低値は、アプリケーション・コンポーネントが正しく機能するために必要な最低メモリー量に基づいて設定することをお薦めします。

次に示すメモリー・キャッシュは手動でサイズ設定されるコンポーネントで、自動共有メモリー管理の制御対象ではありません。

- REDOログ・バッファ

  REDOログ・バッファは、「REDOログ・バッファの構成」で説明されているように、LOG_BUFFER初期化パラメータを使用してサイズ設定されます。

- その他のバッファ・キャッシュ(KEEP、RECYCLEおよび他の非デフォルト・ブロック・サイズなど)

  KEEPプールは、「KEEPプールの構成」で説明されているように、DB_KEEP_CACHE_SIZE初期化パラメータを使用してサイズ設定されます。

  RECYCLEプールは、「RECYCLEプールの構成」で説明されているように、DB_RECYCLE_CACHE_SIZE初期化パラメータを使用してサイズ設定されます。

- 固定SGAおよびその他の内部割当て

  固定SGAおよびその他の内部割当ては、DB_nK_CACHE_SIZE初期化パラメータを使用してサイズ設定されます。

これらのメモリー・キャッシュに割り当てられたメモリーは、自動チューニングされたメモリー・プールの値が自動共有メモリー管理によって計算されるときに、SGA_TARGETパラメータの値から引かれます。

次の各項では、SGA_TARGETパラメータの値へのアクセス方法および設定方法を説明します。

- SGA_TARGETパラメータを設定するためのユーザー・インタフェース
- SGA_TARGETパラメータの設定

関連項目:

- SGAの詳細は、『Oracle Database概要』を参照してください
- SGAの管理の詳細は、『Oracle Database管理者ガイド』を参照してください
- 初期化パラメータの使用方法の詳細は、『Oracle Database管理者ガイド』を参照してください

SGA_TARGETパラメータを設定するためのユーザー・インタフェース

この項では、SGA_TARGETパラメータの値を設定するためのユーザー・インタフェースについて説明します。

この項では、次の項目について説明します。

- Oracle Enterprise Manager Cloud ControlでのSGA_TARGETパラメータの設定
- コマンドライン・インタフェースでのSGA_TARGETパラメータの設定

Oracle Enterprise Manager Cloud ControlでのSGA_TARGETパラメータの設定

メモリー・パラメータSGAページからSGAサイズ・アドバイザにアクセスすることにより、Oracle Enterprise Manager Cloud Control (Cloud Control)でSGA_TARGETパラメータの値を変更できます。

コマンドライン・インタフェースでのSGA_TARGETパラメータの設定

V$SGA_TARGET_ADVICEビューを問い合せ、ALTER SYSTEMコマンドを使用することにより、コマンドライン・インタフェースでSGA_TARGETパラメータの値を変更できます。

SGA_TARGETパラメータの設定

この項では、SGA_TARGETパラメータの値を設定して、自動共有メモリー管理を有効化および無効化する方法を説明します。

この項では、次の項目について説明します。

- 自動共有メモリー管理を使用可能にする方法
- 自動共有メモリー管理の無効化

自動共有メモリー管理の有効化

自動共有メモリー管理を有効化するには、次の初期化パラメータを設定します。

- STATISTICS_LEVELをTYPICALまたはALLに
- SGA_TARGETをゼロ以外の値に

SGA_TARGETパラメータは、SGA_MAX_SIZE初期化パラメータの値以下に設定できます。SGA_TARGETパラメータの値は、SGA専用にするメモリーの容量に設定します。

自動共有メモリー管理の無効化

自動共有メモリー管理を無効化するには、インスタンスの起動時にSGA_TARGETパラメータの値を動的に0に設定します。

これにより、自動共有メモリー管理が無効化され、現在の自動チューニングされたサイズが各メモリー・プールに使用されます。必要な場合は、「手動によるSGAコンポーネントのサイズ設定」の説明に従い、各メモリー・プールのサイズを手動で再設定できます。

統合プログラム・グローバル領域

統合プログラム・グローバル領域(PGA)プールは、マルチテナント・コンテナ・データベース(CDB)と比較してCPUごとのSGAターゲット値が小さくなっている特定のプラガブル・データベース(PDB)によってPGA作業領域に使用される共有グローバル領域(SGA)コンポーネントです。

PGAを使用すると、Autonomous Transaction Processing (ATP)プラガブル・データベース(PDB)をAutonomous Data Warehouse (ADW) PDBに置き換えることができます。そのためには、SGAからADW PDBに一部のPGAを透過的に割り当てます。

ATP-D環境では、メモリーはSGAとPGAの使用量で分割されます。SGAは、仮想マシンまたはホストの起動時またはメモリーが断片化される前の初期段階で一般的に予約されているラージ・ページによってサポートされます。システム・メモリーのSGA部分は、トランザクション・プロセスの要件に基づいて構成され、通常、SGAの場合は65%、PGAの場合は35%です。ただし、ADWプラガブル・データベースにはより多くのプライベート・メモリーが必要であり、通常、SGAの場合は35%、PGAの場合は65%の範囲となります。残念なことに、システムがATP用に構成されているため、PGAのニーズに対してヒュージ・ページ(SGA用に構成)のメモリーを解放できません。ラージ・ページは断片化されると、将来再利用することは困難です。

統合PGAプールは、共有グローバル領域(SGA)がプラガブル・データベース(PDB)のプログラム・グローバル領域(PGA)に使用できるようになる新しい構成です。統合PGAでは、拡張する必要がある場合はバッファ・キャッシュから完全なグラニュルのみをリクエストし、不要になった場合は完全なグラニュルを放棄します。インスタンスの起動時に、PGAには、最小サイズ0 (ゼロ)または_SIZEパラメータで定義されたサイズを指定できます。

ノート:

完全なグラニュルのみが統合 PGA プールで使用できるため、グラニュルが拡張のためにリクエストされる頻度およびシステム・リストに戻されるまでの時間について注意してください。

手動によるSGAコンポーネントのサイズ設定

システムで自動メモリー管理または自動共有メモリー管理が使用されていない場合は、次のSGAコンポーネントのサイズを手動で構成する必要があります。

- データベース・バッファ・キャッシュ

データベース・バッファ・キャッシュは、「データベース・バッファ・キャッシュの構成」で説明されているように、DB_CACHE_SIZE初期化パラメータを使用してサイズ設定されます。

- 共有プール

  共有プールは、「共有プールの構成」で説明されているように、SHARED_POOL_SIZE初期化パラメータを使用してサイズ設定されます。

- ラージ・プール

  ラージ・プールは、「ラージ・プールの構成」で説明されているように、LARGE_POOL_SIZE初期化パラメータを使用してサイズ設定されます。

- Javaプール

  Javaプールは、JAVA_POOL_SIZE初期化パラメータを使用してサイズ設定されます。

- Streamsプール

  Streamsプールは、STREAMS_POOL_SIZE初期化パラメータを使用してサイズ設定されます。

- IM列ストア

  IM列ストアは、INMEMORY_SIZE初期化パラメータを使用してサイズ設定されます。

これらのパラメータの値は、ALTER SYSTEM文を使用して動的に構成することも可能です。

これらのSGAコンポーネントのサイズを構成する前に、次の内容を考慮してください。

- SGAのサイズ設定単位
- SGAの最大サイズ
- アプリケーションの考慮事項
- オペレーティング・システムのメモリー使用量
- 構成での繰返し

関連項目:

- Javaのメモリー使用量およびJAVA_POOL_SIZE初期化パラメータの詳細は、『Oracle Database Java開発者ガイド』を参照してください
- INMEMORY_SIZE初期化パラメータの詳細は、Oracle Database In-Memoryガイドを参照してください

SGAのサイズ設定単位

バッファ・キャッシュ、共有プール、ラージ・プールおよびJavaプールのメモリーは、グラニュル単位で割り当てられます。SGAのサイズが1GBより少ない場合、グラニュル・サイズは4MBです。SGAサイズが1GBを超えている場合、グラニュル・サイズは16MBに変化します。グラニュル・サイズは、データベース・インスタンスの起動時に計算されて固定されます。このサイズは、インスタンスの存続期間中は変化しません。

SGAで現在使用されているグラニュル・サイズを表示するには、V$SGA_DYNAMIC_COMPONENTSビューを使用します。それと同じグラニュルのサイズがSGAのすべての動的コンポーネントで使用されます。

SGAの最大サイズ

データベース・インスタンスで使用できる最大メモリー量は、インスタンス起動時にSGA_MAX_SIZE初期化パラメータで決定されます。SGAの総サイズは、SGA_MAX_SIZEパラメータの値まで拡張できます。SGA_MAX_SIZEパラメータの値は、すべてのSGAコンポーネントの集計にデフォルト設定されています。

SGA_MAX_SIZEパラメータの値が設定されていない場合は、必要であれば、1つのキャッシュのサイズを減らして、そのメモリーを別のキャッシュに再割当てします。または、SGA_MAX_SIZEパラメータの値を、バッファ・キャッシュや共有プールなど、すべてのSGAコンポーネントの合計より大きく設定することも可能です。こうすることにより、別のキャッシュ・サイズを減らさずに、キャッシュ・サイズを動的に増加できます。

ノート:

SGA_MAX_SIZE パラメータの値は、動的にサイズ変更できません。

アプリケーションの考慮事項

メモリーを構成する場合は、アプリケーションの必要性に基づいて、メモリー・キャッシュを適切にサイズ設定してください。逆に、アプリケーションのメモリー・キャッシュの使用率をチューニングすると、リソース要件を大幅に削減できます。メモリー・キャッシュを効率的に使用すると、ラッチ、CPU、I/Oシステムなどの関連リソースに対する負荷も軽減できます。

最適なパフォーマンスを得るために、次のことを考慮してください。

- オペレーティング・システムおよびデータベース・リソースを、最も効率的に使用するようキャッシュを設計してください。
- アプリケーションの必要性を最もよく反映するように、Oracle Databaseメモリー構造にメモリーを割り当ててください。
- 既存のアプリケーションに変更または追加を行う場合は、変更されたアプリケーションの必要性に対応するようOracle Databaseのメモリー構造のサイズを変更してください。
- アプリケーションがJavaを使用する場合、Javaプールのデフォルト構成を変更する必要があるかどうかを調べてください。

関連項目:

Javaのメモリー使用量の詳細は、『Oracle Database Java開発者ガイド』を参照してください。

オペレーティング・システムのメモリー使用量

大半のオペレーティング・システムでは、メモリーを構成する際に次のことを考慮することが重要です。

- ページングの削減
- メイン・メモリーへのSGAの格納
- 個々のユーザーへの十分なメモリーの割当て

関連項目:

オペレーティング・システムのメモリー使用方法のチューニングの詳細は、オペレーティング・システムのハードウェアとソフトウェアのマニュアル、およびオペレーティング・システム固有のOracleマニュアルを参照してください。

ページングの削減

ページングは、新しいページをメモリーにロードするため、オペレーティング・システムがメモリー常駐ページをディスクに転送する場合に行われます。多くのオペレーティング・システムは、実メモリーに格納しきれない大量の情報を収容するために、ページングを行います。大半のオペレーティング・システムでは、ページングはパフォーマンスを低下させます。

ホスト・システムで大量のページングが発生しているかどうかを判断するには、オペレーティング・システムのユーティリティを使用して、オペレーティング・システムを調べます。大量のページングが発生している場合は、メモリーが割り当てられているメモリー・キャッシュを保持できるほど、合計システム・メモリーが大きくない可能性があります。システムの総メモリー量を増加するか、割り当てられているメモリー量を低減することを検討してください。

メイン・メモリーへのSGAの格納

SGAの目的は、迅速なアクセスのためにメモリー内にデータを格納することであるため、SGAはメイン・メモリー内に存在する必要があります。SGAのページがディスクにスワップされると、データに迅速にアクセスできなくなります。多くのオペレーティング・システムでは、ページングによる損失は、大規模なSGAがもたらす利益をかなり上回ります。

この項では、次の項目について説明します。

- SGAメモリー割当ての表示
- 物理メモリーへのSGAのロック

SGAメモリー割当ての表示

SGAとその各内部構造に割り当てられているメモリー量を確認するには、次の例に示すように、SQL*PlusでSHOW SGA文を使用します。

```
SQL> SHOW SGA
```

この文の出力を次に示します。

```
Total System Global Area  840205000 bytes
Fixed Size                   279240 bytes
Variable Size             520093696 bytes
Database Buffers          318767104 bytes
Redo Buffers                1064960 bytes
```

物理メモリーへのSGAのロック

SGAのページ・アウトを回避するには、LOCK_SGAパラメータを有効化して、SGAを物理メモリーにロックすることを検討してください。LOCK_SGAパラメータを有効化した場合、MEMORY_TARGETおよびMEMORY_MAX_TARGETパラメータは使用されません。

個々のユーザーへの十分なメモリーの割当て

SGAをサイズ設定する場合は、個々のサーバー・プロセスとその他のプログラムがシステム上で作動するように十分なメモリーを使用できるようにします。

構成での繰返し

メモリーの割当てを構成する場合は、アプリケーションの必要性により異なりますが、Oracle Databaseメモリー構造に使用可能なメモリーを配分します。Oracle Databaseの構造にメモリーを配分すると、Oracle Databaseが正常に動作するために必要な物理I/O量に影響を与える可能性があります。最初にメモリーを適切に構成しておくと、I/Oシステムが効果的に構成されているかどうかがわかります。

メモリー構成プロセスをひととおり実行した後で、メモリー割当てのステップを繰り返すことが必要となる可能性もあります。実行を繰り返すことによって、後のステップの変更に基づいて前のステップの調整が可能となります。たとえば、バッファ・キャッシュのサイズを小さくすると、共有プールなど別のメモリー構造のサイズを大きくできます。

共有メモリー管理の監視

表12-1に、SGAのサイズ変更操作に関する情報が表示されるビューを示します。

表12-1 共有メモリー管理ビュー

ビュー	説明
V$SGA_CURRENT_RESIZE_OPS	現在進行中の SGA のサイズ変更操作に関する情報が表示されます。
V$SGA_RESIZE_OPS	最後に完了した 800 件の SGA のサイズ変更操作に関する情報が表示されます。これには現在進行中の操作は含まれません。
V$SGA_DYNAMIC_COMPONENTS	SGA の動的コンポーネントに関する情報が表示されます。このビューでは、インスタンスの起動後に発生したすべての実行済 SGA サイズ変更操作に関する情報が要約されます。
V$SGA_DYNAMIC_FREE_MEMORY	将来の動的な SGA サイズ変更操作に使用可能な SGA メモリーの容量に関する情報が表示されます。

関連項目:

これらのビューについては、『Oracle Databaseリファレンス』を参照してください。

インメモリー列ストアによる問合せパフォーマンスの改善

インメモリー列ストア(IM列ストア)は任意のシステム・グローバル領域(SGA)です。ここには、表、パーティションおよびその他のデータベース・オブジェクトのコピーが列形式で格納されています。この列データは迅速にスキャンできるように最適化されています。IM列ストアはデータベース・オブジェクトをメモリー内に格納するため、ディスク上に格納されたデータに対してスキャン、問合せ、結合、およびそのデータの集計を実行した場合と比較して、Oracle Databaseはこれらの操作をはるかに速く実行できます。

ノート:

- IM 列ストアおよびデータベース・バッファ・キャッシュには、同じデータが異なるフォーマットで保存されます。IM 列ストアはデータベース・バッファ・キャッシュの行ベースのストレージを置き換えるものではなく、問合せのパフォーマンスを改善するための補足的な役割を果たします。
- IM 列ストアは、Oracle Database 12c リリース 1(12.1.0.2)から使用可能です。

関連項目:

IM列ストアの詳細は、Oracle Database In-Memoryガイドを参照してください

Memoptimizeされた行ストアを使用した高パフォーマンス・データ・ストリーミングの有効化

Memoptimizeされた行ストアを使用すると、モノのインターネット(IoT)などのアプリケーションで高パフォーマンス・データ・ストリーミングが可能になります。

この項では、次の項目について説明します。

- Memoptimizeされた行ストアについて
- 高速収集の使用
- 高速参照の使用

Memoptimizeされた行ストアについて

Memoptimizeされた行ストアを使用すると、モノのインターネット(IoT)アプリケーションなどのアプリケーションで高パフォーマンス・データ・ストリーミングが可能になります。通常、IoTでは、多数のクライアントからの単一行挿入の少量のデータを同時にストーリミングし、非常に高頻度でクライアントのデータを問い合せます。

Memoptimizeされた行ストアには次の機能があります。

- 高速収集

  高速収集では、データベースへの高頻度の単一行データの挿入処理が最適化されます。高速収集では、ディスクに書き込む前に挿入をバッファリングするためにラージ・プールが使用され、データ挿入パフォーマンスが向上します。

- 高速参照

  高速参照を使用すると、高頻度の問合せのためにデータベースからデータを高速取得できます。高速参照では、表から問い合せたデータをバッファリングするためにMemoptimizeプールと呼ばれるSGA内の別のメモリー領域が使用され、問合せのパフォーマンスが向上します。

ノート:

高速参照を使用する場合は、MEMOPTIMIZE_POOL_SIZE 初期化パラメータを使用して、Memoptimize プールに適切なメモリー・サイズを割り当てる必要があります。

関連項目:

- 高速収集の使用
- 高速参照の使用

高速収集の使用

高速収集では、モノのインターネット(IoT)アプリケーションなどのアプリケーションからデータベースに高頻度で単一行データを挿入する処理が最適化されます。

高速収集では、MEMOPTIMIZE_WRITEヒントを使用して、MEMOPTIMIZE FOR WRITEとして指定された表にデータを挿入します。データベースは、これらの挿入をラージ・プールに一時的にバッファリングし、これらのバッファリングされた挿入をディスクに書き込むときに変更を自動的にコミットします。この変更はロールバックできません。

高速収集を使用した挿入は、遅延挿入とも呼ばれます。これは、最初はラージ・プールにバッファリングされ、後でバックグラウンド・プロセスによって非同期にディスクに書き込まれるためです。

表にデータを挿入するために高速収集を使用するステップ

表にデータを挿入するために高速収集を使用するステップは、次のとおりです。

1. 表で高速収集を有効にするには、CREATE TABLE文またはALTER TABLE文でMEMOPTIMIZE FOR WRITE句を指定します。

   ```
   SQL> create table test_fast_ingest (
   id number primary key,
   test_col varchar2(15))
   segment creation immediate
   memoptimize for write;
   Table created.
   ```

   詳細は、高速収集用の表の有効化を参照してください。

2. INSERT文でMEMOPTIMIZE_WRITEヒントを指定することで、挿入での高速収集を有効にします。

   次に、高速収集について、使用方法ではなくメカニズムを説明します。

   ```
   SQL> insert /*+ memoptimize_write */ into test_fast_ingest values (1, 'test');
   1 row created
   SQL> insert /*+ memotimize_write */ into test_fast_ingest values (2, 'test');
   1 row created
   ```

   詳細は、データ挿入に高速収集を使用するためのヒントの指定を参照してください。

前述の2つの挿入の結果として、SGAのラージ・プール内の収集バッファにデータが書き込まれます。ある時点で、そのデータがTEST_FAST_INGEST表にフラッシュされます。それが起こるまでは、そのデータは永続的ではありません。

高速収集の目的は高パフォーマンスなデータ・ストリーミングをサポートすることであるため、より現実的なアーキテクチャには、1つ以上のアプリケーションまたは収集サーバーでデータを収集してデータベースに挿入をバッチ処理することが含まれます。

初めて挿入を実行すると、ラージ・プールから高速収集領域が割り当てられます。割り当てられたメモリーの量は、alert.logに書き込まれます。

```
Memoptimize Write allocated 2055M from large pool
```

リクエストで最小メモリー要件さえ割当てに失敗した場合、エラー・メッセージがalert.logに書き込まれます。

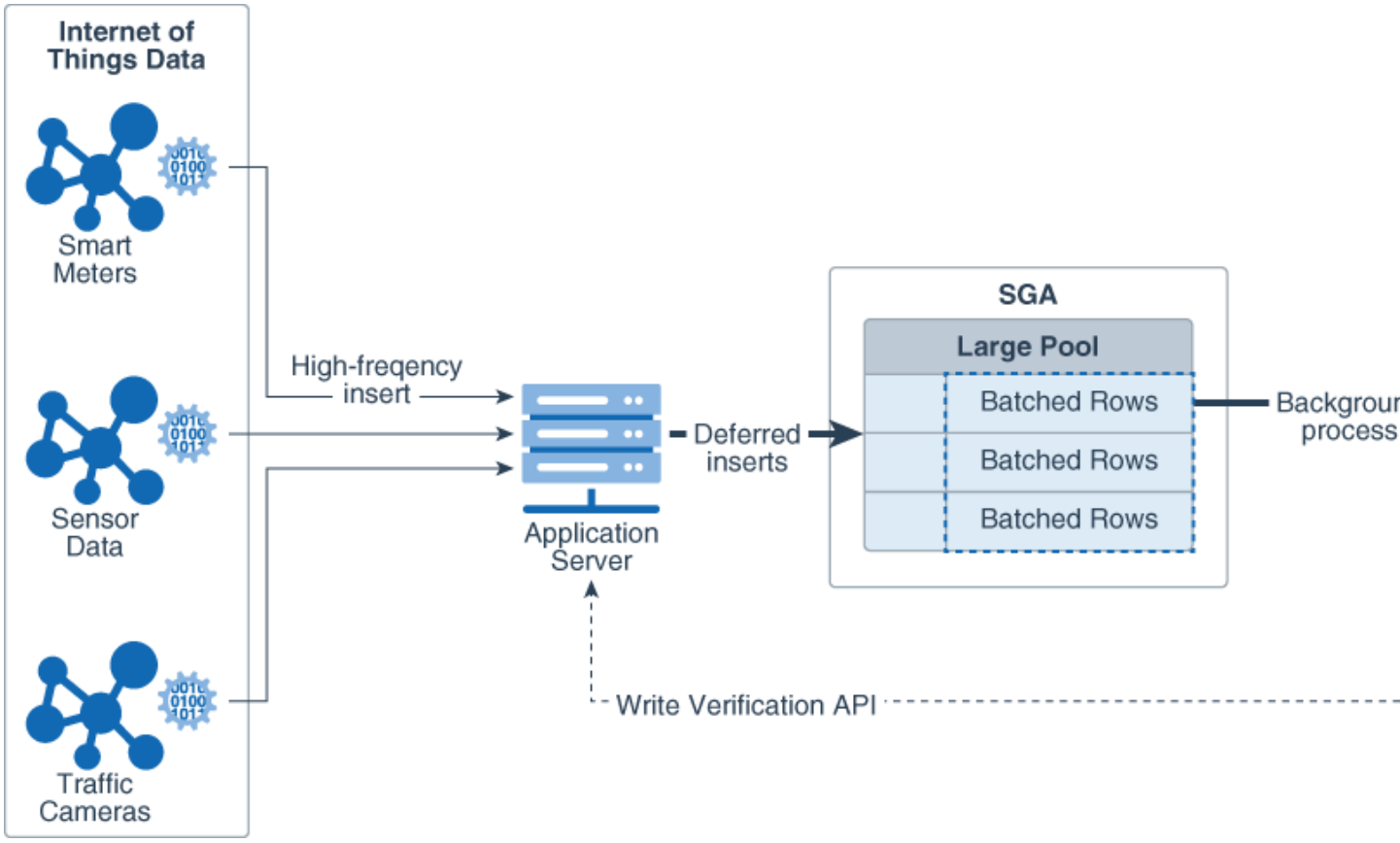
```
Memoptimize Write disabled.  Unable to allocate sufficient memory from large pool.
```

高速収集の詳細

高速収集の目的は、集計に重要な値が含まれている多くの情報データを生成するが必ずしもすべてのACID保証が必要なわけではないアプリケーションをサポートすることです。モノのインターネット(IoT)における多くのアプリケーションには、センサー・データ、スマート・メーター・データまたはトラフィック・カメラなど、時間のかからないボタンを押すだけでよいタイプのワークロードがあります。これらのアプリケーションでは、データは、収集され、その後の分析のために大量にデータベースに書き込まれる場合があります。

次の図では、Memoptimizeされた行ストアの高速収集機能がこのようなケースでどのように使用されるかを示します。

図12-1 高頻度の挿入での高速収集



収集されたデータはラージ・プールにバッチされ、すぐにはデータベースに書き込まれません。このため、収集プロセスは非常に高速になります。個々の行を処理する必要なく、非常に大量のデータを効率的に収集できます。ただし、収集されたデータがデータベース・ファイルに書き込まれる前にデータベースが停止すると、データが失われる可能性があります。

高速収集は、通常のOracle Databaseトランザクション処理とはまったく異なります。この機能では、データはログ記録され、データベースに一旦書き込まれると(つまり、コミットされると)失われることがありません。最大限の収集処理能力を実現するために、通常のOracleトランザクション・メカニズムは無視されます。すべてのデータが実際にデータベースに書き込まれているかどうかは、そのアプリケーションで確認する必要があります。データがデータベースに書き込まれたかどうかを確認するために呼び出すことができる、特別なAPIが追加されました。

高速収集のコンテキストの場合、commit操作は意味を持ちません。これはOracleにおける従来の意味でのトランザクションではないためです。挿入をロールバックする機能はありません。高速収集バッファからディスクにフラッシュされるまで、データの問合せもできません。ビューV$MEMOPTIMIZE_WRITE_AREAを問い合せることで、高速収集バッファに関する管理情報を確認できます。

また、パッケージDBMS_MEMOPTIMIZEおよびDBMS_MEMOPTIMIZE_ADMINを使用して、ラージ・プールから高速収集データをフラッシュする機能や、フラッシュされたデータの順序IDを決定する機能などを実行できます。

索引操作および制約チェックは、データがラージ・プール内の高速収集領域からディスクに書き込まれるときにのみ実行されます。バックグラウンド・プロセスでデータがディスクに書き込まれるときに主キー違反が発生した場合は、データベースによってこれらの行がデータベースに書き込まれることはありません。

すべてではありませんがほとんどのアプリケーションについては、挿入したすべてのデータをデータベースに書き込む必要があると仮定すると、アプリケーションの挿入プロセスで、挿入したデータが実際にデータベースに書き込まれたことを確認してからそのデータを破棄するようにすることが重要です。その確認が行われた場合のみ、挿入側プロセスからデータを削除できます。

関連項目:

- 高速収集表の前提条件
- 表への高速収集の有効化
- データ挿入に高速収集を使用するためのヒントの指定
- ラージ・プール内の高速収集データの管理
- 表への高速収集の無効化
- 遅延挿入メカニズムの詳細は、『Oracle Database概要』を参照してください

高速収集表の前提条件

特定の特性やオブジェクト、パーティション化がある表では、高速収集はサポートされません。

自律型データベースでアイテムがサポートされ、その制限がない場合はその旨が記載されています。

- 次の特性を持つ表では高速収集を使用できません。
    - ディスク圧縮
    - インメモリー圧縮
    - ファンクション索引
    - ドメイン索引
    - ビットマップ索引

    - ビットマップ結合索引
    - ref型
    - varray型
    - OID$型
    - 未使用列
    - LOB
    - トリガー
    - バイナリ列
    - 外部キー
    - 行のアーカイブ
    - 非表示列[自律型データベースは仮想列をサポートしています。]
- 次のオブジェクトでは高速収集を使用できません。
    - 一時表
    - ネストした表
    - 索引構成表
    - 外部表
    - オンデマンド・リフレッシュがあるマテリアライズド・ビュー
    - サブパーティション化はサポートされていません。[自律型データベースはサブパーティション化をサポートしています。]
- 次のパーティション化タイプはサポートされていません。
    - REFERENCE
    - SYSTEM
    - INTERVAL [自律型データベースはこれをサポートしています。]
    - AUTOLIST [自律型データベースはこれをサポートしています。]

高速収集に関するその他の考慮事項を次に示します。

- すべてではありませんがほとんどのアプリケーションについては、挿入したすべてのデータをデータベースに書き込む必要があると仮定すると、アプリケーションの実装プロセスで、挿入したデータが実際にデータベースに書き込まれたことを確認してからそのデータを破棄するようにすることが重要です。その確認が行われた場合のみ、挿入側プロセスからデータを削除できます。
- 高速収集では、ラージ・プールにデータをバッファリングするため、システム障害の発生時にデータが失われる可能性があります。データの損失を回避するには、データがディスクに書き込まれる前にシステム障害が発生した場合に挿入を再現できるように、クライアントは挿入実行後にデータのローカル・コピーを保持する必要があります。クライアントは、DBMS_MEMOPTIMIZEパッケージのサブプログラムを使用して、挿入の永続性を追跡できます。挿入がディスクに書き込まれたら、クライアントは挿入されたデータのローカル・コピーを破棄できます。
- 問合せはラージ・プールからデータを読み取らないため、高速収集を使用して挿入されたデータはディスクに書き込まれるまで問合せできません。
- 索引操作は、通常の挿入と同様に高速収集によってサポートされます。ただし、高速収集の場合、データベースはデータをディスクに書き込むときに索引操作を実行し、ラージ・プールにデータを書き込むときには実行しません。
- ラージ・プール内の高速収集バッファに割り当てられたサイズは、作成後は固定となります。バッファがいっぱいになると、バックグラウンド・プロセスでバッファが排出されるまで、それ以降の収集は待機状態になります。

ノート:

高速収集と高速参照の両方を使用するように表を構成できます。

表への高速収集の有効化

表で高速収集を有効にするには、CREATE TABLE文またはALTER TABLE文でMEMOPTIMIZE FOR WRITE句を指定します。

表で高速収集を有効にするには:

1. SQL*Plusで、ALTER TABLE権限を持つユーザーとしてデータベースにログインします。
2. MEMOPTIMIZE FOR WRITE句を指定したCREATE TABLEまたはALTER TABLE文を実行します。

   次の例では、新しい表test_fast_ingestを作成し、高速収集を有効にしています。

   ```
   CREATE TABLE test_fast_ingest (
        id        NUMBER(5) PRIMARY KEY,
        test_col  VARCHAR2(15))
     SEGMENT CREATION IMMEDIATE
     MEMOPTIMIZE FOR WRITE;
   ```

   次の例では、既存の表hr.employeesで高速収集を有効にしています。

   ```
   ALTER TABLE hr.employees MEMOPTIMIZE FOR WRITE;
   ```

データ挿入に高速収集を使用するためのヒントの指定

INSERT文でMEMOPTIMIZE_WRITEヒントを指定すると、データ挿入に高速収集を使用できます。

前提条件

このタスクでは、次のことを前提としています。

- 表ですでに高速収集が有効になっています。
- オプティマイザでヒントの使用が許可されています(optimizer_ignore_hints=FALSE)。

データ挿入に高速収集を使用するには:

1. SQL*Plusで、データを表に挿入する権限を持つユーザーとしてデータベースにログインします。
2. 高速収集がすでに有効になっている表に対して、MEMOPTIMIZE_WRITEヒントを指定してINSERT文を実行します。

   次に例を示します。

   ```
   INSERT  /*+ MEMOPTIMIZE_WRITE */ INTO test_fast_ingest VALUES (1,'test');
   ```

関連項目:

表への高速収集の有効化

表への高速収集の無効化

ALTER TABLE文でNO MEMOPTIMIZE FOR WRITE句を指定すると、表で高速収集を無効にできます。

表で高速収集を無効にするには:

1. SQL*Plusで、ALTER TABLE権限を持つユーザーとしてデータベースにログインします。
2. NO MEMOPTIMIZE FOR WRITE句を指定したALTER TABLE文を実行します。

   次の例では、表hr.employeesで高速収集を無効にしています。

   ```
   ALTER TABLE hr.employees NO MEMOPTIMIZE FOR WRITE;
   ```

ラージ・プール内の高速収集データの管理

V$MEMOPTIMIZE_WRITE_AREAビューを使用すると、ラージ・プール内の高速収集データを表示できます。パッケージDBMS_MEMOPTIMIZEおよびDBMS_MEMOPTIMIZE_ADMINのサブプログラムを使用して、ラージ・プール内の高速収集データを表示および制御することもできます。

V$MEMOPTIMIZE_WRITE_AREAビューの概要

V$MEMOPTIMIZE_WRITE_AREAビューは、高速収集によるラージ・プールでのメモリー使用量およびデータ挿入に関する次の情報を提供します。

- ラージ・プール内で高速収集データ用に割り当てられているメモリー量の合計
- ラージ・プール内で高速収集データに現在使用されているメモリー量の合計
- ラージ・プールに高速収集データを格納するために現在空いているメモリー量の合計
- データがラージ・プールにあり、まだディスクに書き込まれていない高速収集の挿入操作の数
- データベースにデータを挿入するために現在高速収集を使用しているクライアントの数

関連項目:

V$MEMOPTIMIZE_WRITE_AREAビューの詳細は、『Oracle Databaseリファレンス』を参照してください

DBMS_MEMOPTIMIZEパッケージのサブプログラムの概要

DBMS_MEMOPTIMIZEパッケージの次のサブプログラムを使用して、ラージ・プールの高速収集データを表示および制御できます。

サブプログラム	説明
GET_APPLY_HWM_SEQID	すべてのセッションでディスクに正常に書き込まれたデータ・レコードの低い最高水位標

サブプログラム	説明
	(低いHWM)の順序番号を返します。
GET_WRITE_HWM_SEQID	現在のセッションでラージ・プールに書き込まれたデータ・レコードの最高水位標(HWM)の順序番号を返します。
WRITE_END	現行セッションのすべての高速収集データをラージ・プールからディスクにフラッシュします。

関連項目:

DBMS_MEMOPTIMIZEパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照してください

DBMS_MEMOPTIMIZE_ADMINパッケージのサブプログラムの概要

DBMS_MEMOPTIMIZE_ADMINパッケージの次のサブプログラムを使用して、ラージ・プールの高速収集データを制御できます。

サブプログラム	説明
WRITES_FLUSH	すべてのセッションのすべての高速収集データをラージ・プール

サブプログラム	説明
	からディスクにフラッシュします。

関連項目:

DBMS_MEMOPTIMIZE_ADMINパッケージの詳細は、『Oracle Database PL/SQLパッケージおよびタイプ・リファレンス』を参照してください

高速参照の使用

高速参照により、モノのインターネット(IoT)アプリケーションなどのアプリケーションでデータベース表からの高速データ取得が可能になります。

高速参照では、Memoptimizeプールと呼ばれるSGAバッファ領域に格納されているハッシュ索引を使用し、バッファ・キャッシュに永続的に固定されている表のブロックに高速にアクセスできるため、ディスクI/Oが回避され、問合せのパフォーマンスが向上します。

表の高速参照を使用するステップ

表の高速参照を使用するステップは、次のとおりです。

1. Memoptimizeプールの有効化

   このタスクは、MEMOPTIMIZE_POOL_SIZE初期化パラメータをゼロ以外の値に設定することによって実現されます。

   詳細は、Memoptimizeプールの有効化を参照してください。

2. 表の高速参照の有効化

   このタスクを実現するには、CREATE TABLE文またはALTER TABLE文でMEMOPTIMIZE FOR READ句を指定します。

   詳細は、表の高速参照の有効化を参照してください。

高速参照の使用に関する制限事項

高速参照の使用に関する制限事項を以下に示します。

- 高速参照が有効な表は圧縮できません。
- 高速参照が有効な表には主キー制約を有効にする必要があります。

ノート:

高速収集と高速参照の両方を使用するように表を構成できます。

関連項目:

- Memoptimizeプールの有効化
- 表の高速参照の有効化
- 表の高速参照の無効化
- Memoptimizeプールの高速参照データの管理
- Memoptimizeプールのメモリー・アーキテクチャの詳細は、Oracle Database概要を参照してください
- MEMOPTIMIZE_POOL_SIZE初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください

Memoptimizeプールの有効化

高速参照を使用する前に、Memoptimizeプールを有効にする必要があります。MemoptimizeプールはSGAに存在し、高速参照が有効にされた表のデータとハッシュ索引を格納します。

前提条件

このタスクは、COMPATIBLE初期化パラメータが18.0.0以上に設定されていることを想定しています。

Memoptimizeプールを有効にするには:

1. SQL*Plusで、管理権限を持つユーザーとしてデータベースにログインします。
2. MEMOPTIMIZE_POOL_SIZE初期化パラメータをゼロ以外の値に設定します。最小設定は100 MBです。ALTER SYSTEM文を使用してサーバー・パラメータ・ファイル(SPFILE)でこの初期化パラメータを設定する場合、SCOPE=SPFILEを指定する必要があります。

   たとえば、次の文では、Memoptimizeプールのサイズを10 GBに設定します。

   ```
   ALTER SYSTEM SET MEMOPTIMIZE_POOL_SIZE = 10G SCOPE=SPFILE;
   ```

3. データベースを再起動して、変更を有効にします。

例: Memoptimizeプールの有効化

MEMOPTIMIZE_POOL_SIZE初期化パラメータが0に初期設定されていることが前提です。次の例では、MEMOPTIMIZE_POOL_SIZEを10 GBに設定することで、Memoptimizeプールを有効にしています。

```
SQL> SHOW PARAMETER MEMOPTIMIZE_POOL_SIZE
NAME                    TYPE         VALUE
----------------------  -----------  -----
memoptimize_pool_size   big integer  0
SQL> ALTER SYSTEM SET MEMOPTIMIZE_POOL_SIZE=10G SCOPE=SPFILE;
System altered.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
Total System Global Area 1.1832E+10 bytes
Fixed Size                  9010864 bytes
Variable Size            1.1799E+10 bytes
Database Buffers           16777216 bytes
```

```
Redo Buffers                7766016 bytes
Database mounted.
Database opened.
SQL> SHOW PARAMETER MEMOPTIMIZE_POOL_SIZE
NAME                   TYPE         VALUE
---------------------  -----------  -----
memoptimize_pool_size  big integer  10G
```

> ノート:
>
> MEMOPTIMIZE_POOL_SIZE 値は SGA_TARGET の対象ですが、データベースは memoptimize プールを自動では拡大および縮小しません。たとえば、SGA_TARGET が 10 GB で、 MEMOPTIMIZE_POOL_SIZE が 1 GB の場合、使用可能な SGA メモリーは memoptimize プール以外の合計 9 GB です。

関連項目:

- Memoptimizeプールのメモリー・アーキテクチャの詳細は、Oracle Database概要を参照してください
- MEMOPTIMIZE_POOL_SIZE初期化パラメータの詳細は、Oracle Databaseリファレンスを参照してください

表の高速参照の有効化

表の高速参照を有効にするには、CREATE TABLE文またはALTER TABLE文にMEMOPTIMIZE FOR READ句を指定します。

前提条件

このタスクはMemoptimizeプールが有効であることを前提としています。

表の高速参照を有効にするには:

1. SQL*Plusで、ALTER TABLE権限を持つユーザーとしてデータベースにログインします。
2. 高速参照を有効にする必要がある表に対して、MEMOPTIMIZE FOR READ句を指定したCREATE TABLE文またはALTER TABLE文を実行します。

   次の例では、新しい表test_fast_lookupを作成し、高速参照を有効にしています。

   ```
   CREATE TABLE test_fast_lookup (
       id        NUMBER(5) PRIMARY KEY,
       test_col  VARCHAR2(15))
     MEMOPTIMIZE FOR READ;
   ```

   次の例では、既存のhr.employees表の高速参照が有効になります。

   ```
   ALTER TABLE hr.employees MEMOPTIMIZE FOR READ;
   ```

関連項目:

- Memoptimizeプールの有効化
- 表の高速参照の無効化

- Memoptimizeプールの高速参照データの管理

表の高速参照の無効化

表の高速参照を無効にするには、ALTER TABLE文にNO MEMOPTIMIZE FOR READ句を指定します。

前提条件

このタスクでは、表で高速参照がすでに有効にされていることを前提としています。

表の高速参照を無効にするには:

1. SQL*Plusで、ALTER TABLE権限を持つユーザーとしてデータベースにログインします。
2. 高速参照を無効にする必要がある表に対して、NO MEMOPTIMIZE FOR READ句を指定してALTER TABLE文を実行します。

   次の例では、hr.employees表の高速参照を無効にしています。

   ```
   ALTER TABLE hr.employees NO MEMOPTIMIZE FOR READ;
   ```

関連項目:

表の高速参照の有効化

Memoptimizeプールの高速参照データの管理

Memoptimizeプールには、高速参照が有効にされているすべての表のデータ(高速参照データ)が格納されます。DBMS_MEMOPTIMIZEパッケージのサブプログラムを使用して、Memoptimizeプールの表の高速参照データを明示的に削除または移入できます。

DBMS_MEMOPTIMIZEパッケージのサブプログラムの概要

Memoptimizeプール内の表の高速参照データを削除または移入するために使用できるDBMS_MEMOPTIMIZEパッケージのサブプログラムを次に示します。

サブプログラム	説明
DROP_OBJECT	Memoptimize プールから表の高速参照データを削除します。
POPULATE	Memoptimize プールに表の高速参照データを移入し

サブプログラム	説明
	ます。

関連項目:

- 表の高速参照の有効化
- Memoptimizeプールの有効化
- DBMS_MEMOPTIMIZEパッケージの詳細は、Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンスを参照してください
- Memoptimizeプールのメモリー・アーキテクチャの詳細は、Oracle Database概要を参照してください

13 データベース・バッファ・キャッシュのチューニング

この章では、データベース・バッファ・キャッシュのチューニング方法を説明します。自動メモリー管理を使用してシステムのデータベース・メモリーを管理している場合は、この章で説明されている、手動でのメモリー・キャッシュのチューニングを行う必要ありません。

この章のトピックは、次のとおりです:

- データベース・バッファ・キャッシュについて
- データベース・バッファ・キャッシュの構成
- 複数バッファ・プールの構成
- REDOログ・バッファの構成
- データベース・キャッシュ・モードの構成

データベース・バッファ・キャッシュについて

様々なタイプの操作について、Oracle Databaseではバッファ・キャッシュを使用してディスクから読み取られたデータ・ブロックを格納します。ソートやパラレル読取りなどの特定の操作の場合には、Oracle Databaseではバッファ・キャッシュはバイパスされます。

データベース・バッファ・キャッシュを効果的に使用するには、不要なリソース使用を回避するようにアプリケーションのSQL文をチューニングします。この目的を達成するには、頻繁に実行されるSQL文と、多数のバッファ読取りを実行するSQL文が適切にチューニングされていることを検証します。

パラレル問合せを使用する場合は、直接プログラム・グローバル領域(PGA)に読み込むのではなく、データベース・バッファ・キャッシュを使用するようにデータベースを構成することを検討します。この構成は、システムのメモリー容量が大きい場合に適しています。

関連項目:

- SQL文のチューニングの詳細は、Oracle Database SQLチューニング・ガイドを参照してください
- パラレル実行の詳細は、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください

データベース・バッファ・キャッシュの構成

新規にデータベース・インスタンスを構成する場合は、バッファ・キャッシュの適切なサイズがわかっていません。通常、データベース管理者はキャッシュ・サイズの最初の見積りを行い、次にインスタンス上で代表的なワークロードを実行し、関連する統計を調べて、キャッシュが過小構成か過大構成かを確認します。

この項では、データベース・バッファ・キャッシュの構成方法を説明します。自動共有メモリー管理を使用して共有グローバル領域(SGA)を構成する場合は、この章で説明されているように、手動でのデータベース・バッファ・キャッシュのチューニングを行う必要

ありません。

この項では、次の項目について説明します。

- V$DB_CACHE_ADVICEビューの使用
- バッファ・キャッシュ・ヒット率の計算
- バッファ・キャッシュ・ヒット率の解釈
- データベース・バッファ・キャッシュに割り当てられたメモリーの増加
- データベース・バッファ・キャッシュに割り当てられたメモリーの削減

