

Oracle9i DatabaseからOracle
Database 11gへのアップグレード：
実際のお客様事例

Oracle ホワイト・ペーパー
2008年10月

Oracle Database 11gへのアップグレード

概要	3
"11gへの移行"プロジェクトの3つのフェーズ	4
参照システムの設定	6
Oracle Database 11gへのアップグレード	7
テスト実行1 - 11.1.0.6のワークロード - パラメータ変更および新規機能なし	11
パフォーマンスの最適化	12
初期化パラメータに関する推奨事項	12
テスト実行2 - init.oraファイル内のOracle Database 11gのデフォルト・パラメータ	13
テスト実行3 - システム統計作成による影響	13
SQL Plan Management	14
既知の実行計画の転送	14
Oracle9i DatabaseからOracle Database 11gへのアップグレード	14
Oracle Database 10gからOracle Database 11gへのアップグレード	15
SQL Plan Managementの維持	17
結論	17
Oracle Real Application Testingによる最適化とロード・シミュレーション	18
SQL Performance Analyzer - 変更された実行計画の簡単な検出を可能にするツール	19
テストの目的	19
テストの設定	19
SQL文の取得	19
SQL Tuning Setへの文のロード	20
SQL Tuning Setの維持	21
SPA分析タスクの作成とパラメータ化	22
Oracle9iのパフォーマンス・データによるSPA追跡の実行	22
Oracle Database 11gのパフォーマンス・データによるSPA追跡の実行	23
レポート/比較	23
SQL Performance Analyzerの結果	23
テスト実行4 - SPAによるinit.oraの最適化	25
SQL Performance Analyzerに関する結論	26
Database Replay	27
テストの目的	27
テストの設定	27
ワークロードの取得	28
前処理	29
ワークロードの再生	29
レポート作成	31
結論	31
PL/SQLのネイティブ・コンパイル	33
テスト実行5 - PL/SQLのネイティブ・コンパイル	33
結論	33
SQL Tuning AdvisorおよびAutomatic SQL Tuning	34
テスト実行6 - SQL Tuning AdvisorおよびAutomatic SQL Tuning	37
結論	37
最終テスト実行 - すべての自動化機能および統計機能の無効化	38
最終的な結論	39
参考資料	40

概要

Oracleデータベースのアップグレードは、データベースITの担当者からリスクの高いタスクとみなされることが少なくありません。アップグレードを成功させるには、アップグレード・テストを実行しなければならないだけでなく、すべてのアプリケーションの動作を新しいデータベース環境で検証してアップグレードの正当性を立証し、アップグレード前と同じかそれ以上のパフォーマンスが得られるようにする必要があります。

Oracle9i DatabaseからOracle Database 11gへの直接的なアップグレードは、2012年の8月まではプレミアム・サポートでのみ保証されます。このサポート・レベルでは、9iから11gへのアップグレードで使用できる実用的な機能が数多く提供されています。これらの機能を利用することで、データベースのアップグレード処理において、かつてないほどの高い予測可能性とリスク削減効果を手に入れることができ、新しいOracle Database 11g環境で極めて高いパフォーマンスが実現します。

このOracleテクニカル・ホワイト・ペーパーでは、世界最大の小売企業の1つで実際に行われた全体的な機能検証 (PoC) を取り上げます。この企業では、"Oracle Database 11gへの移行"におけるPoCとして、400のデータベースをOracle 9.2.0.8からOracle Database 11.1.0.6に直接アップグレードするもっとも簡単な方法をテストすること、そしてOracle Database 11gのパフォーマンスの評価を行うことの2つを設定しました。具体的な評価の対象として挙げられたのは、Oracle Tuning Pack、SQL Plan Management、Database ReplayとSQL Performance Analyzerの2つの機能を備えたOracle Real Application Testing、PL/SQLのネイティブ・コンパイルなどです。このケースでは、特に夜間のOLTPバッチ処理の実行を、現在の本番環境で行うのと同じ時間か、それよりも短い時間で完了する必要がありました。

結果は実に驚くべきものでした。Oracle Database 11gではOracle9i Databaseと比べて高速なパフォーマンスが実現されただけでなく、アプリケーションを変更する必要がなかったために、アップグレード・プロジェクト全体に必要な作業時間やリソースも以前のアップグレードより少なくて済んだのです。すべての機能検証を評価した後、この企業では、400のOracle9i DatabaseすべてをOracle Database 11.1.0.7に直接アップグレードする作業を開始しました。"11gへの移行"プロジェクトすべてが、2009年の春までに完了する予定となっています。このプロジェクトには、実際のロールアウトと、150を超えるOracle Application ServerインスタンスのApplication Server 10.1.2.0.3へのアップグレードも含まれています。データベースおよびアプリケーション・サーバー・ソフトウェアのロールアウトは、Oracle Grid ControlのProvisioning Packを使用して行われる予定です。

"11gへの移行"プロジェクトの3つのフェーズ

"Oracle Database 11gへの移行"プロジェクトは、世界でも最大規模の小売企業において2008年5月から9月にかけて行われたもので、この企業では、それまでOracle Database 9.2.0.8上で約400のOracleデータベースを運用していました。プロジェクトにおいて定められていた目標は、以下のとおりです。

1. 手動による介入をできる限り抑えながら、最小限の時間で、140のサーバー上に配置された400のOracle9iを直接Oracle Database 11gにアップグレードするための最善のソリューションを特定する。データベースの平均サイズは120GBです。
2. 極めて重要性の高い夜間のバッチ作業を、Oracle9iで実行するのと同じ時間内に終了できるようにする。この夜間ジョブは、数千におよぶ小売店舗用の商品計画すべてを処理する、累積されたOLTPの実行です。リファレンス実行の完了時間は1時間45分です。これらのジョブはCPUへの負荷が極めて高いため、ジョブの実行中は新しいCPUサーバー16台を最大限に使用しています。また、毎晩データベースごとに25GBのREDOログ情報が生成されます。
3. Oracle Database 11gへの移行のためにアプリケーションの変更が必要かどうかを判断する。過去に行ったOracle8iからOracle9iへのUnicode移行などでは、この判断を下すために多くの作業が必要でした。
4. Oracle Database 11gの新しいパフォーマンス機能を評価する。評価対象となる機能は、SQL Tuning Advisor、SQL Access Advisor、SQL Plan Management (SPM)、SQL Performance Analyzer (SPA)、Database Replay、PL/SQLのネイティブ・コンパイルなどです。また、それぞれの機能に対し、Oracle Database 11gへのアップグレードによるパフォーマンス改善効果と手作業削減効果の両面からも評価します。

プロジェクトは、次の3つのフェーズにわけて行われました。

1. 参照システムの設定、および繰り返し利用可能な本番ワークロードの作成
2. Oracle Database 11gへのアップグレード、および最適化前のパフォーマンス値の比較
3. Oracle Database 11gの機能によるパフォーマンスの最適化
 - a. 初期化パラメータのチューニング
 - b. SQL Performance Analyzerによる実行計画変更の検出
 - c. Oracle Real Application Testingによるロード・シミュレーション：完全なワークロードを取得してアップグレードしたデータベースで再生
 - d. SQL Plan ManagementによるOracle9iと同じレベルの実行計画の維持

- e. PL/SQLのネイティブ・コンパイル機能によるPL/SQLパフォーマンス全体の改善
- f. SQL Tuning AdvisorとSQL Access AdvisorによるSQLの問合せとワークロードの最適化
- g. Oracle Database 11gのチューニング拡張機能すべてを無効化した結果とOracle9iとの比較

参照システムの設定

このケースでは、テスト用として、16のCPUと32GBのRAMが搭載されたIBM P670システムが1つ用意されました。このシステムは、本番システムと同じものです。オペレーティング・システムはAIX 5.3 TL8、ストレージ・サブシステムはEMC DMX2000です。

本番環境では、各サーバーで、それぞれ10の小売店舗をホストしている3つのデータベースを平行で実行しています。プロジェクト開始時の本番データベースのリリースはOracle Database 9.2.0.8です。

参照システムには、Oracle Database 9.2.0.8を含むORACLE_HOMEと、Oracle Database 11.1.0.6を含むもう1つのホームをインストールしました。また、新しいOracle Real Application Testingパックを利用するため、『Metalink Note : 560977.1』に従って以下のパッチを適用しました。

- Oracle Database 11.1.0.6
パッチ6865809:SQL Performance Analyzer - 11gにおける9iトレース情報の抽出機能
- Oracle Database 9.2.0.8
パッチ6973309 : 9.2.0.8における取得機能

Oracle Recovery Manager (Oracle RMAN) を使用して、9.2.0.8リリースの3つの異なる本番データベースの参照システムへのコピーおよびリストアを行いました。また、繰り返し可能な夜間バッチ実行も用意しました。このバッチ実行は、数時間に渡って累積されたOLTPの実行で、実環境で使用される代表的なシナリオです。

Oracle Database 11gへのアップグレード

Oracle Database 11gへの直接的なアップグレードは、Oracle 9.2.0.4とそれ以降の全リリースでサポートされています。アップグレード処理は、グラフィカル・ユーザー・インターフェース・ツールの1つであるOracle Database Upgrade Assistant (Oracle DBUA) を使用して行うか、またはアップグレード・スクリプトを使用して手動で実行できます。オラクルでは、必要なチェックと変更すべてを実行できるOracle DBUAによるアップグレードを推奨しています。

ただし、本書で取り上げるケースでは、Oracle DBUAではなく手動によるアップグレード方法を使用しています。Oracle DBUAを使用すると、オペレーティング・システムのチェックを実行する場合を除き、アップグレード・プロセス全体をサイレント・モードで自動的に完了できるうえ、Oracle RMANによるバックアップ制御や、必要なデータベース・パッチの適用も可能となります。今回のケースでは、以前のアップグレード・プロジェクトで使用されていた複雑なシェル・スクリプトを、Oracle Database 11gの要件に合わせてカスタマイズしています。

アップグレードをスクリプトとコマンドラインにより手動で行う場合、以下の手順に分けることができます。

- Oracle9iインストールの健全性の維持
 - SYS.PLAN_TABLE表がある場合は、この表を削除します。
 - パッチ6973309を適用してOracle9iでデータベース取得機能を使用できるようにします。詳細は『Metalink Note : 560977.1』を参照してください。
 - OBJ\$ of type=10のエントリすべてを削除します。これらは、過去に作成してから削除されたデータベース・リンクの残りです。

```
delete from obj$ where (name,owner#) in (
  select o.name,u.user# from user$ u, obj$ o
  where u.user# (+)=o.owner# and o.type#=10
  and not exists
  (select p_obj#
   from dependency$
   where p_obj# = o.obj#)
);
```