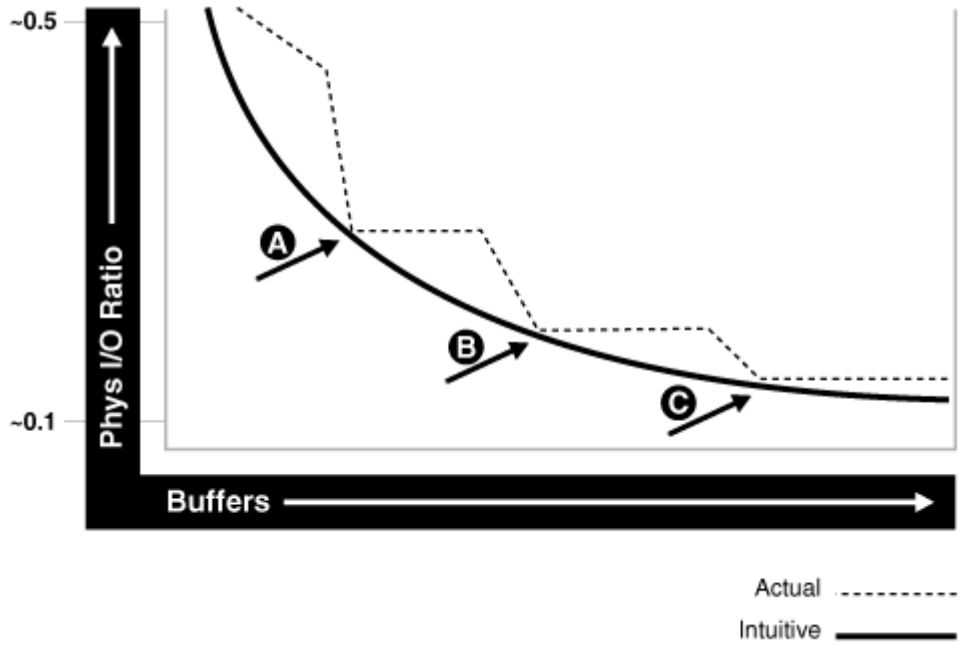
V$DB_CACHE_ADVICEビューの使用

V$DB_CACHE_ADVICEビューは、潜在的なバッファ・キャッシュ・サイズ範囲のシミュレーションによるミス率を示します。このビューは、潜在的な各キャッシュ・サイズの物理読取り数を予測する情報を提供して、キャッシュのサイズ設定を支援します。このデータには物理読取り係数も含まれています。これは、バッファ・キャッシュが所定の値にサイズ変更された場合、現行の物理読取り回数がその分のみ変化すると予測される係数です。

ただし、物理読取りはファイル・システム・キャッシュからの読取りで完了するため、物理読取りが必ずしもOracle Databaseのディスク読取りを意味するわけではありません。このため、キャッシュ内でのブロックの検出成功とキャッシュのサイズ間の関係は、必ずしも滑らかな分布を示しません。バッファ・プールをサイズ設定するときは、キャッシュ・ヒット率の向上に貢献しない(または、ほとんど貢献しない)追加バッファは使用しないでください。

次の図に、物理I/O率とバッファ・キャッシュ・サイズの関係を示します。

図13-1 物理I/O率とバッファ・キャッシュ・サイズ



前述の図に示されている例を調べると、次のことがわかります。

- バッファの数が増加すると、物理I/O率が減少します。
- ポイントAとBおよびポイントBとCとの間の物理I/Oの減少は、グラフの点線で示されるように滑らかではありません。

- ポイントAからポイントBへバッファを増やす場合の利点は、ポイントBからポイントCへバッファを増やす場合よりかなり大きくなります。
- バッファの数が増えるに従い、バッファを増やすメリットが低減します。

このアドバイザ・ビューの使用には、多少のオーバーヘッドが伴います。アドバイザを有効にすると、追加の記録が必要なため、CPUの使用量はわずかに増加します。ブックキーピングに関連するCPUおよびメモリーのオーバーヘッドの両方を削減するために、Oracle Databaseでは、サンプリングを行ってキャッシュ・アドバイザ統計を収集します。サンプリングは、開始時のバッファの数が少ないバッファ・プールでは使用しません。

V$DB_CACHE_ADVICEビューを使用するには:

1. DB_CACHE_ADVICE初期化パラメータの値をONに設定します。

   これにより、アドバイザ・ビューが有効になります。DB_CACHE_ADVICEパラメータは動的であるため、特定のワークロードのアドバイザ・データを収集できるように、アドバイザを動的に有効にしたり、無効にできます。

2. データベース・インスタンスで代表的なワークロードを実行します。

   V$DB_CACHE_ADVICEビューを問い合せる前にワークロードを安定化できるようにします。

3. V$DB_CACHE_ADVICEビューを問い合せます。

次の例に、様々なキャッシュ・サイズについてデフォルト・バッファ・プールに対するI/O要件の予測を戻すこのビューの問合せを示します。

```
COLUMN size_for_estimate          FORMAT 999,999,999,999 heading 'Cache Size (MB)'
COLUMN buffers_for_estimate       FORMAT 999,999,999 heading 'Buffers'
COLUMN estd_physical_read_factor  FORMAT 999.90 heading 'Estd Phys|Read Factor'
COLUMN estd_physical_reads        FORMAT 999,999,999 heading 'Estd Phys| Reads'
SELECT size_for_estimate, buffers_for_estimate, estd_physical_read_factor,
       estd_physical_reads
FROM   V$DB_CACHE_ADVICE
WHERE  name = 'DEFAULT'
  AND  block_size = (SELECT value FROM V$PARAMETER WHERE name = 'db_block_size')
  AND  advice_status = 'ON';
```

この問合せの出力例を次に示します。

```
                                Estd Phys    Estd Phys
 Cache Size (MB)      Buffers Read Factor        Reads
---------------- ------------ ----------- ------------
              30        3,802       18.70  192,317,943      10% of Current Size
              60        7,604       12.83  131,949,536
              91       11,406        7.38   75,865,861
             121       15,208        4.97   51,111,658
             152       19,010        3.64   37,460,786
             182       22,812        2.50   25,668,196
             212       26,614        1.74   17,850,847
             243       30,416        1.33   13,720,149
             273       34,218        1.13   11,583,180
             304       38,020        1.00   10,282,475      Current Size
             334       41,822         .93    9,515,878
             364       45,624         .87    8,909,026
             395       49,426         .83    8,495,039
             424       53,228         .79    8,116,496
             456       57,030         .76    7,824,764
             486       60,832         .74    7,563,180
```

```
       517      64,634        .71    7,311,729
       547      68,436        .69    7,104,280
       577      72,238        .67    6,895,122
       608      76,040        .66    6,739,731      200% of Current Size
```

この例の出力は、キャッシュが現行サイズの304MBではなく212MBである場合、物理読取りの予測数が1.74倍、つまり74%増加することを示しています。そのため、キャッシュ・サイズを212MBに減少させることは望ましくありません。

ただし、キャッシュ・サイズを334MBに増やすと、読取り数は0.93倍、つまり7%減少することになります。システム上でさらに30MBのメモリーを使用可能で、SGA_MAX_SIZEパラメータの値から増分が可能な場合は、デフォルトのバッファ・キャッシュ・プール・サイズを334MBに増やすことをお薦めします。

バッファ・キャッシュ・ヒット率の計算

バッファ・キャッシュ・ヒット率では、ディスク・アクセスを行わずにバッファ・キャッシュ内で要求されたブロックが検出された頻度を計算します。この率は、V$SYSSTATパフォーマンス・ビューから選択したデータを使用して計算されます。バッファ・キャッシュ・ヒット率を使用して、V$DB_CACHE_ADVICEビューで予測されたように物理I/Oを検証します。

表13-1に、バッファ・キャッシュ・ヒット率の計算に使用したV$SYSSTATビューの統計を示します。

表13-1 バッファ・キャッシュ・ヒット率を計算するための統計

統計	説明
consistent gets from cache	バッファ・キャッシュからのブロックに対して読取り一貫性が要求された回数。
db block gets from cache	バッファ・キャッシュから CURRENT ブロックが要求された回数。
physical reads cache	ディスクからバッファ・キャッシュへ読み取られたデータ・ブロックの合計数。

例13-1に、このビューの問合せを示します。

例13-1 V$SYSSTATビューの問合せ

```
SELECT name, value
FROM V$SYSSTAT
WHERE name IN ('db block gets from cache', 'consistent gets from cache',
'physical reads cache');
```

この例では、特定期間の値は選択せず、V$SYSSTATビューから直接選択した値を使用して問合せを単純化しています。アプリケーションの実行中のある期間にわたるこれらの統計の差分を計算し、それらの差分の値を使用してバッファ・キャッシュ・ヒット率を判断することをお薦めします。ある期間の統計を収集する方法の詳細は、「自動パフォーマンス診断」を参照してください。

この問合せの出力の値を使用し、次の計算式でバッファ・キャッシュ・ヒット率を計算します。

```
1 - (('physical reads cache') / ('consistent gets from cache' +
'db block gets from cache'))
```

関連項目:

V$SYSSTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください。

バッファ・キャッシュ・ヒット率の解釈

バッファ・キャッシュ・サイズの増減を決定する前に、バッファ・キャッシュ・ヒット率を調べる必要があります。

キャッシュ・ヒット率が低いことは、バッファ・キャッシュのサイズを大きくすることがパフォーマンスに有益であることを必ずしも意味しません。また、キャッシュ・ヒット率の高いことが、ワークロードに対してバッファ・キャッシュが適切にサイズ設定されていることを示しているとはかぎりません。

バッファ・キャッシュ・ヒット率を解釈する場合は、次の点を考慮する必要があります。

- 1つのパスで処理を実行するか、SQL文を最適化して、頻繁にアクセスされるデータを繰り返しスキャンしないようにします。

  同じ大きな表や索引を繰り返しスキャンすると、キャッシュ・ヒット率を低下させる可能性があります。バッファ読取り数が多く、頻繁に実行されるSQL文を調べて、実行計画が最適なものであるか確認します。

- 頻繁にアクセスされるデータをクライアント・プログラムまたは中間層にキャッシュして、同じデータを再問合せしないようにします。
- OLTPアプリケーションを実行する大容量データベースでは、ほとんどの行は1回しか(または一度も)アクセスされません。このため、使用後にブロックをメモリーに保持していても意味がありません。
- バッファ・キャッシュ・サイズを連続して増加しないでください。

  バッファ・キャッシュ・サイズを連続して増加しても、データベースが全表スキャンやバッファ・キャッシュを使用しない操作を実行している場合は効果がありません。

- 大規模な全表スキャンが行われている場合は、ヒット率が低くなることを考慮してください。

  長い全表スキャン中にアクセスされたデータベース・ブロックは、最低使用頻度(LRU)リストの最後に配置され、リストの先頭には配置されません。そのため、これらのブロックは、索引参照または小規模な表スキャンを実行するときに読み取られるブロックよりも早く除去されます。

ノート:

小規模表のスキャンは、一定のサイズのしきい値を使用して、表に対して実行されるスキャンです。小規模表とは、最大でバッファ・キャッシュの 2%か 20 のいずれか大きい方と定義されます。

データベース・バッファ・キャッシュに割り当てられたメモリーの増加

キャッシュ・ヒット率が低く、全表スキャンを実行しないようにアプリケーションがチューニングされている場合は、バッファ・キャッシュのサイズを増やすことを検討してください。可能な場合は、インスタンスを停止せずに、バッファ・プールを動的にサイズ変更してこの変更を行います。

データベース・バッファ・キャッシュのサイズを大きくするには:

1. DB_CACHE_ADVICE初期化パラメータの値をONに設定します。
2. バッファ・キャッシュ統計の安定化を可能にします。
3. 「V$DB_CACHE_ADVICEビューの使用」の説明を参照し、V$DB_CACHE_ADVICEビュー内のアドバイザ・データを調べて、実行する物理I/Oの量を大幅に減少させるために必要な次の増分を決定します。
4. システムにページングさせずに、バッファ・キャッシュに必要なメモリーを追加で割り当てることができる場合は、このメモリーを割り当てます。
5. バッファ・キャッシュに割り当てられたメモリーの量を増やすには、DB_CACHE_SIZE初期化パラメータの値を増やします。

   DB_CACHE_SIZEパラメータは、データベースの標準ブロック・サイズのデフォルト・キャッシュのサイズを指定します。データベースの標準ブロック・サイズ以外のブロック・サイズを持つ表領域(トランスポータブル表領域など)を作成して使用するには、使用するブロック・サイズごとに個別のキャッシュを構成します。DB_nK_CACHE_SIZEパラメータを使用して、必要な標準以外のブロック・サイズを構成します(nは2、4、8、16または32のいずれかで、標準ブロック・サイズではありません)。

ノート:

- キャッシュ・サイズを選択するプロセスは、キャッシュがデフォルトの標準ブロック・サイズ・キャッシュ、KEEP または RECYCLE キャッシュ、標準以外のブロック・サイズ・キャッシュのいずれかにかかわらず同様です。
- キャッシュを大幅に(20%以上)サイズ変更すると、古いキャッシュ・アドバイザ値は破棄されて、新しいサイズに設定されます。大幅にサイズ変更しない場合は、古いキャッシュ・アドバイザ値は既存の値を補間することで新しいサイズに調整されます。

関連項目:

DB_nK_CACHE_SIZEパラメータの詳細は、次を参照してください。

- 『Oracle Database管理者ガイド』
- Oracle Databaseリファレンス

データベース・バッファ・キャッシュに割り当てられたメモリーの削減

キャッシュ・ヒット率が高い場合は、バッファ・キャッシュが十分大きく、最も頻繁にアクセスされるデータを格納できる可能性が高いです。このような場合に、別のメモリー構造にメモリーが必要なときは、バッファ・キャッシュのサイズを小さくすることを検討してください。

データベース・バッファ・キャッシュのサイズを小さくするには:

1. 「V$DB_CACHE_ADVICEビューの使用」の説明を参照し、V$DB_CACHE_ADVICEビュー内のアドバイザ・データを調べて、バッファ・キャッシュのサイズを小さくした場合に、物理I/Oの量が大幅に減少するかどうかを判断します。
2. バッファ・キャッシュに割り当てられたメモリーの量を減らすには、DB_CACHE_SIZE初期化パラメータの値を減らします。

複数バッファ・プールの構成

一般に、ほとんどのシステムでは1つのデフォルト・バッファ・プールが適切です。ただし、アプリケーションのバッファ・プールについて詳しい知識を持つデータベース管理者であれば、複数バッファ・プールを構成することが有益な場合もあります。

非定型アクセス・パターンを持つセグメントの場合、それらのセグメントからのブロックを2つの個別のバッファ・プールであるKEEPプールとRECYCLEプールに格納することを検討してください。セグメントのアクセス・パターンは、常にアクセスされるか(ホットとも呼ばれる)、ほとんどアクセスされない(1日に1回のみバッチ・ジョブでアクセスされる大きなセグメントなど)というように、非定型である可能性があります。

複数バッファ・プールを使用すると、こうした不規則性に対応できます。KEEPプールを使用してバッファ・キャッシュ内の頻繁にアクセスされるセグメントを保持し、RECYCLEプールを使用してオブジェクトがバッファ・キャッシュ内の領域を不必要に占有するのを防ぐことができます。オブジェクトがバッファ・キャッシュに関連付けられると、そのオブジェクトのすべてのブロックがそのキャッシュに配置されます。Oracle Databaseには、特定のバッファ・プールに割り当てられていないオブジェクトのために、DEFAULTバッファ・プールがあります。デフォルト・バッファ・プールのサイズは、DB_CACHE_SIZE初期化パラメータによって決まります。各バッファ・プールでは、同じLRU置換ポリシーが使用されます。たとえば、KEEPプールのサイズが十分ではなく、プールに割り当てられたすべてのセグメントを格納できない場合、最も古いブロックがキャッシュから除去されます。

オブジェクトを適切なバッファ・プールに割り当てると、次の操作を実行できます。

- I/Oの低減または排除
- 個別のキャッシュに対するオブジェクトの隔離または制限

この項では、複数バッファ・プールの構成方法を説明しており、内容は次のとおりです。

- 複数バッファ・プールを使用する際の考慮事項
- 複数バッファ・プールの使用方法
- 個別のバッファ・プールへのV$DB_CACHE_ADVICEビューの使用
- 個別のバッファ・プールへのバッファ・プール・ヒット率の計算
- バッファ・キャッシュ使用パターンの調査
- KEEPプールの構成
- RECYCLEプールの構成

複数バッファ・プールを使用する際の考慮事項

複数バッファ・プールを使用する際は、次の考慮事項を確認してください。

- 大きいセグメントへのランダム・アクセス
- Oracle Real Application Clustersのインスタンス

大きいセグメントへのランダム・アクセス

(バッファ・キャッシュのサイズと比較して)非常に大きいセグメントに大きい索引レンジ・スキャンまたは非有界索引レンジ・スキャンでアクセスすると、LRU除外方法では問題が発生する可能性があります。非順次物理読取りのかなりの割合(10%を超える

割合)を1つのセグメントが占有する場合、そのセグメントは大規模であると考えられます。大規模セグメントに対するランダム読取りは、他のセグメントのデータを含むバッファがキャッシュから除去される原因となります。大規模セグメントは、バッファ・キャッシュの大きな割合を消費しますが、キャッシングによる利益はありません。

非常に頻繁にアクセスされるセグメントは、バッファが頻繁にアクセスされるのでバッファ・キャッシュから除去されないため、大規模セグメントの読取りの影響を受けません。ただし、その問題は、大規模セグメントの読取りによるバッファの除外を免れるほど頻繁にはアクセスされないウォーム・セグメントに影響を与えます。この問題を解決するオプションは、次の3つです。

- アクセスされたオブジェクトが索引である場合は、索引に選択性があるかどうかを特定します。選択性がない場合は、さらに選択性のある索引を使用するようにSQL文をチューニングします。
- SQL文をチューニングすると、大きいセグメントが個別のRECYCLEキャッシュに移動されるので、その他のセグメントに影響を与えません。RECYCLEキャッシュはDEFAULTバッファ・プールよりも小さくし、迅速にバッファを再利用する必要があります。
- または、大規模セグメントでは使用されない別のKEEPキャッシュに小さなウォーム・セグメントを移動することも検討してください。キャッシュでのミスが最小限になるように、KEEPキャッシュをサイズ設定します。特定の問合せによってアクセスされるセグメントをKEEPキャッシュに格納し、除去されないようにすることで、その問合せのレスポンス時間をより予測可能にすることができます。

Oracle Real Application Clustersのインスタンス

Oracle Real Application Clusters (Oracle RAC)環境では、各データベース・インスタンスに複数バッファ・プールを作成することを検討してください。データベースの各インスタンスに同じセットのバッファ・プールを定義する必要はありません。インスタンスごとにバッファ・プールのサイズを変えることも、バッファを定義しないこともできます。それぞれのアプリケーション要件に従って、各インスタンスをチューニングします。

複数バッファ・プールの使用方法

オブジェクトのデフォルト・バッファ・プールを定義するには、STORAGE句のBUFFER_POOLキーワードを使用します。この句は、次のSQL文に有効です。

- CREATE TABLE
- CREATE CLUSTER
- CREATE INDEX
- ALTER TABLE
- ALTER CLUSTER
- ALTER INDEX

バッファ・プールを定義すると、そのオブジェクトに対して読み取られたブロックは、すべてそのプールに配置されます。バッファ・プールがパーティション表または索引に対して定義されている場合、オブジェクトの各パーティションは、特定のバッファ・プールで上書きされないかぎり、表または索引定義からバッファ・プールを継承します。

オブジェクトのバッファ・プールがALTER文を使用して変更された場合、変更されたセグメントのブロックを現在格納しているすべてのバッファは、ALTER文を発行する前にあったバッファ・プールに残ります。新たにロードされたブロック、および除去されてリロードされたブロックは、新しいバッファ・プールに配置されます。

関連項目:

STORAGE句でのBUFFER_POOLの指定の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

個別のバッファ・プールへのV$DB_CACHE_ADVICEビューの使用

デフォルト・バッファ・プールと同様、V$DB_CACHE_ADVICEビューを、その他のプールのキャッシュ・サイズ設定の参考に使用できます。初期キャッシュ・サイズを見積り、代表的なワークロードを実行したら、使用するプールのV$DB_CACHE_ADVICEビューを問い合せます。

V$DB_CACHE_ADVICEビューの使用方法の詳細は、「V$DB_CACHE_ADVICEビューの使用」を参照してください。

例13-2に、KEEPプールのデータを問い合せるこのビューの問合せを示します。

例13-2 KEEPプールのV$DB_CACHE_ADVICEビューの問合せ

```
SELECT size_for_estimate, buffers_for_estimate, estd_physical_read_factor,
       estd_physical_reads
  FROM V$DB_CACHE_ADVICE
 WHERE name = 'KEEP'
   AND block_size = (SELECT value FROM V$PARAMETER WHERE name = 'db_block_size')
   AND advice_status = 'ON';
```

個別のバッファ・プールへのバッファ・プール・ヒット率の計算

V$SYSSTATビューのデータは、すべてのバッファ・プールに対する論理読取りと物理読取りを1つの統計セットとして表します。バッファ・プールのヒット率を個々に確認するには、V$BUFFER_POOL_STATISTICSビューを問い合せます。このビューは、プールごとの論理読取りと論理書込みの回数に関する統計を維持します。

関連項目:

- ヒット率の詳細は、「バッファ・キャッシュ・ヒット率の計算」を参照してください
- V$BUFFER_POOL_STATISTICSビューの詳細は、『Oracle Databaseリファレンス』を参照してください

次の問合せでは、V$BUFFER_POOL_STATISTICSビューを使用してヒット率を計算しています。

例13-3 V$BUFFER_POOL_STATISTICSビューの問合せ

```
SELECT name, physical_reads, db_block_gets, consistent_gets,
       1 - (physical_reads / (db_block_gets + consistent_gets)) "Hit Ratio"
  FROM V$BUFFER_POOL_STATISTICS;
```

バッファ・キャッシュ使用パターンの調査

V$BHビューは、SGA内に現在存在するすべてのブロックのデータ・オブジェクトIDを示します。プールに多数のバッファがあるセグメントを特定するには、このビューを使用してバッファ・キャッシュ使用パターンを調査します。次の各項で説明するように、すべてのセグメントまたは特定のセグメントのバッファ・キャッシュ使用パターンを調査できます。

- すべてのセグメントのバッファ・キャッシュ使用パターンの調査
- 特定セグメントのバッファ・キャッシュ使用パターンの調査

すべてのセグメントのバッファ・キャッシュ使用パターンの調査

プールに多数のバッファがあるセグメントを特定する方法の1つは、ある時点でバッファ・キャッシュに存在するすべてのセグメントのブロック数を問い合せることです。バッファ・キャッシュ・サイズによっては、このカウントに多数のソート領域を必要とする可能性があります。

例13-4に、すべてのセグメントのブロック数を数える問合せを示します。

例13-4 すべてのセグメントのブロック数の問合せ

```
COLUMN object_name FORMAT A40
COLUMN number_of_blocks FORMAT 999,999,999,999
SELECT o.object_name, COUNT(*) number_of_blocks
  FROM DBA_OBJECTS o, V$BH bh
 WHERE o.data_object_id = bh.OBJD
   AND o.owner != 'SYS'
 GROUP BY o.object_Name
 ORDER BY COUNT(*);
```

この問合せの出力例を次に示します。

```
OBJECT_NAME                              NUMBER_OF_BLOCKS
---------------------------------------- ----------------
OA_PREF_UNIQ_KEY                                        1
SYS_C002651                                             1
..
DS_PERSON                                              78
OM_EXT_HEADER                                         701
OM_SHELL                                            1,765
OM_HEADER                                           5,826
OM_INSTANCE                                        12,644
```

特定セグメントのバッファ・キャッシュ使用パターンの調査

プールに多数のバッファがあるセグメントを特定する別の方法は、ある時点で個々のオブジェクトにより使用されているバッファ・キャッシュの割合を計算することです。

個々のオブジェクトにより使用されているバッファ・キャッシュの割合を計算するには:

1. DBA_OBJECTSビューを問い合せ、セグメントのOracle Database内部オブジェクト数を検出します。

   ```
   SELECT data_object_id, object_type
   FROM DBA_OBJECTS
   WHERE object_name = UPPER('segment_name');
   ```

   2つのオブジェクトが同じ名前を持つことがあるため(異なる型のオブジェクトの場合)、OBJECT_TYPE列を使用して目的のオブジェクトを識別します。

2. SEGMENT_NAMEに対するバッファ・キャッシュ内のバッファ数を検出します。

   ```
   SELECT COUNT(*) buffers
   FROM V$BH
   ```

```
WHERE objd = data_object_id_value;
```

data_object_id_valueについては、前のステップのDATA_OBJECT_IDの値を使用します。

3. データベース・インスタンスのバッファ数を検出します。

```
SELECT name, block_size, SUM(buffers)
FROM V$BUFFER_POOL
GROUP BY name, block_size
HAVING SUM(buffers) > 0;
```

4. バッファの総数に対するバッファの比率を計算し、現在SEGMENT_NAMEで使用されているキャッシュの割合を取得します。

```
% cache used by segment_name = [buffers(Step2)/total buffers(Step3)]
```

ノート:

この方法は、1 つのセグメントに対してのみ有効です。パーティション・オブジェクトについては、パーティションごとに問合せを実行します。

KEEPプールの構成

KEEPバッファ・プールの目的は、メモリー内にオブジェクトを保存して、I/O操作を避けることにあります。各オブジェクトをメモリー内に保持するとトレードオフが発生します。頻繁にアクセスされるブロックをキャッシュに保持しておくとより有益です。よりアクティブな他のブロックのためのスペースが減ることになりるため、頻繁に使用しないブロックはキャッシュに保持しないようにしてください。

アプリケーションに頻繁に参照されるセグメントがある場合は、KEEPバッファ・プールにそれらのセグメントのブロックを格納することを検討してください。KEEPプールに保持される一般的なセグメントは、サイズが小さく、頻繁に使用される参照表です。どの表が候補であるかを判別するには、「バッファ・キャッシュ使用パターンの調査」で説明されているように、V$BHビューを問い合せて、候補となる表のブロック数を確認します。

KEEPプールを構成するには:

1. KEEPバッファ・プールのおおよそのサイズを計算します。

   KEEPバッファ・プールのサイズは、バッファ・キャッシュに保持するオブジェクトによって異なります。サイズを見積るには、このプールに割り当てられたすべてのオブジェクトで使用されるブロックを追加します。

   セグメントに関する統計を収集した場合、DBA_TABLES.BLOCKSとDBA_TABLES.EMPTY_BLOCKSを問い合せて使用されるブロック数を判断します。

2. 違う時間にシステム・パフォーマンスの2つのスナップショットを取得します。

   「個々のバッファ・プールへのV$DB_CACHE_ADVICEビューの使用」で説明されているように、V$DB_CACHE_ADVICEビューを使用して、各スナップショットのKEEPプールからデータを問い合せます。

3. 物理読取り(physical reads)、ブロック取得(block gets)および一貫性読取り(consistent gets)について、古い値から新しい値を引いて、これらの結果を使用してヒット率を計算します。

   100%のバッファ・プール・ヒット率が最適とはかぎりません。KEEPバッファ・プールのサイズを減らしても、十分に高いヒット率が維持されることがよくあります。KEEPバッファ・プールから除去されたブロックは、別のバッファ・プールに割り当ててくだ

さい。

4. DB_KEEP_CACHE_SIZEパラメータの値を必要なサイズに設定して、KEEPバッファ・プールにメモリーを割り当てます。

   KEEPプールのメモリーは、デフォルト・プールのサブセットではありません。

ノート:

オブジェクトのサイズが大きくなった場合に、KEEP バッファ・プールに入りきらなくなることがあります。この場合、キャッシュからブロックが失われ始めます。

RECYCLEプールの構成

メモリーに残さないセグメントに属するブロック用のRECYCLEバッファ・プールを構成できます。RECYCLEプールの目的は、ほとんどスキャンされないか頻繁に参照されないセグメントを保持することです。アプリケーションが非常に大規模なラージ・オブジェクトのブロックにランダムにアクセスする場合、バッファ・プールに格納されているブロックが除去される前に再利用されることはほとんどありません。これは(使用可能物理メモリーの量の制約により)バッファ・プールのサイズに関係なくあてはまります。したがって、そのオブジェクトのブロックをキャッシュする必要はありません。これらのキャッシュ・バッファは、他のオブジェクトに割り当てることができます。

あまり早くメモリーからブロックを破棄しないでください。バッファ・プールが小さすぎると、トランザクションまたはSQL文が実行を完了する前に、ブロックがキャッシュから除外されてしまう可能性があります。たとえば、アプリケーションが表から値を選択し、その値を使用してデータを処理し、レコードを更新する場合があります。SELECT文の後でブロックがキャッシュから削除された場合は、更新を実行するために再度ディスクから読み取る必要があります。ブロックは、ユーザー・トランザクションの所要時間中は保存される必要があります。

RECYCLEプールを構成するには:

- DB_RECYCLE_CACHE_SIZEパラメータの値を必要なサイズに設定して、RECYCLEバッファ・プールにメモリーを割り当てます。

  RECYCLEプールのメモリーは、デフォルト・プールのサブセットではありません。

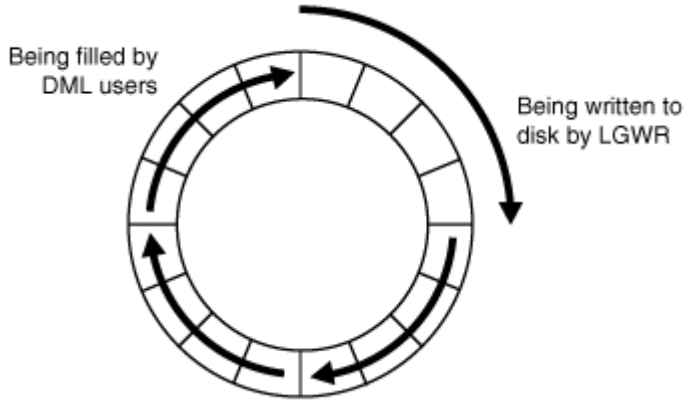
REDOログ・バッファの構成

バッファ・キャッシュ内のデータ・ブロックに変更を行うサーバー・プロセスでは、REDOデータをログ・バッファに生成します。ログ・ライター・プロセス(LGWR)は、次のいずれかの条件に当てはまる場合に、REDOログ・バッファからオンラインREDOログにエントリをコピーする書込みを開始します。

- REDOログ・バッファの少なくとも1/3が満たされる
- LGWRが、COMMITまたはROLLBACKを実行するサーバー・プロセスによって通知される
- データベース・ライター・プロセス(DBWR)がLGWRに実行するよう通知する

LGWRがREDOログ・ファイルまたはディスクにREDOログ・バッファからREDOエントリを書き込むとき、ユーザー・プロセスは、次の図に説明されているように、ディスクに書き込まれたメモリー内のエントリ上に新しいエントリをコピーできます。

図13-2 REDOログ・バッファ



LGWRは、REDOログ・バッファに新しいエントリ用の領域が残るよう、頻繁にアクセスされる場合でも、できるだけ迅速な書込みを試行します。REDOログ・バッファが大きいほど、新しいエントリ用の領域が確保される可能性が高く、LGWRも効率的にREDOレコードを処理できます。大規模な更新が行われるシステムでは、REDOログ・バッファが小さすぎると、LGWRがREDOを継続的にディスクにフラッシュするため、2/3が空のままになります。

高速のプロセッサと比較的低速のディスクを持つシステムでは、REDOログ・ライターによってREDOログ・バッファの一部がディスクに移動される時間に、プロセッサがREDOログ・バッファの残りにデータを挿入していることがあります。この状況では、大きいREDOログ・バッファは低速のディスクの影響を一時的に隠すことがあります。または、次のいずれかを改善することを検討してください。

- チェックポイント機能またはアーカイブ・プロセス
- LGWRのパフォーマンス(すべてのオンライン・ログを高速のRAWデバイスに移動)

REDOログ・バッファのパフォーマンスを改善するには、次のことを確認してください。

- LGWRがREDOログ・エントリを効率的に書き込めるようにする、バッチ・ジョブに対する一括コミット操作
- 大量のデータをロードするときのNOLOGGING操作の使用

この項では、REDOログ・バッファの構成方法を説明しており、内容は次のとおりです。

- REDOログ・バッファのサイズ設定
- REDOログ・バッファ統計の使用

REDOログ・バッファのサイズ設定

REDOログ・バッファのデフォルト・サイズは、次のようにして計算されます。

```
MAX(0.5M, (128K * number of cpus))
```

大量のデータの挿入、変更または削除を行うアプリケーションでは、デフォルトのREDOログ・バッファのサイズ変更が必要になる場合があります。REDOログ・バッファのサイズは、最低8MBに設定することをお薦めします。フラッシュバック機能を使用し、SGAが4GB以上のデータベースでは、最低64MBに設定してください。Oracle Data Guardの非同期REDO転送で使用していて、REDO生成速度が高い場合は、最低256MBに設定します。

REDOログ・バッファのサイズが小さすぎるかどうかを判断するには、「REDOログ・バッファ統計の使用」で説明されているように、

REDOログ・バッファ統計を監視します。また、log buffer space待機イベントがデータベース・インスタンスの待機時間における重要な要因であるかどうかもチェックできます。そうでない場合は、ログ・バッファ・サイズが適切にサイズ設定されていると考えられます。

REDOログ・バッファをサイズ設定するには:

- LOG_BUFFER初期化パラメータの値を必要なサイズに設定して、REDOログ・バッファのサイズを設定します。

  このパラメータの値はバイト単位で表されます。

ノート:

REDO ログ・バッファのサイズは、インスタンスの起動後には変更できません。

REDOログ・バッファ統計の使用

REDO BUFFER ALLOCATION RETRIES統計は、ユーザー・プロセスがREDOログ・バッファ内の領域の使用を待機した回数を反映します。この統計は、V$SYSSTATパフォーマンス・ビューを使用して問い合せることができます。

アプリケーションの実行中、一定期間にわたってredo buffer allocation retries統計を監視する必要があります。この統計の値は、ある時間間隔に対してゼロに近い値である必要があります。この値が継続的に増加する場合は、REDOログ・バッファの領域が使用可能になるまで、ユーザー・プロセスが待機する必要があったことを意味します。この待機は、REDOログ・バッファが小さすぎること、あるいはチェックポイント機能が原因となっていることがあります。この場合は、次のいずれかの方法を検討してください。

- 「REDOログ・バッファのサイズ設定」で説明されているように、REDOログ・バッファのサイズを大きくする
- チェックポイント機能またはアーカイブ・プロセスを改善する。

例13-5に、この統計のV$SYSSTATビューの問合せを示します。

例13-5 V$SYSSTATビューの問合せ

```
SELECT name, value
  FROM V$SYSSTAT
 WHERE name = 'redo buffer allocation retries';
```

データベース・キャッシュ・モードの構成

Oracle Database 12cリリース1 (12.1.0.2)から、2つのデータベース・キャッシュ・モードが存在します。旧バージョンのOracle Databaseで使用されるデフォルト・データベース・キャッシュ・モードと、このリリースの新機能である強制全データベース・キャッシュ・モードです。デフォルト・キャッシュ・モードでは、ユーザーが大きな表を問い合せたときにOracle Databaseは基礎データを常にキャッシュするわけではありません。強制全データベース・キャッシュ・モードでは、バッファ・キャッシュが全データベースをキャッシュできるくらい十分大きいとOracle Databaseで想定され、問合せでアクセスされたすべてのブロックのキャッシュが試行されます。

この項では、次の項目について説明します。

- デフォルト・データベース・キャッシュ・モード
- 強制全データベース・キャッシュ・モード
- 強制全データベース・キャッシュ・モードの使用時の判断
- データベース・キャッシュ・モードの検証

ノート:

強制全データベース・キャッシュ・モードは、Oracle Database 12c リリース 1 (12.1.0.2)以上で使用可能です。

デフォルト・データベース・キャッシュ・モード

デフォルトでは、全表スキャンの実行時にOracle Databaseではデフォルト・データベース・キャッシュ・モードが使用されます。デフォルト・キャッシュ・モードでは、ユーザーが大きな表を問い合せたときにOracle Databaseは基礎データを常にキャッシュするわけではありません。この操作によってより有用なデータがバッファ・キャッシュから削除される可能性があるためです。

全データベースをキャッシュするのに十分な領域がバッファ・キャッシュにあり、その操作が有効であるとOracle Databaseインスタンスによって判断された場合、インスタンスでは全データベースがバッファ・キャッシュに自動的にキャッシュされます。

全データベースをキャッシュするのに十分な領域がバッファ・キャッシュにないとOracle Databaseインスタンスによって判断された場合は、次のようになります。

- 小さい表は、表サイズがバッファ・キャッシュ・サイズの2%未満である場合のみメモリー内にロードされます。
- 中程度の表は、Oracle Databaseによって最後の表スキャンとバッファ・キャッシュの経過タイムスタンプ間の間隔が分析されます。最後の表スキャンで再利用された表のサイズが、残りのバッファ・キャッシュのサイズよりも大きい場合は、表がキャッシュされます。
- 大きい表は、KEEPバッファ・プールの表を明示的に宣言しないかぎり、通常はメモリー内にロードされません。

ノート:

デフォルト・キャッシュ・モードでは、Oracle Database インスタンスは NOCACHE LOB をバッファ・キャッシュにキャッシュしません。

関連項目:

デフォルト・データベース・キャッシュ・モードの詳細は、『Oracle Database概要』を参照してください

強制全データベース・キャッシュ・モード

メモリーがデータベースに追加されるに従い、バッファ・キャッシュ・サイズも継続的に拡大する可能性があります。バッファ・キャッシュのサイズが非常に大きくなったため、全データベースがメモリーに合致するようなケースもあります。全データベースをメモリー内

にキャッシュする機能は、全表スキャンの実行時やLOBのアクセス時にデータベース・パフォーマンスを劇的に改善します。

強制全データベース・キャッシュ・モードでは、データベースのサイズがデータベース・バッファ・キャッシュのサイズより小さい場合、Oracle Databaseが全データベースをメモリー内にキャッシュします。NOCACHE LOBおよびSecureFilesを使用するLOBなどのすべてのデータ・ファイルが、アクセスされているときにバッファ・キャッシュ内にロードされます。

関連項目:

- Oracle Database概要
- 『Oracle Database管理者ガイド』

強制全データベース・キャッシュ・モードの使用時の判断

特にI/Oスループットまたはレスポンス時間で制限されたワークロードにおいて、表スキャンおよびLOBデータ・アクセスのデータベース・パフォーマンスを改善するため、データベース・バッファ・キャッシュのサイズがデータベースのサイズを上回る場合は常に、強制全データベース・キャッシュ・モードの使用を検討します。

強制全データベース・キャッシュ・モードの使用は、次のような状況で検討します。

- 論理データベースのサイズ(または実際の使用済領域)が、Oracle RAC環境内の各データベース・インスタンスの個別バッファ・キャッシュより小さい場合。これはOracle RACデータベース以外にも同様に適用可能です。
- 論理データベースのサイズが、Oracle RAC環境内の(インスタンス・アクセスによって)有効にパーティション化されたワークロードに対する全データベース・インスタンスの合計バッファ・キャッシュ・サイズの80%より小さい場合。
- データベースがSGA_TARGETまたはMEMORY_TARGETを使用する場合。
- NOCACHE LOBをキャッシュする必要がある場合。強制全データベース・キャッシュが使用されないかぎり、NOCACHE LOBがキャッシュされることはありません。

最初の3つの状況では、システム・パフォーマンスを定期的に監視して、パフォーマンスの数値が予想どおりかを検証する必要があります。

あるOracle RACデータベース・インスタンスが強制全データベース・キャッシュ・モードを使用すると、Oracle RAC環境内のその他すべてのデータベース・インスタンスも強制全データベース・キャッシュ・モードを使用します。

マルチテナント環境では、強制全データベース・キャッシュ・モードはコンテナ・データベース(CDB)全体に適用され、そのすべてのプラガブル・データベース(PDB)も含まれます。

データベース・キャッシュ・モードの検証

デフォルトでは、Oracle Databaseではデフォルト・データベース・キャッシュ・モードが実行されます。

強制全データベース・キャッシュ・モードが有効かどうかを検証するには:

- V$DATABASEビューを次のように問い合せます。

```
SELECT FORCE_FULL_DB_CACHING FROM V$DATABASE;
```

問合せでYESの値が返される場合は、強制全データベース・キャッシュ・モードがデータベース上で有効です。問合せでNOの値が返される場合は、強制全データベース・キャッシュ・モードは無効で、データベースはデフォルト・データベース・キャッシュ・モードの状態です。

ノート:

強制全データベース・キャッシュ・モードを有効にするには、次の ALTER DATABASE コマンドを使用します。

```
ALTER DATABASE FORCE FULL DATABASE CACHING;
```

関連項目:

- 強制全データベース・キャッシュ・モードの有効化および無効化の詳細は、『Oracle Database管理者ガイド』を参照してください
- V$DATABASEビューの詳細は、『Oracle Databaseリファレンス』を参照してください

14 共有プールおよびラージ・プールのチューニング

この章では、共有プールおよびラージ・プールのチューニング方法を説明します。自動メモリー管理を使用してシステムのデータベース・メモリーを管理している場合、または自動共有メモリー管理を使用して共有グローバル領域(SGA)を構成している場合は、この章で説明されているように、共有プールおよびラージ・プールを手動でチューニングする必要はありません。

この章のトピックは、次のとおりです:

- 共有プールについて
- 共有プールの使用
- 共有プールの構成
- ラージ・プールの構成

共有プールについて

異なるタイプのデータをキャッシュするには、共有プールを使用します。キャッシュされたデータには、PL/SQLブロックおよびSQL文のテキストおよび実行可能フォーム、ディクショナリ・キャッシュ・データ、結果キャッシュ・データおよびその他のデータが含まれています。

この項では、共有プールについて説明しており、内容は次のとおりです。

- 共有プールを使用する利点
- 共有プールの概念

共有プールを使用する利点

共有プールを適切な大きさにして使用すると、次の4つの方法でリソース使用量を低減できます。

- SQL文が共有プールにある場合は、解析のオーバーヘッドを回避できるため、システムのCPUリソースやエンド・ユーザーの経過時間が低減します。
- リソース使用のラッチングが大幅に減少して、スケーラビリティがさらに増大します。
- すべてのアプリケーションがSQL文およびディクショナリ・リソースの同一プールを使用するので、共有プール・メモリーの必要量が低減されます。
- 共有プールのディクショナリ要素はディスク・アクセスが不要なので、I/Oが減少します。

共有プールの概念

共有プールの主なコンポーネントは、次のとおりです。

- ライブラリ・キャッシュ

  ライブラリ・キャッシュは、最近参照されたSQLとPL/SQLコードの実行可能な(解析またはコンパイルされた)形式を格納します。

- データ・ディクショナリ・キャッシュ

データ・ディクショナリ・キャッシュは、データ・ディクショナリから参照されたデータを格納します。

- サーバー結果キャッシュ(構成により異なります)

  サーバーの結果キャッシュは、問合せの結果とPL/SQLファンクションの結果を格納する共有プール内のオプションのキャッシュです。サーバー結果キャッシュの詳細は、「結果キャッシュについて」を参照してください。

ライブラリ・キャッシュやディクショナリ・キャッシュなどの共有プール内のキャッシュの多くは、必要に応じてサイズを自動的に増減します。共有プールに空き領域がない場合は、新しいエントリを受け入れるために古いエントリが除去されます。

ライブラリ・キャッシュやデータ・ディクショナリ・キャッシュでのキャッシュ・ミスは、バッファ・キャッシュでのミスよりも影響が大きくなります。このため、共有プールには、頻繁に使用されるデータが確実にキャッシュされるように、サイズを設定する必要があります。

共有サーバー、パラレル問合せ、Recovery Managerなど、共有プールに大量のメモリーを割り当てる必要がある機能は複数あります。これらの機能によって使用されるシステム・グローバル領域(SGA)メモリーを分離するために、個別のメモリー領域(ラージ・プール)を使用することをお薦めします。

共有プールからのメモリーの割当ては、チャンクで行われます。このチャンクによって、1つの連続領域を必要とせずにラージ・オブジェクト(5KBより多い)をキャッシュにロードできます。この方法では、フラグメント化のために連続メモリーが不足する可能性が少なくなります。

Java、PL/SQLまたはSQLカーソルによって、5KBを超える領域が共有プールから割り当てられることがあります。このような割当てをより効率よく行うために、Oracle Databaseでは、少量の共有プールを分離します。共有プールの領域が不足すると、予約プールと呼ばれる分離されたメモリーが使用されます。

次の各項では、共有プールの主なコンポーネントの詳細を説明します。

- ライブラリ・キャッシュの概念
- データ・ディクショナリ・キャッシュの概念
- SQL共有基準

ライブラリ・キャッシュの概念

ライブラリ・キャッシュには、総称してアプリケーション・コードと呼ばれる、実行可能な形式のSQLカーソル、PL/SQLプログラムおよびJavaクラスが格納されます。アプリケーション・コードに関連するため、この項ではチューニングを中心に説明します。

アプリケーション・コードが実行されると、既存のコードが以前に実行されており、共有できる場合は、Oracle Databaseによりそのコードの再利用が試行されます。解析されたSQL文の表現がライブラリ・キャッシュ内に存在し、共有できる場合は、既存のコードが再利用されます。これは、ソフト解析またはライブラリ・キャッシュ・ヒットと呼ばれています。既存のコードを使用できない場合は、アプリケーション・コードの新しい実行可能バージョンを作成する必要があります。これは、ハード解析またはライブラリ・キャッシュ・ミスと呼ばれています。SQL文およびPL/SQL文を共有できる場合の詳細は、「SQL共有基準」を参照してください。

ハード解析を実行するには、ソフト解析の実行時より多くのリソースを使用します。ソフト解析に使用するリソースには、CPUおよびライブラリ・キャッシュ・ラッチ取得が含まれます。ハード解析に必要なリソースには、追加のCPU、ライブラリ・キャッシュ・ラッチ取得および共有プール・ラッチ取得が含まれます。ハード解析は、SQL文を処理するときの解析ステップまたは実行ステップのいずれかで発生します。

アプリケーションがSQL文の解析コールを行うとき、解析された文の表現がライブラリ・キャッシュにまだ存在しない場合、Oracle Databaseはその文を解析し、共有プールに解析されたフォームを格納します。解析コールのライブラリ・キャッシュ・ミスを低減するため、可能な場合には、すべての共有可能なSQL文が共有プールに格納されていることを確認してください。

アプリケーションでSQL文の実行コールを行ったときに、SQL文の実行可能部分が、別の文の領域を確保するためにライブラリ・キャッシュから除去(または割当て解除)されていた場合、Oracle Databaseはその文を暗黙的に再解析し、新しい共有SQL領域を作成してその文を実行します。この場合も、ハード解析が発生します。実行コールでのライブラリ・キャッシュ・ミスを低減するには、ライブラリ・キャッシュに割り当てるメモリーを増やします。

ハード解析およびソフト解析の詳細は、「SQLの実行効率」を参照してください。

データ・ディクショナリ・キャッシュの概念

データ・ディクショナリ・キャッシュに格納される情報は、次のとおりです。

- ユーザー名
- セグメント情報
- プロファイル・データ
- 表領域情報
- 順序番号

また、データ・ディクショナリ・キャッシュはスキーマ・オブジェクトを説明する情報、すなわちメタデータも格納します。メタデータが使用されるのは、SQLカーソルの解析時かPL/SQLプログラムのコンパイル時です。

SQL共有基準

Oracle Databaseでは、SQL文またはPL/SQLブロックを発行する際に、共有プールに同じ文が存在するかどうかを自動的に確認します。

SQL文のテキストと、共有プール内の既存のSQL文を比較するために、Oracle Databaseでは次のステップが実行されます。

1. SQL文のテキストがハッシュされます。

   一致するハッシュ値がない場合、SQL文は共有プール内に現在存在せず、ハード解析が実行されます。

2. 共有プール内の既存のSQL文に一致するハッシュ値がある場合は、一致した文のテキストが、ハッシュされた文のテキストと比較され、それらが同一であるかどうかが確認されます。

   SQL文やPL/SQLブロックのテキストは、空白、大文字小文字の区別、コメントも含め、完全に同一である必要があります。たとえば、次の文は同じ共有SQL領域を使用できません。

   ```
   SELECT * FROM employees;
   SELECT * FROM Employees;
   SELECT *  FROM employees;
   ```

   また、リテラルのみ異なるSQL文は同じ共有SQL領域を使用できません。たとえば、次の文は同じSQL領域に変換されません。

   ```
   SELECT count(1) FROM employees WHERE manager_id = 121;
   SELECT count(1) FROM employees WHERE manager_id = 247;
   ```

このルールの唯一の例外は、CURSOR_SHARINGパラメータがFORCEに設定されている場合で、類似の文でSQL領域を共有できます。カーソル共有に関わるデメリットとメリットの詳細は、「カーソルの共有」を参照してください。

3. 発行された文で参照されたオブジェクトは、共有プール内のすべての既存の文の参照済オブジェクトと比較され、両方のオブジェクトが同一であるかどうかが確認されます。

   SQL文やPL/SQLブロック内でスキーマ・オブジェクトを参照する際には、同じスキーマ内の同じオブジェクトである必要があります。たとえば、2人のユーザーが次のSQL文を発行したが、各ユーザーに独自のemployees表がある場合、文はユーザーごとに異なる表を参照するので、この文は同一とみなされません。

   ```
   SELECT * FROM employees;
   ```

4. SQL文の中のバインド変数は、名前、データ型および長さで一致している必要があります。

   たとえば、次の文で同じ共有SQL領域を使用できないのは、バインド変数名が異なるためです。

   ```
   SELECT * FROM employees WHERE department_id = :department_id;
   SELECT * FROM employees WHERE department_id = :dept_id;
   ```

   多くのOracle製品(Oracle Formsやプリコンパイラなど)は、文をデータベースに渡す前にSQLを変換します。首尾一貫したSQL文の集合が生成されるように、文字は大文字に統一して変換され、空白は圧縮され、バインド変数は改名されます。

5. セッションの環境は同一である必要があります。

   たとえば、SQL文は、同一の最適化目標を使用して最適化する必要があります。

関連項目:

CURSOR_SHARING初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください

共有プールの使用

共有プールの重要な目的は、SQL文とPL/SQL文の実行可能バージョンをキャッシュすることです。これにより、ハード解析にリソースを使用することなく、同じSQLまたはPL/SQLコードを複数回実行できるので、CPU、メモリーおよびラッチの使用が大幅に減少します。

また、共有プールは、データ・ウェアハウス・アプリケーションで非共有SQLをサポートできます。これらのアプリケーションでは、同時実行性が低くリソース使用率の高いSQL文が実行されます。このような状況では、リテラル値を持つ非共有SQLを使用することをお薦めします。バインド変数ではなくリテラル値を使用すると、オプティマイザは優れた列選択性の見積りを行えるので、最適なデータ・アクセス・プランが提供されます。

広く普及しているオンライン・トランザクション処理(OLTP)システムでは、共有プールを効率的に使用すると、解析関連アプリケーションでスケーラビリティの問題が発生する確率を大幅に低減できます。OLTPシステムには、共有プールと関連リソースの効率的な使用を実現するいくつかの方法があります。

- 共有カーソルの使用
- シングル・ユーザー・ログオンおよび修飾表の参照の使用

- PL/SQLの使用
- DDL操作の実行の回避
- キャッシュ順序番号
- カーソル・アクセスの制御
- 永続的な接続の維持

関連項目:

共有プールでパラレル問合せを実行した場合の影響の詳細は、Oracle Database VLDBおよびパーティショニング・ガイドを参照してください

共有カーソルの使用

同じアプリケーションを実行する複数のユーザーのために共有SQLを再利用すると、ハード解析が回避されます。ソフト解析は、共有プール・ラッチやライブラリ・キャッシュ・ラッチなどのリソース使用量を大幅に減少させます。

共有カーソルを使用するには:

- 可能な場合、SQL文ではリテラルではなくバインド変数を使用します。

  たとえば、次の2つのSQL文は字面が完全には一致しないので、同じ共有領域は使用できません。

  ```
  SELECT employee_id FROM employees WHERE department_id = 10;
  SELECT employee_id FROM employees WHERE department_id = 20;
  ```

  リテラルをバインド変数と置換すると、2回実行可能なSQL文が1つのみ生成されます。

  ```
  SELECT employee_id FROM employees WHERE department_id = :dept_id;
  ```

  バインド変数を使用するためにコードをリライトすることが実際的ではない既存のアプリケーションについては、「カーソルの共有」で説明されているように、CURSOR_SHARING初期化パラメータを使用してハード解析のオーバーヘッドをある程度回避できます。

- 多数のユーザーが動的な非共有のSQL文を発行するようなアプリケーションを設計しないようにしてください。

  通常、大半のユーザーが必要とするデータの大部分は、事前に設定されている問合せを使用して満たすことができます。そのような機能が必要な場合は、動的SQLを使用します。

- アプリケーションのユーザーが最適化アプローチと目標を各セッションに対して変更しないことを確認してください。
- アプリケーション開発者に対し、次のポリシーを設定します。
  - SQL文とPL/SQLブロックに対して、バインド変数のネーミング規則と、スペーシング規定を標準化します。
  - 可能な場合、ストアド・プロシージャを使用することを考慮してください。

    そうすれば、同じストアド・プロシージャを発行している複数のユーザーが、同じ共有PL/SQL領域を自動的に使用します。ストアド・プロシージャは解析済フォームに格納されるため、ストアド・プロシージャを使用するとランタイム解析が減少します。
- 同一であっても共有されていないSQL文についてV$SQL_SHARED_CURSORビューを問い合せて、カーソルが共有されな

い理由を判断します。

これには、オプティマイザの設定とバインド変数の不整合などがあります。

関連項目:

カーソル共有の詳細は、Oracle Database SQLチューニング・ガイドを参照してください

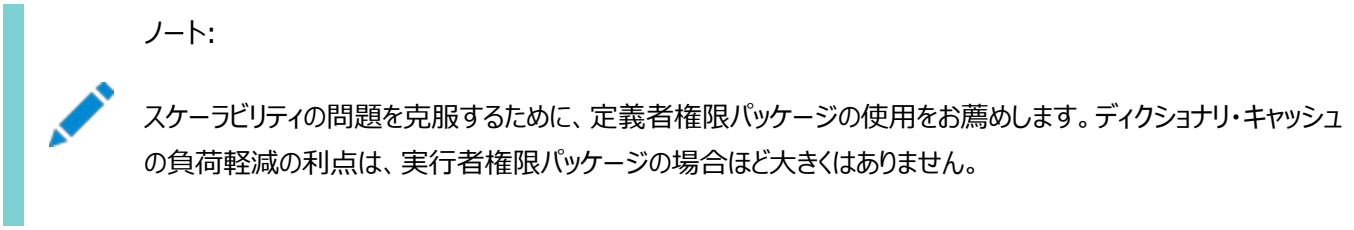
シングル・ユーザー・ログオンおよび修飾表の参照の使用

ユーザーがそれぞれ固有のユーザー・ログインを使用してデータベースにログインする大規模なOLTPシステムでは、パブリック・シノニムを使用するかわりにセグメント所有者を明示的に認定することで、ディクショナリ・キャッシュのエントリ数が大幅に少なくなります。

表名を認定する別の方法として、個々のユーザー・ログオンではなくシングル・ユーザー・ログオンでデータベースに接続します。ユーザー・レベルの検証は、中間層でローカルに行われます。

PL/SQLの使用

ストアドPL/SQLパッケージを使用すると、多数のユーザーが個々にユーザー・ログオンとパブリック・シノニムを持つシステムにおける、スケーラビリティの問題を克服できます。これは、パッケージがコール元ではなく所有者として実行されるため、ディクショナリ・キャッシュの負荷がかなり削減されるためです。



ノート:

スケーラビリティの問題を克服するために、定義者権限パッケージの使用をお薦めします。ディクショナリ・キャッシュの負荷軽減の利点は、実行者権限パッケージの場合ほど大きくはありません。

DDL操作の実行の回避

ピーク時間に使用率の高いセグメントでDDL操作を実行しないようにします。これらのセグメントでDDL操作を実行すると、多くの場合、依存SQLが無効になり、以降の実行で再度解析されることになります。

順序番号のキャッシュ

頻繁に更新される順序番号に十分なキャッシュ領域を割り当てると、ディクショナリ・キャッシュ・ロックの回数が大幅に減るため、スケーラビリティが向上します。

各順序に対するキャッシュ・エントリ数を構成するには:

- CREATE SEQUENCEまたはALTER SEQUENCE文で、CACHEキーワードを使用します。

カーソル・アクセスの制御

使用しているアプリケーション・ツールによっては、カーソル・アクセスを制御することで、アプリケーションの解析コールの実行頻度を

制御できます。

アプリケーションが、カーソルをクローズする、または新しいSQL文に既存のカーソルを再利用する頻度は、セッションで使用されるメモリー量と、時には、そのセッションで実行される解析の量にも影響を与えます。(異なるSQL文の)カーソルをクローズまたは再利用するアプリケーションは、カーソルをオープンした状態を保つアプリケーションほどセッション・メモリーを必要としません。逆に、そのようなアプリケーションでは、解析コールがより多く実行されるため、より多くのCPUおよびデータベース・リソースを使用する可能性があります

頻繁に実行されないSQL文に関連するカーソルをクローズしたり、他の文に再利用できるのは、その文を再実行(および再解析)する可能性が低いからです。再実行されるSQL文を含むカーソルがクローズまたは別の文に再利用される場合は、追加の解析コールが必要になります。カーソルがオープンされた状態であれば、解析コールを発行するためのオーバーヘッドを発生させずに、カーソルを再利用できます。

カーソル・アクセスを制御する方法は、アプリケーション開発ツールにより異なります。この項では、Oracle Databaseツールで使用される方法を説明します。

- OCIを使用したカーソル・アクセスの制御
- Oracleプリコンパイラを使用したカーソル・アクセスの制御
- SQLJを使用したカーソル・アクセスの制御
- JDBCを使用したカーソル・アクセスの制御
- Oracle Formsを使用したカーソル・アクセスの制御

関連項目:

各ツールの詳細は、ツール固有のマニュアルを参照してください

OCIを使用したカーソル・アクセスの制御

Oracle Call Interface (OCI)を使用する場合、再実行するカーソルはクローズおよび再オープンしないでください。そのかわりに、カーソルをオープンしたままにし、実行前にリテラル値をバインド変数に変更してください。

既存のSQL文が今後再実行される場合は、新しいSQL文に文ハンドルを再利用しないようにしてください。

Oracleプリコンパイラを使用したカーソル・アクセスの制御

Oracleプリコンパイラを使用する場合、プリコンパイラ句を設定して、いつカーソルをクローズするかを制御できます。Oracleモードでは、句は次のようになります。

- HOLD_CURSOR = YES
- RELEASE_CURSOR = NO
- MAXOPENCURSORS = desired_value

プリコンパイラ句は、プリコンパイラ・コマンドライン上またはプリコンパイラ・プログラム内で指定できます。ANSIモードでは、HOLD_CURSORとRELEASE_CURSORの値が切り替えられますが、これはお薦めしません。

関連項目:

これらの句の詳細は、使用している言語のプリコンパイラ・マニュアルを参照してください

SQLJを使用したカーソル・アクセスの制御

SQL文を用意し、バインド変数に新しい値を使用して文を再実行します。カーソルは、セッション中はオープンのままです。

ノート:

Oracle Database 12c リリース 2 (12.2)以降、サーバー側の SQLJ コードはサポートされません。つまり、ストアド・プロシージャ、関数およびトリガー内で SQLJ コードを使用することはできません。

JDBCを使用したカーソル・アクセスの制御

新しいリテラル値は、再実行のためにカーソルにバインドできるので、カーソルを再実行する場合は、カーソルをクローズしないでください。別の方法として、JDBCはsetStmtCacheSize()メソッドを使用してJDBCクライアント内にSQL文キャッシュを提供しています。このメソッドを使用して、JDBCはJDBCプログラムに対してローカルなSQL文キャッシュを作成します。

関連項目:

JDBC SQL文キャッシュの使用方法の詳細は、『Oracle Database JDBC開発者ガイド』を参照してください

Oracle Formsを使用したカーソル・アクセスの制御

Oracle Formsを使用すると、実行時、トリガー・レベルまたはフォーム・レベルで、カーソル・アクセスの一部の局面を制御できます。

永続的な接続の維持

中間層を持つ大きなOLTPアプリケーションでは、データベース・リクエストごとに接続と切断を行うのではなく、接続を維持するようにします。永続的な接続を維持することで、ラッチなどのCPUリソースとデータベース・リソースが節約されます。

共有プールの構成

この項では、共有プールの構成方法を説明しており、内容は次のとおりです。

- 共有プールのサイズ設定
- カーソルの割当て解除
- セッション・カーソルのキャッシュ
- カーソルの共有
- 除去防止のためのラージ・オブジェクトの保存
- 予約プールの構成

共有プールのサイズ設定

新規にデータベース・インスタンスを構成する場合は、共有プール・キャッシュの適切なサイズを把握することは困難です。通常、DBAはキャッシュ・サイズの最初の見積りを行い、次にインスタンス上で代表的なワークロードを実行し、関連する統計を調べて、キャッシュが過小構成か過大構成かを調べます。

OLTPアプリケーションの多くでは、共有プール・サイズはアプリケーション・パフォーマンスにとって重要な要因です。意思決定支援システム(DSS)のような、ごく少数の不連続なSQL文を発行するアプリケーションでは、共有プールのサイズはそれほど重要ではありません。

共有プールが小さすぎると、使用可能領域の限度を補うために、追加リソースを使用することになります。このため、CPUとラッチングのリソースが使用され、競合が発生します。共有プールは、頻繁にアクセスされるオブジェクトをキャッシュするためにちょうど十分な大きさであることが理想的です。共有プールに大量の空きメモリーを持つことは、メモリーの無駄になります。データベースの稼働後に統計を調べる際、これらの点についてワークロード内に該当する箇所がないか確認してください。

この項では、共有プールのサイズ設定方法を説明しており、内容は次のとおりです。

- ライブラリ・キャッシュ統計の使用
- 共有プールのアドバイザ統計の使用
- ディクショナリ・キャッシュ統計の使用
- 共有プールに割り当てられたメモリーの増加
- 共有プールに割り当てられたメモリーの低減

ライブラリ・キャッシュ統計の使用

共有プールをサイズ設定するときの目標は、メモリーを割り当てすぎずに、複数回実行されるSQL文をライブラリ・キャッシュにキャッシュすることです。この目標を達成するには、次のライブラリ・キャッシュ統計を調査します。

- RELOADS

  V$LIBRARYCACHEビューのRELOADS列には、以前にキャッシュされ、キャッシュから除去されたSQL文のリロード(または解析)の量が表示されます。アプリケーションで効果的にSQLが再利用され、共有プール・サイズが最適なシステムで実行されている場合、この統計はゼロに近い値を示します。

- INVALIDATIONS

  V$LIBRARYCACHEビューのINVALIDATIONS列は、ライブラリ・キャッシュのデータが無効にされ、再解析された回数を示しています。この統計は、ピーク・ロード中のOLTPシステム上では特に、ゼロに近い値となります。つまり、共有できるSQL文が、ある操作(DDLなど)により無効にされたことを意味します。

- ライブラリ・キャッシュ・ヒット率

  ライブラリ・キャッシュ・ヒット率は、ライブラリ・キャッシュの状態に関する全般的なインジケータです。この値は、ハード解析率や、共有プールまたはライブラリ・キャッシュのラッチ競合があるかどうかなど、その他の統計とともに考慮する必要があります。

- 共有プールの空きメモリー量

  共有プールの空きメモリー量を表示するには、V$SGASTATパフォーマンス・ビューを問い合せます。空きメモリーは、システ

ム上に再解析を発生させない程度で、できるだけ少なくすることが理想的です。

次の各項では、これらのライブラリ・キャッシュ統計を表示および調査する方法を説明します。

- V$LIBRARYCACHEビューの使用
- ライブラリ・キャッシュ・ヒット率の計算
- 共有プールの空きメモリー量の表示

V$LIBRARYCACHEビューの使用

ライブラリ・キャッシュ・アクティビティを反映する統計を監視するには、V$LIBRARYCACHEビューを使用します。これらの統計は、最新のデータベース・インスタンス起動後のすべてのライブラリ・キャッシュのアクティビティを反映しています。

このビューの各行には、ライブラリ・キャッシュ内に保持される項目の1つに対応する統計が収録されます。各行ごとに記述される項目は、NAMESPACE列の値によって識別されます。次のNAMESPACE値を持つ行は、SQL文とPL/SQLブロックのライブラリ・キャッシュのアクティビティを反映します。

- SQL AREA
- TABLE/PROCEDURE
- BODY
- TRIGGER

他のNAMESPACE値を持つ行は、Oracle Databaseが依存関係のメンテナンスのために使用するオブジェクト定義に対するライブラリ・キャッシュのアクティビティを反映します。

例14-1に、各ネームスペースを個別に調査するためのこのビューの問合せを示します。

例14-1 V$LIBRARYCACHEビューの問合せ

```
SELECT namespace, pins, pinhits, reloads, invalidations
  FROM V$LIBRARYCACHE
 ORDER BY namespace;
```

この問合せの出力例を次に示します。

```
NAMESPACE             PINS    PINHITS    RELOADS INVALIDATIONS
--------------- ---------- ---------- ---------- -------------
BODY                  8870       8819          0             0
CLUSTER                393        380          0             0
INDEX                   29          0          0             0
OBJECT                   0          0          0             0
PIPE                 55265      55263          0             0
SQL AREA          21536413   21520516      11204             2
TABLE/PROCEDURE   10775684   10774401          0             0
TRIGGER               1852       1844          0             0
```

この例の出力は、次のことを示しています。

- SQL AREAのネームスペースには、21,536,413回の実行があります。
- これらの実行のうち11,204回ではライブラリ・キャッシュ・ミスが発生しており、データベースで、文またはブロックを暗黙的に再解析するか、ライブラリ・キャッシュから除去されているため、オブジェクト定義をリロードする必要があります。

- SQL文は2回無効化されたため、再度ライブラリ・キャッシュ・ミスが発生しています。

ノート:

この問合せにより、インスタンスの起動からデータが戻されます。かわりに、ある期間に収集された統計を使用すると、より的確に問題を特定できます。ある期間の情報を収集する方法の詳細は、「自動パフォーマンス診断」を参照してください。

関連項目:

V$LIBRARYCACHEビューの詳細は、『Oracle Databaseリファレンス』を参照してください

ライブラリ・キャッシュ・ヒット率の計算

ライブラリ・キャッシュ・ヒット率を計算するには、次の計算式を使用します。

```
Library Cache Hit Ratio = sum(pinhits) / sum(pins)
```

ライブラリ・キャッシュ・ヒット率の式を例14-1に適用すると、次のライブラリ・キャッシュ・ヒット率になります。

```
SUM(PINHITS)/SUM(PINS)
----------------------
            .999466248
```

この例では、ヒット率が約99.94%で、実行の0.06%のみが再解析されたことを意味します。

共有プールの空きメモリー量の表示

共有プールの空きメモリー量は、V$SGASTATビューでレポートされます。

例14-2に、このビューの問合せを示します。

例14-2 V$SGASTATビューの問合せ

```
SELECT *
  FROM V$SGASTAT
 WHERE name = 'free memory'
   AND pool = 'shared pool';
```

この問合せの出力例を次に示します。

```
POOL        NAME                            BYTES
----------- -------------------------- ----------
shared pool free memory                   4928280
```

共有プール内に使用可能な空きメモリーが常にある場合は、サイズを増やしても、効果はほとんど、またはまったくありません。ただし、共有プールがいっぱいというだけでは、問題があるとはいえません。これは、適切に構成されたシステムであることを示している場合があります。

共有プールのアドバイザ統計の使用

ライブラリ・キャッシュに使用可能なメモリー量は、Oracle Databaseの解析率に大きな影響を与えます。ライブラリ・キャッシュの適切なサイズ設定に役立つよう、Oracle Databaseには、次の共有プール・アドバイザ・ビューがあります。

- V$SHARED_POOL_ADVICE
- V$LIBRARY_CACHE_MEMORY
- V$JAVA_POOL_ADVICE
- V$JAVA_LIBRARY_CACHE_MEMORY

これらの共有プール・アドバイザ・ビューから、ライブラリ・キャッシュ・メモリーについての情報を得ることができ、共有プールのサイズ変更が共有プール内のオブジェクトの除去にどのように影響するかを予測できます。これらのビューの共有プール・アドバイザ統計では、共有プール・メモリーにおけるライブラリ・キャッシュの使用率が追跡され、異なるサイズの共有プールでライブラリ・キャッシュがどのように動作するかが予測されます。これらのビューを使用することで、次のことを判断できます。

- ライブラリ・キャッシュが使用しているメモリー量
- 現在確保されているメモリー量
- 共有プールの最低使用頻度(LRU)リストのメモリー量
- 共有プールのサイズを変更することで失われる、または得られる時間

共有プール・アドバイザが有効化されていると、これらのビューには共有プール・アドバイザ統計が表示されます。アドバイザを無効化すると、統計がリセットされます。

次の各項では、これらのビューについてさらに詳しく説明します。

- V$SHARED_POOL_ADVICEビューについて
- V$LIBRARY_CACHE_MEMORYビューについて
- V$JAVA_POOL_ADVICEビューおよびV$JAVA_LIBRARY_CACHE_MEMORYビューについて

V$SHARED_POOL_ADVICEビューについて

V$SHARED_POOL_ADVICEビューは、様々なサイズの共有プールでの解析時間の見積りに関する情報を示します。サイズの範囲は、同じ時間間隔で、現在の共有プール・サイズまたは確保されたライブラリ・キャッシュ・メモリー量の10%のうち大きい方の値から、現在の共有プール・サイズの200%までです。間隔値は、現行の共有プール・サイズに応じて異なります。

関連項目:

V$SHARED_POOL_ADVICEビューの詳細は、Oracle Databaseリファレンスを参照してください

V$LIBRARY_CACHE_MEMORYビューについて

V$LIBRARY_CACHE_MEMORYビューは、様々なネームスペース内のライブラリ・キャッシュ・メモリー・オブジェクトに割り当てられたメモリーに関する情報を示します。メモリー・オブジェクトは、効率的に管理するために内部でメモリーをグループ化したものです。ライブラリ・キャッシュ・オブジェクトは、1つ以上のメモリー・オブジェクトで構成されます。

関連項目:

V$LIBRARY_CACHE_MEMORYビューの詳細は、Oracle Databaseリファレンスを参照してください

V$JAVA_POOL_ADVICEビューおよびV$JAVA_LIBRARY_CACHE_MEMORYビューについて

V$JAVA_POOL_ADVICEおよびV$JAVA_LIBRARY_CACHE_MEMORYビューには、Javaに使用されるライブラリ・キャッシュ・メモリーについての情報を追跡し、Javaプールのサイズ変更が解析率に及ぼす影響を予測する、Javaプール・アドバイザ統計が含まれます。

V$JAVA_POOL_ADVICEビューは、様々なプール・サイズのJavaプール内での見積り解析時間に関する情報を示します。サイズの範囲は、同じ時間間隔で、現在のJavaプール・サイズまたは確保されたJavaライブラリ・キャッシュ・メモリー量の10%のうち大きい方の値から、現在のJavaプール・サイズの200%までです。間隔値は、現行のJavaプール・サイズに応じて異なります。

関連項目:

V$JAVA_POOL_ADVICEビューおよびV$JAVA_LIBRARY_CACHE_MEMORYビューの詳細は、Oracle Databaseリファレンスを参照してください

ディクショナリ・キャッシュ統計の使用

共有プールがライブラリ・キャッシュに対して適切にサイズ設定されている場合、その設定はディクショナリ・キャッシュ・データに対しても適切なサイズであるのが普通です。

場合によっては、データ・ディクショナリ・キャッシュ上でミスが発生します。データベース・インスタンスが起動すると、データ・ディクショナリ・キャッシュにはデータがなくなります。したがって、発行されたSQL文からキャッシュ・ミスが発生する可能性があります。キャッシュに読み取られるデータが増えると、キャッシュ・ミスの可能性は減少します。最終的に、データベースは、最も頻繁に使用されるディクショナリ・データがキャッシュ内に存在する安定状態に到達します。この時点で、キャッシュ・ミスはほとんど発生しません。

V$ROWCACHEビューの各行は、データ・ディクショナリ項目について単一のタイプの統計を収録します。これらの統計は、直前のインスタンス起動以降のデータ・ディクショナリ・アクティビティを反映しています。

表14-1は、データ・ディクショナリ・キャッシュの使用と有効性を反映するV$ROWCACHEビューの中の列を示しています。

表14-1 V$ROWCACHEビューのデータ・ディクショナリ列

列	説明
PARAMETER	特定のデータ・ディクショナリ項目を識別します。各行で、この列の値は接頭辞 dc_が付いた項目です。たとえば、ファイル記述の統計を含む行では、この列の値は dc_files です。
GETS	対応する項目に関する情報へのリクエストの総数を示します。たとえば、ファイル記述の統計を含む行では、この列の値はファイル記述データへのリクエストの総数になりま

列	説明
	す。
GETMISSES	キャッシュで満たされていないデータ・リクエストで、I/O を必要とするものの個数を示します。
MODIFICATIONS	ディクショナリ・キャッシュ内のデータが更新された回数を示します。

例14-3に、アプリケーションの実行中、一定期間にわたって統計を監視するためのこのビューの問合せを示します。導出された列PCT_SUCC_GETSは、項目固有のヒット率と考えることができます。

例14-3 V$ROWCACHEビューの問合せ

```
column parameter format a21
column pct_succ_gets format 999.9
column updates format 999,999,999
SELECT parameter,
       sum(gets),
       sum(getmisses),
       100*sum(gets - getmisses) / sum(gets) pct_succ_gets,
       sum(modifications) updates
  FROM V$ROWCACHE
 WHERE gets > 0
 GROUP BY parameter;
```

この問合せの出力例を次に示します。

```
PARAMETER              SUM(GETS) SUM(GETMISSES) PCT_SUCC_GETS      UPDATES
-------------------- ---------- -------------- ------------- ------------
dc_database_links            81              1          98.8            0
dc_free_extents           44876          20301          54.8       40,453
dc_global_oids               42              9          78.6            0
dc_histogram_defs          9419            651          93.1            0
dc_object_ids             29854            239          99.2           52
dc_objects                33600            590          98.2           53
dc_profiles               19001              1         100.0            0
dc_rollback_segments      47244             16         100.0           19
dc_segments              100467          19042          81.0       40,272
dc_sequence_grants          119             16          86.6            0
dc_sequences              26973             16          99.9       26,811
dc_synonyms                6617            168          97.5            0
dc_tablespace_quotas        120              7          94.2           51
dc_tablespaces           581248             10         100.0            0
dc_used_extents           51418          20249          60.6       42,811
dc_user_grants            76082             18         100.0            0
dc_usernames             216860             12         100.0            0
dc_users                 376895             22         100.0            0
```

この例の出力は、次のことを示しています。

- 使用済エクステント、空きエクステントおよびセグメントには、多数のミスと更新があります。つまり、データベース・インスタンスに大量の動的な領域の拡張があったことを示しています。

- 取得成功率と実際の取得数の比較により、共有プールがディクショナリ・キャッシュ・データを適切に格納できる大きさがあることが示されています。

また、次の問合せを使用して、総合的ディクショナリ・キャッシュ・ヒット率を計算できますが、すべてのキャッシュにわたるデータを合計すると、より細かいデータの粒度は失われます。

```
SELECT (SUM(gets - getmisses - fixed)) / SUM(gets) "row cache"
  FROM V$ROWCACHE;
```

共有プールに割り当てられたメモリーの増加

共有プールのメモリー量を増やすと、ライブラリ・キャッシュ、ディクショナリ・キャッシュおよび結果キャッシュで使用可能なメモリー量も増加します。実行する前に、共有プールの統計を確認し、次のことを調査してください。

- V$LIBRARYCACHE.RELOADS列の値がゼロに近いかどうか
- ディクショナリ・キャッシュが頻繁にアクセスされる場合、V$ROWCACHE.GETMISSES列の合計に対するV$ROWCACHE.GETSの合計の割合が、10%または15%を下回っているかどうか(アプリケーションによって異なります)

これらのどちらの条件も満たしている場合は、共有プールが適切にサイズ設定されており、メモリーを増やしてもパフォーマンスの向上は見込めません。一方で、これらのどちらかの条件を満たしていない場合は、アプリケーションが「共有プールの使用」の説明に従って効率的に共有プールを使用している場合でも、共有プールのメモリーを増やすことを検討してください。

共有プールのサイズを大きくするには:

- 条件を満たすまで、SHARED_POOL_SIZE初期化パラメータの値を増やします。

  このパラメータの最大値はオペレーティング・システムによって異なります。この処置によって、実行のためのSQL文とPL/SQLブロックの暗黙的な再解析が減少します。

IC - 「初期化パラメータによるサーバーの結果キャッシュ・メモリーの管理」へのリンクが必要

共有プールに割り当てられたメモリーの低減

V$LIBRARYCACHE.RELOADS列の値がゼロに近く、共有プールの空きメモリー量が少ない場合、共有プールは最も頻繁にアクセスされるデータを格納するよう適切にサイズ設定されています。共有プールの空きメモリー量が多く、このメモリーを他の場所に割り当てる場合は、共有プールのサイズを小さくすることを検討してください。

共有プールのサイズを小さくするには、次のようにします。

- パフォーマンスが良好な状態に維持されていることを確認しながら、SHARED_POOL_SIZE初期パラメータの値を減らします。

カーソルの割当て解除

ライブラリ・キャッシュ・ミスがない場合は、CURSOR_SPACE_FOR_TIME初期化パラメータの値をTRUEに設定して、実行コールを高速化することを検討してください。このパラメータは、新しいSQL文の領域を作成するために、ライブラリ・キャッシュからカーソルの割当てを解除するかどうかを指定します。

CURSOR_SPACE_FOR_TIMEパラメータに設定されている値によって、次のようになります。

- FALSE(デフォルト)に設定されていると、SQL文に対応付けられているアプリケーション・カーソルがオープンされているか

どうかにかかわらず、ライブラリ・キャッシュからカーソルの割当てを解除できます。

この場合は、SQL文を含むカーソルがライブラリ・キャッシュ内にあることを検証する必要があります。

- TRUEに設定すると、その文に関連するすべてのアプリケーション・カーソルがクローズされる場合のみカーソルの割当てを解除できます。

  この場合、関連するアプリケーション・カーソルがオープンしている間はそのカーソルの割当てを解除できないため、カーソルがライブラリ・キャッシュ内にあるかどうかを確認する必要はありません。

パラメータの値をTRUEに設定することで、Oracle Database側の時間が少し短縮されるので、いくらか実行コールのパフォーマンスが改善する可能性があります。この値は、対応付けられているアプリケーション・カーソルがクローズされるまでカーソルの割当て解除も防ぎます。

次の場合は、CURSOR_SPACE_FOR_TIMEパラメータの値をTRUEに設定しないでください。

- 実行コールでライブラリ・キャッシュ・ミスが見つかった場合

  ライブラリ・キャッシュ・ミスは、共有プールが十分大きくないので同時にオープンしている全カーソルの共有SQL領域を保持できないことを示しています。共有プールに新しいSQL文のための領域が十分になく、このパラメータの値がTRUEに設定されている場合は、文が解析されず、Oracle Databaseにより、共有メモリーが不足していることを示すエラーが戻されます。

- 各ユーザーがプライベートSQL領域に使用可能なメモリー量が少ない場合

  また、この値は、オープンしているカーソルに対応付けられているプライベートSQL領域の割当て解除も防ぎます。同時にオープンしているすべてのカーソルのプライベートSQL領域によって使用可能メモリーがいっぱいになり、新しいSQL文の領域がない場合は、文が解析されず、Oracle Databaseにより、メモリー不足を示すエラーが戻されます。

共有プールに新しいSQL文のための領域が十分になく、このパラメータの値がFALSEに設定されている場合は、Oracle Databaseにより、既存のカーソルの割当てが解除されます。カーソルの割当てを解除するとライブラリ・キャッシュ・ミスが後で発生しますが(カーソルが再度実行される場合)、SQL文を解析できないためにアプリケーションが停止するというエラーよりも望ましい対処と言えます。

セッション・カーソルのキャッシュ

セッション・カーソル・キャッシュには、SQLおよびPL/SQL(再帰的SQLを含む)のクローズされたセッション・カーソルが含まれます。フォーム間で切替えを行うと、最初のフォームに関連するすべてのセッション・カーソルがクローズされるため、このキャッシュは、Oracle Formsが使用されているアプリケーションで有用です。アプリケーションから何度も同じSQL文で解析コールが発行される場合、セッション・カーソルの再オープンによりパフォーマンスが低下することがあります。カーソルを再使用することにより、データベースの分析時間が削減されるため、結果として全体的な実行時間が短縮されます。

この項では、セッション・カーソル・キャッシュについて説明しており、内容は次のとおりです。

- セッション・カーソル・キャッシュについて
- セッション・カーソル・キャッシュの有効化
- セッション・カーソル・キャッシュのサイズ設定

セッション・カーソル・キャッシュについて

セッション・カーソルは、共有子カーソルのインスタンス化を表し、特定のセッションの共有プールに格納されます。各セッション・

カーソルには、インスタンス化した子カーソルへの参照が格納されます。

Oracle Databaseでは、ライブラリ・キャッシュをチェックして、特定の文に対して3回以上の解析リクエストが発行されたかどうかを識別します。カーソルが3回クローズされている場合は、文に関連するセッション・カーソルをキャッシュする必要があると推定し、カーソルをセッション・カーソル・キャッシュに移動します。

同じセッションでSQL文の解析リクエストが続けて出されると、共有カーソルへのポインタの配列が検索されます。ポインタが見つかると、データベースはポインタを逆参照して共有カーソルが存在するかどうか特定します。カーソルをキャッシュから再使用するために、キャッシュ・マネージャはキャッシュされたカーソルの状態が現在のセッションとシステム環境に適合するかどうか確認します。

ノート:

キャッシュ・カーソルの再使用はハード解析ではありませんが、まだ解析として登録されています。

LRUのアルゴリズムでは、必要に応じてセッション・カーソル・キャッシュ内の項目を除去し、新しい項目のための空間を作成します。また、キャッシュは内部の時間ベースのアルゴリズムを使用して、一定時間アイドル状態になっているカーソルを除去します。

セッション・カーソル・キャッシュの有効化

次の初期化パラメータは、セッション・カーソル・キャッシュに関連します。

- SESSION_CACHED_CURSORS

  このパラメータは、1セッション当たりのクローズされたキャッシュ・カーソルの最大数を設定します。デフォルト値は50です。このパラメータは、同じセッションで繰り返し実行される文について、キャッシュからカーソルを再利用する場合に使用します。

- OPEN_CURSORS

  このパラメータは、セッションで同時に開くことができるカーソルの最大数を指定します。たとえば、値を1000に設定すると、各セッションは一度に最大1000個のカーソルを開くことができます。

これらのパラメータは独立しています。たとえば、SESSION_CACHED_CURSORSパラメータの値をOPEN_CURSORSパラメータの値より大きく設定できますが、これは、セッション・カーソルがオープンの状態でキャッシュされないためです。

セッション・カーソル・キャッシュを有効化するには:

1. キャッシュに保持するセッション・カーソルの最大数を決定します。
2. 次のいずれかの操作を行います。
   - 静的キャッシュを有効化するには、SESSION_CACHED_CURSORSパラメータの値を前のステップで決定した数値に設定します。
   - 動的キャッシュを有効化するには、次の文を実行します。

     ```
     ALTER SESSION SET SESSION_CACHED_CURSORS = value;
     ```

セッション・カーソル・キャッシュのサイズ設定

セッション・カーソル・キャッシュがデータベース・インスタンスに対して適切にサイズ設定されているかどうかを確認するには、V$SESSTATビューを使用します。

セッション・カーソル・キャッシュのサイズを設定するには:

1. V$SESSTATビューを問い合せて、現在特定のセッションにキャッシュされているカーソルの数を特定します。
2. V$SESSTATビューを問い合せて、セッション・カーソル・キャッシュでカーソルが検出された解析コールの割合を取得します。
3. 次の内容に当てはまる場合は、SESSION_CACHED_CURSORSパラメータの値を増やすことを検討してください。
    - セッション・カーソル・キャッシュ数が最大値に近い
    - セッション・カーソル・キャッシュのヒットの割合が合計解析数と比較して低い
    - アプリケーションが同じ問合せに対して繰り返し解析コールを実行する

例14-4に、このビューの問合せを2つ示します。

例14-4 V$SESSTATビューの問合せ

次の問合せでは、現在特定のセッションにキャッシュされているカーソルの数がわかります。

```
SELECT a.value curr_cached, p.value max_cached,
       s.username, s.sid, s.serial#
  FROM v$sesstat a, v$statname b, v$session s, v$parameter2 p
 WHERE a.statistic# = b.statistic#  and s.sid=a.sid and a.sid=&sid
   AND p.name='session_cached_cursors'
   AND b.name = 'session cursor cache count';
```

この問合せの出力例を次に示します。

```
CURR_CACHED MAX_CACHED USERNAME   SID    SERIAL#
----------- ---------- -------- ----- ----------
         49 50         APP         35        263
```

この結果は、セッション35で現在キャッシュされているカーソルの数が最大値に近いことを示しています。

次の問合せでは、セッション・カーソル・キャッシュでカーソルが検出された解析コールの割合がわかります。

```
SELECT cach.value cache_hits, prs.value all_parses,
       round((cach.value/prs.value)*100,2) as "% found in cache"
  FROM v$sesstat cach, v$sesstat prs, v$statname nm1, v$statname nm2
 WHERE cach.statistic# = nm1.statistic#
   AND nm1.name = 'session cursor cache hits'
   AND prs.statistic#=nm2.statistic#
   AND nm2.name= 'parse count (total)'
   AND cach.sid= &sid and prs.sid= cach.sid;
```

この問合せの出力例を次に示します。

```
CACHE_HITS ALL_PARSES % found in cache
---------- ---------- ----------------
        34        700             4.57
```

この結果は、セッション35のセッション・カーソル・キャッシュでのヒット数が、解析の合計数と比較して低いことを示しています。

この例では、SESSION_CACHED_CURSORSパラメータの値を100に設定すると、パフォーマンスが向上する可能性があります。

カーソルの共有

SQL解析の内容では、同一文とは、テキストが他の文と空白、大文字小文字の区別、コメントも含めて完全に同一であるSQL文を指します。類似文とは、一部のリテラル値を除いて同一である文です。

解析フェーズでは、文のテキストを共有プールの文と比較して、その文を共有できるかどうかを判断します。CURSOR_SHARING初期化パラメータの値がEXACT (デフォルト値)に設定されていて、共有プールの文が同一でない場合、データベースではSQL領域は共有されません。かわりに、各SQL文には、その文のリテラルに基づいた独自の親カーソルと独自の実行計画があります。

この項では、カーソルの共有方法を説明しており、内容は次のとおりです。

- カーソル共有について
- カーソル共有の強制

カーソル共有について

SQL文がバインド変数よりもリテラルを使用する場合、CURSOR_SHARING初期化パラメータの値をFORCEに設定すると、データベースはリテラルをシステム生成のバインド変数と置き換えることができます。この方法を使用すると、データベースで共有SQL領域の親カーソルの数を低減できる場合があります。

CURSOR_SHARINGパラメータの値がFORCEに設定されている場合、解析フェーズにおいて、データベースで次のステップが実行されます。

1. 共有プールでの同一文の検索。

   同一文が見つかった場合、データベースは次のステップをスキップし、ステップ3に進みます。そうでない場合は、データベースは次のステップに進みます。

2. 共有プールでの類似文の検索。

   類似文が見つからない場合は、データベースはハード解析を実行します。類似文が見つかった場合は、データベースは次のステップに進みます。

3. 続けて解析フェーズの残りのステップを行い、既存の文の実行計画を新しい文に適用できることを確認します。

   計画を適用できない場合は、データベースはハード解析を実行します。計画が適用できる場合は、データベースは次のステップに進みます。

4. 文のSQL領域の共有。

データベースにより実行される様々なチェックの詳細は、「SQL共有基準」を参照してください。

カーソル共有の強制

ベスト・プラクティスは、共有可能なSQLを記述し、CURSOR_SHARING初期化パラメータにデフォルト値のEXACTを使用することです。Oracle Databaseではデフォルトで、適応カーソル共有機能が使用され、バインド変数を含む1つのSQL文が複数の実行計画を使用できるようになります。ただし、バインド変数ではなくリテラルを使用する類似文が多数あるアプリケーションでは、CURSOR_SHARINGパラメータの値をFORCEに設定するとカーソル共有が改善され、結果としてメモリー使用量が減少し、解析時間が削減され、ラッチの競合が減少します。共有プールの文がリテラルの値のみ異なる場合で、ライブラリ・キャッシュ・ミスの回数が多いためにレスポンス時間が遅くなっている場合には、このアプローチを検討してください。この場合は、CURSOR_SHARINGパラ

メータの値をFORCEに設定することによりカーソル共有が最大化され、適応カーソル共有を使用して、異なるリテラル値の範囲に基づき、複数の実行パスを生成できます。

CURSOR_SHARINGパラメータの値をEXACTに設定してストアド・アウトラインを生成すると、データベースはリテラルを使用して生成されたストアド・アウトラインを使用しません。この問題を回避するには、CURSOR_SHARINGをFORCEに設定してアウトラインを生成し、CREATE_STORED_OUTLINESパラメータを使用します。

CURSOR_SHARINGパラメータの値をFORCEに設定すると、次のデメリットがあります。

- データベースは、共有プール内で類似文を検索するために、ソフト解析の間に追加作業を実行する必要があります。
- SELECT文にリテラルを含む選択された式の最大長(DESCRIBEからの戻り値)が増加します。ただし、戻されたデータの実際の長さは変わりません。
- スター型変換はサポートされません。

CURSOR_SHARINGパラメータの値をFORCEに設定すると、データベースは各個別のSQL文に1つの親カーソルと1つの子カーソルを使用します。同じ文の各実行に同じ計画が使用されます。たとえば、次のSQL文について検討します。

```
SELECT *
  FROM hr.employees
 WHERE employee_id = 101;
```