- アップグレード前に、パッチを使用して新しいタイムゾーン・ファイルV4を適用します。このファイルを適用しない場合は、Oracle Database 11gへのアップグレードを開始できません。2007年に、7つのタイムゾーンで夏時間 (DST) の開始日と終了日が変更されました。このタイムゾーンの定義は、Oracleサーバー内でTIMEZONEデータ型を適切に処理するために必要となります。各リリースの適切なパッチ番号とDSTパッチについてのあらゆる疑問に対する回答は、『Metalink Note : 413671.1』に記載されています。
- Oracle Database 11gのインストールで提供されている、\$ORACLE_HOME/rdbms/adminディレクトリ内のutlu111i.sqlスクリプトを実行します。このスクリプトは、SQL*PlusにユーザーSYS AS SYSDBAとして接続し、ソース・データベースの環境内で実行する必要があります。このスクリプトを実行すると、アップグレー

ドするデータベースのチェックが行われ、必要なパラメータ変更または削除に関する推奨事項が表示されます。

- **SPFILE**を使用している場合は、`CREATE PFILE='...' FROM SPFILE='...'`;のように、編集が可能な**init.ora**ファイルを作成する必要があります。

この**init.ora**ファイルには、**Oracle Database 11g**の要件に適合させるため、推奨された変更すべてを適用する必要があります。また、既存のアンダースコア・パラメータとデータベース・イベントを、初期化ファイルから削除する必要もあります。

Oracle9iデータベースを**Oracle Database 11g**に直接アップグレードする場合、初期化パラメータ**COMPATIBLE**を最低でも**10.0.0**に設定する必要があります。また、**Oracle Database 11g**の新規機能すべてを有効化し、新しいオブティマイザを使用するためには、**COMPATIBLE**を**11.1.0**に設定します。

- 安全上の理由から、`TZ_VERSION=4`の**REGISTRY\$DATABASE**をチェックすることを強くお奨めします。タイムゾーンのパッチが適用されていない**Oracle 9.2**データベースを**Oracle Database 11g**に直接アップグレードした場合、新しい環境では**COMPATIBLE**を高く設定して、データベースを**STARTUP UPGRADE**モードで開く必要があります（**COMPATIBLE**は**10.0.0**以上に設定する必要があります）。この時点では、タイムゾーンのパッチが適用されていないので、アップグレードを正常に行うことはできません。しかし、ダウングレードについては**Oracle Database 10g**への実行しかサポートされていないため、**Oracle 9.2**データベースをクリーンアップしてアップグレード・プロセスを再開することは不可能です。このような場合は、データベース全体をバックアップからリストアする必要があります。
- アップグレード直前に、データ・ディクショナリ全体とデフォルトのユーザー・スキーマすべてを分析します。これにより、アップグレードが大幅に迅速化されます。また、ユーザー**SYS**、**SYSTEM**、**MDSYS**、**DBSNMP**、**OUTLN**、**SI_INFORMTN_SCHEMA**、**ORDPLUGINS**、**ORDSYS**、**LBACSYS**、**WKSYS**、**XDB**、**CTXSYS**、および**WMSYS**がデータベース内に存在する場合は、それぞれに対してこの分析を実行する必要があります。

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SYS',  
options => 'GATHER',  
estimate_percent=>DBMS_STATS.AUTO_SAMPLE_SIZE,  
method_opt => 'FOR ALL COLUMNS SIZE AUTO',  
cascade => TRUE);
```

- アップグレード・プロセス時の停止時間を最小化するため、データベースを**NOARCHIVELOG**モードに切り替えることができます。ただし、アップグレードが正常に完了した後は、再度**ARCHIVELOG**モードに戻す必要があります。
- **Oracle9i**から**Oracle Database 10g/11g**にアップグレードする場合、アップグレード・プロセス時にすべてのシノニムが再コンパイルされます。この処理は、シノニムごとに約**0.1**秒かかります。このため、データベース内で多数のシノニムが作成されている場合

は、すべてのシノニムを再コンパイルするために必要となる時間をアップグレードの時間に追加する必要があります。

- アップグレードを開始する前に、ソース・データベースの完全なオンライン・バックアップを行っておく必要があります。データベースをバックアップするためのツールには、**Oracle Recovery Manager**をお奨めします。処理を進める前に、作成したバックアップを任意の時点でリストアおよびリカバリできることを確認しておいてください。アップグレード・プロセスにおいては、リストアおよびリカバリの操作をフォールバックのシナリオとしてテスト、検証することが極めて重要となります。
- 健全性を維持するために前述の操作を実行する場合、**Oracle Database 11g**ソフトウェアがインストールされている必要があります。また、**Metalink**の推奨パッチに関するセクションを参照すると同時に、『**Metalink Note : 738538.1**』で、発生する可能性のあるアラートと重要な確認事項について把握しておいてください。ターゲット・リリース（アップグレード対象のリリース）で使用可能なパッチ・セットがある場合は、アップグレードの前に適用する必要があります。これは、1回だけ実行する必要がある再コンパイルのための停止時間を短縮することと、**Oracle Database 10g**へのダウングレードが必要となる可能性があることの2つの理由によるものです。
- 新しいリスナーを作成する必要があります。この処理は、**Oracle Network Configuration Assistant (Oracle NETCA)** で実行できます。
- 新しい**Oracle Database 11g**環境に切り替える場合は、**LIBPATH**と**ORA_NLS10**が新しい**\$ORACLE_HOME**を示すように適切に設定されていることを確認します。
- これで、データベースは**STARTUP UPGRADE**モードで起動します。これにより、システム・トリガーとレプリケーションにおける依存性の追跡が無効化され、ORA--904: table or view does not existなどの不必要なメッセージが出力されなくなります。この結果、アップグレード時のエラーに対するspoolファイルを簡単に確認できるようになります。

```
SQL> spool /tmp/upgrade.log
SQL> startup upgrade pfile='<new-init.ora>'
```
- **Oracle9i**データベースを**STARTUP UPGRADE**モードで起動した後は、新しい**SYSAUX**表領域を作成する必要があります。これは、**Oracle9i Database**からのアップグレード時にのみ必要な処理で、**Oracle Database 10g**の場合、**SYSAUX**表領域はすでに提供されています。

```
CREATE TABLESPACE SYSAUX
  DATAFILE '/oradata/sysaux_01.dbf' size 2048M
  EXTENT MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO
  ONLINE;
```
- スクリプト**catupgrd.sql**を実行します。これにより、すべてのデータ・ディクショナリ・オブジェクトと、データベースにインストールしたコンポーネントがすべてアップグレードされます。

```
SQL> @?/rdbms/admin/catupgrd.sql
```

この処理が完了すると、アップグレード・スクリプトによりデータベースが

停止されるので、次にSTARTUP処理が必要となります。
SQL> startup pfile='<new-init.ora>'

- 以前にSPFILEを使用したことがある場合は、SPFILEを新しいinit.oraから作成する必要があります。

```
SQL> create spfile='...' from pfile='<new-init.ora>';
```

- Oracle Database 10gからOracle Database 11gへのアップグレード時には、Automatic Workload Repository (AWR) を新しいデータベース・リリースにアップグレードするためのスクリプトを実行する必要があります。

```
SQL> @?/rdbms/admin/catuppst.sql
```

AWRはOracle Database 10gで初めて導入されたため、Oracle9i Databaseからのアップグレード時にはこの手順を実行する必要はありません。

- アップグレード・プロセスの最後の手順は再コンパイルです。再コンパイルは、システムで使用可能なCPUの数に基づいて、パラレルで自動処理されます。

```
SQL> @?/rdbms/admin/utlprp.sql
```

デフォルトでは、システム内にインストールされているCPUコアの数よりも1つ少ないパラレル・スレッドで再コンパイルされるように設定されています。同一のサーバー上で他のデータベースやアプリケーションを実行している場合は、以下のようにパラレル・スレッドの数をさらに制限すると、有効な場合があります。

```
SQL> @?/rdbms/admin/utlprp 7
```

この例の場合、7つのパラレル・スレッドで再コンパイルのためのパラレル処理が開始されます。

- 最後に、ポスト・アップグレード・ステータス・スクリプトutlul11s.sqlを実行して、すべてのコンポーネントがOracle Database 11gに正常にアップグレードされたことを検証する必要があります。

```
SQL> @?/rdbms/admin/utlul11s.sql
```

```
Oracle Database 11.1 Post-Upgrade Status Tool          08-13-2008 10:07:44

Component                Status      Version  HH:MM:SS
Oracle Server              VALID      11.1.0.6.0 03:06:48
JServer JAVA Virtual Machine  VALID      11.1.0.6.0 00:15:43
Oracle Workspace Manager   VALID      11.1.0.6.0 00:01:38
Oracle XDK                  VALID      11.1.0.6.0 00:35:35
Oracle XML Database        VALID      11.1.0.6.0 00:04:28
Oracle Database Java Packages  VALID      11.1.0.6.0 00:00:33
Oracle Multimedia          VALID      11.1.0.6.0 00:05:49
Spatial                     VALID      11.1.0.6.0 00:08:53
Gathering Statistics       VALID      11.1.0.6.0 00:08:18