CURSOR_SHARINGパラメータの値がFORCEに設定されている場合、データベースはこの文をバインド変数を含むかのように最適化し、バインド照合を使用してカーディナリティを予測します。

ノート:

Oracle Database 11g リリース 2 以上では、CURSOR_SHARING パラメータの値を SIMILAR に設定することは推奨されていません。かわりに、適応カーソル共有を使用することを検討してください。

関連項目:

- CURSOR_SHARING初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください
- 適応カーソル共有の詳細は、Oracle Database SQLチューニング・ガイドを参照してください

除去防止のためのラージ・オブジェクトの保存

エントリが共有プールにロードされた後は、それを移動することはできません。エントリがロードされ、除去されると、空きメモリーが断片化されることもあります。共有SQL領域と共有PL/SQL領域は、データベース・バッファに類似のLRUアルゴリズムに従って共有プールから除去されます。パフォーマンスを改善したり、再解析が行われないようにするために、サイズの大きいSQL領域またはPL/SQL領域が古くなって共有プールから除去されないようにすることが可能です。

DBMS_SHARED_POOLパッケージを使用すると、オブジェクトが共有メモリー内に維持されるため、これらのオブジェクトは通常のLRUメカニズムによって除去されることはありません。DBMS_SHARED_POOLパッケージを使用し、SQL領域とPL/SQL領域をメモリーの断片化が発生する前にロードすると、オブジェクトはメモリー内に維持されます。こうすることによって、メモリーが確実に使用

可能になり、SQL領域とPL/SQL領域の除去後にこれらの領域にアクセスする場合に、ユーザーのレスポンス時間が突然低下するのを防ぐことができます。

次のような場合には、DBMS_SHARED_POOLパッケージの使用を検討してください。

- STANDARDやDIUTILパッケージなどの大きなPL/SQLオブジェクトをロードする場合。

  大きなPL/SQLオブジェクトがロードされる場合で、大きなオブジェクト用に領域を確保するために小さなオブジェクトを共有プールから除去する必要がある場合は、ユーザーのレスポンス時間に影響が現れます。場合によっては、ラージ・オブジェクトをロードするためのメモリーが不足している場合があります。

- コンパイルしたトリガーを共有プールで頻繁に使用される表に維持する場合。
- 順序をサポートする場合。

  共有プールから順序が除去されると、順序番号が失われます。DBMS_SHARED_POOLパッケージは、共有プール内に順序を保持し、順序番号の消失を防ぎます。

SQL領域またはPL/SQL領域を共有メモリー内に維持するには:

1. メモリー内に確保しておくパッケージまたはカーソルを決定します。
2. データベースを起動します。
3. DBMS_SHARED_POOL.KEEPパッケージをコールしてオブジェクトを確保します。

   この手順により、確保されているオブジェクトがロードされる前にシステムの共有メモリーが不足しないことが保証されます。オブジェクトをデータベース・インスタンスの早い時期に確保することにより、大きなメモリー領域を共有プールの中央に保持するために発生する可能性のある、メモリーの断片化を防ぐことができます。

関連項目:

DBMS_SHARED_POOLプロシージャの使用方法の詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

予約プールの構成

Oracle Databaseでは、非常に大きいメモリーのリクエストは小さいチャンクに分割されますが、システムによっては、(デフォルトの最小予約プール割当てが4,400バイトであるのに対して5KBを超えるような)メモリーの連続チャンクの検索が必要な場合があります。

共有プールに十分な空き領域がない場合は、このリクエストを満たすための十分な空きメモリーを検索する必要があります。この操作では、長期間にわたってラッチ・リソースが保持されるため、メモリー割当てで他の同時動作に対して多少の影響が生じる可能性があります。

これを回避するため、共有プールに十分な領域がない場合に、Oracle Databaseが使用できる小さいメモリー領域がデフォルトで、共有プールに内部的に予約されます。この予約プールによって、大きいチャンクの割当てがより効率的に行われます。このメモリーは、PL/SQLおよびトリガーのコンパイルなどの操作や、Javaオブジェクトのロード時の一時領域に使用できます。予約プールから割り当てられたメモリーが解放されると、予約プールに戻されます。

大きい割当ての場合、Oracle Databaseは次の順序で共有プールへの領域の割当てを試行します。

1. 共有プールの予約されていない部分。
2. 予約プール。

   共有プールの予約されていない部分に十分な領域がない場合は、予約プールに十分な領域があるかどうかチェックされます。
3. メモリー。

   共有プールの予約されていない部分と予約された部分に十分な領域がない場合は、Oracle Databaseは割当てのために十分なメモリーの解放を試みます。次に、データベースにより、共有プールの予約されていない部分と予約されている部分が再試行されます。

この項では、予約プールの構成方法を説明しており、内容は次のとおりです。

- 予約プールのサイズ設定
- 予約プールに割り当てられたメモリーの増加
- 予約プールに割り当てられたメモリーの低減

予約プールのサイズ設定

通常、Oracle Databaseが予約プールで予約するデフォルトの領域量を変更する必要はありません。ただし、極端に大きなメモリー割当てのために、共有プールに領域を確保しておくことが必要な場合もあります。

予約プール・サイズは、SHARED_POOL_RESERVED_SIZE初期化パラメータの値を設定することで指定できます。SHARED_POOL_RESERVED_SIZEパラメータのデフォルト値は、SHARED_POOL_SIZEパラメータの5%です。

予約プールのメモリーを過剰に予約することはできないため、SHARED_POOL_RESERVED_SIZEパラメータの値をSHARED_POOL_SIZEパラメータの半分を超える値に設定すると、Oracle Databaseからエラーが戻されます。使用可能なオペレーティング・システムのメモリー容量も共有プールのサイズを制約する場合があります。一般的に、SHARED_POOL_RESERVED_SIZEパラメータの値は、SHARED_POOL_SIZEパラメータの10%以下に設定します。ほとんどのシステムにおいて、共有プールが適切にチューニングされていれば、この値で十分です。この値を大きくすると、データベースにより追加のメモリーが共有プールから取得され、より小さな割当てに使用可能な予約されていない共有プールのメモリー量が減少します。

これらのパラメータをチューニングする際は、V$SHARED_POOL_RESERVEDビューの統計を使用します。SGAのサイズを大きくするための空きメモリーが豊富にあるシステムでは、REQUEST_MISSES統計の値をゼロにする必要があります。オペレーティング・システム・メモリーによる制約があるシステムの場合は、REQUEST_FAILURES統計をゼロにするか、少なくともこの値が増加しないようにすることが目標です。これらの目標値を達成できない場合は、SHARED_POOL_RESERVED_SIZEパラメータの値を増やしてください。また、予約リストは共有プールから取られるため、SHARED_POOL_SIZEパラメータの値も同じだけ増やします。

V$SHARED_POOL_RESERVED固定ビューを使用すると、SHARED_POOL_SIZEパラメータの値が小さすぎる場合を示すこともできます。これは、REQUEST_FAILURES統計がゼロより大きい場合および増加しているような場合です。予約リストが有効化されている場合は、SHARED_POOL_RESERVED_SIZEパラメータの値を減らします。予約リストが有効化されていない場合は、「共有プールに割り当てられたメモリーの増加」で説明されているように、SHARED_POOL_SIZEパラメータの値を増やします。

予約プールに割り当てられたメモリーの増加

REQUEST_FAILURES統計の値がゼロより大きい場合および増加している場合は、予約プールが小さすぎます。この場合は、予約プールに使用可能なメモリー量を増やします。

ノート:

予約リストで使用可能なメモリー量を増やしても、予約リストからメモリーを割り当てないユーザーに影響はありません。

予約プールのサイズを大きくするには:

- SHARED_POOL_RESERVED_SIZEおよびSHARED_POOL_SIZE初期化パラメータの値を適宜増やします。

  これらのパラメータに関して選択する値は、「予約プールのサイズ設定」で説明されているように、システムのSGAのサイズ制約によって異なります。

予約プールに割り当てられたメモリーの低減

次のような場合は、予約プールが大きすぎます。

- REQUEST_MISSES統計がゼロの場合または増加しない場合
- FREE_SPACE統計の最小値がSHARED_POOL_RESERVED_SIZEの50%以上になる場合

これらの条件のいずれかに当てはまる場合は、予約プールに使用可能なメモリー量を減らします。

予約プールのサイズを小さくするには:

- SHARED_POOL_RESERVED_SIZE初期化パラメータの値を減らします。

ラージ・プールの構成

共有プールとは異なり、ラージ・プールにはLRUリストがありません。Oracle Databaseは、ラージ・プールからオブジェクトを除去しようとしません。データベース・インスタンスがOracle Databaseの次の機能のいずれかを使用する場合は、ラージ・プールの構成を検討してください。

- 共有サーバー

  共有サーバー・アーキテクチャでは、各クライアント・プロセスのセッション・メモリーが共有プールに含まれています。

- パラレル問合せ

  パラレル問合せでは、共有プール・メモリーを使用してパラレル実行メッセージ・バッファをキャッシュします。

- Recovery Manager

  Recovery Manager (RMAN)は、バックアップおよびリストア操作時に共有プールを使用してI/Oバッファをキャッシュします。I/Oサーバー・プロセス、バックアップおよびリストア操作では、Oracle Databaseは数百KB単位でバッファを割り当てます。

この項では、共有サーバー・アーキテクチャにラージ・プールを構成する方法を説明しており、内容は次のとおりです。

- 共有サーバー・アーキテクチャでのラージ・プールの構成
- パラレル問合せ用のラージ・プールの構成
- ラージ・プールのサイズ設定
- ユーザー・セッションへのメモリー使用の制限
- 3層の接続を使用したメモリー使用の低減

関連項目:

- ラージ・プールの詳細は、『Oracle Database概要』を参照してください
- Recovery Managerを使用する場合のラージ・プールのサイズ設定の詳細は、『Oracle Databaseバックアップおよびリカバリ・ユーザーズ・ガイド』を参照してください

共有サーバー・アーキテクチャでのラージ・プールの構成

Oracle Databaseでは共有サーバー・セッションに共有プールからメモリーを割り当てるため、ライブラリ・キャッシュとデータ・ディクショナリ・キャッシュに使用可能な共有プール・メモリーの量が減少します。別のプールから共有サーバー・セッション・メモリーを割り当てると、共有SQLのキャッシュ用に共有プールを予約できます。

共有サーバー・アーキテクチャのユーザー・グローバル領域(UGA)の割当てには、ラージ・プールの使用をお薦めします。共有プールのかわりにラージ・プールを使用すると、共有プールの断片化が減少し、共有SQLキャッシュの縮小によるパフォーマンス・オーバーヘッドがなくなります。

デフォルトでは、ラージ・プールは構成されません。ラージ・プールを構成しない場合、共有プールは、共有サーバーのユーザー・セッション・メモリーに使用されます。共有サーバー・アーキテクチャを使用する際は、ラージ・プールを構成した場合でも、Oracle Databaseでは、共有プールから各構成セッションに一定量のメモリー(約10KB)が割り当てられます。どちらの場合も、共有プールのサイズを適宜大きくすることを検討してください。

ノート:

共有サーバーの使用により共有メモリーの使用が増加するとしても、合計のメモリー使用量は減少します。これは、プロセス数が減少するので、専用サーバー環境と比較した場合に共有サーバーでは PGA メモリーの使用量が減るためです。

ヒント:

データベースで許可される同時共有サーバー・セッションの最大数を指定するには、CIRCUITS 初期化パラメータを使用します。

ヒント:

共有サーバーを使用したソート操作でパフォーマンスが最大になるよう、SORT_AREA_SIZE および SORT_AREA_RETAINED_SIZE 初期化パラメータの値を同じ値に設定します。これによって、ソート結果をディスクに書き込むのではなくラージ・プールに留めておけます。

パラレル問合せ用のラージ・プールの構成

パラレル問合せは、自動メモリー管理または自動共有メモリー管理が有効ではない場合、共有プール・メモリーを使用してパラレル実行メッセージ・バッファをキャッシュします。パラレル実行メッセージ・バッファの共有プールでのキャッシュによってそのワークロードが増加し、断片化が引き起こされるおそれがあります。

パフォーマンスへのマイナスの影響をできるかぎり避けるため、パラレル問合せの使用時は、SGAメモリーを手動で管理しないことをお薦めします。かわりに、自動メモリー管理または自動共有メモリー管理を有効にして、パラレル実行メモリー・バッファのキャッシュにラージ・プールが使用されるようにします。

関連項目:

- 自動メモリー管理
- 自動共有メモリー管理
- 『Oracle Database VLDBおよびパーティショニング・ガイド』

ラージ・プールのサイズ設定

ラージ・プールに共有サーバー関連のUGAを格納する際は、Oracle Databaseにより使用されるUGAの正確な量はアプリケーションによって異なります。各アプリケーションは、必要なセッション情報メモリー量がそれぞれ異なり、ラージ・プールの構成はメモリー要件を反映する必要があります。

Oracle Databaseでは、セッションに使用されたメモリーの統計が収集され、V$SESSTATビューに格納されます。表14-2に、セッションUGAメモリーを反映するこのビューの統計を示します。

表14-2 V$SESSTATビューのメモリー統計

統計	説明
session UGA memory	セッションに割り当てられたメモリー量(バイト単位)が表示されます。
session UGA memory max	これまでにセッションに割り当てられた最大メモリー量(バイト単位)が表示されます。

このビューを使用してラージ・プールの適切なサイズを判断する方法は2つあります。1つは、同時にアクティブとなるセッションの数を基準にラージ・プールのサイズを構成する方法です。これを実行するには、一般的なユーザーのUGAメモリー使用状況を観察し、その量にユーザー・セッションの見積り数を乗算します。たとえば、一般的なユーザー・セッションのセッション情報を格納するために、共有サーバーに200Kから300Kが必要で、同時にアクティブになるユーザー・セッションが100になると見込まれる場合、ラージ・プールを30MBに構成します。

もう1つは、すべてのユーザー・セッションによって使用される合計メモリーおよび最大メモリーを計算する方法です。例14-5に、これを実行するためのV$SESSTATビューおよびV$STATNAMEビューにおける2つの問合せを示します。

例14-5 V$SESSTATビューおよびV$STATNAMEビューの問合せ

アプリケーションの実行中に、次の問合せを発行します。

```
SELECT SUM(value) || ' bytes' "total memory for all sessions"
  FROM V$SESSTAT, V$STATNAME
 WHERE name = 'session uga memory'
   AND V$SESSTAT.STATISTIC# = V$STATNAME.STATISTIC#;

SELECT SUM(value) || ' bytes' "total max mem for all sessions"
  FROM V$SESSTAT, V$STATNAME
 WHERE name = 'session uga memory max'
   AND V$SESSTAT.STATISTIC# = V$STATNAME.STATISTIC#;
```

また、これらの問合せでは、V$STATNAMEビューから選択して、session memoryとmax session memoryの内部識別子を取得します。

これらの問合せの出力例を次に示します。

```
TOTAL MEMORY FOR ALL SESSIONS
-----------------------------
157125 BYTES

TOTAL MAX MEM FOR ALL SESSIONS
------------------------------
417381 BYTES
```

最初の問合せの結果は、現在、全セッションに割り当てられているメモリーは157,125バイトであることを示しています。この値は、セッションがデータベースに接続されている方法にその位置が依存するメモリーの全体量です。セッションが専用サーバー・プロセスで接続されている場合、このメモリーはユーザー・プロセスのメモリーの一部です。セッションが共有サーバー・プロセスで接続されている場合、このメモリーは共有プールの一部です。

2番目の問合せの結果は、全セッションのメモリーの最大サイズの合計が417,381バイトであることを示しています。2番目の結果は、いくつかのセッションが最大の容量を割り当てた後でメモリーを割当て解除したため、最初の結果よりも大きくなっています。

いずれかの問合せの結果を使用して、共有プールの適切なサイズを判断します。全セッションが同時にそれらの最大割当てに到達すると予測する場合を除き、2番目の値よりも最初の値の方がより適切な見積りになります。

ラージ・プールのサイズを設定するには:

1. V$SGASTATビューでPOOL列をチェックし、オブジェクト用のメモリーが存在するプール(共有プールまたはラージ・プール)を確認します。
2. LARGE_POOL_SIZE初期化パラメータの値を設定します。

   このパラメータの最小値は300Kです。

ユーザー・セッションへのメモリー使用の制限

各クライアント・セッションによるSGAのメモリー使用量を制限するには、PRIVATE_SGAを使用してリソース制限を設定します。

PRIVATE_SGAによって、1セッションでSGAから使用されるメモリーのバイト数が定義されます。ただし、ほとんどのDBAはユーザー単位でのSGA消費量の制限は行わないため、このパラメータを使用することはほとんどありません。

関連項目:

PRIVATE_SGAリソース制限の設定の詳細は、『Oracle Database SQL言語リファレンス』を参照してください

3層の接続を使用するメモリー使用の低減

接続ユーザーが非常に多数の場合は、3層の接続を実装してメモリー使用を低減することを検討してください。ロックやコミットされていないDML操作を複数のコールにわたって保持できないため、トランザクション処理(TP)モニターの使用は、純粋なトランザクション・モデルでしか実現できません。

共有サーバー環境を使用すると、次のようになります。

- TPモニターに比べてアプリケーション設計の制限が大幅に少なくなります。
- ユーザーがサーバーのプールを共有できるので、オペレーティング・システム・プロセス数とコンテキストの切替えが大幅に減ります。
- 共有サーバー・モードでさらに多くのSGAが使用される場合でも総メモリー使用量が大幅に減ります。

15 結果キャッシュのチューニング

この章では、結果キャッシュのチューニング方法を説明しており、内容は次のとおりです。

- 結果キャッシュについて
- 結果キャッシュの構成
- 結果をキャッシュする問合せの指定
- 結果キャッシュの監視

結果キャッシュについて

結果キャッシュは共有グローバル領域(SGA)またはクライアント・アプリケーション・メモリー内のメモリー領域で、データベースの問合せまたは問合せブロックの結果を再利用するために格納します。キャッシュされた行は、失効しないかぎりSQL文およびセッション間で共有されます。

この項では、2つのタイプの結果キャッシュについて説明しており、内容は次のとおりです。

- サーバー結果キャッシュの概念
- クライアント結果キャッシュの概念

サーバー結果キャッシュの概念

サーバーの結果キャッシュは共有プール内のメモリー・プールです。このメモリー・プールは、SQL問合せの結果を格納するSQL問合せの結果キャッシュと、PL/SQLファンクションによって戻される値を格納するPL/SQLファンクションの結果キャッシュで構成されています。

この項では、サーバー結果キャッシュについて説明しており、内容は次のとおりです。

- サーバー結果キャッシュを使用する利点
- サーバー結果キャッシュの機能について

関連項目:

- サーバー結果キャッシュの詳細は、『Oracle Database概要』を参照してください
- PL/SQLファンクション結果キャッシュの詳細は、『Oracle Database PL/SQL言語リファレンス』を参照してください

サーバー結果キャッシュを使用する利点

サーバー結果キャッシュを使用する利点は、アプリケーションによって異なります。OLAPアプリケーションでは、使用により多大なメリットを得ることができます。たとえばデータ・ウェアハウスのように、アクセスする行数が多く、戻す行数の少ない問合せはキャッシュの対象として適しています。たとえば、表を使用するかわりに、同値化を伴う拡張クエリー・リライトを使用して、結果キャッシュに問合せをマテリアライズするマテリアライズド・ビューを作成できます。

関連項目:

結果キャッシュおよび同値化を伴う拡張問合せリライトの使用方法の詳細は、『Oracle Databaseデータ・ウェアハウス・ガイド』を参照してください

サーバー結果キャッシュの機能について

問合せを実行すると、データベースはキャッシュ・メモリーを検索してその結果が結果キャッシュに存在するかどうかを判断します。結果が存在する場合、問合せを実行せずにメモリーから結果を取得します。結果がキャッシュされていない場合、データベースは問合せを実行して結果を出力として戻し、その結果を結果キャッシュに格納します。

ユーザーが問合せとファンクションを繰り返し実行する場合、データベースはキャッシュから行を取得するためレスポンス時間が短縮されます。依存するデータベース・オブジェクトのデータが変更された場合、キャッシュされた結果は無効になります。

次の各項には、サーバー結果キャッシュから結果を取得する方法の例が含まれます。

- 問合せで結果が取得される仕組み
- ビューで結果が取得される仕組み

問合せで結果が取得される仕組み

次の例に、RESULT_CACHEヒントを使用して、サーバー結果キャッシュの行を取得するhr.employeesの問合せを示します。

```
SELECT /*+ RESULT_CACHE */ department_id, AVG(salary)
  FROM hr.employees
 GROUP BY department_id;
```

この問合せの実行計画の一部は次のようになります。

```
------------------------------------------------------------
| Id | Operation          | Name                        |Rows
------------------------------------------------------------
| 0 | SELECT STATEMENT    |                             | 11
| 1 |  RESULT CACHE       | 8fpza04gtwsfr6n595au15yj4y |
| 2 |   HASH GROUP BY     |                             | 11
| 3 |    TABLE ACCESS FULL| EMPLOYEES                   | 107
------------------------------------------------------------
```

実行計画のステップ1に示されているように、この例では、結果がキャッシュから直接取得されています。Name列の値は、結果のキャッシュIDです。

次の例に、キャッシュされた結果に関する詳細な統計を取得するためのV$RESULT_CACHE_OBJECTSビューの問合せを示します。

```
SELECT id, type, creation_timestamp, block_count,
       column_count, pin_count, row_count
  FROM V$RESULT_CACHE_OBJECTS
 WHERE cache_id = '8fpza04gtwsfr6n595au15yj4y';
```

この例において、CACHE_IDの値は、前述の例の実行計画から取得されたキャッシュIDです。この問合せの出力例を次に示します。

```
        ID TYPE       CREATION_ BLOCK_COUNT COLUMN_COUNT  PIN_COUNT  ROW_COUNT
---------- ---------- --------- ----------- ------------ ---------- ----------
```

```
         2 Result      06-NOV-11           1            2           0          12
```

ビューで結果が取得される仕組み

例15-1に、WITH句のビュー内でRESULT_CACHEヒントを使用する問合せを示します。

例15-1 WITHビューに指定されたRESULT_CACHEヒント

```
WITH summary AS
( SELECT /*+ RESULT_CACHE */ department_id, avg(salary) avg_sal
    FROM hr.employees
   GROUP BY department_id )
SELECT d.*, avg_sal
  FROM hr.departments d, summary s
 WHERE d.department_id = s.department_id;
```

この問合せの実行計画の一部は次のようになります。

```
-------------------------------------------------------------------------------------------------
| Id| Operation             | Name                          | Rows | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|  0| SELECT STATEMENT      |                               |   11 |   517 |     7  (29)| 00:00:01 |
|* 1|  HASH JOIN            |                               |   11 |   517 |     7  (29)| 00:00:01 |
|  2|   VIEW                |                               |   11 |   286 |     4  (25)| 00:00:01 |
|  3|    RESULT CACHE       | 8nknkh64ctmz94a5muf2tyb8r     |      |       |            |          |
|  4|     HASH GROUP BY     |                               |   11 |    77 |     4  (25)| 00:00:01 |
|  5|      TABLE ACCESS FULL| EMPLOYEES                     |  107 |   749 |     3   (0)| 00:00:01 |
|  6|   TABLE ACCESS FULL   | DEPARTMENTS                   |   27 |   567 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------
```

実行計画のステップ3に示されているように、この例では、summaryビューの結果がキャッシュから直接取得されています。Name列の値は、結果のキャッシュIDです。

クライアント結果キャッシュの概念

Oracle Call Interface (OCI)クライアントの結果キャッシュは、OCIアプリケーションのSQL問合せの結果セットをキャッシュする、クライアント・プロセス内のメモリー領域です。クライアント・キャッシュは、各クライアント・プロセスに存在し、プロセス内のすべてのセッションによって共有されます。クライアント結果キャッシュは、読取り専用またはほぼ読取り専用の表の問合せにお薦めします。

ノート:

クライアントの結果キャッシュは、SGA 内に存在するサーバーの結果キャッシュとは異なります。クライアントの結果キャッシュが使用可能な場合、クライアント、サーバー、またはその両方で問合せ結果セットをキャッシュできます。クライアントのキャッシュは、サーバーの結果キャッシュが無効の場合も使用できます。

この項では、クライアント結果キャッシュについて説明しており、内容は次のとおりです。

- クライアント結果キャッシュを使用する利点
- クライアント結果キャッシュの機能について

クライアント結果キャッシュを使用する利点

OCCI、JDBC OCIドライバ、ODP.NETなどのOCIドライバは、クライアント結果キャッシュをサポートします。クライアント結果キャッシュを使用するパフォーマンスの利点は次のとおりです。

- 問合せのレスポンス時間の短縮

  問合せが繰り返し実行される場合、アプリケーションは結果をクライアントのキャッシュ・メモリーから直接取得するため、問合せのレスポンス時間が短縮されます。

- データベース・リソースの使用効率の向上

  サーバーのラウンドトリップが減少し、サーバーのCPUやI/Oなどのサーバー・リソースのパフォーマンスが大幅に節約されます。これらのリソースは他のタスクに解放されるため、サーバーのスケーラビリティが向上します。

- メモリー・コストの削減

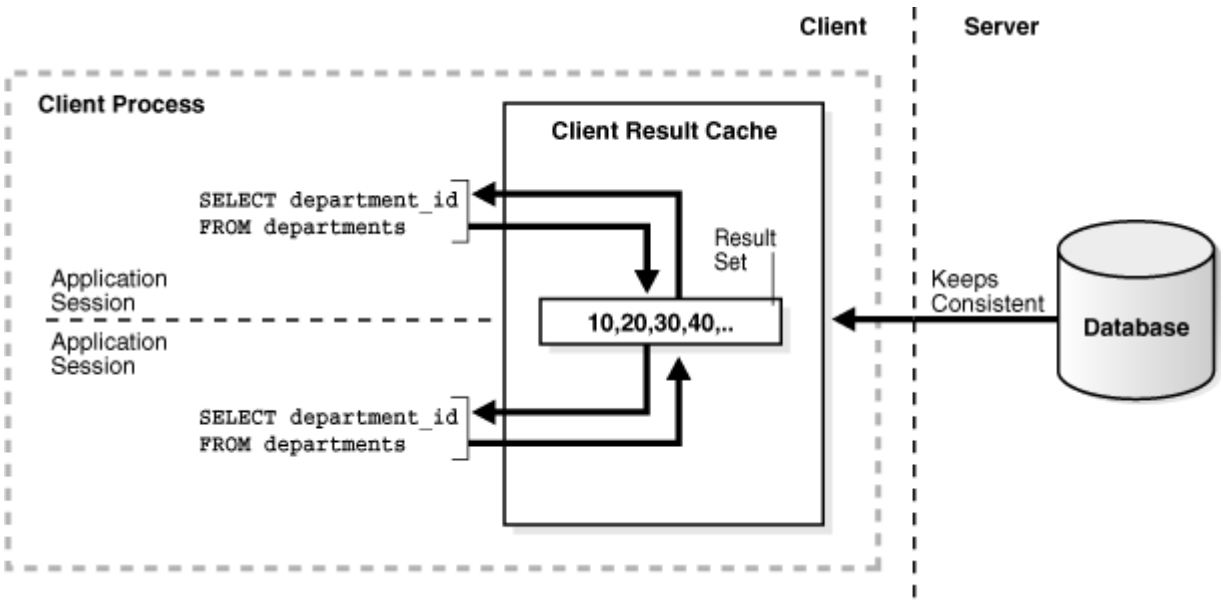
  結果キャッシュでは、サーバー・メモリーよりコストがかからないクライアント・メモリーが使用されます。

クライアント結果キャッシュの機能について

クライアントの結果キャッシュは、一番外側の問合せの結果(OCIアプリケーションで定義される列)を格納します。副問合せと問合せブロックはキャッシュされません。

次の図は、データベースのログイン・セッションのクライアント・プロセスを示しています。このクライアント・プロセスにはクライアント結果キャッシュが1つあり、これはクライアント・プロセスで実行中の複数のアプリケーション・セッション間で共有されます。最初のアプリケーション・セッションが問合せを実行すると、データベースから行が取得され、クライアントの結果キャッシュにキャッシュされます。その他のアプリケーション・セッションが同じ問合せを実行する場合も、行はクライアントの結果キャッシュから取得されます。

図15-1 クライアントの結果キャッシュ



クライアントの結果キャッシュは、セッション状態、またはセッション状態に影響を与えるデータベースの変更と一致するように結果セットを透過的に維持します。キャッシュされた結果の構築に使用するデータベース・オブジェクトのデータまたはメタデータがトランザクションによって変更される場合、データベースは、サーバーへの次のラウンドトリップ時にOCIクライアントに無効化を送信します。

関連項目:

クライアントの結果キャッシュの詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください。

結果キャッシュの構成

この項では、サーバーおよびクライアントの結果キャッシュの構成方法を説明しており、内容は次のとおりです。

- サーバー結果キャッシュの構成
- クライアント結果キャッシュの構成
- 結果キャッシュのモードの設定
- 結果キャッシュの要件

サーバー結果キャッシュの構成

デフォルトでは、Oracle Databaseによってデータベースの起動時に共有プール内のサーバーの結果キャッシュにメモリーが割り当てられます。割り当てられるメモリー・サイズは、共有プールのメモリー・サイズと選択されたメモリー管理システムに応じて変化します。

- 自動共有メモリー管理

  SGA_TARGET初期化パラメータを使用して共有プールのサイズを管理している場合は、Oracle Databaseにより、SGA_TARGETパラメータの値の0.50%が結果キャッシュに割り当てられます。

- 手動共有メモリー管理

  SHARED_POOL_SIZE初期化パラメータを使用して共有プールのサイズを管理している場合は、Oracle Databaseにより、共有プールのサイズの1%が結果キャッシュに割り当てられます。

ノート:

Oracle Database では、サーバーの結果キャッシュに共有プールの 75%を超える容量が割り当てられることはありません。

サーバー結果キャッシュのサイズは、最大サイズに達するまで大きくなります。キャッシュ内の使用可能領域よりも大きな問合せ結果は、キャッシュされません。データベースは最低使用頻度(LRU)アルゴリズムを使用してキャッシュ済の結果を除去しますが、除去しない場合、メモリーはサーバー結果キャッシュから自動的に解放されません。

この項では、サーバー結果キャッシュの構成方法を説明しており、内容は次のとおりです。

- 初期化パラメータを使用したサーバー結果キャッシュのサイズ設定
- DBMS_RESULT_CACHEを使用したサーバー結果キャッシュの管理

初期化パラメータを使用したサーバー結果キャッシュのサイズ設定

表15-1に、サーバー結果キャッシュを制御するデータベース初期化パラメータを示します。

表15-1 サーバー結果キャッシュの初期化パラメータ

パラメータ	説明
RESULT_CACHE_MAX_SIZE	サーバー結果キャッシュに割り当てられるメモリーを指定します。サーバー結果キャッシュを無効化するには、このパラメータを 0 に設定します。
RESULT_CACHE_MAX_RESULT	1 つの結果に使用可能なサーバー結果キャッシュの最大メモリー量(割合)を指定します。有効な値は 1 から 100 です。デフォルト値は 5%です。このパラメータは、システム・レベルまたはセッション・レベルで設定できます。
RESULT_CACHE_REMOTE_EXPIRATION	サーバー結果キャッシュ内で、リモート・データベース・オブジェクトに依存する結果の有効期限(分単位)を指定します。デフォルト値は 0 で、これは、リモート・オブジェクトを使用する結果がキャッシュされないことを指定します。このパラメータに 0 以外の値が設定されている場合、サーバー結果キャッシュは、リモート・データベースに対する DML では無効化されません。

関連項目:

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

サーバー結果キャッシュに割り当てられたメモリーを変更するには:

- RESULT_CACHE_MAX_SIZE初期化パラメータの値を必要なサイズに設定します。

  Oracle Real Application Clusters (Oracle RAC)環境では、結果キャッシュは各データベース・インスタンスに固有であり、インスタンスごとに個別にサイズ設定できます。ただし、無効化は、すべてのインスタンスにわたって機能します。クラスタ内のサーバー結果キャッシュを無効にするには、各インスタンスを起動する際に、このパラメータを0に明示的に設定する必要があります。

DBMS_RESULT_CACHEを使用したサーバー結果キャッシュの管理

DBMS_RESULT_CACHEパッケージでは、サーバーの結果キャッシュのメモリー割当てを管理できる統計、情報および演算子が提供されます。キャッシュ・メモリーの使用に関する統計の取得や、キャッシュのフラッシュなどの操作を実行するには、DBMS_RESULT_CACHEパッケージを使用します。

この項では、DBMS_RESULT_CACHEパッケージを使用したサーバー結果キャッシュの管理方法を説明しており、内容は次のとおりです。

- サーバー結果キャッシュのメモリー使用統計の表示
- サーバー結果キャッシュのフラッシュ

サーバー結果キャッシュのメモリー使用統計の表示

この項では、DBMS_RESULT_CACHEパッケージを使用して、結果キャッシュのメモリー割当て統計を表示する方法を説明します。

結果キャッシュのメモリー使用統計を表示するには:

- DBMS_RESULT_CACHE.MEMORY_REPORTプロシージャを実行します。

  例15-2に、このプロシージャの実行を示します。

例15-2 DBMS_RESULT_CACHEパッケージの使用

```
SQL> SET SERVEROUTPUT ON
SQL> EXECUTE DBMS_RESULT_CACHE.MEMORY_REPORT
```

このコマンドの出力を次に示します。

```
R e s u l t  C a c h e  M e m o r y  R e p o r t
[Parameters]
Block Size = 1024 bytes
Maximum Cache Size = 950272 bytes (928 blocks)
Maximum Result Size = 47104 bytes (46 blocks)
[Memory]
Total Memory = 46340 bytes [0.048% of the Shared Pool]
... Fixed Memory = 10696 bytes [0.011% of the Shared Pool]
... State Object Pool = 2852 bytes [0.003% of the Shared Pool]
... Cache Memory = 32792 bytes (32 blocks) [0.034% of the Shared Pool]
....... Unused Memory = 30 blocks
....... Used Memory = 2 blocks
........... Dependencies = 1 blocks
........... Results = 1 blocks
............... SQL = 1 blocks
PL/SQL procedure successfully completed.
```

サーバー結果キャッシュのフラッシュ

この項では、DBMS_RESULT_CACHEパッケージを使用して、既存の結果をすべて削除し、結果キャッシュ・メモリーを消去する方法を説明します。

サーバー結果キャッシュをフラッシュするには:

- DBMS_RESULT_CACHE.FLUSHプロシージャを実行します。

関連項目:

DBMS_RESULT_CACHE パッケージの詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください

クライアント結果キャッシュの構成

表15-2に、クライアントの結果キャッシュの有効化またはキャッシュの動作に影響を与えるデータベース初期化パラメータを示します。

表15-2 クライアントの結果キャッシュの初期化パラメータ

パラメータ	説明
CLIENT_RESULT_CACHE_SIZE	クライアント・プロセスごとにクライアント結果キャッシュの最大サイズを指定します。クライアントの結果キャッシュを有効にするには、サイズを 32768 バイト以上に設定します。これよりも少ない値(デフォルトの 0 を含む)を設定すると、クライアント結果キャッシュは無効になります。 ノート: CLIENT_RESULT_CACHE_SIZE の設定によってクライアントのキャッシュを無効にする場合、クライアント・ノードではクライアントのキャッシュを有効にできません。ただし、CLIENT_RESULT_CACHE_SIZE の設定によってクライアントのキャッシュを有効にすると、クライアント・ノードでその設定を上書きできます。たとえば、クライアント・ノードで、クライアントの結果キャッシュを無効にしたり、そのキャッシュ・サイズを増やしたりすることができます。
CLIENT_RESULT_CACHE_LAG	クライアント結果キャッシュのタイム・ラグ長(ミリ秒単位)を指定します。デフォルト値は 3000(3 秒)です。OCI アプリケーションが一定期間データベース・コールを実行しない場合、この設定によって、次の文の実行コールが検証のために確認されます。 OCI アプリケーションによるデータベースへのアクセス頻度が低い場合、このパラメータの値を小さくすると、クライアントの結果キャッシュとデータベースとの同期を維持する目的で、OCI クライアントからデータベースに対するラウンドトリップが増加します。
COMPATIBLE	互換性を維持しておく Oracle Database のリリースを指定します。クライアントの結果キャッシュを有効にするには、このパラメータを 11.0.0.0 以上に設定する必要があります。クライアントのビューでのキャッシュの場合、このパラメータを 11.2.0.0.0 以上に設定する必要があります。

オプションのクライアント構成ファイルは、サーバー・パラメータ・ファイルに設定されているクライアント結果キャッシュの初期化パラメータを上書きします。

ノート:

クライアント結果キャッシュ・ラグを設定できるのは、CLIENT_RESULT_CACHE_LAG 初期化パラメータと一緒の場合のみです。

関連項目:

- クライアント構成ファイルに設定可能なパラメータの詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください

- 前述のクライアント結果キャッシュの初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください

結果キャッシュ・モードの設定

結果キャッシュ・モードは、どの問合せがサーバーおよびクライアントの結果キャッシュに結果セットを格納する対象になるかを決定するデータベース設定です。問合せがキャッシュの対象である場合、アプリケーションは結果キャッシュを確認し、問合せ結果セットがキャッシュ内に存在するかどうかを調べます。存在する場合、結果キャッシュから結果を直接取得します。存在しない場合、データベースは問合せを実行して結果を出力として戻し、その結果を結果キャッシュに格納します。結果キャッシュは、読取り専用またはほぼ読取り専用のデータベース・オブジェクトの問合せにお薦めします。

結果キャッシュが有効化されている場合はデータベースでも、確定的ではないPL/SQLファンクションを呼び出す問合せがキャッシュされます。そのようなファンクションを呼び出すSELECT文をキャッシュする際、結果キャッシュは、PL/SQLファンクションおよびデータベース・オブジェクトに対するデータの依存性を追跡します。ただし、ファンクションが使用するデータが追跡されていない場合(順序、SYSDATE、SYS_CONTEXT、パッケージ変数など)、こうしたファンクションを呼び出す問合せで結果キャッシュを使用すると失効します。この点に関して、結果キャッシュの動作はPL/SQLファンクションのキャッシュと同一です。そのため、結果キャッシュを有効化する際は、パフォーマンスだけでなく、データの精度も常に考慮してください。

結果キャッシュ・モードを設定するには:

- RESULT_CACHE_MODE初期化パラメータの値を設定し、結果キャッシュの動作を決定します。

  このパラメータは、インスタンス(ALTER SYSTEM)、セッション(ALTER SESSION)またはサーバー・パラメータ・ファイルで設定できます。

  表15-3に、このパラメータの値を説明します。

表15-3 RESULT_CACHE_MODEパラメータの値

値	説明
MANUAL	問合せヒントまたは表注釈を使用した場合のみ問合せの結果を結果キャッシュに格納できます。これがデフォルトで、推奨値です。
FORCE	すべての結果が結果キャッシュに格納されます。結果がキャッシュ内に存在しない場合、データベースは問合せを実行して結果をキャッシュに格納します。それ以降は、結果キャッシュ・ヒントを含め、同じ SQL 文が実行されると、キャッシュのデータが取得されます。可能な場合、これらの結果がセッションで使用されます。キャッシュからの問合せ結果を除外するには、/*+ NO_RESULT_CACHE */問合せヒントを使用する必要があります。 ノート: FORCE モードでは、データベースおよびクライアントがすべての問合せをキャッシュしようとして、パフォーマンスおよびラッチの著しいオーバーヘッドが生じるため、お薦めできません。さらに、確定的でない PL/SQL ファンクションを呼び出す問合せもキャッシュされるため、そのように対象範囲が広い場合の結果キャッシュは、大幅に結果を変える原因になる可能性があります。

関連項目:

RESULT_CACHE_MODE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください

結果キャッシュの要件

結果キャッシュを有効化しても、サーバーまたはクライアントの結果キャッシュに特定の結果セットが含まれることは保証されません。結果がキャッシュされるには、次の要件が満たされている必要があります。

- 読取り一貫性の要件
- 問合せパラメータの要件
- 結果キャッシュの制限

読取り一貫性の要件

スナップショットを再利用可能にするには、スナップショットの読取り一貫性を維持する必要があります。結果セットがキャッシュの対象になるには、少なくとも、次の条件のいずれか1つに当てはまる必要があります。

- 結果の構築に使用される読取り一貫性スナップショットで、データの最新のコミット済状態を取得していること。
- 問合せが、フラッシュバック問合せを使用して明示的な時点を示していること。

現在のセッションで、問合せ内にアクティブなトランザクションの参照オブジェクトがある場合、この問合せの結果はキャッシュの対象となりません。

問合せパラメータの要件

キャッシュの結果は、問合せが等価であり、パラメータ値が同じである場合、変数の値とともにパラメータ化されていれば再利用できます。値やバインド変数の名前が異なる場合、キャッシュ・ミスが生じる場合があります。次のいずれかの要素が問合せで使用されている場合、結果はパラメータ化されます。

- バインド変数
- SQLファンクションDBTIMEZONE、SESSIONTIMEZONE、USERENV/SYS_CONTEXT(定数変数を含む)、UIDおよびUSER。
- NLSパラメータ

結果キャッシュの制限

次のオブジェクトまたはファンクションが問合せ内にある場合、結果をキャッシュできません。

- 一時表およびSYSまたはSYSTEMスキーマ内の表
- 順序のCURRVALおよびNEXTVAL擬似列
- SQLファンクションCURRENT_DATE、CURRENT_TIMESTAMP、LOCAL_TIMESTAMP、USERENV/SYS_CONTEXT (非定数変数を含む)、SYS_GUID、SYSDATEおよびSYSTIMESTAMP

クライアント結果キャッシュには、その他にも結果をキャッシュする際の制限があります。

ノート:

結果キャッシュは、読取り専用モードでオープンされた Active Data Guard スタンバイ・データベースでは機能しません。

関連項目:

クライアント結果キャッシュの追加の制限事項の詳細は、『Oracle Call Interfaceプログラマーズ・ガイド』を参照してください

結果をキャッシュする問合せの指定

この項では、結果キャッシュの問合せの指定方法を説明しており、内容は次のとおりです。

- SQLの結果キャッシュ・ヒントの使用
- 結果キャッシュの表注釈の使用

SQLの結果キャッシュ・ヒントの使用

キャッシュの動作を制御するには、アプリケーション・レベルで結果キャッシュ・ヒントを使用します。SQLの結果キャッシュ・ヒントは、結果キャッシュのモードおよび表注釈よりも優先されます。

この項では、SQL結果キャッシ・ヒントの使用方法を説明しており、内容は次のとおりです。

- RESULT_CACHEヒントの使用
- NO_RESULT_CACHEヒントの使用
- ビューでのRESULT_CACHEヒントの使用

関連項目:

RESULT_CACHEおよびNO_RESULT_CACHEヒントの詳細は、『Oracle Database SQL言語リファレンス』を参照してください

RESULT_CACHEヒントの使用

結果キャッシュ・モードがMANUALの場合、/*+ RESULT_CACHE */ヒントは、問合せブロックの結果をキャッシュに格納し、キャッシュされた結果を今後の実行で使用するようデータベースに指示します。

例15-3に、RESULT_CACHEヒントを使用する問合せを示します。

例15-3 RESULT_CACHEヒントの使用

```
SELECT /*+ RESULT_CACHE */ prod_id, SUM(amount_sold)
  FROM sales
 GROUP BY prod_id
 ORDER BY prod_id;
```

この例では、sales表の問合せの行をキャッシュするよう、問合せがデータベースに指示します。

NO_RESULT_CACHEヒントの使用

/*+ NO_RESULT_CACHE */ヒントは、データベースがサーバーまたはクライアントの結果キャッシュに結果をキャッシュしないように指示します。

例15-4に、NO_RESULT_CACHEヒントを使用する問合せを示します。

例15-4 NO_RESULT_CACHEヒントの使用

```
SELECT /*+ NO_RESULT_CACHE */ prod_id, SUM(amount_sold)
  FROM sales
 GROUP BY prod_id
 ORDER BY prod_id;
```

この例では、sales表の問合せの行をキャッシュしないよう、問合せがデータベースに指示します。

ビューでのRESULT_CACHEヒントの使用

RESULT_CACHEヒントは、ヒントが指定された問合せブロックにのみ適用されます。ビューにのみヒントが指定されている場合は、その結果のみがキャッシュされます。ビュー・キャッシュには次の特性があります。

- ビューは次のタイプである必要があります。
  - 標準ビュー(CREATE ... VIEW文で作成されたビュー)
  - SELECT文のFROM句に指定されたインライン・ビュー
  - WITH句を使用して作成されたインライン・ビュー
- 相関列(外側の問合せブロックへの参照)を使用するビュー問合せの結果は、キャッシュできません。
- 問合せの結果は、クライアントの結果キャッシュではなく、サーバーの結果キャッシュに格納されます。
- キャッシュ・ビューは、外側の(または参照している)問合せブロックにマージされません。