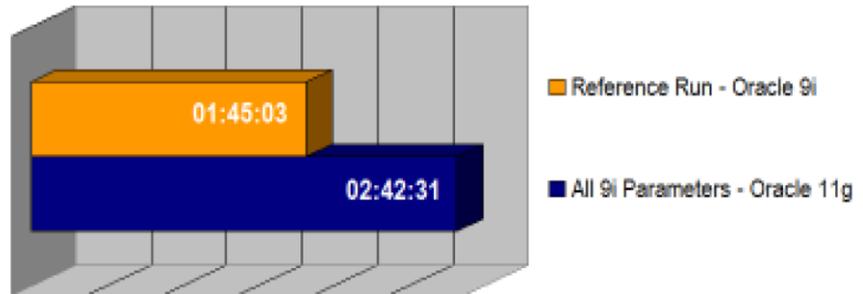
Total Upgrade Time:04:27:49
```

これで、データベースのOracle Database 11gへのアップグレードが正常に完了しました。

テスト実行1 - 11.1.0.6のワークロード - パラメータ変更および新規機能なし

テスト実行1では、diagnostic_destなどの必要なinit.oraパラメータ以外は変更せずに、Oracle9iからOracle Database 11gへのデータベース・アップグレードを行いました。Oracle9i環境で使用されていた特別なアンダースコア・パラメータについても再利用されており、削除されていません。

比較の基準としては、リリース9.2.0.8の本番データベースのクローンに対し、同一システムでの5つの個別テスト実行を平均したリファレンス実行を使用しました。



Oracle9iでの平均実行時間は1時間45分03秒でした。

調整を一切行わないケースとして、Oracle Database 11gへのアップグレード直後に、新しい統計機能やチューニング機能を使用せず、Oracle9i関連のすべての初期化パラメータ（廃止されたり使われなくなったりしているパラメータも含む）を使用してテストを行った結果、ワークロードの完了には2時間42分31秒を要しました。

アップグレードが何の問題もなく行われたとしても、このプロジェクトに対して設定されたパフォーマンス目標を達成するには、なんらかのチューニングが必要なことは明らかです。

パフォーマンスの最適化

本書のケースで設定されていたパフォーマンスの最適化における目標は、夜間のOLTPバッチ処理実行で同程度の完了時間を達成することです。これを達成するため、実行可能なパフォーマンスの最適化として、以下の3つを指定しました。

- 初期化パラメータに関する推奨事項
- SQL Plan Management
- SQL Tuning AdvisorおよびAutomatic SQL Tuning

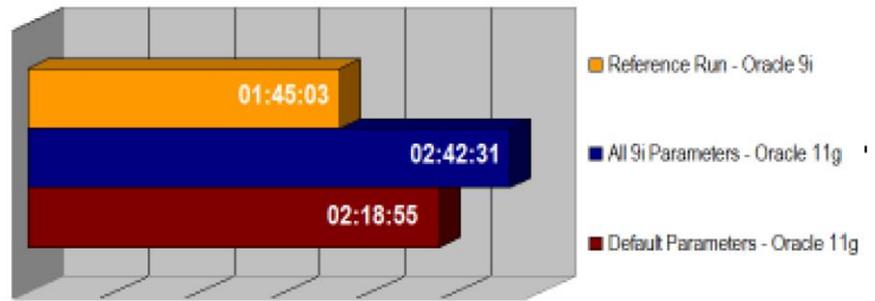
初期化パラメータに関する推奨事項

以下に、このケースでの実施例に基づいて、テスト実行時のinit.oraパラメータ推奨事項の特定と実装について示します。

- 初期化パラメータ・ファイルから、Oracle9i固有のパラメータすべてを削除します。以下は具体的なパラメータの例です。
 - `_ALWAYS_ANTI_JOIN='OFF'`
 - `_ALWAYS_SEMI_JOIN='OFF'`
 - `_SHARED_POOL_RESERVED_MIN_ALLOC=4000`
 - `_UNNEST_SUBQUERY=FALSE`
 - `DISK_ASYNC_IO=FALSE`
 - `SORT_AREA_SIZE=4194304`
- 以下のパラメータをOracle Database 11gで必要な値に変更します。
 - `LOG_ARCHIVE_FORMAT='%r_%t_%s.arc'`
Oracle Database 10gでは、アーカイブ・ログのネーミング形式内でresetlogs IDが指定されるため、'%r' オプションが必要となります。
 - `TIMED_STATISTICS=TRUE`
有益で正確なパフォーマンス情報を得るため、このパラメータは常にTRUEに設定しておきます。
- 新しいパラメータを設定します。
 - `CURSOR_SPACE_FOR_TIME=TRUE`
 - `DIAGNOSTIC_DEST='/ml61/oracle/admin'`
すべてのトレースとダンプに対して、新しいAutomatic Diagnostic Repositoryの場所を指定します。これにより、background_dump_destやuser_dump_destなどのパラメータが置き換えられます。
 - `OPTIMIZER_INDEX_COST_ADJUST=75`
このパラメータについては、索引スキャンではなく全表スキャンが行われるように修正されています。本書のケースでは、ワークロードはCPUバウンドになる傾向があるため、I/Oは問題とはなりません。
 - `OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES=FALSE`
 - `OPTIMIZER_USE_SQL_PLAN_BASELINES=FALSE`
 - `PGA_AGGREGATE_TARGET=1000M`
 - `RECYCLEBIN=OFF`
ごみ箱は、このケースでは本番環境で使用されることはないため、無効化されています。
 - `SESSION_CACHED_CURSORS=500`
 - `SHARED_POOL_SIZE=1250M`
 - `_DISABLE_FLASHBACK_ARCHIVER=1`
Oracle Total Recallは本番環境で使用されないため、バックグラウンド・プロセスFBDAは無効化されています。

テスト実行2 - init.ora ファイル内のOracle Database 11gのデフォルト・パラメータ

第2のテスト実行では、不必要な初期化パラメータすべてをinit.oraファイルから削



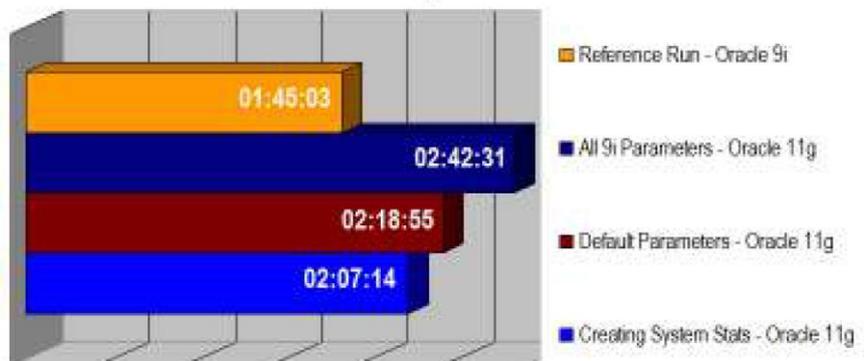
除し、削除しなかったパラメータをOracle Database 11g用に推奨されているデフォルトに設定しました。

この基本手順を実行することで、パフォーマンスは大幅に改善されます。アップグレード後に、以前のリリース固有のデータベース・パラメータを削除することが常に推奨される根拠はここにあります。ただしこのプロジェクトでは、パフォーマンス上の目標を達成するために、他にも作業が必要でした。

このため次の手順として、システム統計を作成し、更新された統計を使用しました。

テスト実行3 - システム統計作成による影響

ワークロード期間内のシステム統計を作成すると、データベース内のシステムのI/Oパフォーマンスについての情報を得ることができます。このシステム統計は、エクスポート、保存、およびテスト・システムへのインポートが可能です。



システム統計は、少なくとも数時間のワークロード期間に基づいて作成する必要があります。

```
exec DBMS_STATS.GATHER_SYSTEM_STATS('start');
```

次に以下を実行します。

```
exec DBMS_STATS.GATHER_SYSTEM_STATS('stop');
```

結果は、SYS.AUX_STATS\$の問合せを行うことで監視できます。

SQL Plan Management

SQL Plan Managementは、Oracle Database 11gの新機能です。SPMには、計画の管理をサポートし、新しいSQL実行計画のパフォーマンス検証を行うために必要なインフラストラクチャとサービスが取り入れられています。オブティマイザでは、このような計画管理を実現するため、1回以上実行された各SQL文の計画履歴が維持されています。SQL計画ベースラインには、実証済みの既知のSQL実行計画のみが保存されます。また、一連のSQL文に対する計画の手動での入力もサポートされています。

- `DBMS_SPM.LOAD_PLANS_FROM_SQLSET('mySQLTuningSet1')`
- `DBMS_SPM.LOAD_PLANS_FROM_CURSOR_CACHE(<sql-id>)`

SQL計画ベースラインとSQL計画履歴は、SQL Management Baseの一部としてSYSAUX表領域内に含まれています。この表領域のサイズと履歴の拡張については、次のように制御できます。

- `DBMS_SPM.CONFIGURE('SPACE_BUDGET_PERCENT',20);`
- `DBMS_SPM.CONFIGURE('PLAN_RETENTION_WEEKS',15);`

SQL計画ベースラインは、以下の2つの初期化パラメータで制御します。

- `optimizer_use_sql_plan_baselines = true | false`
- `optimizer_capture_sql_plan_baselines = true | false`

`optimizer_use_sql_plan_baselines`がデフォルト値のTRUEに設定されている場合は、SQL計画ベースラインに保存されている検証済みの計画または手動でロードされた計画のみが、文を繰り返すために使用されます。

既知の実行計画の転送

SQL実行計画のSQL計画ベースラインへの直接ロードは、データベース・アップグレードのシナリオでは理想的な機能です。この機能により、アップグレードしたデータベースに完全に実証された実行計画を転送できるため、計画でリグレッションが発生するリスクを限りなく少なくできます。

Oracle9i DatabaseからOracle Database 11gへのアップグレード

Oracle9iでは、実行計画が含まれたすべての文をSQL Tuning Set (STS) で収集して保存できます。ただし、アップグレードしたデータベースのSQL計画ベースラインにSQL Tuning Setを直接ロードすることはできません。このため、計画におけるリグレッションのリスクを軽減するために別の対策を講じる必要があります。この場合、オブティマイザのパラメータを再度設定し、繰り返し利用できるSQL文による実行計画をSQL計画ベースラインに移行する機能を使用して、アップグレードしたデータベースを一定時間に渡り実行する必要があります。

- `optimizer_capture_sql_plan_baselines = true`
- `optimzer_features_enable='9.2.0'`

次に、アップグレードしたデータベースでアプリケーションを実行します。ここで重要となるのは高い負荷を生成することではなく、重要な文のすべてが複数回発行されるようにすることです。

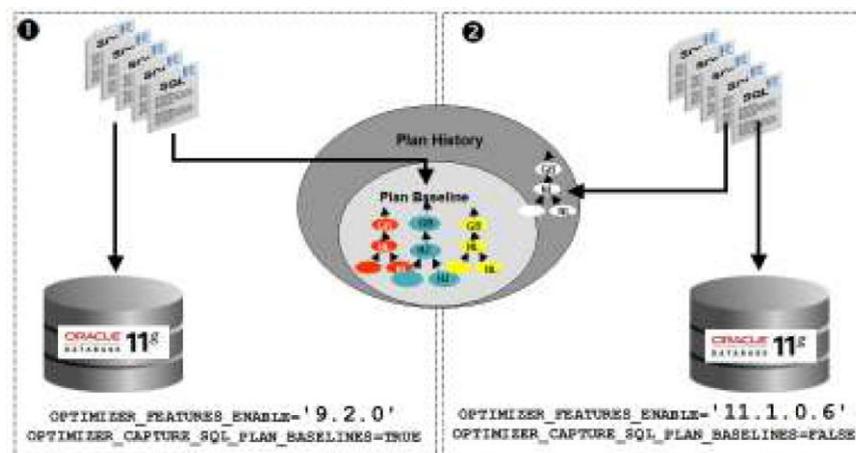


図1 : SQL Plan Managementを使用したOracle9iのアップグレード

第2フェーズでは、オプティマイザのパラメータをリリース11gに一致したレベルに設定しますが、`optimizer_capture_sql_plan_baselines`については無効化します。これにより、新しい実行計画は、SQL計画ベースラインで保存されている計画以上のレベルにあるとみなされ、計画履歴に記録されます。ただし、オプティマイザでは、SQL計画ベースラインに保存されている検証済みの計画が使用されます。この時点で、DBAは計画履歴に記録された計画を検証し、SQL計画ベースラインに含めるか、自動消去の対象とするかを決定します。

- `optimizer_capture_sql_plan_baselines = false`
- `optimzer_features_enable='11.1.0.6'`

Oracle Database 10gからOracle Database 11gへのアップグレード

Oracle Database 10gからOracle Database 11gに移行する場合は、すべての実行計画をSQL Tuning Set内で転送し、アップグレードしたデータベースのSQL計画ベースラインに直接ロードできるため、さらにリスクが減少します。SQL Tuning Setを使用するには、Oracle Tuning PackまたはOracle Real Application Testing Packのライセンスが必要となります。

まず、ソース・データベース上ですべての計画を取得し、STSに転送する必要があります。この処理は、カーソル・キャッシュを取得するか、またはAutomatic Workload Repositoryから文を選択することで実行できます。

```
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('STS102WKLD');
  DBMS_SQLTUNE.CAPTURE_CURSOR_CACHE_SQLSET(
    sqlset_name => 'STS102WKLD',
    time_limit => 120,
    repeat_interval => 5);
END;
/
```

上記の例では、カーソル・キャッシュは120秒間で5回ポーリングされます。その後、実行計画などのSQL文すべてが含まれたSQL Tuning SetのSTS102WKLDをステージング表にパッケージ化する必要があります。

```
BEGIN
  DBMS_SQLTUNE.CREATE_STGTAB_SQLSET (
    table_name => 'STGTAB102');
  DBMS_SQLTUNE.PACK_STGTAB_SQLSET (
    sqlset_name => 'STS102WKLD',
    staging_table_name => 'STGTAB102');
END;
/
```

続いてOracle Data Pumpのエクスポート/インポート機能、またはターゲットとなるOracle Database 11gシステムへのデータベース・リンクを使用して11gに移行します。

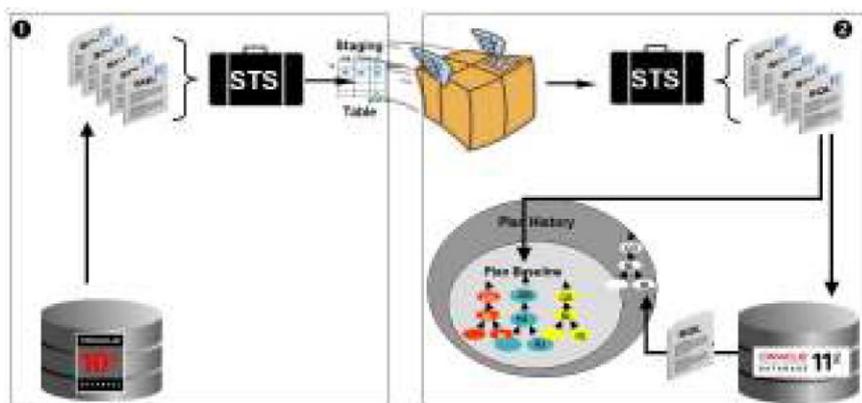


図2 : 10gの実行計画の11gのSQL計画ベースラインへの転送

移行先の11gでアンパックし、そこに含まれている実行計画をSQL計画ベースラインに書き込みます。

```
DECLARE
  my_plans pls_integer;
BEGIN
  DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET (
    sqlset_name => 'STS102WKLD',
    sqlset_owner => '%',
    replace => TRUE,
    staging_table_name => 'STGTAB102');
  my_plans := DBMS_SPM.LOAD_PLANS_FROM_SQLSET (
    sqlset_name => 'STS102WKLD',
    sqlset_owner => 'SYS',
    basic_filter => 'sql_text like ''%'',
    fixed => 'YES',
    enabled => 'YES',
    commit_rows => 1);
END;
/
```

SQL Plan Managementの維持

SQL計画ベースラインを維持しておけば、テスト・システムにロードしたり、別の機会にDatabase Replayを実行するためにリストア済みのデータベースにロードしたりすることで、Oracle Database 11gのパフォーマンスを検証する際にOracle9iと同等の実行計画を使用できるため、役に立つ可能性があります。そこで、ベースラインの計画すべてをステージング表にロードし、そのステージング表をエクスポートします。

```
DECLARE
  spm number;
BEGIN
  DBMS_SPM.CREATE_STGTAB_BASELINE (
    table_name=>'SPM_SAVED',
    table_owner=>'SYSTEM');
  spm := DBMS_SPM.PACK_STGTAB_BASELINE (
    table_name=>'SPM_SAVED',
    table_owner=>'SYSTEM');
END;
/

$ exp system/manager file=bls.dmp tables=SPM_SAVED
```

システムをリセットしてから、DBMS_SPM.UNPACK_STGTAB_BASELINEプロシージャを使用して保存したベースラインをインポートし、再度アンパックします。これで、11.10.6に設定したオプティマイザを使用して取得したワークロードを再生できます。

```
$ imp system/manager file=bls.dmp full=y

DECLARE
  spm number;
BEGIN
  spm := DBMS_SPM.UNPACK_STGTAB_BASELINE (
    table_name=>'SPM_SAVED',
    table_owner=>'SYSTEM');
END;
/
```

結論

SQL Plan Managementは、データベースのアップグレードにおける計画の安定性を実現する高度な新機能です。特にOracle Database 10gからOracle Database 11gへの移行時には、これにより計画におけるリグレッションのリスクがゼロになります。

Oracle Real Application Testingによる最適化と ロード・シミュレーション

Oracle Real Application Testing (Oracle RAT) パックは、以下の2つのコンポーネントで構成されています。

- **SQL Performance Analyzer**
SPAを使用すると、エンドユーザーが影響を受ける前にSQLパフォーマンスの逸脱を予測できます。SPAでは、SQL Tunings Setに保存された各SQL文を本番コンテキストで個別に実行し、実行計画とランタイム統計を実行する‘前’と‘後’の比較を行います。SPAは、パフォーマンスが改善または低下したSQL文のセットを特定することを目的としています。
- **Database Replay**
Database Replayを使用すると、本番システムにおける実際のワークロードを取得して、同時実行性、同期、および依存性などの本番特性を使用してテスト・システム上で再生できます。これにより、ワークロードのスループット変更による影響評価が可能となります。Database Replayの目的は、データベース・サーバーのサブシステムすべてに対する包括的なテストを、本番環境における実際のワークロードを使用して実行することです。

Oracle Real Application Testingのライセンスには、SQL Tuning Setのライセンスも含まれています。

以下の2つの項では、データベース・アップグレード・プロジェクトにおけるSQL Performance AnalyzerとDatabase Replayの使用と利点について説明します。

SQL Performance Analyzer - 変更された実行計画の簡単な検出を可能にするツール

アップグレード・プロジェクトにおける最大の課題の1つは、リリース変更後に既知のSQL文がどのように実行されるかを特定することです。SPAは、パフォーマンスが改善または低下するSQL文と新しいリリースで変更される実行計画を、アップグレード前に特定するための最適なツールです。処理を行うにあたっては、現在のデータベース・システムのSQL文すべてを取得する必要があります。

テストの目的

SQL Performance Analyzerを使用する第1の目的は、新しいリリースで変更される実行計画を特定し、簡単なパフォーマンス比較メカニズムを構築することです。

テストの設定

テスト評価の基盤には、完全な夜間バッチ処理の実行を使用します。この実行が、Oracle 9.2.0.8本番データベースのクローンに対してテストを実行する際のテスト基準となります。必要な情報はすべてSQLトレース・イベント10046を使用して収集します。このトレース情報については、オブジェクト・マッピング表に従って転送および前処理を行います。前処理が終わった後は、この情報を必要に応じて再利用できます。また、設定とパラメータの変更、SQL Profileの作成、およびSQL Plan Managementの使用が可能なおうえ、すべての変更について即座に検証を行うこともできます。

SQL文の取得

まず、統計のタイミングを有効化する必要があります。この処理は、初期化パラメータ・ファイルで設定するか、以下のスクリプトを発行することで実行できます。

```
ALTER SYSTEM SET TIMED_STATISTICS=TRUE;
```

Oracle Database 9.2.0.8におけるSQL文の取得は、SQLのトレース機能を使用して実行します。この場合、DBMS_SUPPORT (『Metalink Note:62294.1』を参照)を使用するか、または以下のようにイベント10046を使用して、すべてのバインド変数も収集します。

```
ALTER SYSTEM SET EVENTS '10046 TRACE NAME CONTEXT  
FOREVER, LEVEL 4';
```

これで、すべての新しく接続されたセッションに対するSQLがすべて取得され、トレース・ファイル内に格納されます。トレース機能は、以下のスクリプトを使用することによって、後で無効化できます。

```
ALTER SYSTEM SET EVENTS '10046 TRACE NAME CONTEXT OFF';
```

注：SQLトレースを開始する前に、USER_DUMP_DESTで指定されたディレクトリを空にしておくことを強くお奨めします。これは、このプロセスの一部として収集されるトレース情報が膨大な量になる可能性があるためです。本書のケースでは、約60～80GBのトレース・ファイルがディスクに書き込まれました。収集されるデータの量は、データベースとワークロードによって異なります。

また、SQLトレースの他にも、マッピング表を作成し、エクスポートまたはインポートを介して評価データベースに転送する必要があります。

```
CREATE TABLE MAPPING_TABLE AS
  SELECT OBJECT_ID ID, OWNER,
         SUBSTR(OBJECT_NAME, 1, 30) NAME
  FROM DBA_OBJECTS
  WHERE OBJECT_TYPE NOT IN
        ('CONSUMER GROUP', 'EVALUATION
        CONTEXT', 'FUNCTION', 'INDEXTYPE',
        'JAVA CLASS', 'JAVA DATA', 'JAVA
        RESOURCE', 'LIBRARY', 'LOB',
        'OPERATOR', 'PACKAGE', 'PACKAGE
        BODY', 'PROCEDURE', 'QUEUE',
        'RESOURCE PLAN', 'SYNONYM',
        'TRIGGER', 'TYPE', 'TYPE BODY')

UNION ALL
  SELECT USER_ID ID, USERNAME OWNER, NULL NAME
  FROM DBA_USERS;
```

SQL Tuning Setへの文のロード

11gデータベースにインポートしたマッピング表は、9iのトレース情報を11gのSTSに合わせて調整するために使用します。

```
DECLARE
  syscur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('SPA_STS');
  OPEN syscur FOR
    SELECT VALUE(p) FROM
      TABLE(DBMS_SQLTUNE.SELECT_SQL_TRACE(
        directory => 'SPA_DIR',
        file_name => '%ora%',
        mapping_table_name => 'MAPPING_TABLE',
        select_mode=>dbms_sqltune.single_execution)) p;
  DBMS_SQLTUNE.LOAD_SQLSET(
    sqlset_name => 'SPA_STS',
    populate_cursor => syscur,
    commit_rows => 5);
  CLOSE syscur;
END;
/
```

収集されたトレース・ファイルが多い場合は、ファイルを個別のサブディレクトリに均等に分割してパラレルでロードすることをお奨めします。一般に、トレース・ファイルを分配するディレクトリの数は、サーバー内のCPUよりも1つ少ない数とするのが理想的です。

本書のケースでは、2,760に及ぶ個別の文すべてをSTSにロードする時間が、パラレル・ロードにより、9.5時間から1.5時間へと短縮されました。この方法を使用してパラレルでSTSを移入した場合、ロード終了時点でマージ処理が必要となります。

```
DECLARE
  cur DBMS_SQLTUNE.SQLSET_CURSOR;
BEGIN
  DBMS_SQLTUNE.CREATE_SQLSET('SPA_MAIN_STS');
```

```

OPEN cur FOR
  SELECT VALUE(p)
    FROM TABLE (
      DBMS_SQLTUNE.SELECT_SQLSET('SQLSET1')) p;
DBMS_SQLTUNE.LOAD_SQLSET (
  sqlset_name => 'SPA_MAIN_STS',
  populate_cursor => cur,
  load_option => 'MERGE',
  update_option => 'ACCUMULATE');
END;
/

```

この操作はSQLSET1からSQLSETnまでのすべてに対して実行する必要があります。
以下の問合せを使用すると、STSにすでにロードされている一意のSQL文の数を監視できます。