  RESULT_CACHEヒントをインライン・ビューに追加すると、キャッシュ済結果の再利用性を最大化するために、外部問合せとインライン・ビュー間の最適化は無効化されます。

次の例に、インライン・ビューview1の問合せを示します。

```
SELECT *
  FROM ( SELECT /*+ RESULT_CACHE */ department_id, manager_id, count(*) count
           FROM hr.employees
          GROUP BY department_id, manager_id ) view1
 WHERE department_id = 30;
```

この例において、view1のSELECT文は外側のブロックで、employeesのSELECT文は内側のブロックです。RESULT_CACHEヒントは内側のブロックにのみ指定されているため、内部問合せの結果はサーバー結果キャッシュに格納されますが、外側の問合せの結果はキャッシュされません。

次の例に示すように、同じセッションで、ビューview2の問合せが実行される場合を考えます。

```
WITH view2 AS
( SELECT /*+ RESULT_CACHE */ department_id, manager_id, count(*) count
    FROM hr.employees
   GROUP BY department_id, manager_id )
SELECT *
  FROM view2
 WHERE count BETWEEN 1 and 5;
```

この例では、WITH句の問合せブロックにのみRESULT_CACHEヒントが指定されているため、employeesの問合せ結果はキャッシュ

の対象となります。これらの結果は最初にあげた例の問合せからキャッシュされているため、2番目にあげた例のWITH句のSELECT文では、キャッシュされた行を取得できます。

結果キャッシュの表注釈の使用

結果キャッシュの制御には、表注釈を使用することもできます。表注釈は、問合せセグメントではなく、問合せ全体に影響します。表注釈を使用する主な利点は、結果キャッシュ・ヒントをアプリケーション・レベルで問合せに追加する必要がないことです。表注釈の優先度はSQL結果キャッシュ・ヒントより低いため、問合せレベルでヒントを使用することにより、表およびセッションの設定を上書きできます。

表15-4に、RESULT_CACHE表注釈の有効な値を示します。

表15-4 RESULT_CACHE表注釈の値

値	説明
DEFAULT	問合せの 1 つ以上の表が DEFAULT に設定されている場合、RESULT_CACHE_MODE 初期化パラメータを FORCE に設定するか、RESULT_CACHE ヒントを指定しないかぎり、この問合せの結果キャッシュは表レベルでは有効になりません。これがデフォルト値です。
FORCE	問合せのすべての表が FORCE に設定されている場合、問合せ結果はキャッシュ対象とみなされます。表注釈 FORCE は、セッション・レベルで設定した RESULT_CACHE_MODE パラメータの値 MANUAL よりも優先されます。

この項では、RESULT_CACHE表注釈の使用方法を説明しており、内容は次のとおりです。

- DEFAULT表注釈の使用
- FORCE表注釈の使用

DEFAULT表注釈の使用

DEFAULT表注釈を使用すると、データベースによる表レベルの結果のキャッシュが行われません。

例15-5に、DEFAULT表注釈を使用して表salesおよびこの表の問合せを作成するCREATE TABLE文を示します。

例15-5 DEFAULT表注釈の使用

```
CREATE TABLE sales (...) RESULT_CACHE (MODE DEFAULT);
SELECT prod_id, SUM(amount_sold)
  FROM sales
 GROUP BY prod_id
 ORDER BY prod_id;
```

この例では、sales表は、結果キャッシュを無効化する表注釈を使用して作成されています。この例では、sales表の問合せの結果は、表注釈があるためにキャッシュの対象とみなされません。

関連項目:

CREATE TABLE文およびその構文の詳細は、『Oracle Database SQL言語リファレンス』を参照してください。

FORCE表注釈の使用

FORCE表注釈を使用すると、データベースでは、表レベルで結果がキャッシュされます。

例15-5で作成されたsales表を使用して、この表の結果キャッシュを強制することを決定した場合は、FORCE表注釈を使用してこれを実行できます。

例15-6に、sales表のFORCE表注釈を使用するALTER TABLE文を示します。

例15-6 FORCE表注釈の使用

```
ALTER TABLE sales RESULT_CACHE (MODE FORCE);
SELECT prod_id, SUM(amount_sold)
  FROM sales
 GROUP BY prod_id
HAVING prod_id=136;
SELECT /*+ NO_RESULT_CACHE */ *
  FROM sales
 ORDER BY time_id DESC;
```

この例には、sales表の2つの問合せが含まれています。最初の問合せは使用頻度が高く数行を戻します。これは表注釈があるためキャッシュの対象となります。2番目の問合せは、1回かぎりの問合せで多くの行を戻します。この問合せではヒントを使用して、結果がキャッシュされないようにしています。

結果キャッシュの監視

サーバーおよびクライアントの結果キャッシュに関する情報を表示するには、関連するデータベース・ビューおよび表を問い合せます。

表15-5に、結果キャッシュの監視に最も役に立つビューおよび表を示します。

表15-5 結果キャッシュに関する情報を含むビューおよび表

ビュー/表	説明
V$RESULT_CACHE_STATISTICS	サーバーの結果キャッシュの様々な設定とメモリー使用量の統計のリスト。
V$RESULT_CACHE_MEMORY	サーバーの結果キャッシュのすべてのメモリー・ブロックとそれに対応する統計のリスト。
V$RESULT_CACHE_OBJECTS	結果がその属性とともにサーバーの結果キャッシュ内にあるすべてのオブジェクトのリスト。
V$RESULT_CACHE_DEPENDENCY	サーバー結果キャッシュ内の結果とこれらの結果の依存性間における依存関係の詳細のリスト。

ビュー/表	説明
CLIENT_RESULT_CACHE_STATS$	OCI クライアント・プロセスから取得されたクライアントの結果キャッシュのキャッシュ設定とメモリー使用量の統計を格納します。この統計表には、結果キャッシュを使用する各クライアント・プロセスのエントリが含まれます。クライアント・プロセスの終了後、エントリはこの表から削除されます。クライアント表には、V$RESULT_CACHE_STATISTICS に類似の情報が含まれます。
DBA_TABLES、USER_TABLES、ALL_TABLES	表の結果キャッシュ・モードの注釈を示す RESULT_CACHE 列が含まれます。表に注釈が付いていない場合、この列には DEFAULT と表示されます。この列は、サーバーとクライアントの両方の結果キャッシュに適用されます。

関連項目:

これらのビューおよび表の詳細は、『Oracle Databaseリファレンス』を参照してください。

次の例に、サーバー結果キャッシュの統計を監視するためのV$RESULT_CACHE_STATISTICSビューの問合せを示します。

```
COLUMN name FORMAT a20
SELECT name, value
  FROM V$RESULT_CACHE_STATISTICS;
```

この問合せの出力例を次に示します。

```
NAME                          VALUE
--------------------      ----------
Block Size (Bytes)              1024
Block Count Maximum             3136
Block Count Current               32
Result Size Maximum (Blocks)     156
Create Count Success               2
Create Count Failure               0
Find Count                         0
Invalidation Count                 0
Delete Count Invalid               0
Delete Count Valid                 0
```

次の例に、クライアント結果キャッシュの統計を監視するためのCLIENT_RESULT_CACHE_STATS$表の問合せを示します。

```
SELECT stat_id, SUBSTR(name,1,20), value, cache_id
  FROM CLIENT_RESULT_CACHE_STATS$
 ORDER BY cache_id, stat_id;
```

この問合せの出力例を次に示します。

```
STAT_ID    NAME OF STATISTICS      VALUE   CACHE_ID
=======    ==================      =====   ========
    1      Block Size               256         124
```

```
   2      Block Count Max          256         124
   3      Block Count Current      128         124
   4      Hash Bucket Count       1024         124
   5      Create Count Success      10         124
   6      Create Count Failure       0         124
   7      Find Count                12         124
   8      Invalidation Count         8         124
   9      Delete Count Invalid       0         124
  10      Delete Count Valid         0         124
```

CLIENT_RESULT_CACHE_STATS$表には、クライアントの結果をキャッシュするアクティブな各クライアント・プロセスの統計エントリがあります。すべてのクライアント・プロセスには、一意のキャッシュIDがあります。

クライアントのキャッシュを実行しているセッションのクライアント接続情報を取得するには:

1. CLIENT_RESULT_CACHE_STATS$表のCACHE_ID列に対応するGV$SESSION_CONNECT_INFOビューのCLIENT_REGID列からセッションIDを取得します。
2. GV$SESSION_CONNECT_INFOおよびGV$SESSIONビューで関連する列を問い合せます。

サーバーとクライアントのどちらの結果キャッシュ統計の場合も、結果キャッシュ用に最適化されているデータベースでは、Create Count FailureおよびDelete Count Valid統計の値が比較的小さく、Find Count統計の値が比較的大きくなります。

16 プログラム・グローバル領域のチューニング

この章では、プログラム・グローバル領域(PGA)のチューニング方法を説明します。自動メモリー管理を使用してシステムのデータベース・メモリーを管理している場合は、この章で説明されているように、PGAを手動でチューニングする必要はありません。

この章のトピックは、次のとおりです:

- プログラム・グローバル領域について
- 自動メモリー管理を使用したプログラム・グローバル領域のサイズ設定
- ハード制限を使用したプログラム・グローバル領域のサイズ設定

プログラム・グローバル領域について

プログラム・グローバル領域(PGA)は、サーバー・プロセスのデータおよび制御情報が含まれるプライベート・メモリー領域です。PGAにアクセスできるのはサーバー・プロセスのみです。Oracle Databaseは、サーバー・プロセスのかわりにPGAに対する情報の読取りおよび書込みを行います。そのような情報の例として、カーソルのランタイム領域があります。カーソルを実行するたびに、そのカーソルを実行するサーバー・プロセスのPGAメモリー領域内に、そのカーソルのための新しいランタイム領域が作成されます。

ノート:

ランタイム領域の一部は、共有サーバーを使用するときに共有グローバル領域(SGA)内に配置できます。

複雑な問合せ(意思決定支援の問合せなど)の場合、ランタイム領域の大部分が、次のようなメモリー集中型演算子で割り当てられた作業領域に使用されます。

- ソート・ベース演算子(たとえば、ORDER BY、GROUP BY、ROLLUPおよびウィンドウ・ファンクション)
- ハッシュ結合
- ビットマップ・マージ
- ビットマップ作成
- 一括ロード操作で使用される書込みバッファ

ソート演算子は、作業領域(ソート領域)を使用して一連の行のインメモリー・ソートを実行します。同様に、ハッシュ結合演算子は作業領域(ハッシュ領域)を使用して、ハッシュ表を左側から入力して作成します。

作業領域のサイズ

Oracle Databaseを使用すると、作業領域のサイズを制御およびチューニングできます。一般に、作業領域を大きくすると、メモリー消費量は増えますが特定の演算子のパフォーマンスを大幅に向上できます。次に、使用可能な作業領域のサイズを示します。

- 最適

  最適なサイズは、作業領域のサイズが、関連するSQL演算子で割り当てられた入力データや補助メモリー構造を処

理できるほど十分に大きなサイズである場合です。これは、作業領域の理想的なサイズです。

- ワン・パス

  ワン・パス・サイズは、作業領域のサイズが最適なサイズを下回っており、入力データの一部で余分なパスが実行される場合です。ワン・パス・サイズでは、レスポンス時間が長くなります。

- マルチ・パス

  マルチ・パス・サイズは、作業領域のサイズがワン・パスしきい値を下回っており、入力データに対する複数のパスが必要な場合です。入力データのサイズに比べて作業領域のサイズが小さすぎるため、マルチ・パス・サイズでは、レスポンス時間が大幅に長くなります。

たとえば、10GBのデータをソートするシリアル・ソート操作では、最適なサイズで実行するには10GBよりも少し多めにする必要があり、ワン・パス・サイズで実行するには少なくとも40MBが必要です。作業領域が40MBに満たない場合は、入力データに対し、ソート操作で複数のパスを実行する必要があります。

作業領域をサイズ設定する際の目標は、最適なサイズ(90%を超える、またはOLTPシステム固有の場合は100%)で大半の作業領域を動作させ、ごく一部をワン・パス・サイズ(10%未満)で動作させることです。次の理由から、マルチ・パス実行はお薦めしません。

- マルチ・パス実行では、パフォーマンスが大幅に低下する可能性があります。

  マルチ・パスの作業領域が多いと、関連付けられたSQL演算子のレスポンス時間に大きな悪影響があります。

- ワン・パス実行を行うために、大量のメモリーは必要ありません。

  ワン・パス・サイズで1GBのデータのソートに必要なのは22MBのみです。

大きなソートとハッシュ結合を実行するDSSシステムの場合であっても、ワン・パスの実行のメモリー要件は相対的に少ない量です。妥当なPGAメモリー量で構成されたシステムは、入力データに対してマルチ・パスを実行する必要がありません。

自動メモリー管理を使用したプログラム・グローバル領域のサイズ設定

自動PGAメモリー管理により、PGAメモリーの割当て方法が単純化され改善されます。デフォルトでは、PGAメモリー管理は有効化されます。このモードでは、SGAメモリー・サイズの20%をベースとして、作業領域専用のPGAメモリーの一部が動的に調整され、Oracle Databaseにより自動的にPGAが設定されます。最小値は10MBです。

> ノート:
>
> 下位互換性のために、PGA_AGGREGATE_TARGET 初期化パラメータを 0(ゼロ)に設定して、自動 PGA メモリー管理を無効にできます。自動 PGA メモリー管理が無効になっている場合は、SORT_AREA_SIZE 初期化パラメータなどの関連_AREA_SIZE パラメータを使用して作業領域の最大サイズを設定できます。

この項では、自動PGAメモリー管理を使用してPGAをサイズ設定する方法を説明しており、内容は次のとおりです。

- 自動PGAメモリー管理の構成
- PGA_AGGREGATE_TARGETの初期値の設定
- 自動PGAメモリー管理の監視

- PGA_AGGREGATE_TARGETのチューニング

自動PGAメモリー管理の構成

Oracle Databaseを自動PGAメモリー管理モードで実行すると、すべてのセッションの作業領域のサイズ設定は自動になり、*_AREA_SIZEパラメータはそのモードで動作するすべてのセッションで無視されます。アクティブな作業領域に使用できるPGAメモリーの総量は、Oracle Databaseにより、PGA_AGGREGATE_TARGET初期化パラメータから自動的に導出されます。PGAのメモリー量は、システムの他のコンポーネントに割り当てられたPGAメモリー(セッションで割り当てられたPGAメモリーなど)の量をPGA_AGGREGATE_TARGETの値から減算した値に設定されます。その後、Oracle Databaseにより、結果のPGAメモリーが、それぞれの特定のメモリー要件に基づいて個々のアクティブな作業領域に割り当てられます。

Oracle Databaseは、作業領域に割り当てられるPGAのメモリー量を動的に制御することで、DBAが設定したPGA_AGGREGATE_TARGETの値への準拠を試行します。これを達成するために、Oracle Databaseではまず、メモリー集中型のすべてのSQL操作に対する最適な作業領域数の最大化が試行されます。DBAにより(PGA_AGGREGATE_TARGETパラメータを使用して)設定されたPGAメモリーの制限が低すぎるため、マルチ・パスを実行してメモリー消費を削減し、PGAのターゲット制限を満たす必要がある場合を除き、残りの作業領域はワン・パス・モードで実行されます。

データベース・インスタンスを新しく構成する際は、PGA_AGGREGATE_TARGETの適切な設定を判断することが困難な場合があります。

自動PGAメモリー管理を構成するには:

1. 「PGA_AGGREGATE_TARGETの初期値の設定」で説明されているように、PGA_AGGREGATE_TARGETパラメータの値の初期見積りを行います。
2. 「自動PGAメモリー管理の監視」で説明されているように、データベース・インスタンスで代表的なワークロードを実行し、パフォーマンスを監視します。
3. 「PGA_AGGREGATE_TARGETのチューニング」で説明されているように、Oracle PGAアドバイス統計を使用してPGA_AGGREGATE_TARGETパラメータの値をチューニングします。

関連項目:

PGA_AGGREGATE_TARGET初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください。

PGA_AGGREGATE_TARGETの初期値の設定

Oracleデータベース・インスタンスに使用できるメモリー量に基づいて、PGA_AGGREGATE_TARGET初期化パラメータの初期値を設定します。この値は後からインスタンス・レベルでチューニングしたり動的に変更できます。デフォルトで、Oracle Databaseは、この値のSGAサイズの20%を使用します。ただし、この設定は、大規模DSSシステムには小さすぎる場合があります。

PGA_AGGREGATE_TARGETの初期値を設定するには:

1. オペレーティング・システムと、同じシステムで実行されているその他のOracle以外のアプリケーション用に予約する、合計物理メモリー量を決定します。

   たとえば、オペレーティング・システムとその他のOracle以外のアプリケーション用に合計物理メモリーの20%を予約する

場合、システムのメモリーの80%をOracleデータベース・インスタンスで使用することになります。

2. 使用可能な残りのメモリーをSGAとPGAで分割します。

   - OLTPシステムの場合、使用可能なメモリーにおけるPGAメモリーの割合は通常ごくわずかで、残りのメモリーの大部分がSGAに充てられます。

     最初は、使用可能なメモリーの20%をPGAに使用し、80%をSGAに使用することをお薦めします。そのため、OLTPシステムのPGA_AGGREGATE_TARGETパラメータの初期値は、次のようにして計算できます。

     PGA_AGGREGATE_TARGET = (total_mem * 0.8) * 0.2 (ここで、total_memは、システムで使用可能な物理メモリーの合計量です。)

   - メモリー集中型の大量の問合せを実行するDSSシステムでは、通常、PGAメモリーは使用可能なメモリーの70%までを使用します。

     最初は、使用可能なメモリーの50%をPGAに使用し、50%をSGAに使用することをお薦めします。そのため、DSSシステムのPGA_AGGREGATE_TARGETパラメータの初期値は、次のようにして計算できます。

     PGA_AGGREGATE_TARGET = (total_mem * 0.8) * 0.5 (ここで、total_memは、システムで使用可能な物理メモリーの合計量です。)

たとえば、物理メモリーが4GBのシステムで実行するようOracleデータベース・インスタンスが構成されている場合、およびOracleデータベース・インスタンスがメモリーの80% (または3.2GB)を使用している場合、OLTPシステムではPGA_AGGREGATE_TARGETを640MBに、DSSシステムでは1,600MBに設定します。

自動PGAメモリー管理の監視

チューニング・プロセスを開始する前に、データベース・インスタンスで代表的なワークロードを実行し、パフォーマンスを監視してください。Oracle Databaseにより収集されるPGA統計を使用すると、PGAの最大サイズが小さく構成されているか大きく構成されているかを判断できます。これらの統計を監視すると、自動PGAメモリー管理のパフォーマンスを評価でき、PGA_AGGREGATE_TARGETパラメータの値を適宜チューニングできます。

この項では、パフォーマンス・ビューを使用して自動PGAメモリー管理を監視する方法を説明しており、内容は次のとおりです。

- V$PGASTATビューの使用
- V$PROCESSビューの使用
- V$PROCESS_MEMORYビューの使用
- V$SQL_WORKAREA_HISTOGRAMビューの使用
- V$WORKAREA_ACTIVEビューの使用
- V$SQL_WORKAREAビューの使用

V$PGASTATビューの使用

V$PGASTATビューは、PGAメモリー使用量および自動PGAメモリー・マネージャに関するインスタンス・レベルの統計を示します。

次の例に、このビューの問合せを示します。

```
SELECT *
  FROM V$PGASTAT;
```

この問合せの出力例を次に示します。

```
NAME                                                       VALUE UNIT
---------------------------------------------------------- ---------- ------------
aggregate PGA target parameter                           41156608 bytes
aggregate PGA auto target                                21823488 bytes
global memory bound                                       2057216 bytes
total PGA inuse                                          16899072 bytes
total PGA allocated                                      35014656 bytes
maximum PGA allocated                                   136795136 bytes
total freeable PGA memory                                 524288 bytes
PGA memory freed back to OS                             1713242112 bytes
total PGA used for auto workareas                             0 bytes
maximum PGA used for auto workareas                     2383872 bytes
total PGA used for manual workareas                           0 bytes
maximum PGA used for manual workareas                    8470528 bytes
over allocation count                                        291
bytes processed                                         2124600320 bytes
extra bytes read/written                                  39949312 bytes
cache hit percentage                                         98.15 percent
```

表16-1は、V$PGASTATビューに表示される主な統計を説明しています。

表16-1 V$PGASTATビューの統計

統計	説明
aggregate PGA target parameter	この統計は、PGA_AGGREGATE_TARGET パラメータの現在の値を示します。デフォルト値は、SGA サイズの 20%です。このパラメータを 0 に設定すると、自動 PGA メモリー管理が無効化されます。
aggregate PGA auto target	この統計は、Oracle Database が、自動モードで実行中の作業領域に使用できる PGA メモリーの量を示します。この量は、PGA_AGGREGATE_TARGET パラメータの値と現在の作業領域のワークロードから動的に導出されます。したがって、Oracle Database により継続的に調整されます。この値が PGA_AGGREGATE_TARGET の値と比較して小さいと、システムの他のコンポーネント(PL/SQL または Java など)により PGA メモリーの大部分が使用され、作業領域用にはわずかしか残りません。自動モードで実行されている作業領域用に十分な PGA メモリーが残るようにしてください。
global memory bound	この統計は、自動モードで実行される作業領域の最大サイズを示します。この値は、作業領域のワークロードの現在の状態を反映するように継続的に調整されます。通常、システム内でアクティブな作業領域の数が増加すると、グローバル・メモリーの上限は低くなります。一般的には、グローバル・バウンドの値は 1MB を下回らないようにする必要があります。そうなった場合は、PGA_AGGREGATE_TARGET パラメータの値を増やします。
total PGA allocated	この統計は、データベース・インスタンスにより割り当てられた PGA メモリーの現在の量を示します。この値は Oracle Database により、PGA_AGGREGATE_TARGET の値未満に維持されます。ただし、作業領域のワークロードが急速に増えている場合や、

統計	説明
	PGA_AGGREGATE_TARGET パラメータが小さすぎる値に設定されている場合は、少量かつ短期間、この値を超過した PGA が割り当てられることがあります。
total freeable PGA memory	この統計は、割り当てられた PGA メモリーのうち解放可能な量を示します。
total PGA used for auto workareas	この統計は、自動モードで実行中の作業領域により現在消費されている PGA メモリーの量を示します。この数値を使用して、PGA メモリーの他のコンシューマ(PL/SQL や Java など)で消費されるメモリーの量を判断できます。 PGA other = total PGA allocated - total PGA used for auto workareas
over allocation count	この統計は、インスタンス起動時から累積されます。PGA_AGGREGATE_TARGET の値が小さすぎて、PGA other コンポーネントと、作業領域のワークロードを実行するために必要な最小メモリーに対応できない場合は、PGA メモリーの過剰割当てとなる可能性があります。この場合、Oracle Database は PGA_AGGREGATE_TARGET の値を満たすことができないため、追加の PGA メモリーを割り当てる必要があります。過剰割当てが発生した場合は、「V$PGA_TARGET_ADVICE ビューの使用」で説明されているように、V$PGA_TARGET_ADVICE ビューにより提供される情報を使用して、PGA_AGGREGATE_TARGET パラメータの値を増やします。
total bytes processed	この統計は、インスタンスの起動後にメモリー集中型 SQL 演算子によって処理されたバイト数を示します。たとえば、ソート操作の入力サイズが、処理されたバイト数によって示されます。この数値は cache hit percentage メトリックを計算する場合に使用します。
extra bytes read/written	作業領域が最適に実行できない場合は、1 つ以上の余分なパスが入力データで実行されています。この統計は、インスタンスの起動後の追加パスの間に処理されたバイト数を表します。この数値は cache hit percentage メトリックを計算する場合にも使用します。total bytes processed に比べて小さい値であることが理想的です。
cache hit percentage	このメトリックは Oracle Database によって計算され、PGA メモリー・コンポーネントのパフォーマンスを反映します。この値はインスタンスの起動時から累積されます。値 100%は、インスタンス起動後にシステムで実行されたすべての作業領域で、最適な量の PGA メモリーが使用されていることを意味します。これが理想的ですが、純粋な OLTP システムを除き、そのようになることはほとんどありません。通常は、PGA メモリーの総サイズに応じて、一部の作業領域でワン・パスやマルチ・パスが実行されます。作業領域を最適に実行できない場合は、1 つ以上の余分なパスが入力データで実行されています。これにより、入力データのサイズと追加実行されたパス数に比例して、cache hit percentage が低下します。このメトリックの計算方法の例は、例 16-1 を参照し

統計	説明
	てください。

例16-1に、余分なパスがcache hit percentageメトリックにどのように影響するかを示します。

例16-1 キャッシュ・ヒット率の計算

4つのソート操作が実行され、そのうちの3つは小さく(1MBの入力データ)、1つは大きい(100MBの入力データ)操作です。4つの操作で処理される合計バイト数(BP)は103MBです。小さなソートの1つがワン・パスを実行すると、1MBの入力データで余分なパスが実行されます。この1MBという値は、extra bytes read/writtenまたはEBPの数を示します。

cache hit percentageは次の計算式を使用して計算されます。

```
BP x 100 / (BP + EBP)
```

この例では、cache hit percentageは99.03%です。この値は、余分なパスが実行されたのは小さいソート操作の1つのみで、その他すべてのソート操作は最適なサイズで実行できたことを反映しています。そのため、cache hit percentageはほぼ100%になりますが、それはこの1MB超の余分なパスがわずかなオーバーヘッドであるためです。ただし、大きなソート操作がワン・パス・サイズで実行される場合、EBPは1MBではなく100MBとなり、cache hit percentageは50.73%に低下しますが、それはこの余分なパスがもたらす影響がはるかに大きくなるためです。

V$PROCESSビューの使用

V$PROCESSビューでは、データベース・インスタンスに接続されているOracleプロセスごとに1行が表示されます。これらのプロセスのPGAメモリー使用量を監視するには、このビューの次の列を使用します。

- PGA_USED_MEM
- PGA_ALLOC_MEM
- PGA_FREEABLE_MEM
- PGA_MAX_MEM

例16-2に、このビューの問合せを示します。

例16-2 V$PROCESSビューの問合せ

```
SELECT program, pga_used_mem, pga_alloc_mem, pga_freeable_mem, pga_max_mem
  FROM V$PROCESS;
```

この問合せの出力例を次に示します。

```
PROGRAM                  PGA_USED_MEM PGA_ALLOC_MEM PGA_FREEABLE_MEM PGA_MAX_MEM
------------------------ ------------ ------------- ----------------- -----------
PSEUDO                              0             0                 0           0
oracle@examp1690 (PMON)        314540        685860                 0      685860
oracle@examp1690 (MMAN)        313992        685860                 0      685860
oracle@examp1690 (DBW0)        696720       1063112                 0     1063112
oracle@examp1690 (LGWR)      10835108      22967940                 0    22967940
oracle@examp1690 (CKPT)        352716        710376                 0      710376
oracle@examp1690 (SMON)        541508        948004                 0     1603364
oracle@examp1690 (RECO)        323688        685860                 0      816932
```

```
oracle@examp1690 (q001)         233508          585128               0          585128
oracle@examp1690 (QMNC)         314332          685860               0          685860
oracle@examp1690 (MMON)         885756         1996548          393216         1996548
oracle@examp1690 (MMNL)         315068          685860               0          685860
oracle@examp1690 (q000)         330872          716200           65536          716200
oracle@examp1690 (CJQ0)         533476         1013540               0         1144612
```

V$PROCESS_MEMORYビューの使用

V$PROCESS_MEMORYビューは、それぞれのOracleプロセスの動的なPGAメモリー使用量を名前付きコンポーネント・カテゴリ別に示します。このビューでは、各Oracleプロセスに最大6行が表示され、1つは次のための行です。

- 各名前付きコンポーネント・カテゴリ:
  - Java
  - PL/SQL
  - OLAP
  - SQL
- 解放可能

  オペレーティング・システムによって、特定のカテゴリにではなくプロセスに割り当てられたメモリー
- その他

  名前を付けられたカテゴリではなく、1つのカテゴリに割り当てられたメモリー

6つのカテゴリのそれぞれに対するOracleプロセスのPGAメモリー使用量を動的に監視するには、このビューの次の列を使用します。

- CATEGORY
- ALLOCATED
- USED
- MAX_ALLOCATED

ノート:

V$PROCESS_MEMORY_DETAIL ビューには、PGA 使用量が 500MB を超える Oracle プロセスの動的 PGA メモリー使用量が表示されます。Oracle Database 12c リリース 2 以降では、V$PROCESS_MEMORY_DETAIL ビューを使用できます。

関連項目:

V$PROCESS_MEMORYビューおよびV$PROCESS_MEMORY_DETAILビューの詳細は、Oracle Databaseリファレンスを参照してください

V$SQL_WORKAREA_HISTOGRAMビューの使用

V$SQL_WORKAREA_HISTOGRAMビューには、インスタンス起動後に、最適なメモリー・サイズ、ワン・パス・メモリー・サイズおよびマルチ・パス・メモリー・サイズで実行された作業領域の総数が示されます。このビューの統計は、バケットに分割されます。バケットは、作業領域の最適なメモリー要件によって定義されます。各バケットは、LOW_OPTIMAL_SIZEおよびHIGH_OPTIMAL_SIZE列の値で指定された最適メモリー要件の範囲によって識別されます。

たとえば、ソート操作を最適なサイズ(キャッシュ)で実行するには、3MBのメモリーが必要です。このソート操作により使用された作業領域に関する統計は、次のようにして定義されるバケットに配置されます。

- LOW_OPTIMAL_SIZE = 2097152 (2MB)
- HIGH_OPTIMAL_SIZE = 4194303 (4MBから1バイト減算)

最適、ワン・パスまたはマルチ・パスのサイズで作業領域を実行する場合のパフォーマンスの影響は、その作業領域のサイズに大きく依存するため、統計は作業領域のサイズでセグメント化されます。3MBが最適なサイズのその範囲に含まれるため、この例では、作業領域の統計はこのバケットに配置されています。

例16-3および例16-4は、このビューを問い合せる2つの方法を示します。

例16-3 V$SQL_WORKAREA_HISTOGRAMビューの問合せ: 空でないバケット

次の問合せでは、空でないすべてのバケットの統計が示されます。

```
SELECT low_optimal_size/1024 low_kb,
       (high_optimal_size+1)/1024 high_kb,
       optimal_executions, onepass_executions, multipasses_executions
  FROM V$SQL_WORKAREA_HISTOGRAM
 WHERE total_executions != 0;
```

この問合せの結果を次に示します。

```
LOW_KB HIGH_KB OPTIMAL_EXECUTIONS ONEPASS_EXECUTIONS MULTIPASSES_EXECUTIONS
------ ------- ------------------ ------------------ ----------------------
     8      16             156255                  0                      0
    16      32                150                  0                      0
    32      64                 89                  0                      0
    64     128                 13                  0                      0
   128     256                 60                  0                      0
   256     512                  8                  0                      0
   512    1024                657                  0                      0
  1024    2048                551                 16                      0
  2048    4096                538                 26                      0
  4096    8192                243                 28                      0
  8192   16384                137                 35                      0
 16384   32768                 45                107                      0
 32768   65536                  0                153                      0
 65536  131072                  0                 73                      0
131072  262144                  0                 44                      0
262144  524288                  0                 22                      0
```

この例の出力は、1MBから2MBのバケットで551の作業領域が最適なサイズで実行されている一方で、16がワン・パス・サイズで実行され、マルチ・パス・サイズで実行されている作業領域はないことを示しています。また、1MB未満のすべての作業領域が最適なサイズで実行できたことも示されています。

例16-4 V$SQL_WORKAREA_HISTOGRAMビューの問合せ: 最適パーセント

次の問合せは、起動後に作業領域が最適なサイズ、ワン・パス・サイズまたはマルチ・パス・サイズで実行された回数の割合を示しています。この問合せでは、一定のサイズ(最適メモリー要件が最低64KB)の作業領域のみ考慮されます。

```
SELECT optimal_count, ROUND(optimal_count*100/total, 2) optimal_perc,
       onepass_count, ROUND(onepass_count*100/total, 2) onepass_perc,
       multipass_count, ROUND(multipass_count*100/total, 2) multipass_perc
FROM
 (SELECT DECODE(SUM(total_executions), 0, 1, SUM(total_executions)) total,
         SUM(optimal_executions) optimal_count,
         SUM(onepass_executions) onepass_count,
         SUM(multipass_executions) multipass_count
    FROM V$SQL_WORKAREA_HISTOGRAM
   WHERE low_optimal_size >= 64*1024);
```

この問合せの出力例を次に示します。

```
OPTIMAL_COUNT OPTIMAL_PERC ONEPASS_COUNT ONEPASS_PERC MULTIPASS_COUNT MULTIPASS_PERC
------------- ------------ ------------- ------------ ---------------- --------------
         2239        81.63           504        18.37                0              0
```

この例の出力は、81.63%の作業領域が最適なサイズで実行できたことを示しています。残りの作業領域(18.37%)はワン・パス・サイズで実行され、マルチ・パス・サイズで実行されたものはありませんでした。

V$WORKAREA_ACTIVEビューの使用

V$WORKAREA_ACTIVEビューには、データベース・インスタンスでアクティブ(または実行中)な作業領域が表示されます。小さいアクティブなソート操作(64KB未満)はこのビューから除外されます。すべてのアクティブな作業領域のサイズを正確に監視したり、それらのアクティブな作業領域が一時セグメントに流用されているかどうかを判断するには、このビューを使用します。

例16-5に、このビューの問合せを示します。

例16-5 V$WORKAREA_ACTIVEビューの問合せ

```
SELECT TO_NUMBER(DECODE(sid, 65535, null, sid)) sid,
       operation_type operation,
       TRUNC(expected_size/1024) esize,
       TRUNC(actual_mem_used/1024) mem,
       TRUNC(max_mem_used/1024) "max mem",
       number_passes pass,
       TRUNC(TEMPSEG_SIZE/1024) tsize
  FROM V$SQL_WORKAREA_ACTIVE
 ORDER BY 1,2;
```

この問合せの出力例を次に示します。

```
SID          OPERATION     ESIZE       MEM   MAX MEM  PASS   TSIZE
--- ----------------- --------- --------- --------- ----- -------
  8   GROUP BY (SORT)       315       280       904     0
  8         HASH-JOIN      2995      2377      2430     1   20000
  9   GROUP BY (SORT)     34300     22688     22688     0
 11         HASH-JOIN     18044     54482     54482     0
 12         HASH-JOIN     18044     11406     21406     1  120000
```

この例の出力は、次のことを示しています。

- 作業領域がワン・パス・サイズ(PASS列)で動作しているハッシュ結合操作(OPERATION列)を、セッション12(SID列)が実行しています
- PGAメモリー・マネージャが予測する、このハッシュ結合操作で使用する最大メモリー量は18044 KB (ESIZE列)です
- 作業領域は現在11406KBのメモリー(MEM列)を使用しています
- 作業領域は、過去に最大21406KBのPGAメモリー(MAX MEM列)を使用しました
- 作業領域は、120000KBの一時セグメント (TSIZE列)に流用されました

作業領域を割当て解除されたとき、すなわち、関連するSQL演算子の実行が完了したときに、このビューから自動的に削除されます。

V$SQL_WORKAREAビューの使用

実行計画が1つ以上の作業領域を使用するカーソルがロードされるたびに、累積された作業領域の統計がメンテナンスされます。作業領域が割当て解除されるたびに、V$SQL_WORKAREAビューがその作業領域の実行統計で更新されます。

V$SQL_WORKAREAビューをV$SQLビューと結合して、作業領域をカーソルに関連付けることができ、V$SQL_PLANビューと結合した場合は、計画のどの演算子が作業領域を使用するかを正確に判断できます。

例16-6に、このビューの問合せを3つ示します。

例16-6 V$SQL_WORKAREAビューの問合せ

次の問合せでは、最もキャッシュ・メモリーを必要とする上位10個の作業領域を検索します。

```
SELECT *
FROM   (SELECT workarea_address, operation_type, policy, estimated_optimal_size
        FROM V$SQL_WORKAREA
        ORDER BY estimated_optimal_size DESC)
 WHERE ROWNUM <= 10;
```

次の問合せでは、ワン・パスまたはマルチ・パスで実行された1つ以上の作業領域を持つカーソルを検索します。

```
col sql_text format A80 wrap
SELECT sql_text, sum(ONEPASS_EXECUTIONS) onepass_cnt,
       sum(MULTIPASSES_EXECUTIONS) mpass_cnt
FROM V$SQL s, V$SQL_WORKAREA wa
WHERE s.address = wa.address
GROUP BY sql_text
HAVING sum(ONEPASS_EXECUTIONS+MULTIPASSES_EXECUTIONS)>0;
```

特定のカーソルのハッシュ値とアドレスを使用することにより、関連する作業領域に関する情報を含むカーソル実行計画が次の問合せで表示されます。

```
col "O/1/M" format a10
col name format a20
SELECT operation, options, object_name name, trunc(bytes/1024/1024) "input(MB)",
       TRUNC(last_memory_used/1024) last_mem,
       TRUNC(estimated_optimal_size/1024) optimal_mem,
       TRUNC(estimated_onepass_size/1024) onepass_mem,
       DECODE(optimal_executions, null, null,
```

```
             optimal_executions||'/'||onepass_executions||'/'||
             multipasses_executions) "O/1/M"
  FROM V$SQL_PLAN p, V$SQL_WORKAREA w
 WHERE p.address=w.address(+)
   AND p.hash_value=w.hash_value(+)
   AND p.id=w.operation_id(+)
   AND p.address='88BB460C'
   AND p.hash_value=3738161960;
```

この問合せの出力例を次に示します。

```
OPERATION    OPTIONS  NAME     input(MB) LAST_MEM OPTIMAL_ME ONEPASS_ME O/1/M
------------ -------- -------- --------- -------- ---------- ---------- ------
SELECT STATE
HASH         GROUP BY              4582        8         16         16 16/0/0
HASH JOIN    SEMI                  4582     5976       5194       2187 16/0/0
TABLE ACCESS FULL     ORDERS          51
TABLE ACCESS FUL      LINEITEM      1000
```

アドレスおよびハッシュ値は、次の問合せに示すように、問合せでパターンを指定することによりV$SQLビューから取得できます。

```
SELECT address, hash_value
  FROM V$SQL
 WHERE sql_text LIKE '%my_pattern%';
```

PGA_AGGREGATE_TARGETのチューニング

PGA_AGGREGATE_TARGET初期化パラメータの値のチューニングを容易にするため、Oracle Databaseには、V$PGA_TARGET_ADVICEおよびV$PGA_TARGET_ADVICE_HISTOGRAMの2つのPGAパフォーマンス・アドバイザ・ビューが提供されています。これらのビューを使用することで、経験的な方法でPGA_AGGREGATE_TARGETパラメータの値をチューニングする必要がありません。かわりに、これらのビューを使用して、PGA_AGGREGATE_TARGETパラメータの値を変更すると、主要なPGA統計にどのような影響があるかを予測できます。

この項では、PGA_AGGREGATE_TARGET初期化パラメータの値をチューニングする方法を説明しており、内容は次のとおりです。

- PGAパフォーマンス・アドバイザ・ビューの自動生成の有効化
- V$PGA_TARGET_ADVICEビューの使用
- V$PGA_TARGET_ADVICE_HISTOGRAMビューの使用
- V$SYSSTATおよびV$SESSTATビューの使用
- チュートリアル: PGA_AGGREGATE_TARGETのチューニング方法

PGAパフォーマンス・アドバイザ・ビューの自動生成の有効化

Oracle Databaseでは、ワークロードの履歴を記録し、その履歴をPGA_AGGREGATE_TARGETパラメータの様々な値でシミュレートすることによって、V$PGA_TARGET_ADVICEおよびV$PGA_TARGET_ADVICE_HISTOGRAMビューを生成します。可能性のある高い値および低い値を評価するため、PGA_AGGREGATE_TARGETパラメータの値は、その現在の値の分数または倍数から導出されます。これらの値は予測に使用され、範囲は10MBから最大256GBまでです。このシミュレーション・プロセスはバックグラウンドで実行され、ワークロードの履歴を継続的に更新してシミュレーション結果を生成します。これらのビューを問い合せれば、いつ

でも結果を表示できます。

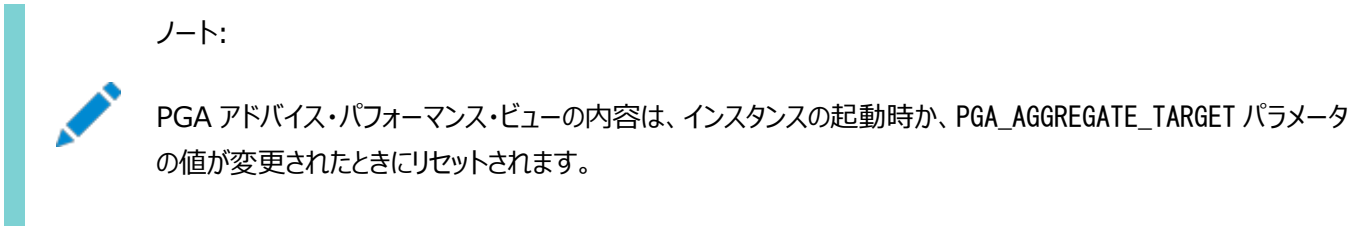
PGAパフォーマンス・アドバイザ・ビューの自動生成を有効化するには:

1. PGA_AGGREGATE_TARGETパラメータを設定して、自動PGAメモリー管理を有効化します。

   このパラメータを0に設定すると、自動PGAメモリー管理を無効化できますが、推奨されません。このパラメータの設定方法の詳細は、「PGA_AGGREGATE_TARGETの初期値の設定」を参照してください。

2. STATISTICS_LEVELパラメータをTYPICAL (デフォルト)またはALLに設定します。

   このパラメータをBASICに設定すると、PGAパフォーマンス・アドバイス・ビューの生成を無効化できますが、推奨されません。



ノート:

PGA アドバイス・パフォーマンス・ビューの内容は、インスタンスの起動時か、PGA_AGGREGATE_TARGET パラメータの値が変更されたときにリセットされます。

V$PGA_TARGET_ADVICEビューの使用

V$PGA_TARGET_ADVICEビューでは、PGA_AGGREGATE_TARGET初期化パラメータの値を変更した場合に、V$PGASTATビューの次の統計にどのような影響があるかを予測できます。

- cache hit percentage
- over allocation count

次の例に、このビューの問合せを示します。

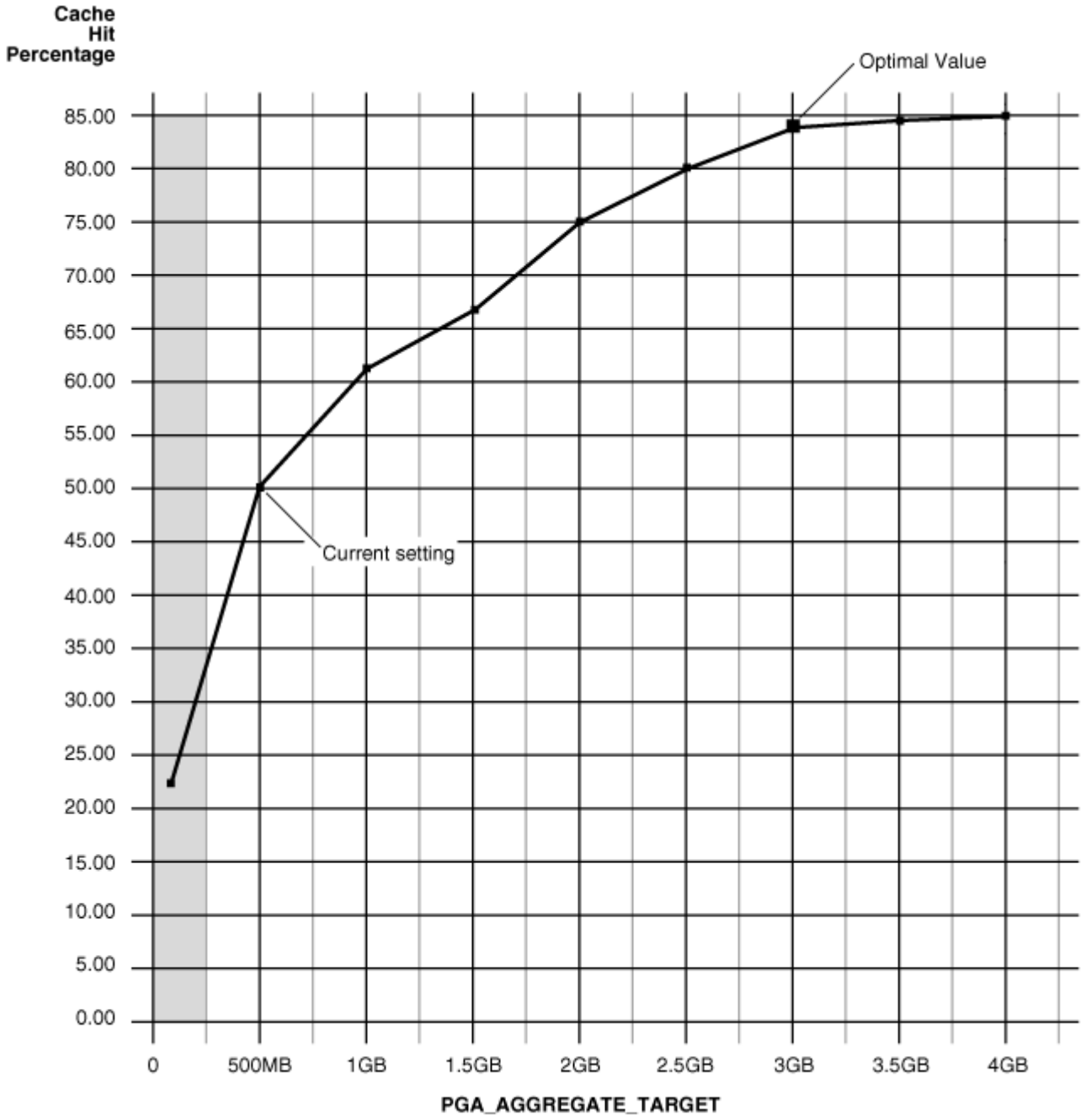
```
SELECT ROUND(pga_target_for_estimate/1024/1024) target_mb,
       estd_pga_cache_hit_percentage cache_hit_perc,
       estd_overalloc_count
  FROM V$PGA_TARGET_ADVICE;
```