```

SELECT STATEMENT_COUNT
  FROM DBA_SQLSET
 WHERE NAME='SPA_STS';

```

また、ビューDBA_SQLSET_STATEMENTSに対する問合せを行い、STSにロードされた特定のSQLに関する詳細を監視することもできます。

SQL Tuning Setの維持

SQL Tuning Setへの移入を実行するのは、1回のみとする必要があります。このため、このSTSを、エクスポート可能なステージング表に保存する必要があります。テスト・システムを特定の時点に設定できる場合は、保存したSTSを必要に応じて何度でも再利用できます。

```

-- Create the staging table
-- Please note that the name of the mapping table
-- currently cannot exceed 19 characters
BEGIN
  DBMS_SQLTUNE.CREATE_STGTAB_SQLSET (
    table_name => 'SPA_STS_STGTAB' );
END;
/
-- Load the SQL Set into the mapping table
BEGIN
  DBMS_SQLTUNE.PACK_STGTAB_SQLSET (
    sqlset_name => 'SPA_STS',
    staging_table_name => 'SPA_STS_STGTAB');
END;
/

```

これで、ステージング表SPA_STS_STGTABをエクスポートして保存場所に格納し、いつでもテスト・システムに再インポートできます。インポートしたステージング表は、以下のスクリプトによりアンパックできます。

```

BEGIN
  DBMS_SQLTUNE.UNPACK_STGTAB_SQLSET (
    sqlset_name => '%',
    replace => TRUE,
    staging_table_name => 'SPA_STS_STGTAB');
END;
/

```

SPA分析タスクの作成とパラメータ化

SPAを使用してSTSに保存されたSQL文を分析する場合は、まず分析タスクを作成する必要があります。

```
DECLARE
  tname VARCHAR2(100);
BEGIN
  tname := DBMS_SQLPA.CREATE_ANALYSIS_TASK(
    sqlset_name=>'SPA_STS',
    task_name=>'SPA_TASK_9i_11g',
    description=>'Move on from 9i to 11g');
END;
/
```

分析タスクでは、以下のようにパラメータを設定し、分析の制御で使用できるようにする必要があります。

```
BEGIN
  DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(
    task_name => 'SPA_TASK_9i_11g',
    parameter => 'workload_impact_threshold',
    value => 0);
  DBMS_SQLPA.SET_ANALYSIS_TASK_PARAMETER(
    task_name => 'SPA_TASK_9i_11g',
    parameter => 'sql_impact_threshold',
    value => 10);
end;
/
```

WORKLOAD_IMPACT_THRESHOLDは、SPAによりパフォーマンスの改善または低下のいずれかのラベルが付けられる前に、ワークロードに影響するSQLについての最小のしきい値を、パーセント値として定義するものです。SPA用として使用している9iのSTSには実行頻度の情報が含まれていないため、このパラメータはゼロに設定してかまいません。

SQL_IMPACT_THRESHOLDは、SPAによりパフォーマンスの改善または低下のいずれかのラベルが付けられる前に、影響を受けるSQLの各実行パフォーマンスについての最小しきい値を、パーセント値として（ワークロードに対する影響のしきい値に加えて）設定するものです。このパラメータのデフォルト値は1です。ただし、本書のケースでは、関係のないわずかなパフォーマンスの問題に関するレポートによりデータが多くなりすぎることを避けるために、10に設定されています。この2つのしきい値を使用することで、個々のパフォーマンスに10%以上の増減が見られるSQLすべてに対して検出された改善とリグレーションがSPAにおいて正当化され、残りのワークロードは無視されることとなります。

Oracle9iのパフォーマンス・データによるSPA追跡の実行

Oracle9iデータベースからSTS内のSQL情報処理を行う場合、STSからSPAタスクに統計をコピーすることになります。つまり、SQL文は実行されないため、この操作は極めて軽く、迅速に完了できます。

```
BEGIN
  DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
    task_name => 'SPA_TASK_9i_11g',
    execution_name => 'EXEC_SPA_TASK_9i',
    execution_type => 'CONVERT_SQLSET',
    execution_desc => 'Convert 9i Workload');
END;
/
```

Oracle Database 11gのパフォーマンス・データによるSPA追跡の実行

Oracle Database 11gのパフォーマンス・データを使用してSPA追跡を作成した場合は、個々の文が新しく準備したテスト・データベースで実行されます。

```
BEGIN
  DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
    task_name => 'SPA_TASK_9i_11g',
    execution_name => 'EXEC_SPA_TASK_11g',
    execution_type => 'TEST_EXECUTE',
    execution_desc => 'Test 9i Workload in 11g');
END;
/
```

この手順では、実際にすべての文が実行されるため、リソースを大量に消費することになります。これについては、以下の問合せを実行することで監視できます。

```
select sofar, totalwork from v$advisor_progress
where task_id = <tid>;
```

レポート/比較

タスク実行後は、次のような複数のメトリックに基づいて結果を比較できます。

PARSE_TIME, ELAPSED_TIME, CPU_TIME, USER_IO_TIME, BUFFER_GETS, DISK_READS, DIRECT_WRITES and OPTIMIZER_COST.

```
BEGIN
  DBMS_SQLPA.EXECUTE_ANALYSIS_TASK(
    task_name => 'SPA_TASK_9i_11g',
    execution_name => 'Compare 9i 11g CPU_TIME',
    execution_type => 'COMPARE PERFORMANCE',
    execution_params =>
      DBMS_ADVISOR.ARGLIST(
        'comparison_metric',
        'cpu_time',
        'execution_name1','EXEC_SPA_TASK',
        'execution_name2','TEST 9i 11g CPU'),
    execution_desc => 'Compare 9i to 11g CPU_TIME');
END;
/
```

この比較に関するレポートを表示するには、以下を実行します。

```
SELECT
  xmltype(DBMS_SQLPA.REPORT_ANALYSIS_TASK
    ('SPA_TASK_9i_11g', 'html', 'typical', 'summary',
    null, 350, 'Compare 9i to 11g CPU_TIME'))
  .getclobval(2,2)
FROM dual;
```

この操作は、さまざまなメトリックに対して実行できます。CPU_TIMEおよびBUFFER_GETSの2つは、Oracle9iのデータを使用してテストを実行する場合にもっとも重要な意味を持つメトリックです。

SQL Performance Analyzerの結果

このプロジェクトでは、パフォーマンス評価フェーズ全体をとおしてSQL Performance Analyzerを使用しました。SPAでは、ソース・データベースからロードされた文は、その後の処理で必要に応じて何度でも繰り返すことができます。

これにより、結果を検証するための処理がSPAタスクの再実行だけで済むため、SPAはチューニング時に非常に役立つツールとなっています。

以下に、SPAによるオブティマイザのパラメータ最適化に関する例を示します。ここでは、異なる初期化パラメータ・セットを使用してSPAタスクを繰り返し再実行することで、最善の設定を特定しました。この方法を行うには、アプリケーションとSQL文に関する知識が必要となるためご注意ください。

本書のケースでは、特にパフォーマンスが低下した高負荷の実行計画に対し、SPAレポートでの詳細なチェックが行われました。ここで使用された評価用のSQL Tuning Setには、2,397の一意のSQL文が含まれています。BUFFER_GETSは、このワークロードに対しては比較的重要ではないため、最適化はCPU_TIMEを対象として行われました。

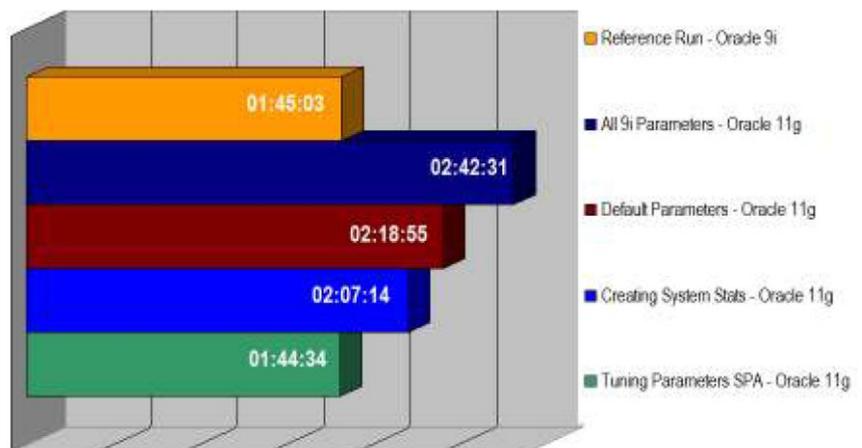
また、_optimizer_cost_based_transformation=offおよび_new_initial_join_order=falseなどの、10g Release 2の推奨パラメータ処理についての検証も実行されました。ただし、ここでの検証結果では、これらのパラメータはOracle Database 11gでは必要がなく、使用し続ければかえって実行結果の悪化につながる事が明らかとなっています。

Reports on CPU_TIME	Report on BUFFER_GETS																														
optimizer_features_enable=9.2.0																															
<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>2127</td> </tr> <tr> <td>Improved</td> <td>750</td> <td>734</td> </tr> <tr> <td>Regressed</td> <td>302</td> <td>260</td> </tr> <tr> <td>Unchanged</td> <td>1235</td> <td>1133</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	2127	Improved	750	734	Regressed	302	260	Unchanged	1235	1133	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>2127</td> </tr> <tr> <td>Improved</td> <td>1219</td> <td>1125</td> </tr> <tr> <td>Regressed</td> <td>116</td> <td>80</td> </tr> <tr> <td>Unchanged</td> <td>952</td> <td>922</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	2127	Improved	1219	1125	Regressed	116	80	Unchanged	952	922
SQL Category	SQL Count	Plan Change Count																													
Overall	2397	2127																													
Improved	750	734																													
Regressed	302	260																													
Unchanged	1235	1133																													
SQL Category	SQL Count	Plan Change Count																													
Overall	2397	2127																													
Improved	1219	1125																													
Regressed	116	80																													
Unchanged	952	922																													
optimizer_features_enable=11.1.0.6																															
<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1704</td> </tr> <tr> <td>Improved</td> <td>775</td> <td>640</td> </tr> <tr> <td>Regressed</td> <td>207</td> <td>173</td> </tr> <tr> <td>Unchanged</td> <td>1305</td> <td>891</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1704	Improved	775	640	Regressed	207	173	Unchanged	1305	891	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1704</td> </tr> <tr> <td>Improved</td> <td>1139</td> <td>899</td> </tr> <tr> <td>Regressed</td> <td>170</td> <td>143</td> </tr> <tr> <td>Unchanged</td> <td>978</td> <td>662</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1704	Improved	1139	899	Regressed	170	143	Unchanged	978	662
SQL Category	SQL Count	Plan Change Count																													
Overall	2397	1704																													
Improved	775	640																													
Regressed	207	173																													
Unchanged	1305	891																													
SQL Category	SQL Count	Plan Change Count																													
Overall	2397	1704																													
Improved	1139	899																													
Regressed	170	143																													
Unchanged	978	662																													
_optimizer_cost_based_transformation=off																															
<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1699</td> </tr> <tr> <td>Improved</td> <td>757</td> <td>632</td> </tr> <tr> <td>Regressed</td> <td>245</td> <td>176</td> </tr> <tr> <td>Unchanged</td> <td>1285</td> <td>891</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1699	Improved	757	632	Regressed	245	176	Unchanged	1285	891	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1699</td> </tr> <tr> <td>Improved</td> <td>1058</td> <td>821</td> </tr> <tr> <td>Regressed</td> <td>156</td> <td>122</td> </tr> <tr> <td>Unchanged</td> <td>1073</td> <td>756</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1699	Improved	1058	821	Regressed	156	122	Unchanged	1073	756
SQL Category	SQL Count	Plan Change Count																													
Overall	2397	1699																													
Improved	757	632																													
Regressed	245	176																													
Unchanged	1285	891																													
SQL Category	SQL Count	Plan Change Count																													
Overall	2397	1699																													
Improved	1058	821																													
Regressed	156	122																													
Unchanged	1073	756																													

<code>_new_initial_join_order=false</code>	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1699</td> </tr> <tr> <td>Improved</td> <td>771</td> <td>639</td> </tr> <tr> <td>Regressed</td> <td>205</td> <td>155</td> </tr> <tr> <td>Unchanged</td> <td>1311</td> <td>905</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1699	Improved	771	639	Regressed	205	155	Unchanged	1311	905	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1699</td> </tr> <tr> <td>Improved</td> <td>1058</td> <td>821</td> </tr> <tr> <td>Regressed</td> <td>155</td> <td>122</td> </tr> <tr> <td>Unchanged</td> <td>1074</td> <td>756</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1699	Improved	1058	821	Regressed	155	122	Unchanged	1074	756
SQL Category	SQL Count	Plan Change Count																														
Overall	2397	1699																														
Improved	771	639																														
Regressed	205	155																														
Unchanged	1311	905																														
SQL Category	SQL Count	Plan Change Count																														
Overall	2397	1699																														
Improved	1058	821																														
Regressed	155	122																														
Unchanged	1074	756																														
<code>optimizer_index_cost_adj=75</code>	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1174</td> </tr> <tr> <td>Improved</td> <td>827</td> <td>649</td> </tr> <tr> <td>Regressed</td> <td>181</td> <td>100</td> </tr> <tr> <td>Unchanged</td> <td>1279</td> <td>425</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1174	Improved	827	649	Regressed	181	100	Unchanged	1279	425	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1174</td> </tr> <tr> <td>Improved</td> <td>1175</td> <td>881</td> </tr> <tr> <td>Regressed</td> <td>142</td> <td>110</td> </tr> <tr> <td>Unchanged</td> <td>970</td> <td>183</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1174	Improved	1175	881	Regressed	142	110	Unchanged	970	183
SQL Category	SQL Count	Plan Change Count																														
Overall	2397	1174																														
Improved	827	649																														
Regressed	181	100																														
Unchanged	1279	425																														
SQL Category	SQL Count	Plan Change Count																														
Overall	2397	1174																														
Improved	1175	881																														
Regressed	142	110																														
Unchanged	970	183																														
<code>optimizer_mode=first_rows_10</code>	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1707</td> </tr> <tr> <td>Improved</td> <td>821</td> <td>687</td> </tr> <tr> <td>Regressed</td> <td>158</td> <td>126</td> </tr> <tr> <td>Unchanged</td> <td>1308</td> <td>894</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1707	Improved	821	687	Regressed	158	126	Unchanged	1308	894	<p>SQL Statement Count</p> <table border="1"> <thead> <tr> <th>SQL Category</th> <th>SQL Count</th> <th>Plan Change Count</th> </tr> </thead> <tbody> <tr> <td>Overall</td> <td>2397</td> <td>1707</td> </tr> <tr> <td>Improved</td> <td>1085</td> <td>845</td> </tr> <tr> <td>Regressed</td> <td>145</td> <td>118</td> </tr> <tr> <td>Unchanged</td> <td>1057</td> <td>744</td> </tr> </tbody> </table>	SQL Category	SQL Count	Plan Change Count	Overall	2397	1707	Improved	1085	845	Regressed	145	118	Unchanged	1057	744
SQL Category	SQL Count	Plan Change Count																														
Overall	2397	1707																														
Improved	821	687																														
Regressed	158	126																														
Unchanged	1308	894																														
SQL Category	SQL Count	Plan Change Count																														
Overall	2397	1707																														
Improved	1085	845																														
Regressed	145	118																														
Unchanged	1057	744																														

テスト実行4 - SPAによるinit.oraの最適化

SPAは、init.oraの最適化を実行するための理想的なツールです。SPAを使用した場合、ロードの実行が1回で済むうえ、SPAタスクを何度でも繰り返すこともできます。



このケースでは、SPAにより、ワークロード処理の最適化用として以下のパラメータが検出されました。

```
optimizer_index_cost_adj=75  
optimizer_mode=first_rows_10
```

これにより、実行時間が最適なものとなり、この機能検証で初めて、Oracle9iのワークロード完了を上回る結果が得られました。

SQL Performance Analyzerに関する結論

SQL Performance Analyzerは、アップグレードを実行して各種の問題解決策に対する評価を行う前に、計画におけるリグレッションを検知できる優れたツールです。SPAは、ソース・システムに対するパッチ・アプリケーションを必要としないため、使用するために本番システムで余分な停止時間が発生することはなく、実際のデータが変更されることもありません。すべての文をSQL Tuning Setに収集し、ターゲット・システムに転送すると、特別な処理をしなくても、それらの文の評価を繰り返し実行できるようになります。

大きな問題ではありませんが、現在SPAには、各文の本来の実行頻度が考慮されないという弱点があります。このため、個々の文のベンチマーク評価をワークロード全体との比較に基づいて行うことはできません。

Database Replay

データベースの取得および再生は、Oracle Database 11gで導入された機能です。このオプションを使用すると、データベースにおける実行ワークロード全体を取得し、前処理を行った後に、任意のプラットフォーム上にあるOracle Database 11gでそのワークロードを再生できます。取得は、Oracle 9.2.0.8、Oracle Database 10.2.0.2、10.2.0.3、10.2.0.4、またはOracle Database 11.1環境で実行できますが、10.2.0.4より前のリリースのデータベースで取得機能を有効化するにはパッチが必要となります。必要なパッチの番号は、『Metalink Note : 560977.1』に記載されています。

Oracle Database 10.2.0.4では、ワークロードの取得機能を有効化するには、次のスクリプト

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

を実行する必要があります。これにより、PRE_11G_ENABLE_CAPTURE初期化パラメータが自動的にTRUEに設定されます。取得機能は、初期化パラメータPRE_11G_ENABLE_CAPTUREをFALSEに変更することで、いつでも無効化できます。

テストの目的

Database Replayを使用する第1の目的は、新しいシステムで夜間バッチの完全なロードがどのように実行されるかを特定することです。このため、Oracle 9.2.0.8データベースで実行したときのワークロード全体を取得する必要があります。取得したワークロードは、前処理の完了後、Oracle Database 11g環境で評価を行うために、必要に応じて使用できます。ただし、取得したワークロードによって実際にデータが変更されるため、アップグレードしたデータベースを、完全なワークロードの処理を繰り返し行える状態にリストアすることが重要となります。

テストの設定

テスト評価の基盤には、完全な夜間バッチ処理の実行を使用します。この実行が、Oracle 9.2.0.8本番データベースのクローンに対してテストを実行する際のテスト基準となります。必要な情報はすべて、取得用のバイナリ・ファイル内に格納されます。これらのファイルに対しては、前処理を1回行う必要があります。前処理の終わったファイルは再利用可能でプラットフォームに依存しないため、データベース評価のためだけでなく、OSの機能や、ストレージ・サブシステム、または運用環境に対するその他の変更の評価にもワークロード情報を使用できます。

取得したファイルを再生するには、Workload Replay Client (WRC) をOSレベルで起動する必要があります。データベースは、記録されたワークロードのうち正常に処理が完了した特定の時点にリセットする必要があります。この処理には、以下の複数のオプションを使用できます。

- Oracle RMANによるアップグレード・データベースのバックアップ、および再生処理ごとのリストア
 - 利点
 - 十分にテストが行われており、テストを実行するために追加のオーバーヘッドを必要としない
 - 短所
 - 完了までに時間がかかる (約1時間)

- フラッシュバック・データベースの有効化 - この処理を行う場合、アーカイブ・ログ・モードに切り替えて、保証されたリストア・ポイントを作成する必要があります。これによってワークロードの再生が可能となり、その後、保証されたリストア・ポイントにデータベースをフラッシュバックできるようになります。
 - 利点
設定が簡単で極めて高速
 - 短所
テスト・ロード時にフラッシュバック・ログ作成のための追加のオーバーヘッド、および追加のディスク容量が必要（推定容量：アーカイブ・ログ作成と同程度）
- スナップショット・スタンバイ - フィジカル・スタンバイ・データベースが、スナップショット・スタンバイ・データベースに一時的に変換されます。ワークロード再生後は、元のフィジカル・スタンバイに変換されます。その後、Oracle Data Guardにより、本番データベースの変更すべてがスタンバイに同期されます。
 - 利点
設定が簡単で極めて高速
 - 短所
テストのために使用できるフィジカル・スタンバイ・データベースが必要。本書で取り上げているプロジェクトでは、使用可能なフィジカル・スタンバイ・データベースがありませんでした。このようなケースでデータベースを取得実行の時点でロールバックする場合は、完全なワークロード再生のみが可能となります。

ワークロードの取得

ワークロードの取得を正常に行うには、空の取得用ディレクトリを定義して、取得を開始する前にデータベースを制限付きモードで起動する必要があります。これにより、取得プロセスを有効化して、すべてのユーザー・コールを取得できるようになります。取得を開始すると、その直後にデータベースは無制限モードに自動的に切り替わり、ユーザーが接続できるようになります。

```
CREATE OR REPLACE DIRECTORY RAT_DIR as '/oracle/rat';

STARTUP RESTRICTED;
```

特定のユーザーを除外したり、含めたりする場合は、取得に対してフィルタを追加できます。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.ADD_FILTER(
    fname => 'FILTER_FOR_USER_SYS',
    fattribute => 'USER',
    fvalue => 'SYS');
END;
/
```

以下のスクリプトにより、ワークロード取得を開始して監視できます。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.START_CAPTURE(
    name => 'RAT_9208',
    dir => 'RAT_DIR');
END;
/
SELECT * FROM DBA_WORKLOAD_CAPTURES;
```

特定の期間のみの実行に制限されていない場合、以下を実行することでワークロードの取得を停止できます。

```
EXECUTE DBMS_WORKLOAD_CAPTURE.FINISH_CAPTURE;
```

Oracle Database 10gデータベースまたはOracle Database 11gデータベースでの取得が完了したら、後で行う比較のためにAutomatic Workload Repositoryをエクスポートする必要があります。