この問合せの出力例を次に示します。

```
 TARGET_MB  CACHE_HIT_PERC  ESTD_OVERALLOC_COUNT
----------  --------------  --------------------
        63              23                   367
       125              24                    30
       250              30                     3
       375              39                     0
       500              58                     0
       600              59                     0
       700              59                     0
       800              60                     0
       900              60                     0
      1000              61                     0
      1500              67                     0
      2000              76                     0
      3000              83                     0
      4000              85                     0
```

次の図は、この問合せの結果をプロットする方法を示しています。

図16-1 V$PGA_TARGET_ADVICEのグラフ



この曲線は、PGA_AGGREGATE_TARGETパラメータの値が増加するにつれて、PGAのcache hit percentageが上昇する様子を示しています。グラフの陰付きのゾーンは、ESTD_OVERALLOCATION_COUNT列の値がゼロではない、over allocationゾーンを表します。この領域は、PGA_AGGREGATE_TARGETパラメータの値が小さすぎるため、PGAメモリーの最低要件を満たしていないことを示します。PGA_AGGREGATE_TARGETパラメータの値をover allocationゾーン内に設定すると、メモリー・マネージャによってメモリーが過剰に割り当てられ、実際に消費されたPGAメモリーが設定された制限を超過します。したがって、PGA_AGGREGATE_TARGETパラメータの値をそのゾーンに設定しても意味がありません。この例では、PGA_AGGREGATE_TARGETパラメータは最低375MBに設定する必要があります。

over allocationゾーンを超えると、PGAのcache hit percentageの値が急速に増加します。これは最適、またはワン・パスの作業領域の数が増加し、マルチ・パス実行の数が減少するためです。この例では、500MB付近で曲線の変化が発生してい

ますが、これはほとんどの(場合によってはすべての)作業領域が最適なサイズ、または少なくともワン・パス・サイズで実行可能になるポイントに対応しています。このポイントを超えてcache hit percentageは緩やかに増加し、先細りが始まってPGA_AGGREGATE_TARGETパラメータの値の増加による上昇がわずかしかないポイントに達します。図では、この状態はPGA_AGGREGATE_TARGETが3GBに達したときに発生しています。この時点で、cache hit percentageは83%で、PGAメモリーに1GBを追加してもわずかしか向上(2%)しません。この例では、PGA_AGGREGATE_TARGETパラメータの最適値は3GBです。

ノート:

PGA の cache hit percentage の理論的な最大値が 100%の場合でも、作業領域の実際の最大サイズには制限があるため、PGA_AGGREGATE_TARGET パラメータの値をさらに増やしても、理論的な最大値に達しない場合があります。これが発生するのは、最適メモリー要件が大きく、cache hit percentage の値が低いヒット率(90%など)に下がる可能性のある大規模 DSS システムのみです。

PGA_AGGREGATE_TARGETパラメータの値は、最適値に設定するか、少なくともover allocationゾーンを超えた範囲で可能な最大値に設定するのが理想的です。一般的には、PGAのcache hit percentageは60%以上に設定します。60%の時点でシステムは、理想的な状況で実際に処理する必要のあるバイト数のほぼ2倍の量を処理するためです。この例では、PGA_AGGREGATE_TARGETパラメータの値は最低500MBで、可能なかぎり3GBに近い値に設定する必要があります。ただし、PGA_AGGREGATE_TARGETパラメータの適切な設定は、PGAコンポーネントにどれだけのメモリーを使用できるかによって異なります。一般的に、PGAメモリーを追加する場合は、共有プールまたはバッファ・キャッシュなど、一部のSGAコンポーネントのメモリーを減らす必要がありますが、これはデータベース・インスタンスに使用する全メモリーが、システムで使用できる物理メモリー量の制約を受ける場合があるためです。したがって、システム内で使用可能なメモリーと、様々なSGAコンポーネントのパフォーマンス(共有プール・アドバイザの統計およびバッファ・キャッシュ・アドバイザの統計で監視)という、より大きな視点で、PGAメモリーの増量の決定を下す必要があります。メモリーをSGAコンポーネントから減らせない場合は、システムへの物理メモリーの追加を検討してください。

V$PGA_TARGET_ADVICE_HISTOGRAMビューの使用

V$PGA_TARGET_ADVICE_HISTOGRAMビューでは、PGA_AGGREGATE_TARGET初期化パラメータの値を変更した場合に、V$SQL_WORKAREA_HISTOGRAMビューの統計にどのような影響があるかが予測されます。このビューを使用すると、予測に使用するPGA_AGGREGATE_TARGET値における、最適、ワン・パス、マルチ・パスの各作業領域の予測実行回数に関する詳細情報を表示できます。

V$PGA_TARGET_ADVICE_HISTOGRAMビューはV$SQL_WORKAREA_HISTOGRAMビューと同じであり、予測に使用するPGA_AGGREGATE_TARGET値を示す2つの追加列があります。したがって、PGA_AGGREGATE_TARGETパラメータの希望値を選択するための追加の述語を使用し、V$SQL_WORKAREA_HISTOGRAMビューに対して実行される任意の問合せをこのビューで使用できます。

例16-7に、PGA_AGGREGATE_TARGETパラメータの値を現在の値の2倍に設定したときの、V$SQL_WORKAREA_HISTOGRAMビューの予測内容を表示するこのビューの問合せを示します。

例16-7 V$PGA_TARGET_ADVICE_HISTOGRAMビューの問合せ

```
SELECT low_optimal_size/1024 low_kb, (high_optimal_size+1)/1024 high_kb,
```

```
       estd_optimal_executions estd_opt_cnt,
       estd_onepass_executions estd_onepass_cnt,
       estd_multipasses_executions estd_mpass_cnt
  FROM V$PGA_TARGET_ADVICE_HISTOGRAM
 WHERE pga_target_factor = 2
   AND estd_total_executions != 0
 ORDER BY 1;
```

この問合せの出力例を次に示します。

```
LOW_KB   HIGH_KB   ESTD_OPTIMAL_CNT   ESTD_ONEPASS_CNT   ESTD_MPASS_CNT
------   -------   ----------------   ----------------   --------------
     8        16             156107                  0                0
    16        32                148                  0                0
    32        64                 89                  0                0
    64       128                 13                  0                0
   128       256                 58                  0                0
   256       512                 10                  0                0
   512      1024                653                  0                0
  1024      2048                530                  0                0
  2048      4096                509                  0                0
  4096      8192                227                  0                0
  8192     16384                176                  0                0
 16384     32768                133                 16                0
 32768     65536                 66                103                0
 65536    131072                 15                 47                0
131072    262144                  0                 48                0
262144    524288                  0                 23                0
```

この例の出力は、PGA_AGGREGATE_TARGETパラメータの値を2倍ずつ増やしていくと、16MB未満のすべての作業領域を最適サイズで実行できることを示しています。

V$SYSSTATおよびV$SESSTATビューの使用

V$SYSSTATビューとV$SESSTATビューの統計は、最適、ワン・パスおよびマルチ・パスの各メモリー・サイズで実行された作業領域の総数を示します。これらの統計は、インスタンスまたはセッションが開始された後から累積されます。

例16-8に、インスタンスの起動後にこれら3つのサイズで作業領域が実行された合計回数と割合を表示するV$SYSSTATビューの問合せを示します。

例16-8 V$SYSSTATビューの問合せ

```
SELECT name profile, cnt, DECODE(total, 0, 0, ROUND(cnt*100/total)) percentage
  FROM (SELECT name, value cnt, (SUM(value) over ()) total
  FROM V$SYSSTAT
 WHERE name
  LIKE 'workarea exec%');
```

この問合せの出力例を次に示します。

```
PROFILE                                  CNT PERCENTAGE
---------------------------------- ---------- ----------
workarea executions - optimal             5395         95
workarea executions - onepass              284          5
workarea executions - multipass              0          0
```

この例の出力では、5,395 (または95%)の作業領域実行が最適なサイズで実行され、284 (または5%)の作業領域実行がワン・パス・サイズで実行されたことを示しています。

チュートリアル: PGA_AGGREGATE_TARGETのチューニング方法

このチュートリアルは、この章で説明されている様々なビューを使用して、PGA_AGGREGATE_TARGETパラメータの値をチューニングするためのガイドラインです。

PGA_AGGREGATE_TARGETをチューニングするには:

1. メモリーを過剰に割り当てないようにPGA_AGGREGATE_TARGETパラメータの値を設定します。

   「V$PGA_TARGET_ADVICEビューの使用」で説明されているように、V$PGA_TARGET_ADVICEビューを使用して、PGA_AGGREGATE_TARGETの値がover-allocationゾーン内に設定されていないことを確認します。例16-8では、PGA_AGGREGATE_TARGETの値は最低375MBに設定する必要があります。

2. レスポンス時間の要件およびメモリーの制約に基づいて、PGAのcache hit percentageを最大化します。

   「V$PGA_TARGET_ADVICEビューの使用」で説明されているように、V$PGA_TARGET_ADVICEビューを使用して、PGA_AGGREGATE_TARGETパラメータの最適値を判断し、パラメータをその値に設定するか可能なかぎり最大の値に設定します。

   PGAに割当て可能なメモリーの制限Xを推定します。

   - 制限Xが最適値より大きい場合は、PGA_AGGREGATE_TARGETパラメータの値を最適値に設定します。

     例16-8で、10GBをPGA専用とする場合は、PGA_AGGREGATE_TARGETパラメータの値を3GBに設定し、残りの7GBをSGAに使用します。

   - 制限Xが最適値より小さい場合は、PGA_AGGREGATE_TARGETパラメータの値をXに設定します。

     例16-8で、2GBのみをPGA専用とする場合は、PGA_AGGREGATE_TARGETパラメータの値を2GBに設定し、75%のcache hit percentageを受け入れます。

3. PGA_AGGREGATE_TARGETパラメータの新しい値が、最適およびワン・パスの作業領域実行の必要な数になることを検証し、マルチ・パス作業領域実行を回避します。

   「V$PGA_TARGET_ADVICE_HISTOGRAMビューの使用」で説明されているように、V$PGA_TARGET_ADVICE_HISTOGRAMビューを使用して、最適、ワン・パスおよびマルチ・パスの各作業領域実行の数を予測します。

4. より多くのPGAメモリーが必要な場合は、SGAコンポーネントからメモリーを減らすか、システムに物理メモリーを追加してPGAメモリーを増やします。

5. 任意の時点で、最適、ワン・パスおよびマルチ・パスの各作業領域実行の数が予測に一致することを確認し、必要な場合はPGA_AGGREGATE_TARGETパラメータの値をチューニングします。

   「V$SYSSTATおよびV$SESSTATビューの使用」で説明されているように、V$SYSSTATおよびV$SESSTATビューを使用して、インスタンスの起動またはセッションの開始以降の最適、ワン・パスおよびマルチ・パスの各メモリー・サイズで実行された作業領域の合計数を確認します。

絶対制限指定によるプログラム・グローバル領域のサイズ設定

自動PGAメモリー管理モードでは、Oracle Databaseは、作業領域に割り当てられるPGAのメモリー量を動的に制御するこ

とで、PGA_AGGREGATE_TARGETの値への準拠を試行します。ただし、次の理由で、PGAメモリー使用量がPGA_AGGREGATE_TARGET設定を超える場合があります。

- PGA_AGGREGATE_TARGET設定が制限ではなく、目標として機能するため。
- PGA_AGGREGATE_TARGETで制御されるのがチューニング可能なメモリーの割当てのみであるため。

PGAを過剰に使用すると、スワッピング率が高くなる可能性があります。そうなった場合は、システムが応答せず、不安定になります。その場合、次のいずれかの方法を使用して、PGAメモリー使用量に絶対制限を指定することを検討してください。

- PGA_AGGREGATE_LIMITパラメータを使用して、全体のPGAメモリーの使用量に対して絶対制限を設定します。

  PGA_AGGREGATE_LIMITパラメータを使用したプログラム・グローバル領域のサイズ設定を参照してください
- リソース・マネージャのプロシージャDBMS_RESOURCE_MANAGER. CREATE_PLAN_DIRECTIVEを使用して、特定のコンシューマ・グループ内の各セッションのPGAメモリー使用量に絶対制限を設定します。

  リソース・マネージャを使用したプログラム・グローバル領域のサイズ設定を参照してください

PGA_AGGREGATE_LIMITパラメータを使用したプログラム・グローバル領域のサイズ設定

PGA_AGGREGATE_LIMIT初期化パラメータを使用すると、PGAメモリー使用量に絶対制限を指定できます。PGA_AGGREGATE_LIMITの値を超過している場合は、Oracle Databaseにより、チューニングできないPGAメモリーを最も消費しているセッションやプロセスが、次の順番で中断または終了されます。

- チューニングできないPGAメモリーを最も消費しているセッションのセルが中断されます。
- PGAのメモリー使用量が依然としてPGA_AGGREGATE_LIMITを超過している場合は、チューニングできないPGAメモリーを最も消費しているセッションおよびプロセスが終了されます。

中断または終了するセッションやプロセスを決定する際、Oracle Databaseでは、パラレル問合せが1つの単位として処理されます。

デフォルトで、PGA_AGGREGATE_LIMITパラメータは2GBより大きく設定されるか、PGA_AGGREGATE_TARGET値の200%、またはPROCESSESパラメータの値の300万倍に設定されます。ただし、物理メモリーのサイズからSGAの合計サイズを引いた数の120%を超えることはありません。デフォルト値はアラート・ログに印刷されます。システムの物理メモリー量を判別できない場合は、警告のメッセージがアラート・ログに印刷されます。

PGA_AGGREGATE_LIMITを設定するには:

- PGA_AGGREGATE_LIMIT初期化パラメータを、バイト単位で必要な値に設定します。

  値は、K (KBの場合)、M (MBの場合)またはG (GBの場合)が続く数値として表されます。値を0に設定すると、PGAメモリーの強い制限が無効化されます。

関連項目:

- PGA_AGGREGATE_LIMIT初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください
- V$PGASTATビューの詳細は、『Oracle Databaseリファレンス』を参照してください

- Oracle Database Resource Managerおよびコンシューマ・グループの詳細は、『Oracle Database管理者ガイド』を参照してください

リソース・マネージャを使用したプログラム・グローバル領域のサイズ設定

Oracle Databaseリソース・マネージャのDBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVEプロシージャでSESSION_PGA_LIMITパラメータを使用して、特定のコンシューマ・グループ内の各セッションに割り当てられるPGAメモリー使用量に絶対制限を設定できます。セッションがコンシューマ・グループに設定されたPGAメモリー制限を超過した場合、そのセッションは終了しORA-10260エラー・メッセージが表示されます。

関連項目:

- Oracle Database管理者ガイドのトピック
  - コンシューマ・グループ内の各セッションに対するPGAメモリーの制限の詳細は、プログラム・グローバル領域(PGA)に関する項を参照してください。
  - DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVEプロシージャを使用したリソース・プラン・ディレクティブの作成の詳細は、リソース・プラン・ディレクティブの作成に関する項を参照してください。
- DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVEプロシージャの構文は、Oracle Database PL/SQLパッケージおよびタイプ・リファレンスを参照してください。

第IV部 システム・リソースの管理

この部は、次の章で構成されています。

- I/O構成および設計
- オペレーティング・システム・リソースの管理

17 I/O構成および設計

I/Oサブシステムは、Oracleデータベースに不可欠なコンポーネントです。基本的なI/O概念を紹介し、データベースの様々な部分のI/O要件について説明し、I/Oサブシステムの設計のための構成例を示します。

この章の内容は次のとおりです。

- I/Oについて
- I/O構成
- データベース内部のI/O測定
- Oracle Orion測定ツールによるI/O測定

I/Oについて

すべてのOracleデータベースでは、ディスク上のデータが読み書きされ、ディスクI/Oが生成されます。多くのソフトウェア・アプリケーションのパフォーマンスは、本質的にディスクI/Oによって制限されます。CPUタイムの大部分をI/Oアクティビティが完了するまでの待機に使用するアプリケーションはI/Oバウンドと呼ばれます。

Oracle Databaseは、適切に作成されたアプリケーションのパフォーマンスが、I/Oで制限されないように設計されています。I/Oシステムが最大限またはそれに近い状態で動作しており、許容時間内にI/Oリクエストに対応できない場合は、I/Oのチューニングを行うと、アプリケーションのパフォーマンスを向上できます。ただし、アプリケーションがI/Oバウンドではない場合(たとえば、CPUが制限要因である場合)、I/Oをチューニングしてもパフォーマンスを改善できません。

I/Oシステムを設計するときは、次のデータベース要件を考慮してください。

- ディスクの最小容量などの記憶域
- 継続的(24時間×7日)または営業時間のみなどの可用性
- I/Oスループットやアプリケーション・レスポンス時間などのパフォーマンス

多くのI/O設計では、パフォーマンスが問題とならないと仮定して記憶域要件および可用性要件の計画を立てています。ただし、これに該当しない場合もあります。構成するディスクおよびコントローラの数はI/Oスループットおよび冗長性の要件で判断することが最適です。この場合、ディスクのサイズは記憶域要件で判断できます。

I/O設計計画を作成する場合は、Oracle Automatic Storage Management(Oracle ASM)の使用を検討します。Oracle ASMは、記憶域の管理を管理者に任せるのではなく、データベースで管理するという方針に基づいた、高いパフォーマンスを実現する統合データベース・ファイル・システムおよびディスク・マネージャです。

データベース・ファイル記憶域には、RAWデバイスやオペレーティング・システムのファイル・システムではなく、Oracle ASMを使用することをお薦めします。Oracle ASMの主な利点を次に示します。

- ストライプ化
- ミラー化
- オンラインでの記憶域の再構成と動的リバランス

- 管理対象ファイルの作成と削除

関連項目:

Oracle ASMの詳細は、『Oracle Automatic Storage Management管理者ガイド』を参照してください。

I/O構成

この項では、収集する基本情報およびシステムのI/O構成を定義するときの決定事項について説明します。必要な可用性、リカバリ能力およびパフォーマンスは維持しながら、できるだけ単純な構成を維持します。構成が複雑になるに従って、管理、メンテナンスおよびチューニングが困難になります。

この項では、次の項目について説明します。

- オペレーティング・システムまたはハードウェアのストライプ化を使用したファイルのレイアウト
- 手動によるI/Oの分散
- ファイルを分割する場合
- 3つの構成サンプル
- Oracle Managed Files
- データ・ブロック・サイズの選択

オペレーティング・システムまたはハードウェアのストライプ化を使用したファイルのレイアウト

オペレーティング・システムにLVMソフトウェアまたはハードウェア・ベースのストライプ化がある場合、これらのツールを使用してI/Oを分散できます。LVMまたはハードウェア・ストライプ化を使用するときの決定事項には、**ストライプ深度**および**ストライプ幅**が含まれます。

- ストライプ深度は、ストライプのサイズで、ストライプ単位とも呼ばれます。
- ストライプ幅は、ストライプ深度とストライプ・セットを構成するドライブ数の積です。

これらの値を適切に選択して、システムが必要なスループットを維持できるようにします。Oracleデータベースにおける適切なストライプ深度は、256KBから1MBです。ストライプ深度によって、得られるアプリケーションの利点の種類が異なります。最適なストライプ深度およびストライプ幅は、次の項目により異なります。

- リクエストされたI/Oサイズ
- I/Oリクエストの同時実行性
- 物理ストライプ境界とブロック・サイズ境界との位置合せ
- 提案されたシステムの管理性

リクエストされたI/Oサイズ

表17-1に、I/Oサイズの設定で使用できるOracle Databaseおよびオペレーティング・システムのパラメータを示します。

表17-1 Oracle Databaseおよびオペレーティング・システム操作パラメータ

パラメータ	説明
DB_BLOCK_SIZE	単一ブロック I/O リクエストのサイズ。また、このパラメータをマルチブロック・パラメータと組み合せて使用して、マルチブロック I/O リクエスト・サイズを決定します。
OS ブロック・サイズ	REDO ログおよびアーカイブ・ログ操作の I/O サイズを決定します。
最大 OS I/O サイズ	単一 I/O リクエストのサイズに上限を設けます。
DB_FILE_MULTIBLOCK_READ_COUNT	全表スキャンの最大 I/O サイズは、このパラメータに DB_BLOCK_SIZE を乗算して計算されます。(上限値は、オペレーティング・システムの制限を受けます。)この値が明示的に設定されていない場合(または 0 に設定されている場合)のデフォルト値は、効率的に実行可能な最大 I/O サイズですが、プラットフォームごとに異なります。
SORT_AREA_SIZE	ソート操作のための I/O サイズおよび同時実行性を決定します。
HASH_AREA_SIZE	ハッシュ操作のための I/O サイズを決定します。

I/Oサイズの他に、同時実行性の程度も理想的なストライプ深度を調べる上で役立ちます。ストライプ幅およびストライプ深度を選択する場合は、次の点を考慮してください。

- 同時実行性の低い(順次)システムでは、単一I/Oが同じディスクに2回アクセスしないようにします。たとえば、ストライプ幅が4つのディスクで、ストライプ深度が32KBであると仮定します。1MBの単一I/Oリクエスト(たとえば、全表スキャンの場合)がOracleサーバー・プロセスで発行された場合、ストライプ内の各ディスクは要求されたデータを戻すために8回I/Oを実行する必要があります。このような状況を回避するために、平均I/Oのサイズは、ストライプ深度とストライプ幅の積より小さいサイズにしてください。そうでない場合は、オペレーティング・システムに対してOracle Databaseが単一I/Oリクエストを行うと、同じディスクに対して複数の物理I/Oリクエストが発生します。
- 同時実行性が高い(ランダム)システムでは、単一I/Oリクエストが複数の物理I/Oコールに分解されないようにしてください。そうでなければ、システムで実行される物理I/Oリクエスト数が何倍にもなり、I/Oレスポンス時間が大幅に下がります。

I/Oリクエストの同時実行性

従来のOLTP環境などの高度な小さい同時I/Oリクエストが存在するシステムでは、ストライプ深度を大きく保つことが有効です。I/Oサイズより大きいストライプ深度を使用することは粗密なストライプ化と呼ばれます。同時実行性の高いシステムのストライプ深度は次のようになります(n > 1)。

```
n * DB_BLOCK_SIZE
```

粗密なストライプ化ではアレイ内の1ディスクで複数のI/Oリクエストに対応できます。この方法では、一連のストライプ化ディスクで多数の同時I/Oリクエストに対応でき、I/Oセットアップ・コストも最小限で済みます。粗密なストライプ化は、全体的なI/Oスループットの最大化を目指します。ストライプ深度が大きく、1つのドライブからサービスできる場合、全表スキャンによるマルチブロック読取りにメリットをもたらします。データ・ウェアハウス環境のパラレル問合せも、粗密なストライプ化の候補です。これは、それぞれ個別のI/Oを発行する個々のプロセスが多数存在するためです。粗密なストライプ化は、同時リクエストが少ないシステ

ムで使用されると、ホット・スポットが生成される可能性があります。

従来のDSS環境や同時実行性の低いOLTPシステムなどの大きいI/Oリクエストが少ないシステムでは、ストライプ深度を小さく保つことが有益です。これは、ファイングレイン・ストライプ化と呼ばれます。このようなシステムでは、ストライプ深度は次のようになります。nはDB_FILE_MULTIBLOCK_READ_COUNTなどのマルチブロック読取りパラメータよりも小さくなります。

```
n * DB_BLOCK_SIZE
```

ファイングレイン・ストライプ化では、複数のディスクで単一I/Oリクエストを処理できます。ファイングレイン・ストライプ化では、個々のI/Oリクエストまたはレスポンス時間のパフォーマンスが最大化されます。

物理ストライプ境界とブロック・サイズ境界との位置合せ

一部のOracle Databaseポートでは、データベース・ブロック境界がストライプと合わない可能性があります。ストライプ深度がデータベースのブロック・サイズと同じである場合、Oracle Databaseから発行される1つのI/Oによって、2つの物理I/O操作が生じる場合があります。

これは、OLTP環境では最適ではありません。1つの論理I/Oで物理I/Oが1つのみ発生する確率が高くなるようにするには、最小のストライプ深度がOracleブロック・サイズの少なくとも2倍である必要があります。表17-2に、ランダム・アクセスおよび順次読取りでお薦めする最小ストライプ深度を示します。

表17-2 最小ストライプ深度

ディスク・アクセス	最小ストライプ深度
ランダム読取りおよび書込み	最小ストライプ深度は、Oracle ブロック・サイズの 2 倍です。
順次読取り	最小ストライプ深度は、Oracle ブロック・サイズを乗算した DB_FILE_MULTIBLOCK_READ_COUNT の値の 2 倍です。

関連項目:

プラットフォームの固有のマニュアル

提案されたシステムの管理性

LVMの場合、最も管理の簡単な構成は、使用可能なすべてのディスク上に単一ストライプ化ボリュームを構成したものです。この場合、ストライプ幅はすべての使用可能なディスクを包含します。すべてのデータベース・ファイルはそのボリューム内に常駐し、効果的に負荷を均等分散します。この単一ボリューム・レイアウトは、ほとんどの状況で適切なパフォーマンスを実現します。

単一ボリューム構成は、RAID 1などの簡単なリカバリを可能にするRAID技術と併用する場合のみ有効です。それ以外の場合、単一のディスクを失うことはすべてのファイルを同時に失うことであり、完全なデータベースのリストアおよびリカバリを実行する必要があることを意味します。

パフォーマンスの他に、管理性の問題があります。システムの設計で、ディスクを簡単に追加できるようにして、データベースの拡

張を可能にする必要があります。課題は、負荷のバランスを均一に維持しながら拡張を行うことです。

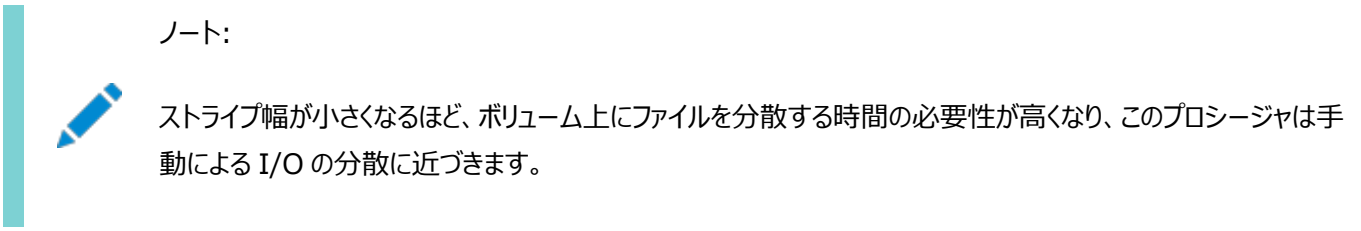
たとえば、初期構成で、64個の16GBのディスク上に単一ストライプ化ボリュームを作成する必要があるとします。これはプライマリ・データの1TBの合計ディスク領域になります。場合によっては、システムが動作した後に、将来のデータベース拡張を可能にするためにさらに80GB(すなわち、5つのディスク)を追加する必要があります。

この領域をデータベースで使用できるようにするオプションには、5つの新しいディスクを含む第2のボリュームの作成があります。ただし、これらの新しいディスクがその上に配置されたファイルに必要なI/Oスループットを保持できない場合、I/Oボトルネックが発生する可能性があります。

もう1つのオプションは、元のボリュームのサイズを増やすことです。LVMは高度になりつつあり、ストライプ幅の動的な再構成を可能にするので、システムがオンライン中にディスクを追加できます。このため、本番環境で単一ストライプ化ボリューム上にすべてのファイルを配置できるようになってきました。

LVMがストライプへのディスクの動的な追加をサポートできない場合は、より小さく管理しやすいストライプ幅を選択する必要があります。そのようにすると、新しいディスクを追加する場合、ストライプ幅だけシステムを拡張できます。

前述の例で、8個のディスクがより管理しやすいストライプ幅といえます。これは、8個のディスクで1秒間に必要な数のI/Oを維持できる場合のみ可能です。したがって、追加のディスク領域が必要なときは、別の8ディスク・ストライプを追加して、ボリューム間でI/Oのバランスを維持できます。



ノート:

ストライプ幅が小さくなるほど、ボリューム上にファイルを分散する時間の必要性が高くなり、このプロシージャは手動による I/O の分散に近づきます。

手動によるI/Oの分散

システムにLVMまたはハードウェアのストライプ化がない場合、各ファイルのI/O要件に従ってファイルを分散することにより、使用可能なディスク間でI/Oを手動でバランス化する必要があります。ファイルの配置に関する決定を行うには、データベース・ファイルのI/O要件およびI/Oシステムの機能についてよく理解している必要があります。このようなデータに慣れておらず、解析対象の代表的なワークロードを取得できない場合は、まず推定を行い、次に使用量がわかったときにこのレイアウトをチューニングします。

ディスクを手動でストライプ化するには、ファイルの記憶域要件をI/O要件と関連付ける必要があります。

1. ファイルおよびディスクのサイズをチェックして、データベースのディスク記憶域要件を評価します。
2. 1ファイル当たりの予測I/Oスループットを識別します。最高のI/O率を持つファイルおよび多数のI/Oを持たないファイルを判断します。I/O率を均等にするために、すべての使用可能なディスク上にファイルをレイアウトします。

手動I/O分散の一般的なアプローチとして、頻繁に使用される表をその索引から分離することがあげられます。これは正しくありません。一連のトランザクション中は、索引が読み取られてから表が読み取られます。これらのI/Oは順次に発生するので、表と索引を同じディスク上に格納しても競合は発生しません。データファイルには索引や表データが含まれているため、これを単純に分離するだけでは十分ではありません。ファイルを分離する決定は、そのファイルのI/O率がデータベースのパフォーマンスに影響

を与える場合にのみ行ってください。

ファイルを分割する場合

オペレーティング・システムのストライプ化または手動I/O分散を使用するかどうかに関係なく、I/OシステムまたはI/Oレイアウトが要求されたI/O率をサポートできない場合は、I/O率の高いファイルをそれ以外のファイルから分離する必要があります。このようなファイルは計画段階かシステムの本稼働後に確認できます。

ファイルを分離する決定は、I/O率、リカバリ能力の問題、管理性の問題によってのみ影響を受けます。(たとえば、LVMがストライプ幅の動的な再構成をサポートしない場合、同一構成の新しいストライプを作成するために一度にn個のディスクを追加できるように、さらに小さいストライプ幅を作成する必要がある場合があります。)

ファイルを分離する前に、ボトルネックが実際にI/Oの問題であるかどうかを検証します。 ボトルネックの調査から生成されたデータでは、最高のI/O率を持つファイルを識別します。

後続の項では、次のファイル・タイプを分離する方法について説明します。

- 表、索引およびTEMP表領域
- REDOログ・ファイル
- アーカイブREDOログ

表、索引およびTEMP表領域

表および索引を含む表領域に属するデータファイルがI/Oの多いファイルである場合は、これらのファイルのI/OをSQLのチューニングまたはアプリケーション・コードのいずれかで削減できるかどうかを識別します。

I/Oの多いファイルがTEMP表領域に属するデータファイルである場合は、このアクティビティを回避するためにディスク・ソートを実行するSQL文をチューニングするか、あるいはソートをチューニングするかを調べます。

不要なI/Oを回避するようにアプリケーションをチューニングした後、I/Oレイアウトが引き続き必要なスループットを維持できない場合は、I/Oの多いファイルの分離を考慮してください。

REDOログ・ファイル

I/Oの多いファイルがREDOログ・ファイルである場合は、REDOログ・ファイルをその他のファイルから分離することを考慮してください。可能な構成には、次のことが含まれています。

- すべてのREDOログを、他のファイルのない1つのディスクに置きます。可用性も考慮します。すなわち、リカバリ可能性の目的で、同じグループのメンバーは異なる物理ディスクおよびコントローラ上にあることが必要です。
- 他のファイルを格納しない個別のディスク上に各REDOログ・グループを置きます。
- オペレーティング・システムのストライプ化ツールを使用して、複数のディスクにまたがってREDOログ・ファイルをストライプ化します。(この場合、手動ストライプ化は不可能です。)
- REDOログにRAID 5を使用しないでください。

REDOログ・ファイルは、ログ・ライター(LGWR)プロセスで順次書き込まれます。同じディスクに対する同時実行アクティビティが存在しない場合、この操作はさらに高速にできます。REDOログ・ファイルに別々の専用ディスクを割り当てると、さらにチューニン

グしなくても通常はLGWRが円滑に実行されます。システムが非同期I/Oをサポートせず、この機能が現在構成されていない場合、この機能を使用することが有効かどうかを確認します。LGWRに関連するパフォーマンス上のボトルネックはめったにありません。

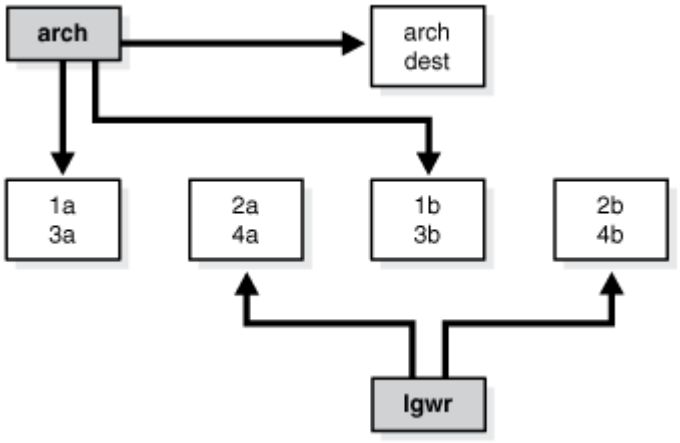
アーカイブREDOログ

アーカイバが遅い場合は、アーカイバの読取りとLFWRの書込みを必ず分けるようにして、アーカイバ・プロセスとLGWRとのI/O競合を防止するのが賢明です。これは、ログを交互のドライブに置くことでアーカイブされます。

たとえば、システムに4つのREDOログ・グループが存在し、各グループが2つのメンバーを持っているとします。個別ディスク・アクセスを作成するには、8つのログ・ファイルにそれぞれ1a、1b、2a、2b、3a、3b、4a、4bのラベルを付けてください。この場合、最低でも4つのディスクと、アーカイブ・ファイル用に1つのディスクが必要です。

次の図は、競合を最小限にするために、ディスク間でREDOメンバーを分散する方法を示しています。

図17-1 ディスク間でのREDOメンバーの分散



この例では、LGWRはログ・グループ1(メンバー1aと1b)から切り替えられ、ログ・グループ2(2aと2b)に書込みを行います。同時に、アーカイバ・プロセスはグループ1から読取りをして、アーカイブ先に書込みを行います。REDOログ・ファイルがどのようにして競合から分離されているかに注意してください。

ノート:

REDO ログ・ファイルをミラー化する、すなわち各 REDO ログ・ファイルの複数のコピーを別々のディスク上に保持することで、LGWR が大幅に遅くはなりません。LGWR は、各ディスクに対して並列して書込みを行い、並列書込みの各部が完了するまで待機します。したがって、並列書込みが、最も長い単一のディスク書込みよりも長くなることはありません。

REDOログはシリアルに書き込まれるので、REDOログ・アクティビティ専用のドライブでは一般にヘッドの移動はわずかです。このため、ログ書込みのスピードが大幅に向上します。

3つの構成サンプル

この項では、I/Oシステムを構成する高水準の例を3つ示します。これらの例には、ディスク・トポロジやストライプ深度などを定義する計算例が含まれています。

- すべてのディスクにまたがったすべての内容のストライプ化
- 異なるディスクへのアーカイブ・ログの移動
- 個別のディスクへのREDOログの移動

すべてのディスクにまたがったすべての内容のストライプ化

I/O構成の最も簡単なアプローチは、すべての使用可能ディスクにまたがってストライプ化された、1つの大きなボリュームを作成することです。リカバリ能力を考慮して、ボリュームがミラー化されます(RAID 1)。ディスク当たりのストライプ化の単位は、頻繁なI/O操作のための最大I/Oサイズより大きくする必要があります。そうすると、ほとんどの場合は十分なパフォーマンスが得られます。

異なるディスクへのアーカイブ・ログの移動

アーカイブREDOログ・ファイルが他のファイルと同じディスク・セット上でストライプ化されている場合、REDOログがアーカイブされる際、これらのディスク上のI/Oリクエストが影響を受ける可能性があります。個別のディスクにアーカイブREDOログ・ファイルを移動する場合の利点は次のとおりです。

- 非常に高速なアーカイブを実行できます(順次I/Oを使用)。
- アーカイブ先のディスク上でレスポンス時間が低下しても、その影響を受けるものは他にありません。

アーカイブ・ログ用のディスク数は、アーカイブ・ログ生成頻度およびアーカイブ記憶域の必要量により決定されます。

個別のディスクへのREDOログの移動

更新の多いOLTPシステムでは、REDOログは書込み中心です。他のディスクおよびアーカイブREDOログ・ファイルから分離されたディスクにREDOログ・ファイルを移動すると、次の利点があります。

- REDOログの書込みは、可能なかぎり高速で行われます。したがって、トランザクション処理のパフォーマンスは最高になります。
- REDOログの書込みが他のI/Oで損われることはありません。

REDOログ用のディスク数は、ほとんどの場合に、現行技術のディスク・サイズと比較して一般に小さいREDOログ・サイズで決定されます。一般に、2つのディスクを持つ構成(フォルト・トレランスのために4つのディスクにミラー化など)で十分です。特に、2つのディスク上でREDOログ・ファイルを交互に使用すると、1つのファイルにREDOログ情報を書き込む場合、アーカイブの完了したREDOログの読取りが妨げられません。

Oracle Managed Files

ファイル・システムにすべてのOracle Databaseデータを取り込むことができる場合、データベース管理はOracle Managed Filesを使用して簡単に行えます。表領域、一時ファイル、オンライン・ログおよび制御ファイルについては、Oracle Databaseは内部的に標準ファイル・システム・インタフェースを使用して、必要に応じてファイルを作成および削除します。管理者は、特定のタイプのファイルに使用するファイル・システム・ディレクトリのみを指定します。データファイルについてはデフォルトの場所を1つ、制御ファイルおよびオンラインREDOログ・ファイルについては複数の場所を最大5つ指定できます。

Oracle Databaseでは、一意のファイルが作成された後、そのファイルが不要になると削除されます。このため、管理者が誤ったファイルを指定したことにより発生する破損、および廃止されたファイルで消費される無駄なディスク領域が減り、テスト・データ

ベースおよび開発データベースの作成が簡素化されます。また、オペレーティング・システム固有のファイル名をSQLスクリプトに設定する必要がないため、ポータブルなサード・パーティのツールの開発を容易にします。

新しいファイルはOracle Managed Filesとして作成できますが、古いファイルは古い方法で管理されます。このため、データベースにはOracle Managed Filesとユーザー管理ファイルが混在する場合があります。

ノート:

Oracle Managed Files は、RAW デバイスでは使用できません。

Oracle Managed Filesをチューニングする場合はいくつかの点に注意する必要があります。

- Oracle Managed Filesではファイル・システムを使用する必要があるため、DBAはデータのレイアウト方法は管理しません。したがって、ファイル・システムを正しく構成することが重要です。
- Oracle Managed Filesのファイル・システムは、ストライプ化をサポートするLVM上に構築します。ロード・バランシングおよびスループットの向上のために、ファイル・システム内のディスクをストライプ化します。
- Oracle Managed Filesは、動的に拡張可能な論理ボリュームをサポートするLVM上で使用するときに最高の機能を発揮します。それ以外の場合は、論理ボリュームをできるだけ大きく構成します。
- Oracle Managed Filesは、ファイル・システムに大きな拡張可能なファイルがある場合、最高の機能を発揮します。

関連項目:

Oracle Managed Filesの使用方法の詳細は、Oracle Database管理者ガイドを参照してください

データ・ブロック・サイズの選択

8KBのブロック・サイズはほとんどのシステムにとって最適です。ただし、OLTPシステムではより小さなブロック・サイズを、DSSシステムではより大きなブロック・サイズを使用することがあります。この項では、最適なパフォーマンスを得るためにデータベース・ブロック・サイズを選択するときの考慮事項を説明します。内容は次のとおりです。

- 読取り
- 書込み
- ブロック・サイズの長所と短所

ノート:

管理性の問題があるため、単一データベース・インスタンスでの複数のブロック・サイズの使用はお薦めしません。

読取り

データのサイズとは関係なく、目標は必要なデータを取り出すために必要な読取り回数を最小にすることです。

- 行が小さく、アクセスがきわめてランダムな場合は、小さなブロック・サイズを選択します。
- 行が小さく、アクセスがきわめて順次である場合は、大きなブロック・サイズを選択します。
- 行が小さく、アクセスがランダムかつ順次である場合は、大きなブロック・サイズを選択するのが有効です。
- 行が大きい(たとえば、ラージ・オブジェクト(LOB)データが含まれている)場合は、大きなブロック・サイズを選択します。

書込み

同時実行性の高いOLTPシステムで、大きなブロック・サイズを使用する場合は、INITRANS、MAXTRANSおよびFREELISTSの適切な値について検討します。これらのパラメータは、ブロック内で許可されている更新の同時実行性の程度に影響を与えます。ただし、自動セグメント領域管理を使用する場合は、FREELISTSの値を指定する必要はありません。

選択する必要があるブロック・サイズが不明な場合、多数のトランザクションを処理するシステムでは、8KBのデータベース・ブロック・サイズを試行してください。これは優れた妥協案であり、通常は有効です。8KBを超えるサイズが必要なのはLOBデータを処理するシステムのみです。

関連項目:

使用しているプラットフォームの最小および最大のブロック・サイズは、オペレーティング・システム固有のOracle Databaseインストール・マニュアルを参照してください。

ブロック・サイズの長所と短所

表17-3に、様々なブロック・サイズの長所と短所を示します。

表17-3 ブロック・サイズの長所と短所

ブロック・サイズ	長所	短所
小さい	行数が少なく大量のランダム・アクセスに適しています。 ブロック競合が低減されます。	メタデータ(すなわち、ブロック・ヘッダー)による比較的大きな領域のオーバーヘッドがあります。 行が大きい場合にはお薦めできません。1 行がブロックに収まらない場合、1 ブロック当たりの格納行数がわずかになったり、さらには行の連鎖が発生する可能性があります。
大きい	オーバーヘッドが少ないため、データを格納する空間が多くなります。 1 回の I/O でバッファ・キャッシュに複数の行を読み取れます(行サイズおよびブロック・サイズにより異なります)。 順次アクセスまたは非常に大きな行(LOB	ブロック・サイズが大きい場合に少数の行にランダム・アクセスすると、バッファ・キャッシュ内の領域が消費されます。たとえば、8KB のブロック・サイズと 50 バイトの行サイズでは、ランダム・アクセスを行うときにバッファ・キャッシュ内の 7,950 バイトが消費されます。 OLTP 環境で使用される索引ブロックには適しません。これは、索引リーフ・ブロック上のブロック競合が増えるためです。