```
BEGIN
  DBMS_WORKLOAD_CAPTURE.EXPORT_AWR(
    capture_id => 5);
END;
/
```

前処理

取得したワークロードをオペレーション・システムに転送して再生できるようにするには、前処理を行う必要があります。

```
EXECUTE DBMS_WORKLOAD_REPLAY.PROCESS_CAPTURE(
  capture_dir => 'RAT_DIR');
```

ワークロードの再生

ワークロードの再生は、Oracle Database 11gシステムでのみ実行可能な機能です。ワークロードの再生を正常に行うには、再生開始時に、ターゲット・データベースを取得開始と同じ時点にリセットする必要があります。このリセット処理を行わない場合は、コールがすべて成功しなくなる可能性があります。エクスポートしたAWRスナップショットは、後で行う比較のためにインポートする必要があります。

```
DECLARE db_id number;
BEGIN
  db_id := DBMS_WORKLOAD_CAPTURE.IMPORT_AWR(
    capture_id => 5,
    staging_schema => 'TEST');
END;
/
```

次に、有効な現在の統計とAWRスナップショットを作成する必要があります。

```
exec dbms_stats.gather_dictionary_stats;
exec dbms_stats.gather_fixed_objects_stats;
exec dbms_stats.gather_system_stats('start');
exec dbms_stats.gather_system_stats('stop');
```

新しいオブジェクト統計を作成します。

```
exec dbms_auto_task_immediate.gather_optimizer_stats;
```

次のスクリプトにより、ジョブを監視できます。

```
SELECT job_name,state
FROM dba_scheduler_jobs
WHERE program_name='GATHER_STATS_PROG';
```

現在の統計を作成すると、再生の初期化が可能となります。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.INITIALIZE_REPLAY(
    replay_name => 'TEST_REPLAY1',
    replay_dir => 'RAT_DIR');
END;
/
```

Workload Replay Clientに対し、オペレーティング・システムのレベルでキャリブレーションを実行する必要があります。これにより、開始するWRC数を予測できます。

```
$ wrc mode=calibrate replaydir=/oracle/rat

Recommendation:
Consider using at least 1 clients divided among 1 CPU(s).

Workload Characteristics:
- max concurrency: 23 sessions
- total number of sessions: 343

Assumptions:
- 1 client process per 50 concurrent sessions
- 4 client process per CPU
- think time scale = 100
- connect time scale = 100
- synchronization = TRUE
```

再生は、以下のモードでパラメータ化することができます。

- `SYNCHRONIZATION = TRUE | FALSE`
取得したワークロードでのCOMMIT命令が使用されます。つまり、依存するCOMMITが完了して初めて、アクションが実行されます。
- `CONNECT_TIME_SCALE = 0 [%] - 100 [%]`
ワークロードの取得が開始された時点からセッションが接続されるまでの経過時間を指定する、オプションのパラメータです。このパラメータはパーセント値として読み取られ、ユーザー数の増減に対して使用されます。
- `THINK_TIME_SCALE = 0 [%] - 100 [%]`
2件のユーザー・コール間の経過時間を制御するオプションのパラメータです。この値を0に設定すると、再生速度が最速になります。
- `THINK_TIME_AUTO_CORRECT = TRUE | FALSE`
再生時のユーザー・コールの完了時間が取得時よりも長い場合に、コール間のTHINK_TIME_SCALEを修正するオプションのパラメータです。

```
BEGIN
  DBMS_WORKLOAD_REPLAY.PREPARE_REPLAY(
    SYNCHRONIZATION => TRUE);
```

```
END;  
/
```

この時点で、データベースは、Workload Replay Clientが接続されるのを待機している状態にあります。Workload Replay Clientは、コマンドライン・プロンプトで開始する必要があります。開始するwrcクライアント数は、以前に実行したキャリブレーションにより提案されます。

```
$ wrc userid=system password=mypwd  
replaydir=/oracle/rat
```

次に、実際のワークロードの再生を開始します。

```
BEGIN  
  DBMS_WORKLOAD_REPLAY.START_REPLAY;  
END;  
/
```

この処理は、Oracle Enterprise Manager Database Controlか、またはDBA_WORKLOAD_REPLAYSのビューで監視できます。

```
SELECT id, name, status,  
       start_time, end_time, num_clients  
FROM dba_workload_replays;
```

ワークロードの再生を取り消すには、以下を実行します。

```
BEGIN  
  DBMS_WORKLOAD_REPLAY.CANCEL_REPLAY ();  
END;  
/
```

レポート作成

統計全体を再生するスクリプトは、次のようになります。

```
set long 10000000  
set lines 128  
column output format a120  
set trimspool on  
spool report_replay_text.txt  
set heading off  
set pages 0  
select dbms_workload_replay.report(&&1,'TEXT') output from dual;  
spool off  
  
set heading off  
set pages 0  
set long 1000000000  
spool /tmp/replay_&&1.html  
select dbms_workload_replay.report(&&1,'HTML') from dual;  
spool off
```

結論

Database Replayは、パッチとリリース全体のどちらのアップグレードに対しても、アプリケーション・レベルでの機能の完全性を検証できる、極めて役に立つ機能です。この機能により、取得したワークロードの再実行によってターゲット・システムを実行する方法についての有益なヒントが数多く提供されます。また、この機能を使用すると、データベースの評価が可能となるだけでなく、システムやストレージ・パフォーマンスを現在の設定と比較して評価することもできます。

本書のケースでは、Database Replayにより、Oracle Database 11gにアップグレードしてもアプリケーションの変更は必要ないことが確認されました。Oracle8iからOracle9iへの移行時など、過去のアップグレード処理では、アプリケーションの変更が必要でした。

このように、Database Replayの結果を利用することで、安心して新しいリリースへの移行を行うことができますようになります。

PL/SQLのネイティブ・コンパイル

本来Oracle PL/SQLはインタープリタ型の言語ですが、Oracle9i以降では、ネイティブ・コンパイルも使用可能となりました。また、Oracle Database 11gからは、外部のコンパイラは必要なくなり、ネイティブ・コンパイルの使用もより効率的で簡単になっています。初期化パラメータは、以下のようにシンプルな設定となります。

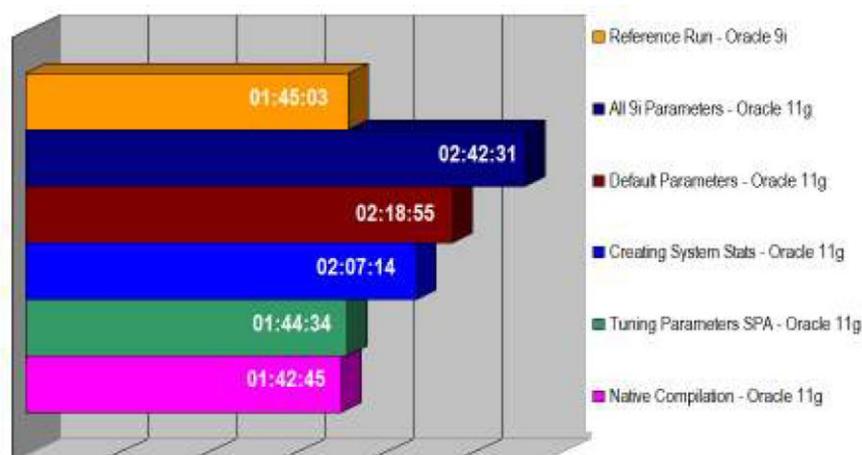
```
plsql_code_type=native  
plsql_optimization_level=3
```

これだけで初期化パラメータは適切な値に設定され、以下のプロシージャのスキーマを使用して、PL/SQLコードのスキーマを再コンパイルできるようになります。

```
exec DBMS_UTILITY.COMPILE_SCHEMA('<username>');
```

テスト実行5 - PL/SQLのネイティブ・コンパイル

本書のケースでは、PL/SQLプロシージャおよびパッケージが含まれたデータベース内のスキーマすべてに対し、再コンパイルを行いました。再コンパイル処理の実行後は、ネイティブ・コンパイルなしの場合と比べてワークロードの完了時間が約2分短縮されました。



結論

本書で取り上げている顧客シナリオにおけるPL/SQLコードのほとんどは、単に多くのSQL文を発行するだけのものですが、それでもPL/SQLコードのネイティブ・コンパイルを使用することで、ワークロード完了時間全体が若干改善されます。使用するプロシージャ、機能、およびパッケージが計算を実行するもので、なおかつPL/SQL処理が多用される場合には、改善効果はより大きなものとなる可能性があります。

SQL Tuning AdvisorおよびAutomatic SQL Tuning

Oracle Database 10gから提供されているデータベース・パッケージOracle Tuning Packは、Oracle Diagnostic Packと密接に連携しており、SQL Profileを作成することで、SQLコードとアプリケーション・コードのどちらも変更することなく、より高速にSQL文を実行できる機能を提供しています。これにより、一意のSQL文それぞれにおいて、正規化の際に割り当てられたハッシュ値が取得され、このsql_id上の文にSQL Profileをバインドできます。SQL Profileを作成するには、オプティマイザを包括モードに設定する必要があります。このモードでは、通常のデータベース操作時に、該当する特定のSQL文を実行する場合の時間短縮の可能性すべてを計算できます。作成されたSQL Profileには、オプティマイザ用の追加情報とノートが含まれています。この情報はストアド・アウトラインのような統計ではなく、永続的な情報です。Oracle Database 10g Release 2では、ステージング表におけるSQL Profileのデータベース間の転送が可能です。

Oracle Database 11g以降は、負荷の高いSQL文を手作業で選択するプロセスや、SQL Tuning Advisorにおいて高負荷SQL文に関する分析のスケジューリングを行うプロセスをAutomatic SQL Tuningで自動化できます。自動化後は、Automatic Workload RepositoryでAWRスナップショットが収集および保存されるたびに（デフォルトでは60秒ごと）、Automatic SQL Tuningのプロセスにおいて、該当するAWRスナップショットでマークされた高負荷SQL文に対するSQL Profileの計算が開始されます。このプロセスでは、すべてのSQL Profileを自動的に実装し、プロファイリング済みの文のいずれかが再実行された場合に即座にSQL Profileが適用されるようにすることができます。

Automatic SQL Tuningは、以下を実行することで有効化および無効化できます。

```
BEGIN
  DBMS_AUTO_TASK_ADMIN.ENABLE(
    client_name => 'sql tuning advisor',
    operation => NULL,
    window_name => NULL);
END;
/

BEGIN
  DBMS_AUTO_TASK_ADMIN.DISABLE(
    client_name => 'sql tuning advisor',
    operation => NULL,
    window_name => NULL);
END;
/
```

Automatic SQL Tuningには、手動による操作を行わなくてもSQL Profileが自動的に受け入れられるようにパラメータを設定できます。

```
BEGIN
  DBMS_SQLTUNE.SET_TUNING_TASK_PARAMETER(
    task_name => 'SYS_AUTO_SQL_TUNING_TASK',
    parameter => 'ACCEPT_SQL_PROFILES',
    value => 'TRUE');
END;
/
```

次のスクリプトにより、後述のレポートにチューニングの結果が表示されます。

```
variable my_rept CLOB;

BEGIN
  :my_rept :=DBMS_SQLTUNE.REPORT_AUTO_TUNING_TASK(
    begin_exec => NULL,
    end_exec => NULL,
    type => 'TEXT',
    level => 'TYPICAL',
    section => 'ALL',
    object_id => NULL,
    result_limit => NULL);
END;
/

print :my_rept
```

表示されるレポートは以下のようになります。ここでは、Automatic SQL Tuning Advisorにより、2つのAWRスナップショットにおいて7つのチューニング処理が候補として検出されています。

```
-----
GENERAL INFORMATION SECTION
-----
Tuning Task Name           : SYS_AUTO_SQL_TUNING_TASK
Tuning Task Owner         : SYS
Workload Type             : Automatic High-Load SQL W
Execution Count           : 2
Current Execution         : EXEC_20
Execution Type            : TUNE SQL
Scope                     : COMPREHENSIVE
Global Time Limit(seconds): 3600
Per-SQL Time Limit(seconds): 1200
MY_REPT

-----
Completion Status         : COMPLETED
Started at                : 08/27/2008 22:00:02
Completed at              : 08/27/2008 22:05:19
Number of Candidate SQLs : 7
Cumulative Elapsed Time of SQL (s) : 1052
-----
```

レポートには、90%の効果が得られたSQL Profile、98%の効果があつた索引、および2つの文に対する再構築のヒントの4つの結果が示されています。

```
-----
Global SQL Tuning Result Statistics
-----
MY_REPT
-----
Number of SQLs Analyzed           : 7
Number of SQLs in the Report      : 4
Number of SQLs with Findings      : 4
Number of SQLs with SQL profiles recommended : 1
Number of SQLs with Index Findings : 1
Number of SQLs with SQL Restructure Findings : 2
-----
SQLs with Findings Ordered by Maximum (Profile/Index) Benefit, Object ID
objID SQL ID statistics profile(benefit) index(benefit) restructure
-----
      13 bgjxn875surdu                               98.10%
      12 6p8uzacmr06kr                               90.37%
      9  a9cq9s90q2k5w
      14 9b87y60nh1jcg
-----
Tables with New Potential Indices (ordered by schema, number of times, tab)
-----
Schema Name      Table Name      Index Name      Nb Time
-----
      USER_ABC M123_AUSZEICH_DAT      IDX$$_00010006      1
```

レポートには、上記のグローバル・チューニングの結果を受けて、チューニングに関する詳細な推奨事項すべてが表示されます。

後続処理で再利用するために、テスト・サイクルにおけるデータベースのリストア操作でSQL Profileを再インポートする機能を使用できるようにするには、SQL Profileをステージング表に書き込んでエクスポートします。エクスポートしたSQL Profileは、テスト・システムのリストアが完了した後に、インポートして再度アンパックします。このプロシージャを使用すると、SQL Profileをテスト・システムから本番データベースに転送することもできます。

この処理を行うには、まずステージング表を作成します。ただし、この表はSYSユーザー・スキーマに配置することはできません。

```
begin
  DBMS_SQLTUNE.CREATE_STGTAB_SQLPROF (
    table_name=>'STAB_PROFILES',
    schema_name=>'SYSTEM');
end;
/
```

次に、すべてのSQL Profileを、作成したステージング表に書き込みます。

```
BEGIN
  DBMS_SQLTUNE.PACK_STGTAB_SQLPROF (
    profile_name => '%',
    profile_category => 'DEFAULT',
    staging_table_name => 'STAB_PROFILES',
    staging_schema_owner => 'SYSTEM');
END;
/
```

これで、Data Pump Exportを使用してすべてのSQL Profileが含まれたステージング表をエクスポートし、その後テスト・システムを再度リストアしたときにインポートできます。このステージング表は、既存のデータベースのリンクを介して他のデータベースに移行することもできます。アンパック後のSQL Profileは、ターゲット・システムに適用されます。

```
BEGIN
  DBMS_SQLTUNE.UNPACK_STGTAB_SQLPROF (
    profile_name => '%',
    profile_category => 'DEFAULT',
    staging_table_name => 'STAB_PROFILES',
    staging_schema_owner => 'SYSTEM',
    replace => TRUE);
end;
/
```

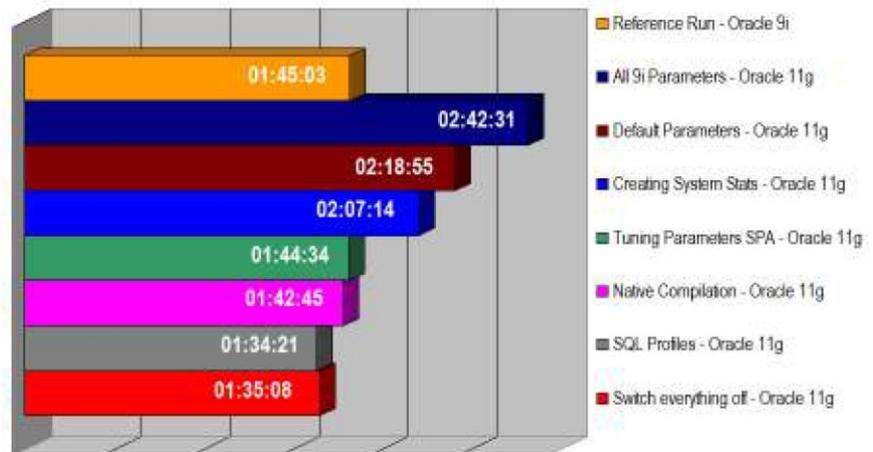

最終テスト実行 - すべての自動化機能および統計機能の無効化

Oracle Database 11gでは、Oracle9iのワークロード完了時間についてはすでに改善されていますが、比較用として確実な根拠を得るため、もう1つ実行しなくてはならない最終的なテストがあります。つまり、`timed_statistics`または`STATSPACK`を有効化せずに、Oracle9iデータベースを実行する必要があります。これは、Oracle Database 11gでは、AWRスナップショット、アドバイザ・ジョブ、およびAutomatic SQL Tuningなど、多くのチューニング・メカニズムがデフォルトで有効化されているためです。最終テストの実行に際しては、これらの機能をすべて無効化しました。

```
statistics_level=basic
_ash_enable=false
timed_statistics=false
```

また、SQL Profileも削除しました。

結果の解釈に誤りがあるてはならないため、このテストでは、理想的なケースの基礎データとしてOracle Database 11gの完了時間に関する初期結果のみを取り上げて比較を実行しました。本番環境での使用を目的とする場合は、診断機能とチューニング機能をすべて有効化しておきます。



診断およびチューニング機能すべてを有効化し、SQL Profileを使用した場合、あらゆるチューニング・メカニズムを無効化した場合のワークロード実行と比較して、パフォーマンス全体に驚くほどの改善効果が認められました。最終テストを実行した際のワークロード完了時間は、テスト実行6の結果ほど良好なものではありませんでした。この結果は、Oracleのパフォーマンス機能すべてを無効化することでデータベースのパフォーマンスが高速化するという考え方が誤りであることを証明すると同時に、データベース・アプリケーションにおいてこれらの機能を使用することで得られるメリットもはっきりと示しています。

最終的な結論

この機能検証を開始したときに設定されたおもな目的は、400のOracleデータベースをOracle 9.2.0.8からOracle 11.1.0.7にアップグレードする最善の方法を特定することと、アプリケーションを変更することなく元のOracle9iでの完了時間を達成できるようにすることでした。パフォーマンス全体で10%の低下があったとしても許容範囲内でしたが、機能検証の結果はそのレベルをはるかに上回り、パフォーマンスは10%も改善しました。

今回検証を行った顧客企業にとって、もっとも大きな意味があったのは、アップグレード時にアプリケーションの変更が一切必要なかったという点です。

アップグレード自体も極めて簡単かつ滞りなく行われ、そのスムーズさは予想をはるかに超えたものでした。自動統計収集やAutomatic SQL Tuningなどの自動化機能により、データベースにおける通常操作時のメンテナンス作業の負担もかなり軽減されました。また、チューニングに関する推奨事項はデータベースによって検出されるため、手動による介入も、追加のチューニング時間も必要ありませんでした。

この企業では、結果は全体的に"すばらしい"ものであったと評価されました。今後は、400のデータベースのうち最初の30のデータベースについての実際の移行が2009年1月に予定されており、2009年5月末までにすべてのデータベースを本稼働させる計画となっています。

参考資料

1. 『Oracle Database 11gへアップグレードする理由』
<http://www.oracle.com/technetwork/jp/database/upgrade/11g-upgrade-benefits-whitepaper-304280-ja.pdf>
2. 『Oracle Databaseアップグレード・ガイド11g』
http://otndnld.oracle.co.jp/document/products/oracle11g/111/doc_dvd/server.111/E05758-02.pdf
3. 『Oracle Database 11g Upgrade Companion』
https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=601807.1
4. Oracle Database 11gへの手動アップグレードに関する完全チェック・リスト：
https://metalink.oracle.com/metalink/plsql/ml2_documents.showDocument?p_database_id=NOT&p_id=429825.1
5. 『Testing the SQL Performance Impact of an Oracle 9i/10g Release 1 to Oracle Database 10g Release 2 upgrade with SQL Performance Analyzer』
http://www.oracle.com/technology/products/manageability/database/pdf/owp_spa_9i_10g.pdf
6. OTNのデータベース・アップグレードに関するWebページ
<http://www.oracle.com/technetwork/jp/database/enterprise-edition/overview/index.html>
7. アップグレードに関するOTNのディスカッション・フォーラム
<http://forums.oracle.com/forums/forum.jspa?forumID=583>
8. 『Upgrading from Oracle Database 9i to 10g - What to Expect from the Optimizer?』
http://www.oracle.com/technology/products/bi/db/10g/pdf/twp_bidw_optimizer_10gr2_0208.pdf

ORACLE

Oracle9i DatabaseからOracle Database 11gへのアップグレード:

実際のお客事例

2008年10月 - V1.1

著者: Mike Dietrich

共著者: Thomas Kempkens, Roy Swonger, Carol Tagliaferri, Carol Palmer

Oracle Corporation

World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

海外からのお問合せ窓口:

電話: +1.650.506.7000

ファックス: +1.650.506.7200

www.oracle.com

Copyright © 2008, Oracle. All rights reserved.

本文書は情報提供のみを目的として提供されており、ここに記載される内容は予告なく変更されることがあります。

本文書は一切間違いがないことを保証するものではなく、さらに、口述による明示または法律による黙示を問わず、特定の目的に対する商品性もしくは適合性についての黙示的な保証を含み、いかなる他の保証や条件も提供するものではありません。オラクル社は本文書に関するいかなる法的責任も明確に否認し、本文書によって直接的または間接的に確立される契約義務はないものとします。

本文書はオラクル社の書面による許可を前もって得ることなく、いかなる目的のためにも、電子または印刷を含むいかなる形式や手段によっても再作成または送信することはできません。Oracle、JD Edwards、PeopleSoft、およびRetekは、米国Oracle Corporationおよびその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標です。