ブロック・サイズ	長所	短所
	データなど)に適しています。	

データベース内部のI/O測定

Oracle DatabaseのI/O測定機能では、ストレージ・サブシステムのパフォーマンスを評価し、I/Oパフォーマンス問題がデータベースとストレージ・サブシステムのどちらを原因とするかを判別できます。I/Oを連続的に発行する外部のI/O測定ツールとは異なり、Oracle DatabaseのI/O測定機能では、Oracleデータファイルを使用してI/Oをランダムに発行し、ストレージ・メディアにアクセスするため、よりデータベースの実際のパフォーマンスに近い結果を得ることができます。

この項では、Oracle DatabaseのI/O測定機能の使用方法について説明します。この項には、次の項目があります。

- I/O測定の前提条件
- I/O測定の実行

Oracle Databaseは、OrionによるI/O測定も提供しています。Orionは、Oracleのインストールやデータベースの作成を行わなくてもOracle databaseのパフォーマンスを予測できるツールです。他のI/O測定ツールとは異なり、Oracle Orionは、Oracleと同じI/Oソフトウェア・スタックを使用してOracleデータベースのI/Oワークロードをシミュレートするように特別に設計されたものです。またOrionは、Oracle Automatic Storage Managementによって実行されるストライプ化の効果もシミュレートすることができます。詳細は、「Oracle Orion測定ツールによるI/O測定」を参照してください。

I/O測定の前提条件

I/O測定を実行する前に、次の前提条件が満たされていることを確認してください。

- SYSDBA権限をユーザーに付与する必要があります。
- timed_statisticsがTRUEに設定されている必要があります。
- 非同期I/Oが有効化されている必要があります。

  ファイル・システムを使用している場合、初期化パラメータFILESYSTEMIO_OPTIONSをSETALLに設定することで非同期I/Oを有効化できます。
- 次の問合せを実行し、データファイルに対して非同期I/Oが有効化されていることを確認してください。

```
COL NAME FORMAT A50
SELECT NAME,ASYNCH_IO FROM V$DATAFILE F,V$IOSTAT_FILE I
WHERE  F.FILE#=I.FILE_NO
AND    FILETYPE_NAME='Data File';
```

また、1つのデータベース・インスタンスに一度に実行できる測定は、1回のみです。

I/O測定の実行

Oracle DatabaseのI/O測定機能には、DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャを使用してアクセスします。このプロシージャは、データベース・ファイルにI/O集中型の読取り専用ワークロード(1MBのランダムI/Oで構成されるワークロード)を発行し、ストレージ・サブシステムが耐えられる最大IOPS(1秒当たりのI/Oリクエスト)とMBPS(1秒当たりのI/OのMB)を判定します。

I/O測定は、次の2つのステップで行われます。

- I/O測定の最初のステップでは、DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャにより、すべてのデータベース・インスタンスのすべてのデータファイルに対してデータベース・ブロック・サイズのランダムな読取り(デフォルトは8KB)を発行します。このステップでは、出力パラメータmax_iopsに、データベースが維持できる最大IOPSが示されます。max_iopsの値は、OLTPデータベースに重要なメトリックです。出力パラメータactual_latencyには、このワークロードの平均待機時間が示されます。特定の目標待機時間が必要な場合は、入力パラメータmax_latencyを使用して目標待機時間を指定できます(データベース・ブロック・サイズのIOリクエストの最大許容待機時間をミリ秒単位で指定)。
- 2番目の測定のステップでは、DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャにより、すべてのデータベース・インスタンスのすべてのデータファイルに対して1MBのランダムな読取りを発行します。2番目のステップでは、出力パラメータmax_mbpsに、データベースが維持できるI/Oの最大MBPSが示されます。このステップは、データ・ウェアハウスに重要なメトリックを提供します。

ユーザーが入力パラメータnum_physical_disksを指定すると(データベース・ストレージ・システム内の物理ディスクの概数を指定)、より効率的に測定を実行できます。

I/Oワークロードの実行によりオーバーヘッドが発生するため、I/O測定はデータベースがアイドル状態のとき(つまりピーク時以外の時間)にのみ実行し、通常のデータベース・ワークロードに対するI/Oワークロードの影響を最小化する必要があります。

I/O測定を実行し、Oracle Databaseで使用されるストレージ・サブシステムのI/O性能を評価するには、次のようにDBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャを使用します。

```
SET SERVEROUTPUT ON
DECLARE
  lat  NUMBER;
  iops INTEGER;
  mbps INTEGER;
BEGIN
-- DBMS_RESOURCE_MANAGER.CALIBRATE_IO (<DISKS>, <MAX_LATENCY>, iops, mbps, lat);
   DBMS_RESOURCE_MANAGER.CALIBRATE_IO (2, 10, iops, mbps, lat);

end;
/
```

DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャを実行する場合、次のことを考慮してください。

- 同じストレージ・サブシステムを使用するデータベースで一度に実行できる測定は1回のみです。同じストレージ・サブシステムを使用する別のデータベースで同時に測定を実行すると、測定は失敗します。
- インスタンスでのI/Oを最小化するため、データベースを停止します。
- Oracle Real Application Clusters(Oracle RAC)構成の場合、ノード間でストレージ・サブシステムを測定できるようすべてのインスタンスがオープン状態にあることを確認します。
- Oracle Real Application Clusters(Oracle RAC)データベースでは、ワークロードはすべてのインスタンスから同時に生成されます。
- 入力パラメータnum_physical_disksはオプションです。num_physical_disksパラメータにデータベースのストレージ・システム内にある物理ディスクの概数を設定すると、より高速かつ正確に測定できます。
- 場合によっては、非同期I/Oがデータファイルで許可されている一方で、非同期I/Oを発行するI/Oサブシステムが最大限度に達し、I/O測定を継続できなくなります。このような場合、ポート固有のドキュメントを参照して、システムの非

同期I/Oの最大限度をチェックしてください。

I/O測定プロセス中は、いつでもV$IO_CALIBRATION_STATUSビューで測定ステータスを問い合せることができます。I/O測定が正常に完了したら、DBA_RSRC_IO_CALIBRATE表でその結果を参照できます。

関連項目:

- DBMS_RESOURCE_MANAGER.CALIBRATE_IOプロシージャの実行方法の詳細は、『Oracle Database PL/SQLパッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- V$IO_CALIBRATION_STATUSビューおよびDBA_RSRC_IO_CALIBRATE表の詳細は、Oracle Databaseリファレンスを参照してください

Oracle Orion測定ツールによるI/O測定

この項では、Oracle Orion測定ツールについて説明します。この項には次の項目があります。

- Oracle Orion測定ツールの概要
- Orionの開始
- Orionの入力ファイル
- Orionのパラメータ
- Orionの出力ファイル
- Orionのトラブルシューティング

Oracle Orion測定ツールの概要

Oracle Orionは、Oracleのインストールやデータベースの作成を行わなくてもOracleデータベースのパフォーマンスを予測できるツールです。他のI/O測定ツールとは異なり、Oracle Orionは、Oracleと同じI/Oソフトウェア・スタックを使用してOracleデータベースのI/Oワークロードをシミュレートするように特別に設計されたものです。またOrionは、Oracle Automatic Storage Managementによって実行されるストライプ化の効果もシミュレートすることができます。

表17-4に、Orionがサポートする I/Oワークロードのタイプを示します。

Orionは、表17-4に示すワークロードの各タイプについて、様々なI/O負荷を使用してテストを実行し、MBPS、IOPS、およびI/O待機時間などのパフォーマンス・メトリックを測定できます。負荷は、未処理の非同期I/Oの数で表されます。内部的には、Orionソフトウェアはこの各負荷レベルに対して、できるだけ高速にI/Oリクエストを発行し続けてI/O負荷をそのレベルに保ちます。ランダム・ワークロードでは、大規模I/Oと小規模I/Oのどちらを使用する場合でも、負荷レベルは未処理I/Oの数です。大規模な順次ワークロードでは、負荷レベルは順次ストリームの数とストリーム当たりの未処理I/Oの数の組合せです。指定されたワークロードを一定範囲の負荷レベルでテストすると、負荷がパフォーマンスに与える影響を理解するのに役立ちます。

Orionを使用する際には、次の点に注意してください。

- Orionはストレージがアイドル状態(またはほぼアイドル状態)のときに実行してください。Orionは、生成したI/O負荷に基づいてストレージのパフォーマンスを測定するため、Orion以外のI/Oワークロードが同時に実行されていると、パ

フォーマンスを適切に評価できません。

- ストレージにすでにデータベースが作成されている場合は、別の方法としてPL/SQLルーチン dbms_resource_manager.calibrate_io()を使用してストレージを測定できます。

表17-4 OrionがサポートするI/Oワークロード

ワークロード	説明
小規模なランダム I/O	OLTP アプリケーションは通常、データベースのブロック・サイズと同じサイズ(通常は 8KB)のランダム読取りおよびランダム書込みを生成します。このようなアプリケーションでは通常、1 秒当たりの I/O(IOPS)で表すスループットと 1 リクエスト当たりの平均待機時間(I/O 応答時間)が重要になります。これらのパラメータは、アプリケーション・レイヤーでトランザクション速度とトランザクション応答時間に変換されます。 Orion は、指定された読取りの書込みに対する割合、指定された I/O サイズ、そして指定された未処理 I/O の数を使用して、ランダム I/O ワークロードをシミュレートします。この Orion のワークロード・シミュレーションでは、I/O はすべてのディスクに配分されます。
大規模な順次 I/O	データ・ウェアハウス・アプリケーション、データのロード、バックアップ、およびリストアでは、複数の 1MB の未処理 I/O からなる順次読取りおよび順次書込みのストリームが生成されます。このようなアプリケーションは、表全体やデータベース全体など、大量のデータを処理し、通常、1 秒当たりの MB(MBPS)で表すデータ全体のスループットが重要になります。 Orion は、指定された数の順次読取りまたは順次書込みストリームを、指定された I/O サイズと指定された未処理 I/O 数でシミュレートできます。Orion は、順次ストリームのテスト時に、オプションで Oracle Automatic Storage Management のストライプ化をシミュレートできます。
大規模なランダム I/O	順次ストリームは通常、他のデータベース通信と同時にディスクにアクセスします。ストライプ化により、順次ストリームは多数のディスクに分散されます。結果として、ディスク・レベルでは、複数の順次ストリームが 1MB のランダム I/O となります。
混合ワークロード	Orion は、2 つの同時ワークロード(小規模なランダム I/O と、大規模な順次 I/O か大規模なランダム I/O のどちらか)をシミュレートできます。このワークロード・タイプにより、たとえば 8KB のランダム読取りおよびランダム書込みの OLTP ワークロードと、4 つの 1 MB I/O の順次読取りストリームのバックアップ・ワークロードを一緒にシミュレートできます。

Orionの各データ・ポイントは、一定期間持続する小規模I/O負荷と大規模I/O負荷の特定の混合に対するテストです。Orionテストは複数のデータ・ポイント・テストからなります。これらのデータ・ポイント・テストは、2次元マトリックスとして表すことができます。マトリックスの各列は、同じ小規模I/O負荷によるデータ・ポイント・テストを表しますが、大規模I/O負荷は異なります。各行は、同じ大規模I/O負荷によるデータ・ポイント・テストを表しますが、小規模I/O負荷は異なります。Orionテストは、1つのポイント、1つの行、1つの列に対して、またはマトリックス全体に対して行うことができます。

Orionテストのターゲット

Orionを使用して、非同期I/Oをサポートするディスクベースのキャラクタ・デバイスをテストできます。Orionは次のターゲット・タイプについて検証済です。

- DAS(直接接続型)ストレージ: Orionを使用して、1つ以上のローカル・ディスク、ボリューム、またはローカル・ホスト上のファイルのパフォーマンスをテストできます。
- SAN(ストレージ・エリア・ネットワーク): Orionは、そのすべてまたは一部のSANストレージがキャラクタ・デバイスとしてマップされている任意のホストで実行できます。デバイスは、ストレージ・アレイによってエクスポートされたストライプ・ボリュームまたは非ストライプ・ボリューム、個々のディスク、または1つ以上のアレイ全体に相当します。
- NAS(ネットワーク接続ストレージ): Orionを使用して、NASストレージ上のデータファイルのパフォーマンスをテストできます。一般的には、NASストレージでのパフォーマンス結果は、データファイルの作成および更新に使用されたI/Oパターンによって異なります。そのため、Orionを実行する前にデータファイルを適切に初期化する必要があります。

Oracle管理者のためのOrion

Oracle管理者は、Orionを使用して、予測されるワークロードに基づいて、様々なストレージ・アレイを評価および比較することができます。またOracle管理者は、Orionを使用して、予測されるピーク・ワークロードに対して、ネットワーク接続、ストレージ・アレイ、ストレージ・アレイ・コントローラ、およびディスクの最適な数を判断できます。

Orionの開始

Orionの使用を開始するには、次のようにします。

1. Orion -testnameパラメータにより、使用するテスト名を選択します。このパラメータにより、Orionの実行に対して固有の識別子が指定されます。たとえば、テスト名「mytest」を使用します。詳細は、「Orionのパラメータ」を参照してください。
2. テスト名に基づいて、Orion入力ファイルを作成します。たとえば、mytest.lunという名前のファイルを作成します。入力ファイルに、テストするRAWボリュームまたはファイルをリストします。1行ごとに1つのボリューム名を追加します。.lunファイルにはコメントなど他のものは入力しないでください。

   たとえば、Orion入力ファイルには次を含めることができます。

   ```
   /dev/raw/raw1
   /dev/raw/raw2
   /dev/raw/raw3
   /dev/raw/raw4
   /dev/raw/raw5
   /dev/raw/raw6
   /dev/raw/raw7
   /dev/raw/raw8
   ```

   詳細は、「Orionの入力ファイル」を参照してください。
3. 入力ファイルで指定されたすべてのボリューム、たとえばmytest.lunが、コマンドddまたは他の同等のファイル表示ユーティリティを使用してアクセスできることを確認します。たとえば、一般的な健全性チェックでは、Linuxシステムで次のことを試してください。

   ```
   $ dd if=/dev/raw/raw1 of=/dev/null bs=32k count=1024
   ```

プラットフォームによって、使用するファイル表示ユーティリティとそのインタフェースは異なる場合があります。

4. プラットフォームに非同期I/Oに必要なライブラリがインストールされていることを確認します。Orionテストは非同期I/Oに完全に依存しています。LinuxおよびSolarisでは、ライブラリlibaioが標準libディレクトリにあるか、またはシェル環境のライブラリパス変数(通常はシェルに応じてLD_LIBRARY_PATHまたはLIBPATH)を使用してこのライブラリにアクセスできることが必要です。Windowsには組込みの非同期I/Oライブラリがあるため、この問題は適用されません。
5. Orionでの最初のテストとして、-runにoltpまたはdssのどちらかのオプションを付けて使用します。データベースが主としてOLTPの場合は、-run oltpを使用します。データベースが主としてデータ・ウェアハウスまたは分析用の場合は、-run dssを使用します。

   たとえば、デフォルトの入力ファイル名、orion.lunを使用してOLTPのようなワークロードを実行するには、次のコマンドを使用します。

   ```
   $ ./orion -run oltp
   ```

   Orionは、テストされるディスク・スピンドルの数(または-num_disksパラメータで指定された数)を考慮してI/O負荷レベルを生成します。スピンドルの数が入力ファイルで指定されたボリュームの数に関係するかどうかは、これらのボリュームがどのようにマップされているかによって異なることを覚えておいてください。

6. 「Orionの出力ファイル」の項に、Orionの出力ファイルを示す結果例を記載しています。手始めとしてサンプル・ファイルmytest_summary.txtを使用すると、入力パラメータの検証や出力の分析に役立ちます。サンプル・ファイルmytest_*.csvには、複数のI/Oパフォーマンス測定の値がカンマ区切りで含まれています。

Orionの入力ファイル

Orion -testname <testname>パラメータを指定すると、Orionの入力および出力ファイル名にテスト名の接頭辞が設定されます。-testnameオプションのデフォルト値は「orion」です。

Orion入力ファイル、<testname>.lunには、改行で区切ったLUNのリストを含める必要があります。

Orionのパラメータ

I/Oワークロードのタイプや他のOrionオプションを指定するには、Orionコマンド・パラメータを使用します。

Orionに必要なパラメータ

Orionコマンドには-runパラメータが必要です。表17-5で-runパラメータについて説明します。

表17-5 必要なOrionパラメータ

オプション	説明	デフォルト
-run level	テスト実行レベルを level に指定します。このオプションは実行レベルを定義するもので、複雑なコマンドに高度なレベルを指定することができます。-run advanced を設定しないで、-cache_size と-verbose 以外の他のパラメータを設定すると、結果はエラーになります。 advanced を除くすべての-run level 設定は、あらかじめ指定されているパラメータ・セットを使用します。	normal

level には次のいずれかを指定します。

- oltp

  小規模な(8K)ランダム I/O で負荷を増加させて最大 IOPS を特定するためのテストです。

  このパラメータは、次の Orion 呼び出しに相当します。

  ```
  %> ./orion -run advanced ¥
  -num_large 0 -size_small 8 -type rand ¥ -simulate concat -write 0
  -duration 60 ¥ -matrix row
  ```

- dss

  大規模な(1M)ランダム I/O で負荷を増加させて最大スループットを特定するためのテストです。

  このパラメータは、次の Orion 呼び出しに相当します。

  ```
  %> ./orion -run advanced ¥ -num_small 0 -size_large 1024 -type
  rand ¥ -simulate concat -write 0 -duration 60 ¥ -matrix column
  ```

- simple

  一定範囲の負荷レベルでの小規模なランダム I/O と大規模なランダム I/O のワークロードを生成します。このオプションでは、小規模 I/O と大規模 I/O が個別にテストされます。この実行レベルで指定できるオプション・パラメータは、-cache_size と-verbose のみです。

  このパラメータは、次の Orion 呼び出しに相当します。

  ```
  %> ./orion -run advanced ¥
  -size_small 8 -size_large 1024 -type rand ¥ -simulate concat -
  write 0 -duration 60 ¥ -matrix basic
  ```

- normal

  simple と同じですが、一定範囲の負荷レベルでの小規模なランダム I/O と大規模なランダム I/O の組合せも生成されます。この実行レベルで指定できるオプション・パラメータは、-cache_size と-verbose のみです。

  このパラメータは、次の Orion 呼び出しに相当します。

  ```
  %> ./orion -run advanced ¥
  -size_small 8 -size_large 1024 -type rand ¥ -simulate concat -
  write 0 -duration 60 ¥ -matrix detailed
  ```

- advanced

オプション	説明	デフォルト
	オプション・パラメータで指定したワークロードをテストします。この実行レベルではどのオプション・パラメータでも指定できます。	

Orionのオプション・パラメータ

表17-6 Orionのオプション・パラメータ

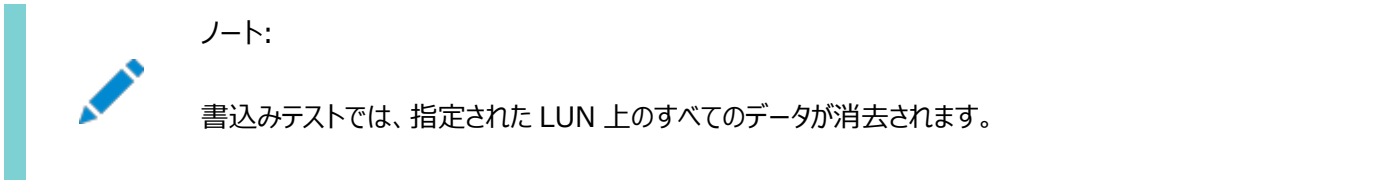
オプション	説明	デフォルト
-cache_size num	ストレージ・アレイの読取りまたは書込みキャッシュのサイズ(MB)。大規模な順次 I/O ワークロードでは、Orion は各データ・ポイントの前に大規模なランダム I/O を行うことでキャッシュをウォーミングします。Orion はキャッシュ・サイズを使用して、このキャッシュ・ウォーミング操作の期間を決定します。0 に設定すると、キャッシュ・ウォーミングは行われません。 このオプションを 0 に設定しない場合は、Orion は大規模な順次データ・ポイントごとに、その前に複数の未測定のランダム I/O を発行します。これらの I/O は、ストレージ・アレイのキャッシュ(存在する場合)をランダム・データで満たし、あるデータ・ポイントからの I/O が次のデータ・ポイントのキャッシュ・ヒットにならないようにします。読取りテストではジャンク読取りを行い、書込みテストではジャンク書込みを行います。このキャッシュ・ウォーミングは、指定されると、I/O の num MB の読取りまたは書込みが終了するまで実行されます。	デフォルト値: 指定しないと、ウォーミングはデフォルト時間(2 分間)で実行されます。つまり、各データ・ポイントの前に、未測定のランダム I/O を 2 分間発行します。
-duration num_seconds	各データ・ポイントをテストする時間を値 num_seconds に秒単位で設定します。	デフォルト値: 60
-help	Orion のヘルプ情報を出力します。help とともに設定されたその他のオプションはすべて無視されます。	
-matrix type	一定範囲の負荷でテストする混合ワークロードのタイプ。Orion テストは複数のデータ・ポイント・テストからなります。データ・ポイント・テストは、2 次元マトリックスとして表すことができます。 マトリックスの各列は、同じ小規模 I/O 負荷によるデータ・ポイント・テストを表しますが、大規模 I/O 負荷は異なります。各行は、同じ大規模 I/O 負荷によるデータ・ポイント・テストを表しますが、小規模 I/O 負荷は異なります。Orion テストは、次のマトリックスの type に応じて、1 つのポイント、1 つの列、1 つの行に対して、またはマトリックス全体に対して行うことができます。 ● basic: 混合ワークロードなし。小規模なランダム・ワークロードと、大規模なランダムまたは順次ワークロードは、個別にテスト	デフォルト値: basic

オプション	説明	デフォルト
	されます。小規模 I/O のみをテストし、次に大規模 I/O のみをテストします。 ● detailed: 小規模なランダム・ワークロードと、大規模なランダムまたは順次ワークロードは、組み合せてテストされます。マトリックス全体をテストします。 ● point: S 未処理の小規模なランダム I/O および L 未処理の大規模なランダム I/O または順次ストリームによるデータポイント。S は、-num_small パラメータで設定されます。L は、-num_large パラメータで設定されます。-num_small により小規模 I/O を、-num_large により大規模 I/O をテストします。 ● col: 大規模なランダムまたは順次ワークロードのみ。-num_small 小規模 I/O を使用して、大規模 I/O の負荷を変化させてテストします。 ● row: 小規模なランダム・ワークロードのみ。-num_large 大規模 I/O を使用して、小規模 I/O の負荷を変化させてテストします。 ● max: detailed と同じですが、パラメータ-num_small と-num_large で指定された最大負荷でのワークロードのみをテストします。-num_small と-num_large の制限値まで負荷を変化させてテストします。	
-num_disks value	テストに使用する物理的なディスクの数を指定します。これを使用して負荷の範囲が生成されます。ディスク(物理的なスピンドル)の数を指定します。この数の value を使用して、Orion がテストする必要のある負荷の範囲を判断します。このパラメータを増やすと、Orion はより高い I/O 負荷を使用します。	デフォルト値: <testname>.lun 内の LUN の数
-num_large value	大規模 I/O 負荷を制御します。 ノート: このオプションは、-matrix が row、point、または max に指定されている場合のみ適用されます。 -type オプションが rand に設定されている場合は、パラメータ引数の value で未処理の大規模 I/O の数を指定します。 -type オプションが seq に設定されている場合は、パラメータ引数の value で順次 I/O ストリームの数を指定します。	デフォルト値: デフォルトなし

オプション	説明	デフォルト
-num_small	小規模なランダム I/O ワークロードに対する未処理 I/O の最大数を指定します。 ノート: このオプションは、-matrix が col、point、または max に指定されている場合のみ適用されます。	デフォルト値: デフォルトなし
-num_streamIO num	1 ストリーム当たりの同時 I/O の数を num に指定します。 ノート: このパラメータは、-type が seq の場合のみ使用されます。	デフォルト値: 4
-simulate type	大規模な順次 I/O ワークロードのシミュレーションのためのデータ・レイアウト。Orion は、これらのいずれかの方法で指定された LUN の組合せによって形成された仮想 LUN 上でテストします。type には次のいずれかを指定します。 ● concat: 仮想ボリュームは、指定された LUN をシリアルにつなぐことによりシミュレートされます。この仮想ボリュームに対する順次テストは、あるポイントから各 LUN の終わりまで行われ、続けて次の LUN の先頭から最後までというように続きます。 ● raid0: 仮想ボリュームは、指定された LUN をまたいでストライプ化してシミュレートされます。各順次ストリームは、raid0 ストライプ化を使用してすべての LUN をまたいで I/O を発行します。ストライプ深度はデフォルトで 1M で、Oracle Automatic Storage Management のストライプ深度に一致しています。これは-stripe パラメータを使用して変更できます。 I/O のオフセットは、次のようにして決定されます。 小規模なランダム・ワークロードおよび大規模なランダム・ワークロードの場合: ● LUN は単一の仮想 LUN(VLUN)に連結され、VLUN 内でランダム・オフセットが選択されます。 大規模な順次ワークロードの場合: ● ストライプ化あり(-simulate raid0)。LUN を使用して単一のストライプ化された VLUN が作成されます。小規模なランダム・ワークロードが同時にない場合は、順次ストリームはストライプ化された VLUN 内の固定のオフセットで開始されます。n ストリームの場合、n が 1 でなければ、ストリーム i はオフセット	デフォルト値: concat

オプション	説明	デフォルト
	VLUNsize * (i + 1) / (n + 1)で開始されます。小規模なランダム・ワークロードが同時にある場合は、ストリームはストライプ化された VLUN 内のランダム・オフセットで開始されます。 ● ストライプ化なし(-simulate CONCAT)。LUN は単一の VLUN に連結されます。ストリームは単一の VLUN 内のランダム・オフセットで開始されます。 このパラメータは通常、-type が seq の場合のみ使用されます。	
-size_large num	大規模なランダムまたは順次 I/O ワークロードの I/O のサイズ(KB)を num に指定します。	デフォルト値: 1024
-size_small num	小規模なランダム I/O ワークロードの I/O のサイズ(KB)を num に指定します。	デフォルト値: 8
-storax type	I/O ワークロードのテストに使用する API。 ● skgfr: オペレーティング・システム I/O レイヤーを使用。 ● oss: Exadata マシン内のセル・サーバーによる I/O に対して OSS API を使用。 ● asmlib: ASMLIB ディスク・デバイス・ベースのストレージ API を I/O に対して使用。 ● odmlib: Direct NFS ストレージ・ベースの API を I/O に対して使用。	デフォルト値: skgfr
-testname tname	tname にテスト実行の識別子を指定します。指定する場合は、LUN ディスクを含む入力ファイルまたはファイル名を<tname>.lun という名前にする必要があります。 出力ファイルは、接頭辞<tname>_を付けた名前になります。	デフォルト値: orion
-type [rand | seq]	大規模 I/O ワークロードのタイプ。 ● rand: ランダムに分配された大規模 I/O。 ● seq: 順次ストリームの大規模 I/O。	デフォルト値: rand

オプション	説明	デフォルト
-verbose	ステータスとトレース情報を標準出力に出力します。	デフォルト値：オプションの設定なし
-write num_write	書込み I/O の割合を num_write に指定します。残りは読取り I/O になります。 このパラメータは大規模 I/O ワークロードと小規模 I/O ワークロードの両方に適用されます。大規模な順次 I/O の場合、各ストリームは読取り専用か書込み専用のどちらかです。そのため、パラメータは書込み専用のストリームの割合を指定します。ディスクに書き込まれるデータは不要データで、ディスク上の既存データとは無関係です。 注意：書込みテストでは、指定された LUN 上のすべてのデータが消去されます。	デフォルト値：0



ノート:

書込みテストでは、指定された LUN 上のすべてのデータが消去されます。

Orionのコマンドライン例

次に様々なタイプのI/Oワークロードに対するOrionコマンドの例を示します。

1. OLTPデータベースのストレージを評価するには:

```
-run oltp
```

2. データ・ウェアハウスのストレージを評価するには:

```
-run dss
```

3. データの基本セット用:

```
-run normal
```

4. 読取り専用の小規模なランダムI/Oワークロードと大規模なランダムI/Oワークロードでのストレージ・パフォーマンスを理解する場合:

```
$ orion -run simple
```

5. 小規模なランダムI/Oワークロードと大規模なランダムI/Oワークロードの混合でのストレージ・パフォーマンスを理解するには:

```
$ orion -run normal
```

6. 32KBと1MBの読取りの組合せをランダムな位置に生成するには:

```
$ orion -run advanced -size_small 32 ¥
```

```
-size_large 1024 -type rand -matrix detailed
```

7. 複数の1MBの順次書込みストリームを生成して、1MB RAID0ストライプ化をシミュレートする場合:

```
$ orion -run advanced -simulate raid0 ¥
-stripe 1024 -write 100 -type seq -matrix col -num_small 0
```

8. 32KBと1MBの読取りの組合せをランダムな位置に生成するには:

```
-run advanced -size_small 32 -size_large 1024 -type rand -matrix detailed
```

9. 複数の1MBの順次書込みストリームを生成して、RAID0ストライプ化をシミュレートする場合:

```
-run advanced -simulate raid0 -write 100 -type seq -matrix col -num_small 0
```

Orionの出力ファイル

テスト実行の出力ファイルには、接頭辞<testname>_<date>が付きます。ここでdateはyyyymmdd_hhmmです。

表17-7にOrionの出力ファイルを示します。

表17-7 Orionで生成される出力ファイル

出力ファイル	説明
<testname>_<date>_hist.csv	I/O 待機時間のヒストグラム。
<testname>_<date>_iops.csv	小規模 I/O のパフォーマンス結果(IOPS)。
<testname>_<date>_lat.csv	小規模 I/O の待機時間(マイクロ秒)。
<testname>_<date>_mbps.csv	大規模 I/O のパフォーマンス結果(MBPS)。
<testname>_<date>_summary.txt	入力パラメータのサマリーが、小規模 I/O の最短待機時間(秒)、最大 MBPS、および最大 IOPS とともに表示されます。
<testname>_<date>_trace.txt	拡張された、未処理の出力。

ノート:

書込みテストを実行する場合は、LUN に格納されたデータが失われることに注意してください。

Orionの出力ファイル例

Orionは、表17-7に示すように、複数の出力ファイルを作成します。「Orionの開始」の項で示した例「mytest」の場合、出力ファイルは次のようになります。

- mytest_summary.txt: このファイルには次が含まれます。
  - 入力パラメータ
  - 大規模なランダムまたは順次ワークロードで観測された最大スループット
  - 小規模なランダム・ワークロードで観測された最大I/O率
  - 小規模なランダム・ワークロードで観測された最大待機時間
- mytest_mbps.csv: 大規模なランダムまたは順次ワークロードのデータ転送速度(MBPS)結果を含むカンマ区切りのファイル。一般的には、このファイルと他のすべてのCSVファイルには、2次元の表が含まれます。表の各行は大規模なI/O負荷レベルに対応し、各列は特定の小規模なI/O負荷レベルに対応します。このため、列のヘッダーは未処理の小規模I/Oの数、行のヘッダーは未処理の大規模I/Oの数(大規模なランダムI/Oテストの場合)または順次ストリームの数(大規模な順次I/Oテストの場合)になります。

  次に、「mytest」に対するOrion MBPSのCSV出力ファイルのデータ・ポイントを先頭からいくつか例として示します。単純なmytestコマンドラインでは、大規模I/Oと小規模I/Oの組合せはテストされません。このため、MBPSファイルには未処理0の小規模I/Oに対応する1列のみが含まれます。この例では、8つの未処理の大規模な読取りと小規模I/Oなしの負荷レベルで、レポート・データは103.06 MBPSのスループットを示しています。

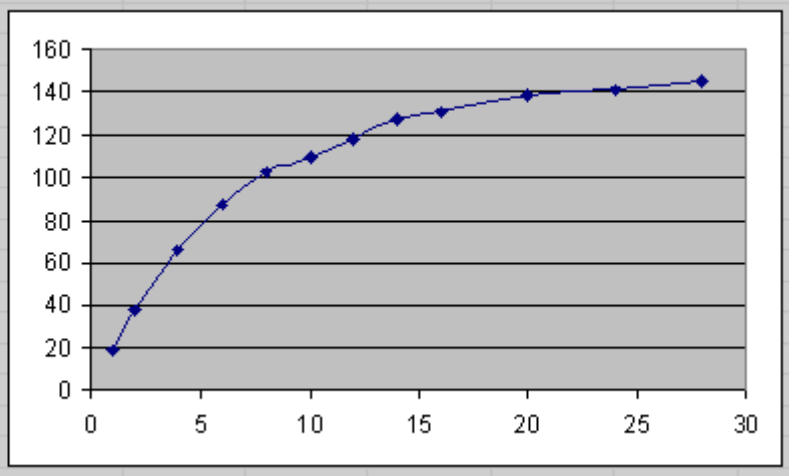
```
Large/Small,        0
1,     19.18
2,     37.59
4,     65.53
6,     87.03
8,    103.06
10,    109.67
.  .     .  .  .
.  .     .  .  .
```

  次のグラフに、様々な大規模I/O負荷レベルで測定されたデータ転送速度の例を示します。このグラフは、mytest_mbps.csvをスプレッドシートに読み込んでデータ・ポイントをグラフ化して生成できます。Orionはこのようなグラフを直接的には生成しません。X軸は未処理の大規模な読取りの数に対応し、Y軸は測定されたスループットに対応します。

  グラフは、標準的なストレージ・システムの動作を示しています。未処理のI/Oリクエストの数が増えるにつれて、スループットは上がります。ただし、特定のポイントでスループット・レベルは変化しなくなり、ストレージ・システムの最大スループット値であることを示します。

  図17-2 I/O負荷レベルの例

Large	MBPS
1	19.18
2	37.59
4	65.53
6	87.03
8	103.06
10	109.67
12	117.7
14	127.06
16	130.55
20	138.42
24	141.21
28	145.5



- mytest_mbps.csv: 小規模なランダム・ワークロードのI/Oスループット(IOPS)結果を含むカンマ区切りのファイル。MBPSファイルと同様に、列のヘッダーは未処理の小規模I/Oの数、行のヘッダーは大規模なランダムI/Oのテストでは未処理の大規模I/Oの数、大規模な順次I/Oのテストでは順次ストリームの数です。

  一般的には、CSVファイルには2次元の表が含まれます。ただし、大規模I/Oと小規模I/Oの組合せをテストしない単純なテストの場合は、結果ファイルは1行のみになります。このため、IOPSの結果ファイルは大規模I/Oを含まない1行のみになります。次の例で示すように、12の未処理の小規模な読取りと大規模I/Oなしのデータ・ポイントでは、スループットは951 IOPSです。

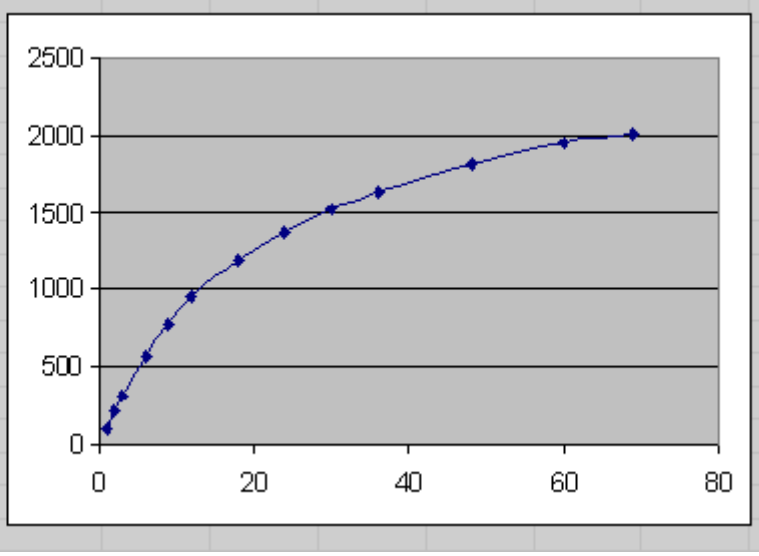
```
Large/Small,      1,      2,      3,      6,      9,     12 . . . .
0,              105,    208,    309,    569,    782,    951 . . . .
```

次のグラフは、mytest_iops.csvをExcelにロードし、データをチャート化することによって生成されます。このグラフは、様々な小規模I/O負荷レベルでのIOPSスループットを示しています。

グラフは、標準的なストレージ・システムの動作を示しています。未処理のI/Oリクエストの数が増えるにつれて、スループットは上がります。ただし、特定のポイントでスループット・レベルは変化しなくなり、ストレージ・システムが最大スループット値に達したことを示します。スループット・レベルが上がるほど、I/Oリクエストの待機時間は大幅に増加します。そのため、このデータをmytest_lat.csvに生成された待機時間結果で提供される待機時間データとともに表示することが重要です。

図17-3 様々な小規模I/O負荷レベルでのI/Oスループット

Small	IOPS
1	105
2	208
3	309
6	569
9	782
12	951
18	1189
24	1369
30	1521
36	1638
48	1815
60	1941
69	2013



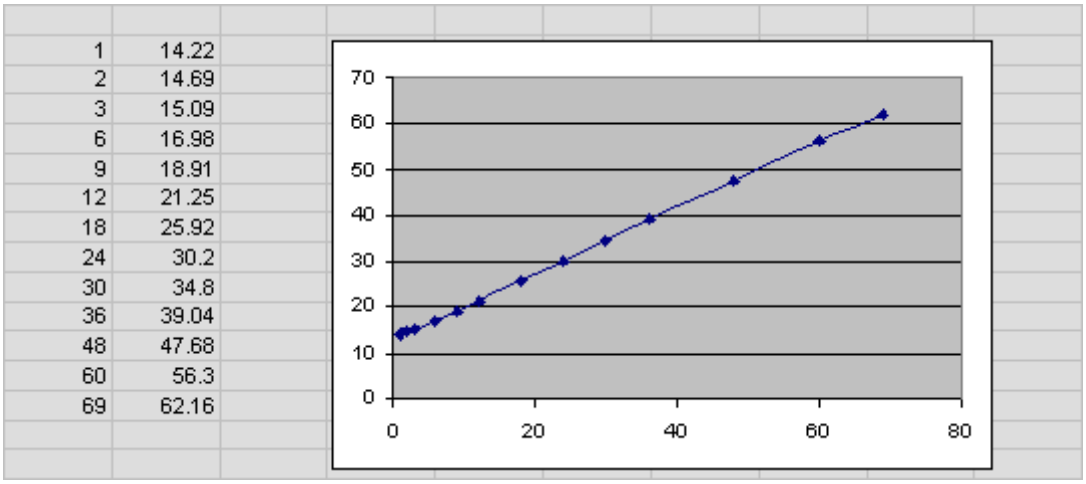
- mytest_lat.csv: 小規模なランダム・ワークロードの待機時間結果を含むカンマ区切りのファイル。MBPSファイルやIOPSファイルと同様に、列のヘッダーは未処理の小規模I/Oの数、行のヘッダーは未処理の大規模I/Oの数(大規模なランダムI/Oをテストする場合)または順次ストリームの数です。

  一般的には、CSVファイルには2次元の表が含まれます。ただし、大規模I/Oと小規模I/Oの組合せをテストしない単純なテストの場合は、結果ファイルは1行のみになります。このため、IOPSの結果ファイルは大規模I/Oを含まない1行のみになります。次の例では、12の未処理の小規模な読取りと大規模I/Oなしの持続する負荷レベルで、生成される結果は22.25ミリ秒のI/O応答待機時間を示しています。

```
Large/Small, 1, 2, 3, 6, 9, 12 . . . .
0, 14.22, 14.69, 15.09, 16.98, 18.91, 21.25 . . . .
```

  次のグラフは、mytest_lat.csvをExcelにロードし、データをチャート化することによって生成されます。このグラフは、mytestの様々な小規模I/O負荷レベルでの小規模I/Oレイテンシを示しています。

  図17-4 小規模I/O負荷レベルでの待機時間



- mytest_trace.txt: 拡張された、未処理のテスト出力を含みます。

ノート:

Orion は、標準出力でのテスト中に発生したエラーを報告します。

Orionのトラブルシューティング

1. <testname>.lunファイルで指定した1つ以上のボリュームでI/Oエラーが発生する場合;
   - ddなどのファイル・コピー・プログラムを使用して、テスト、読取り、または書込みと同じモードでボリュームにアクセスできることを確認してください。
   - ホストのオペレーティング・システムのバージョンが非同期I/Oを行えることを確認してください。
   - LinuxおよびSolarisでは、ライブラリlibaioが標準libディレクトリにあるか、またはシェル環境のライブラリ・パス変数(通常はシェルに対応してLD_LIBRARY_PATHまたはLIBPATH)を使用してこのライブラリにアクセスできることが必要です。
2. NASストレージで実行している場合:
   - Orionを実行するには、ファイル・システムを正しくマウントする必要があります。詳細は、Oracleインストレーション・ガイドを参照してください(たとえば、Databaseインストレーション・ガイド for Linux x86のセクション、

付録B「NASデバイスの使用」など)。

- mytest.lunファイルには既存のファイルの1つ以上のパスを含める必要があります。Orionはディレクトリやマウント・ポイントに対して機能することはありません。意味のあるテストにするには、ファイルは十分な大きさにする必要があります。このファイルのサイズは、データファイルの最終的な予測サイズ(つまり数年使用後のサイズ)にする必要があります。
- Linux(2.6カーネルを含む)のNFSで非同期I/Oを行うと、パフォーマンスが低下することがあります。
- 読取りテストを行っていて、初期化されていないかまだ書込みが行われていないファイルの未使用ブロックを読取りがヒットしている場合、高性能なNASシステムではゼロ設定されたブロックを返すことで読取りに見せかけることがあります。これが発生した場合は、予想外によいパフォーマンスになります。

  これを回避するには、読込みテストを実行する前に、ddなどのツールを使用してすべてのブロックに書込みを行います。

3. OrionをWindowsで実行する場合: RAWパーティションでのテストでは、パーティションを一時的にドライブ文字にマップして、これらのドライブ文字をtest.lunファイルで指定する必要があります。
4. Orion 32ビットLinux/x86バイナリをx86_64システムで実行する場合: 32ビットlibaio.soファイルを、同じLinuxバージョンを実行している32ビットのコンピュータからコピーしてください。
5. 多数のディスク(num_disksが30を超える)を使用してテストする場合:
   - 各データ・ポイントに長時間(120秒以上など)を指定するには、-durationオプションを使用してください(詳細はオプション・パラメータのセクションを参照)。Orionはすべてのスピンドルを特定の負荷レベルで実行を継続させようとするため、各データ・ポイントで立ち上げ時間が必要となり、これによってテストに要する時間が長くなることがあります。
   - 期間の値を増やすように指示する次のエラー・メッセージが表示されることがあります。

     ```
     Specify a longer -duration value.
     ```

     おおよその目安として、期間はスピンドル数の2倍程度にするとよいようです。使用しているディスク技術によって、プラットフォームで必要とする時間は上下します。
6. Orionによって使用されるライブラリに関するエラーが発生した場合:
   - Linux/Solaris: I/Oエラーのトラブルシューティングを参照してください。
   - NT-Only: 配布に含まれているOracleライブラリの移動や削除は行わないでください。これらはorion.exeと同じディレクトリに置く必要があります。
7. 「信じられないほどよい」パフォーマンス数値になった場合:
   - 大規模な読取りまたは書込みキャッシュがあるか、Orionプログラムとディスク・スピンドルの間のどこかでキャッシュの読取りと書込みを行っている場合があります。通常は、最も効果が高いのは、ストレージ・アレイ・コントローラです。このキャッシュのサイズを調べて、-cache_size advancedオプションを使用してそれをOrionに指定してください(詳細はオプション・パラメータのセクションを参照)。
   - ボリュームの合計サイズが、これまでの1つ以上のキャッシュと比較して極端に小さい場合があります。キャッシュを無効にするようにしてください。ストレージを共有する他のボリュームが本番環境で著しいI/Oアクティビティを示す(そして結果的に共有キャッシュの大部分を使用する)場合には、これが必要になります。
8. Orionが長い予測実行時間を報告している場合:

- -num_disksが高いときに実行時間は長くなります。Orionは内部的に線形計算式を使用して、指定された数のディスクを維持するためにかかる時間を特定します。
- -cache_sizeパラメータは、指定していない場合でも、実行時間に影響します。Orionはデフォルトでデータ・ポイントごとに2分間のキャッシュ・ウォーミングを行います。キャッシュを無効にする場合は、-cache_size 0を指定します。
- -durationに大きい値を指定すると、予想通り、実行時間は長くなります。

18 オペレーティング・システム・リソースの管理

この章では、Oracle Databaseのパフォーマンスを最適化するためのオペレーティング・システムのチューニング方法について説明します。

この章の構成は、次のとおりです。

- オペレーティング・システムのパフォーマンスの問題について
- オペレーティング・システムの問題の解決
- CPUについて
- CPUの問題の解決

関連項目:

- オペレーティング・システムのドキュメント
- 各プラットフォームのチューニング情報が記載された特定プラットフォームのOracle Databaseマニュアル

オペレーティング・システムのパフォーマンスの問題について

オペレーティング・システムのパフォーマンスの問題は、一般にプロセス管理、メモリー管理およびスケジューリングに関係します。Oracleデータベース・インスタンスをチューニングした後で、さらにパフォーマンスを改善する必要がある場合は、作業内容の確認またはシステム時間の短縮を試みてください。I/O帯域幅、CPU能力およびスワップ・スペースが十分にあることを確認してください。ただし、オペレーティング・システムをさらにチューニングしても、それがアプリケーションのパフォーマンスに大きな効果を与えることは期待できません。オペレーティング・システムの簡単なチューニングよりも、Oracle Database構成またはアプリケーションでの変更の方が、オペレーティング・システム効率に大きな差異が生じます。

たとえば、アプリケーションでbuffer busy waitsが非常に頻繁に発生する場合は、システム・コールの回数が増加します。アプリケーションをチューニングすることでbuffer busy waitsを削減すると、システム・コールの数が減少します。

この項では、オペレーティング・システムのパフォーマンスの問題に関する次の項目について説明します。

- オペレーティング・システムのキャッシュの使用
- メモリー使用量
- オペレーティング・システムのリソース・マネージャの使用

オペレーティング・システムのキャッシュの使用

オペレーティング・システムおよびデバイス・コントローラには、Oracle Databaseのキャッシュ管理と直接に競合しないデータ・キャッシュがあります。ただし、これらの構造はリソースを消費し、パフォーマンスにおける利点はほとんどありません。この状況は、LinuxまたはUNIXファイル・システムに格納されたデータベース・ファイルにおいて最も顕著です。デフォルトでは、データベースのすべてのI/Oはファイル・システム・キャッシュを使用します。

一部のLinuxおよびUNIXシステムでは、直接I/Oのファイルストアを使用できます。この場合、ファイル・システム・キャッシュをバイパスして、ファイル・システム内でデータベース・ファイルにアクセスできます。直接I/OによるCPUリソースの節約によって、ファイル・システム・キャッシュをプログラム・テキストやスプール・ファイルなどの非データベース・アクティビティ専用にすることができます。

ノート:

この問題は Windows では発生しません。データベースから要求されたファイルは、すべてファイル・システム内のキャッシュをバイパスします。

Oracle Databaseバッファ・キャッシュにブロックが格納されるため、多くの場合、オペレーティング・システムのキャッシュは冗長な機能ですが、場合によっては、データベース・バッファ・キャッシュが使用されない場合もあります。この場合、直接I/OまたはRAWデバイスを使用すると、オペレーティング・システム・バッファを使用する場合よりも、パフォーマンスがさらに低下することがあります。これには次のような例があります。

- TEMP表領域での読取りまたは書込み
- NOCACHE LOBに格納されるデータ
- パラレル実行サーバーがデータを読み込んでいる

ノート:

パラレル問合せのデータは、ディスクから PGA に直接読み込むかわりに、データベース・バッファ・キャッシュにキャッシュできる場合があります。この構成は、データベース・サーバーのメモリー容量が大きい場合に適しています。パラレル実行の使用をさらに学習するには、『Oracle Database VLDB およびパーティショニング・ガイド』を参照してください。

キャッシュを使用する場合でも、オペレーティング・システム・レベルですべてのファイルをキャッシュするのは望ましくない場合があります。

非同期I/O

同期I/Oでは、I/Oリクエストがオペレーティング・システムに発行されても、書込みの完了が確認されないかぎり書込みプロセスによってブロックされます。処理はその後で継続できます。非同期I/Oでは、I/Oリクエストが発行されて処理されている間も処理が継続されます。ボトルネックを回避できる場合は非同期I/Oを使用します。

プラットフォームには、デフォルトで非同期I/Oをサポートしているもの、特殊な構成が必要なもの、基礎となる一定のファイル・システム・タイプでのみ非同期I/Oをサポートしているものがあります。

FILESYSTEMIO_OPTIONS初期化パラメータ

FILESYSTEMIO_OPTIONS初期化パラメータを使用して、ファイル・システムのファイルについて非同期I/Oあるいは直接I/Oを、有効化または無効化することができます。このパラメータは、プラットフォーム固有であり、それぞれのプラットフォームに最適なデフォルト値が設定されています。

FILESYSTEMIO_OPTIONSは、次のいずれかの値に設定できます。

- ASYNCH: ファイル・システム・ファイル上の非同期I/Oを有効にします。非同期I/Oでは、転送に対する時間的な要件はありません。
- DIRECTIO: ファイル・システム・ファイル上の直接I/Oを有効にします。直接I/Oでは、バッファ・キャッシュがバイパスされます。
- SETALL: ファイル・システム・ファイル上の非同期および直接I/Oを有効にします。
- NONE: ファイル・システム・ファイル上の非同期および直接I/Oを無効にします。

関連項目:

詳細は使用しているプラットフォーム固有のマニュアルを参照してください。

NFSサーバー環境における非同期I/Oの制限

一部のネットワーク・ファイル記憶域(NFS)サーバー環境では、短期間に大量の非同期I/Oリクエストが作成されると、パフォーマンスが低下する場合があります。そのような場合には、DNFS_BATCH_SIZE初期化パラメータを使用してパフォーマンスを改善し、Oracleプロセスから発行されるI/O数を制限して、システムの安定性を高めます。

DNFS_BATCH_SIZE初期化パラメータは、Direct NFSクライアントが有効化されている場合に、Oracleフォアグラウンド・プロセスによってキューに入れることが可能な非同期I/Oの数を制御します。NFSサーバーで未処理の非同期I/Oリクエストを大量に処理できない環境では、このパラメータの値を128に設定することをお薦めします。その後、NFSサーバーのパフォーマンスに応じて、この値を増減できます。

ノート:

DNFS_BATCH_SIZE 初期化パラメータのデフォルト設定は 4096 です。推奨値の 128 を適用できるのは、NFS サーバーで大量の非同期 I/O リクエストを処理できず、サーバー遅延が検出されるシステムのみです。

関連項目:

DNFS_BATCH_SIZE初期化パラメータの詳細は、『Oracle Databaseリファレンス』を参照してください

I/Oパフォーマンス向上のためのDirect NFS Clientの使用

Direct NFS Clientは、NFSクライアントの機能をOracle Databaseに直接統合します。Direct NFS ClientはOracle Databaseの専用NFSクライアントであるため、高度に最適化されています。Direct NFS Clientでは、従来のオペレーティング・システムのNFSクライアントと比較して、NFSを介したデータベース・パフォーマンスが大きく向上しています。

Parallel NFSはNFSバージョン4.1で導入された機能であり、Direct NFS Clientではオプションで提供される機能です。Oracle Database 12cリリース2 (12.2)以降でサポートされています。Parallel NFSはスケーラビリティの高い分散ストレージ・プロトコルであり、クライアント、サーバー、およびストレージ・デバイスがそれぞれファイル・アクセスの管理を担当します。4.1より前のバージョンのNFSでは、サーバーのみがファイルのアクセス管理を担当します。したがって、Parallel NFSを使用すると、スケーラビリティの高い分散NASストレージを構築でき、I/Oパフォーマンスが向上します。

Oracle Database 12cリリース2 (12.2)以降では、Direct NFS ClientのDirect NFSディスパッチャ機能を使用することもできます。Direct NFSのディスパッチャによって、データベース・インスタンスからNFSサーバーに作成されるTCP接続が統合されます。大規模なデータベース・デプロイメントでは、Direct NFSディスパッチャを使用すると、スケーラビリティおよびネットワーク・パフォーマンスが向上します。そのため、多数のTCP接続がある場合には、Direct NFS Clientデプロイメントのパラレル NFSとともにDirect NFSディスパッチャを使用することをお薦めします。

関連項目:

- Direct NFS構成ファイルoranfstabでnfs_versionパラメータの値をpNFSに設定することによってDirect NFS ClientのパラレルNFS機能を有効化する方法の詳細は、Oracle Databaseインストレーション・ガイドを参照してください。
- ENABLE_DNFS_DISPATCHER初期化パラメータの値をtrueに設定することによってDirect NFS ClientのDirect NFSディスパッチャ機能を有効化する方法の詳細は、Oracle Databaseリファレンスを参照してください。

メモリー使用量

メモリー使用量は、バッファ・キャッシュの制限と初期化パラメータの両方の影響を受けます。

バッファ・キャッシュの制限

UNIXバッファ・キャッシュはオペレーティング・システムのメモリー・リソースを消費します。あるバージョンのUNIXでは、UNIXバッファ・キャッシュに一定量のメモリーを割り当てることができますが、現在ではより複雑なメモリー管理メカニズムが使用されるのが普通です。通常これらの管理メカニズムでは、I/Oをキャッシュするために空きメモリー・ページを使用できます。そのようなシステムでは、オペレーティング・システムのレポート・ツールは空きメモリーがないことを示すのが普通で、通常は問題になりません。プロセスがより多くのメモリーを必要とする場合、通常はI/Oデータをキャッシュするメモリーが開放されて、そのプロセス・メモリーを割り当てることができます。

メモリー使用量に影響を与えるパラメータ

Oracle Databaseセッションのメモリー要件は、様々な要因に依存します。一般的に、大きな要因には次のものがあります。

- オープン・カーソルの数
- PL/SQLで使用されるメモリー(PL/SQL表など)
- SORT_AREA_SIZE初期化パラメータ

Oracle Databaseでは、PGA_AGGREGATE_TARGET初期化パラメータを使用することで、セッションのメモリー使用量に対する制御が向上します。

オペレーティング・システムのリソース・マネージャの使用

一部のプラットフォームではオペレーティング・システム・リソース・マネージャが提供されます。これらは、システム・リソースへのアクセスに優先順位を付けることによってピーク負荷使用パターンの影響を少なくするように設計されています。これらは通常、ユーザーがアクセス可能なリソースと各ユーザーがこれらのリソースのどの程度を消費可能かを統括する、管理方針を実装します。

オペレーティング・システム・リソース・マネージャは、ドメインや他の類似のファシリティとは異なります。ドメインは、1つのシステム内で1つ以上のまったく別の環境を提供します。ディスク、CPU、メモリーおよびその他すべてのリソースが各ドメイン専用となっており、他のドメインからアクセスできません。他の類似のファシリティは、異なる領域、通常個別のCPUまたはメモリー領域にシステム・リソースの一部のみを完全に分離します。ドメインと同様、個別のリソース領域はその領域に割り当てられた処理専用となっています。プロセスは境界を超えて移行できません。ドメインとは異なり、その他すべてのリソース(通常はディスク)は、システムのすべてのパーティションからアクセスできます。

Oracle Databaseはドメイン内で実行される他、パーティション化されたメモリー(RAM)リソースの割当てが動的でなく、固定されている場合は、その他の不完全なパーティション構造体内で実行されます。

オペレーティング・システム・リソース・マネージャは、リソースのグローバル・プール内、通常はドメインあるいはシステム全体内でのリソース割当てに優先順位を設定します。プロセスはグループに割り当てられ、グループはリソース・プール内の任意の場所にあるリソースに割り当てられます。

ノート:

- 1 つのノードに複数のインスタンスがあり、それらの間でリソースを配分する場合は、各インスタンスを専用のオペレーティング・システム・リソース・マネージャ・グループまたは管理対象エンティティに割り当てる必要があります。管理対象エンティティの複数のインスタンスを実行するには、インスタンス・ケージングを使用して管理対象エンティティ内の CPU リソースをインスタンス間で配分する方法を管理します。Oracle Database Resource Manager は、CPU リソースを管理するときに、各インスタンスに対する CPU リソースが一定量であると予測します。インスタンス・ケージングを使用しない場合は、使用可能な CPU リソースは管理対象エンティティ内の CPU の数と同じであると予測します。インスタンス・ケージングを使用する場合、使用可能な CPU リソースは CPU_COUNT 初期化パラメータの値と同じであると予測します。CPU リソースが予測より少ない場合、Oracle Database Resource Manager でりソース計画のリソース割当てを実行したときに適切な結果が得られません。

- UNIX オペレーティング・システム・リソース・マネージャのメモリー管理および割当て機能では、Oracle Database の使用はサポートされていません。Oracle データベース・インスタンス内のリソース割当て機能を提供する Oracle Database Resource Manager は、任意のオペレーティング・システムのリソース・マネージャでは使用できません。

  Oracle Database および Oracle Database Resource Manager で動作するオペレーティング・システムのリソース管理、リソースの割当ておよび割当て解除機能の完全なリストは、システム・ベンダーおよび Oracle 代理から入手してください。特定のリリース・レベルとの互換性において、これらのシステム機能は動作保証されていません。

関連項目:

- Oracle Database Resource Managerの詳細は、『Oracle Database管理者ガイド』を参照してください
- インスタンス・ケージングの詳細は、『Oracle Database管理者ガイド』を参照してください

オペレーティング・システムの問題の解決

この項では、様々なシステムでのチューニングのヒントを示します。次の項目について説明します。

- UNIXベースのシステムのパフォーマンスに関するヒント
- Windowsシステムのパフォーマンスに関するヒント
- HP OpenVMSシステムのパフォーマンスに関するヒント

プラットフォーム固有の問題をよく理解し、使用しているオペレーティング・システムが提供するパフォーマンス・オプションを認識してください。

関連項目:

使用しているプラットフォーム固有のOracleマニュアルおよび使用しているオペレーティング・システム・ベンダーのマニュアル

UNIXベースのシステムのパフォーマンスに関するヒント

UNIXシステムで、オペレーティング・システムのシステム・コールの処理やプロセス・スケジューリングの実行にかかる時間と、アプリケーションの実行時間とで、妥当な比率の確立を試みます。システム・モードではなく、アプリケーション・モード(ユーザー・モードとも呼ばれる)での実行に大部分の時間を費やすことを目標としてください。

各モードで消費される時間の比率は、潜在する問題の徴候にすぎず、次の項目に関係している可能性があります。

- ページングまたはスワッピング
- 実行しているオペレーティング・システム・コールが多すぎる状態
- 実行しているプロセスが多すぎる状態

このような条件が存在する場合は、アプリケーションの実行に使用できる時間は少なくなります。オペレーティング・システム側の時間を多く解放するほど、アプリケーションが実行できるトランザクションの量が増えます。

Windowsシステムのパフォーマンスに関するヒント

UNIXベースのシステムと同様に、Windowsシステムでは、アプリケーション・モードの時間とシステム・モードの時間の比率を適切に設定してください。Windows管理パフォーマンス・ツールで多数の要因を容易に監視できます。CPU、ネットワーク、I/Oおよびメモリーはすべて、これらの領域でのボトルネックを容易に回避できるように同じグラフ上に表示されます。

HP OpenVMSシステムのパフォーマンスに関するヒント

メインフレームのページング・パラメータを検討し、Oracle Databaseが非常に大きな作業セットを利用できる点に留意します。

HP OpenVMS環境での空きメモリーは、どのオペレーティング・システム・プロセスにも実際にマップされていないメモリーです。ビジーなシステムでは、空きメモリーに、1つ以上の現行のアクティブ・プロセスに属するページが含まれる可能性があります。これがアクセスされるとソフト・ページ・フォルトが発生し、ページはそのプロセスの作業セットに組み込まれます。プロセスが作業セットを拡張できない場合は、プロセスによって現在マップされているページの1つを空きセットに移動する必要があります。

任意の数のプロセスが、作業セット内に共有メモリーのページを持つことができます。したがって、作業セットのサイズの合計が使用可能メモリーを著しく超過することがあります。通常、Oracleサーバーを実行している場合、SGA、Oracle Databaseカーネル・コードおよびOracle Formsランタイム実行可能ファイルはすべて共有可能であり、これらはページ・アクセスのおよそ80%または90%を占めます。

CPUについて

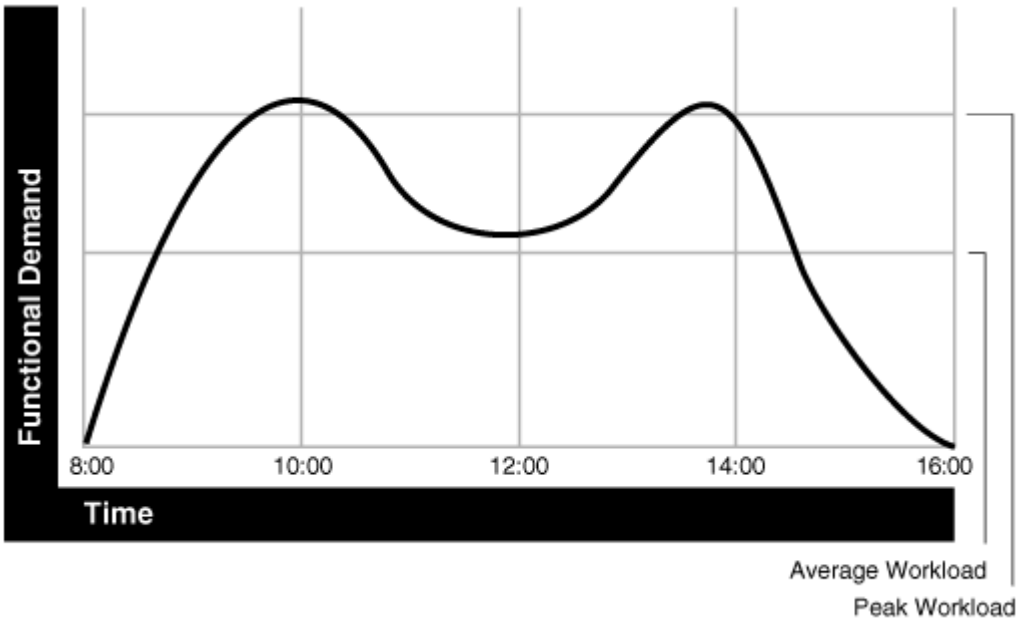
CPUの問題に対処するには、最初に、システムのCPUリソースの使用量について、適切な見積りを設定します。次に、十分なCPUリソースが使用可能かどうかを判断し、システムがいつリソースを使用しすぎているかを認識します。まず、次の3つのシステム状況において、Oracleデータベース・インスタンスのCPUリソースの使用量を判断します。

- システムがアイドル状態で、少数のOracle Databaseおよび非Oracleアクティビティが存在する場合
- システムが平均ワークロードの場合
- システムがピーク・ワークロードの場合

自動ワークロード・リポジトリ、StatspackまたはUTLBSTAT/UTLESTATユーティリティを使用して、様々なワークロードのスナップショットを取得できます。UNIXのvmstat、sarおよびiostatや、Windowsの管理パフォーマンス・モニタリング・ツールなどのオペレーティング・システム・ユーティリティは、V$OSSTATやV$SYSMETRIC_HISTORYビューとともに、自動ワークロード・リポジトリ、Statspack、UTLBSTAT/UTLESTATなどと同じ時間間隔で使用し、統計全体の補完ビューを提供することができます。

システムのCPU使用率のレベルを評価するときには、ワークロードが重要な要因となります。ピーク・ワークロード時には、CPU使用率が90%の場合アイドルは10%であり、待機時間が発生するのは受容できます。低ワークロード時に使用率が30%となるのも理解できます。しかし、システムが標準のワークロードで高い使用率を示している場合は、ピーク・ワークロードに耐える余裕がありません。たとえば、次の図に、午前10:00と午後2:00にピークとなるアプリケーションのワークロードの時間に伴う変化の例を示します。

図18-1 平均のワークロードおよびピーク・ワークロード



この例のアプリケーションでは、1日に8時間作業するユーザーが100人います。各ユーザーが5分ごとに1つのトランザクションを入力すると、1日のトランザクションは9,600になります。8時間にわたってシステムは1時間当たり1,200のトランザクションをサ

ポートする必要があり、1分当たり平均20トランザクションをサポートする必要があります。需要率が一定の場合は、この平均ワークロードを満たすようにシステムを構築します。

ただし、使用率のパターンは一定ではないので、この場合、1分当たり20トランザクションというのは単なる最低条件と考えられます。達成する必要があるピークの割合が1分当たり120トランザクションの場合は、このピーク・ワークロードをサポートできるシステムを構成する必要があります。

この例では、ワークロードがピークのときにOracle DatabaseがCPUリソースの90%を使用するとします。この場合、平均的なワークロード期間にOracle Databaseが使用するのは、次の等式が示すように、使用可能なCPUリソースの約15%以下です。

```
20 tpm / 120 tpm * 90% = 15% of available CPU resource
```

ここで、tpmは1分当たりのトランザクション数を表します。

システムが20 tpmを達成するためにCPUリソースの50%を必要とする場合は、問題があります。これでは、CPUの90%を使用しても1分当たり120トランザクションを処理できません。しかし、CPUの15%のみを使用して20 tpmを達成するようにこのシステムをチューニングした場合は、(線形のスケーラビリティを前提とすると)CPUリソースの90%を使用して1分当たり120トランザクションを処理できます。

アプリケーションを使用するユーザーが増加するにつれて、ワークロードがかつてのピーク・レベルにまで上昇する可能性があります。そのときには、実際には以前よりも高くなった新しいピークの割合に対応できるCPU能力はありません。

CPUの問題の解決

CPU容量の問題は、次のように解決します。

- CPUを過剰に消費する問題を検出し、解決します。詳細は、「CPU使用率の確認およびチューニング」を参照してください。
- Oracle Database Resource Managerを使用してCPUリソース割当ての優先度を付け、ピーク負荷使用パターンの影響を減らします。詳細は、「Oracle Database Resource Managerを使用したCPUリソースの管理」を参照してください。
- インスタンス・ケージングを使用して、複数CPUシステムで複数のデータベース・インスタンスが動作するときに同時に使用可能なCPUの数を制限します。詳細は、「インスタンス・ケージングを使用したCPUリソースの管理」を参照してください。
- ハードウェアの能力を高め、システム・アーキテクチャを改善します。

CPU使用率の確認およびチューニング

システムで実行されているすべてのプロセスは、使用可能なCPUリソースに影響を与えます。したがって、データベース以外の要素をチューニングすることで、データベースのパフォーマンスが向上する場合もあります。

V$OSSTATまたはV$SYSMETRIC_HISTORYビューを使用して、オペレーティング・システムからのシステム使用率統計を監視します。V$OSSTATおよびV$SYSMETRIC_HISTORYに含まれる有用な統計には、次のものがあります。

- CPUの数
- CPU使用率
- 負荷
- ページング
- 物理メモリー

関連項目:

V$OSSTATおよびV$SYSMETRIC_HISTORYの詳細は、『Oracle Databaseリファレンス』を参照してください。

システム全体で実行されているプロセスを確認するには、オペレーティング・システム・モニタリング・ツールを使用します。システムの負荷が非常に高い場合は、この項で後述するメモリー、I/Oおよびプロセス管理の各項目をチェックしてください。

多くのUNIXベースのシステムでは、sar -uなどのツールを使用して、システムのCPU使用率のレベルを確認できます。UNIXでは、統計情報に、ユーザー時間、システム時間、アイドル時間およびI/Oの待機時間が表示されます。標準ワークロードまたは低いワークロードで、アイドル時間およびI/O待機時間が0に近い場合(5%未満)、CPUの問題が存在します。

Windowsでは、管理パフォーマンス・ツールを使用してCPU使用率を監視できます。このユーティリティは、プロセッサ時間、ユーザー時間、特権時間(privileged time)、割込み時間およびDPC時間の統計を提供します。

関連項目:

- メモリー管理のチェック
- I/O管理のチェック
- ネットワーク管理のチェック
- プロセス管理のチェック

ノート:

このドキュメントでは、ほとんどの UNIX ベース・システムおよび Windows システムにおける CPU 使用率のチェック方法を説明します。その他のプラットフォームについては、使用しているオペレーティング・システムのマニュアルを参照してください。

メモリー管理のチェック

次のメモリー管理項目をチェックします。

- ページングとスワッピング
- 大きすぎるページ表

ページングとスワッピング

V$OSSTATビュー、UNIXのsar、vmstatなどのユーティリティ、またはWindowsの管理パフォーマンス・ツールなどを使用して、ページングとスワッピングの原因を調査します。

大きすぎるページ表

UNIXでは、処理領域が大きくなりすぎると、ページ表が大きくなりすぎることがあります。これはWindowsシステムでは問題になりません。

I/O管理のチェック

スラッシングはI/O管理の課題です。コンピュータのスラッシング(メモリー内外のスワッピングおよびページング処理)が発生しないように、ワークロードがメモリーに収まることを確認してください。オペレーティング・システムは固定サイズのタイム・スライスをユーザーのプロセスに割り当て、プロセスはそのタイム・スライス中にCPUリソースを使用できます。プロセスが実行可能かどうか、および必要な構成要素がすべてコンピュータにあるかどうかを確認するために時間の大半を浪費している場合、実際の処理に使用される時間は、割り当てられた時間の50%のみである可能性があります。

ネットワーク管理のチェック

クライアント/サーバーのラウンドトリップをチェックします。メッセージを処理する際にはオーバーヘッドが発生します。アプリケーションで多数のメッセージを生成し、ネットワークを介して送信する必要がある場合、メッセージ送信の待機時間によってCPUのオーバーロードが発生することがあります。この問題を緩和するには、多数のラウンドトリップを実行するのではなく、複数のメッセージを1つにまとめます。たとえば、配列挿入、配列フェッチなどを使用できます。

プロセス管理のチェック

この項で説明するいくつかのプロセス管理の項目をチェックする必要があります。

- スケジューリングとスイッチング
- コンテキストのスイッチング
- オペレーティング・システムの新規プロセスの開始

スケジューリングとスイッチング

オペレーティング・システムはスケジューリングおよびスイッチング処理に大量の時間を消費することがあります。使用中のプロセスが多すぎる可能性があるため、オペレーティング・システムの使用方法を調査してください。Windowsシステムでは、データベース以外のプロセスでサーバーが過負荷にならないようにします。

コンテキストのスイッチング

オペレーティング・システム固有の特性によって、システムがコンテキストのスイッチングに大量の時間を消費することがあります。コンテキストのスイッチングは、特に超大規模SGAの場合には不経済です。インスタンス当たりのプロセスが1つのみであるWindowsでは、コンテキストのスイッチングは問題になりません。すべてのスレッドが同じページ表を共有します。

Oracle Databaseには、複数のコンテキスト・スイッチング機能があります。

- ポスト・ウェイト・ドライバ

  Oracleプロセスは、別のOracleプロセスをポスト(メッセージの送信)でき、さらにポストされるのを待機できる必要があります。たとえば、フォアグラウンド・プロセスがLGWRをポストして、指定の時点までのブロックをすべて書き出してコミットを承認するよう、LGWRに通知する必要があります。

  このポスト・ウェイト・メカニズムは、UNIXのセマフォを使用して実装するのが普通ですが、これらのセマフォがリソース消

費型である場合があります。したがって、一部のプラットフォームでは、ポスト・ウェイト・ドライバを提供しています。通常は、ポスト・ウェイト・インタフェースを軽量に実装できるカーネル・デバイス・ドライバが提供されます。

- メモリーマップ済システム・タイマー

  Oracle Databaseでは、システム時間を問い合せてタイミング情報を得る必要が生じる場合があります。その場合、オペレーティング・システム・コールが実行され、比較的コストのかかるコンテキストのスイッチングが発生することがあります。一部のプラットフォームでは、メモリーマップ済タイマーを実装し、プロセス仮想アドレス空間のアドレスに現在のタイミング情報を収集します。このメモリーマップ済タイマーから時間を読み取る方が、システム・コールのコンテキストのスイッチングのオーバーヘッドよりも経済的です。

- 1回のコールで複数の非同期I/Oを発行するリストI/Oインタフェース

  リストI/Oとは、個別のシステム・コールで複数のI/Oリクエストを発行するかわりに、1回のシステム・コールで複数の非同期I/Oリクエストを発行できるApplication Program Interfaceです。この機能の主な利点は、コンテキストのスイッチングの所要回数が減少することです。

オペレーティング・システムの新規プロセスの開始

オペレーティング・システムの新規プロセスを開始する際には高いコストがかかります。開発者は、単一目的のプロセスを生成し、そのプロセスを終了後に、また新規のプロセスを生成することがよくあります。この方法ではそのつどプロセスの再生成と破壊が行われるため、特に大規模なSGAを持つアプリケーションではCPUが集中して消費されます。CPUは、そのたびにページ表を構築する必要があります。共有メモリーを確保またはロックするときは、すべてのページにアクセスする必要があるため、問題が増大します。

たとえば、1GBのSGAがある場合は、4KBごとにページ表のエントリがあり、1つのページ表のエントリは8バイトになります。エントリのサイズの合計は(1GB÷4KB)*×8バイトになります。ページ表がロードされていることを絶えず確認する必要があるので、コストが高くなります。

Oracle Database Resource Managerを使用したCPUリソースの管理

Oracle Database Resource Managerは、データベース・ユーザーおよびアプリケーション間のCPUリソースの割当ておよび管理を次のように行います。

- CPU飽和の防止

  CPU稼働率が100%の場合、Oracle Database Resource Managerを使用して、各コンシューマ・グループのセッションに最大限のCPUを割り当てることができます。この機能により、優先順位の高いセッションをすぐに実行できるように、優先順位の低いセッションのCPU消費量を削減できます。

- コンシューマ・グループに対するCPU使用量の制限

  リソース・マネージャのディレクティブmax_utilization_limitを使用して、コンシューマ・グループが使用できるCPU使用率を厳密に制限できます。この機能は、優先順位の低いセッションのCPU消費量を制限して、コンシューマ・グループ内のワークロードにより安定したパフォーマンスを提供できるようにします。

- リソース集中型の問合せによるダメージの制限

  Oracle Database 11gリリース2(11.2.0.2)から、Oracle Database Resource Managerは、コールの最大実行時間を制限することで、または長時間実行問合せを低優先順位の各コンシューマ・グループに移動することで、リソース集中型の問合せによるダメージを制限することができます。

- コンシューマ・グループに対するパラレル文アクティビティの制限

  Oracle Database 11gリリース2(11.2.0.2)から、リソース・マネージャのディレクティブparallel_target_percentageを使用して、1つのコンシューマ・グループがすべてのパラレル・サーバーを占有するのを回避できます。データベースは、この制限を超える原因となるパラレル文をキューに入れます。

  たとえば、パラレル・サーバーのターゲット数が64で、コンシューマ・グループETLがこのディレクティブを50%に設定しているとします。コンシューマ・グループETLが30個のパラレル・サーバーを使用しており、新しいパラレル文が4個のパラレル・サーバーを必要とする場合、データベースはこの文をキューに入れます。

関連項目:

- Oracle Database Resource Managerの使用方法を学習するには、『Oracle Database管理者ガイド』を参照してください
- パラレル問合せの使用方法を学習するには、『Oracle Database VLDBおよびパーティショニング・ガイド』を参照してください

インスタンス・ケージングを使用したCPUリソースの管理

1つのシステムで複数のデータベース・インスタンスが実行されている場合、インスタンスはCPUリソースを競い合います。1つのリソース集中型のデータベース・インスタンスが、他のインスタンスのパフォーマンスを大きく低下させる場合があります。この問題を回避するには、インスタンス・ケージングを使用して、各インスタンスが使用できるCPU数を制限します。このとき、Oracle Database Resource Managerは、インスタンスに設定されたリソース計画に基づいて様々なデータベース・セッション間にCPUを割り当て、インスタンスがCPUバインドになる可能性を最小限に抑えます。

関連項目:

インスタンス・ケージングの詳細は、『Oracle Database管理者ガイド』を参照してください。

用語集

ADOポリシー

自動データ最適化(ADO)のためのルールおよび条件を指定するポリシー。たとえば、ADOポリシーにより、オブジェクトが作成の30日後(条件)にNOINMEMORYとマークされる(アクション)よう指定できます。CREATE TABLE文およびALTER TABLE文のILM句を使用してADOポリシーを指定します。

自動データ最適化(ADO)

情報ライフサイクル管理(ILM)計画を実装するために、ポリシーを作成し、それらのポリシーに基づいてアクションを自動化するテクノロジ。

自動インメモリー

IM列ストアからコールド(アクセス頻度の低い)セグメントを自動的に削除し、作業データ・セットが常に移入されるようにする機能。

可用性

要求に応じてアプリケーション、サービスまたは機能を使用可能な程度。

ブルーム・フィルタ

セット内のメンバーシップをテストする低メモリー・データ構造。データベースでは、ハッシュ結合のパフォーマンスを高めるためにブルーム・フィルタが使用されます。

列圧縮単位(CU)

インメモリー圧縮単位(IMCU)内の列のための隣接する記憶域。

列データ・プール

列データを格納する、インメモリー領域内のサブプール。1 MBプールとも呼ばれます。

列形式

インメモリー列ストアにあるオブジェクト用の列ベースの形式。データ・ブロックで使用される、行形式と対照を成す列形式。

共通ディクショナリ

ローカル・ディクショナリから作成された、セグメント・レベルの、インスタンス固有のマスター・ディクショナリ・コードのセット。ローカル・ディクショナリは、列圧縮単位(CU)固有のディクショナリ・コードのソート済リストです。結合グループでは、結合を最適化するために共通ディクショナリが使用されます。

圧縮階層化

アクセス・パターンに基づく、データへの様々なレベルの圧縮の適用。たとえば、管理者は、アクセスがより低速になるという代償

を払い、より高い圧縮率で、非アクティブなデータを圧縮できます。

データ・フロー演算子(DFO)

パラレル問合せのデータ再分散ステージ間の作業単位。

データベース・バッファ・キャッシュ

unresolvable-reference.htmlでデータ・ブロックのコピーを保持している部分。unresolvable-reference.htmlに同時接続されたクライアント・プロセスはすべて、バッファ・キャッシュへのアクセスを共有します。

稠密グループ化キー

グループ化列が特定のファクト表またはディメンションから取得されるすべてのグループ化キーを表すキー。

稠密結合キー

結合列が特定のファクト表またはディメンションから取得されるすべての結合キーを表すキー。

稠密キー

ネイティブ整数として格納され、値の範囲を持つ数値キー。

ダブル・バッファリング

バックグラウンド・プロセスで、変更した最新の行を元の行と組み合せることで新しいインメモリー圧縮単位(IMCU)バージョンを作成する再移入メカニズム。再移入中は、古いIMCUを引き続き問合せに使用できます。

式

1つ以上の値、演算子および値を評価するSQL機能の組合せ。

式の取得間隔

データベースが、取得可能なIM式を検討する時間間隔。

式取得ウィンドウ

DBMS_INMEMORY_ADMINパッケージのIME_OPEN_CAPTURE_WINDOWおよびIME_OPEN_CAPTURE_WINDOWプロシージャの呼出しによって定義される式の取得間隔。

式統計ストア(ESS)

式評価に関する統計を格納する、オプティマイザによって保持されるリポジトリ。セグメントごとに、ESSによって、実行頻度、評価コスト、タイムスタンプ評価などの統計が監視されます。ESSは本来、永続的で、式の高速参照のためのSGA表現を持っています。

ヒート・マップ

ヒート・マップでは、データ・ブロックおよび行の需要が示されます。異なるストレージ層に移動する候補となるセグメントを決定する、自動データ最適化(ADO)。

ホーム・ロケーション

IMCUが存在するデータベース・インスタンス。Oracle RACで自動DOPが有効になっている場合、パラレル問合せコーディネータでは、ホーム・ロケーションを使用して、各IMCUが存在する場所、その大きさなどが判断されます。

インメモリー・ハイブリッド・スキャン

IM列ストアと行ストアの両方をスキャンする問合せ。すべての述語列にINMEMORY属性があり、SELECTリストの一部の列にINMEMORY属性がない場合、オプティマイザはインメモリー・ハイブリッド・スキャンを自動的に考慮します。

ハイブリッド・パーティション表

パーティションの一部がデータ・ファイル・セグメントに格納されたり、外部データ・ソースに格納される表。

IM集計

単一の大きい表から複数の小さい表に結合する複数の問合せに対して集計を促進する最適化。変換では、KEY VECTORおよびVECTOR GROUP BY演算子が使用されます。これが、VECTOR GROUP BY集計とも呼ばれる理由です。

IM列ストア

迅速スキャン用に最適化された表とパーティションのコピーを、列形式で格納するオプションのSGA領域。

IM動的スキャン

インメモリー表スキャンを自動的にパラレル化する軽量スレッドの使用。

IM式

インメモリー列ストアに結果が格納されるSQL式。last_nameが、IM列ストアに格納される列である場合は、IM式がUPPER(last_name)になる場合があります。

IMCUミラー化

Oracle RACでは、複数のIM列ストアでのIMCUの複製。たとえば、インスタンス1およびインスタンス2上のIM列ストアが、同じsales表を使用して移入されます。

IMCUプルーニング

インメモリー列ストアの問合せにおける、各IMCUの高い値および低い値に基づくIMCUの排除。たとえば、100を上回る製品IDが文でフィルタリングされる場合、100を下回る値が含まれるIMCUのスキャン処理がデータベースで回避されます。

IM記憶域索引

IMCU内のすべての列の最小値および最大値を格納する、IMCUヘッダー内のデータ構造。

インメモリー・アドバイザ

データベース内の分析処理ワークロードを分析する、ダウンロード可能なPL/SQLパッケージ。このアドバイザは、インメモリー移入による利益を得られる、IM列ストアのサイズ、およびオブジェクトのリストを推薦します。

インメモリー集計

IM集計を参照してください。

インメモリー領域

IM列ストアが含まれる、オプションのSGAコンポーネント。

インメモリー列ストア

IM列ストアを参照してください。

インメモリー圧縮単位(IMCU)

高速スキャン用に最適化されたインメモリー列ストア内の記憶域単位。インメモリー列ストアでは、各列が表内に別々に格納され、圧縮されます。各IMCUでは、ある行サブセットのすべての列が特定の表セグメント内に含まれます。

IMCUとデータベース・ブロックのセットの間には1対多マッピングが存在します。たとえば、表に列c1およびc2が含まれ、その行がディスク上の100個のデータベース・ブロックに格納されている場合、IMCU 1にはブロック1から50までの両方の列の値が格納され、IMCU 2にはブロック51から100までの両方の列の値が格納されます。

インメモリー・コーディネータ・プロセス(IMCO)

主要タスクが列データのバックグラウンド移入および再移入の開始である、バックグラウンド・プロセス。

In-Memory動的スキャン

IM動的スキャンを参照してください。

インメモリー式

IM式を参照してください。

インメモリー式単位(IMEU)

インメモリー式(IM式)の計算結果を格納するコンテナ。各IMEUは、それぞれに独自の親のインメモリー圧縮単位(IMCU)にリンクされています。

インメモリー・ファスト・スタート

データベース・インスタンス再起動時のIM列ストアへのデータ移入時間を大幅に削減する機能。

インメモリー移入

移入を参照してください。

インメモリー仮想列

インメモリー列ストアに移入される候補となる仮想列。

情報ライフサイクル管理(ILM)

データをその有用期間中ずっと管理するためのプロセスおよびポリシーのセット。

結合グループ

同じ表または異なる表から頻繁に結合された列を指定するユーザー定義オブジェクト。外部表はサポートされていません。

一般的な結合グループ候補は、ファクト表およびディメンション表を結合するために使用される列のセットです。結合グループは、INMEMORY_SIZEがゼロ以外の値の場合のみサポートされます。

キー・ベクター

稠密結合キーと稠密グループ化キー間をマップするデータ構造。

ラージ・プール

バックアップおよびリストア操作、I/Oサーバー・プロセスおよび共有サーバーとOracle XAのセッション・メモリー用に、大量のメモリー割当てを提供するSGA内のオプション領域。

ローカル・ディクショナリ

列圧縮単位(CU)固有のディクショナリ・コードのソート済リスト。

軽量スレッド

In-Memory動的スキャンで使用される実行エンティティ。軽量スレッドは、IMCUのスキャンをパラレル化するために役立ちます。

メタデータ・プール

IM列ストア内に存在するオブジェクトについてのメタデータを格納する、インメモリー領域のサブプール。メタデータ・プールは、64 KBプールとも呼ばれます。

memoptimizeプール

バッファおよびMEMOPTIMIZE FOR READと指定されたヒープ構成表の関連構造を格納するSGAプール。

オンデマンド移入

INMEMORY PRIORITYがNONEに設定されている場合、IM列ストアでは、全体スキャンを通じてアクセスされるときのみオブジェクトが移入されます。オブジェクトは、アクセスされることがないか、索引スキャンまたはROWIDによるフェッチを通じてしかアクセスされ

ない場合、移入されることはありません。

OSON

Oracleの最適化バイナリJSON形式。OSONは、Oracleデータベース・サーバーおよびデータベース・クライアントでのJSONデータ・モデルの高速な問合せおよび更新を可能にします。

OZIP

非常に高速な解凍を提供する、独自の圧縮技術。OZIPは、Oracle Database向けに特別に調整されています。

パーティション交換ロード

表を作成し、データをそれにロードしてから、既存の表パーティションをその表と交換する技術。この交換プロセスは、実際のデータの移動を伴わないDDL操作です。

移入

データ・ファイルから既存のデータ・ブロックを読み取り、行を列形式に変換してから、その列データをIM列ストアに書き込む操作。一方、ロードとは、DMLまたはDDLを使用して新しいデータをデータベースに移すことを指します。

優先度ベース移入

PRIORITYがNONE以外の値に設定されている場合、Oracle Databaseでは、オブジェクトは優先順位付けされた移入キューに追加されます。データベースでは、CRITICALからLOWまで、それらのキュー位置に基づいてオブジェクトが移入されます。優先順位ベースと呼ばれる理由は、IM列ストアではデータベースが再度開かれるときは必ず、優先順位付けられたリストを使用して自動的にオブジェクトが移入されるためです。オンデマンド移入と異なり、オブジェクトは、移入に全体スキャンを必要としません。

再移入

現在移入されているインメモリー圧縮単位(IMCU)の自動リフレッシュ。そのデータが大幅に変更された後に行われます。一方、移入は、IM列ストアでの最初のIMCU作成です。

サービス

同じ属性、パフォーマンスしきい値および優先順位を共有する、アプリケーション・ワークロードの論理表現。単一のサービスをOracle RACデータベースの1つ以上のインスタンスに関連付けると、単一のインスタンスで複数のサービスをサポートできます。

SIMD

単一命令、複数データ。データを別々の命令としてではなく、**ベクトル**と呼ばれる単一の単位として処理する命令。SIMD処理は**ベクトル化**と呼ばれます。

スナップショット・メタデータ単位(SMU)

関連するインメモリー圧縮単位(IMCU)のメタデータおよびトランザクション情報を含む、インメモリー領域内の記憶域単位。

領域管理ワーカー・プロセス(Wnnn)

インメモリー・コーディネータ・プロセス(IMCO)にかわってIM列ストアにデータを移入または再移入するプロセス。

失効しきい値

再移入を開始する、IMCUのトランザクション・ジャーナル内のエントリについて内部的に設定されたパーセンテージ。

ストレージ階層化

アクセス・レベルに応じた、ストレージの様々な層でのデータのデプロイメント。たとえば、管理者は、非アクティブなデータを高パフォーマンスで高コストのストレージから低コストのストレージに移行します。

表スキャン・プロセス

IM動的スキャンを調整するフォアグラウンドまたはPQプロセス。

しきい値ベース再移入

IMCU内の失効エントリの数が内部失効しきい値に達したときのIMCUの自動再移入。

トランザクション・ジャーナル

IM列ストアのトランザクションの一貫性を保つ、スナップショット・メタデータ単位(SMU)内のメタデータ。

トリクル再移入

しきい値ベース再移入の補完機能。インメモリー・コーディネータ・プロセス(IMCO)では、失効エントリが存在するが失効しきい値に満たないIM列ストア内の任意のIMCUに対して、トリクル再移入を自動的に引き起こす場合があります。

ベクター集計

IM集計を参照してください。

仮想列

ディスク上に格納されない列。データベースは、必要に応じて、一連の式またはファンクションを計算することによって仮想列の値を導出します。

作業データ・セット

指定時間にアクティブに問合せされているINMEMORYオブジェクトのサブセット。通常、作業の作業データ・セットは時間の経過とともに変化します。

索引

A

B

C

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

        - direct path 10.3.6
        - enqueue 10.3.7
        - free buffer waits 10.3.9
        - アイドル待機イベント 10.3.10
        - ラッチ 10.3.11
        - ライブラリ・キャッシュ・ラッチ 10.3.11
        - log buffer space 10.3.15
        - log file parallel write 10.3.12
        - log file switch 10.3.16
        - log file sync 10.3.17
        - ネットワーク通信待機イベント 10.3.19
        - rdbms ipc reply 10.3.18
        - リソース待機イベント 10.3.4
- Windowsのパフォーマンス 18.2.2
- workloads 2.6.2, 2.6.3, 2.6.4