Oracle Direct Seminar

ORACLE®

実践!! パフォーマンスチューニング
-モニタリング編-

日本オラクル株式会社

Oracle Direct

Agenda

1. なぜモニタリングが必要か
2. モニタリングを行う方法紹介
3. パフォーマンスの分析方法
4. GUIによるパフォーマンス監視・チューニング

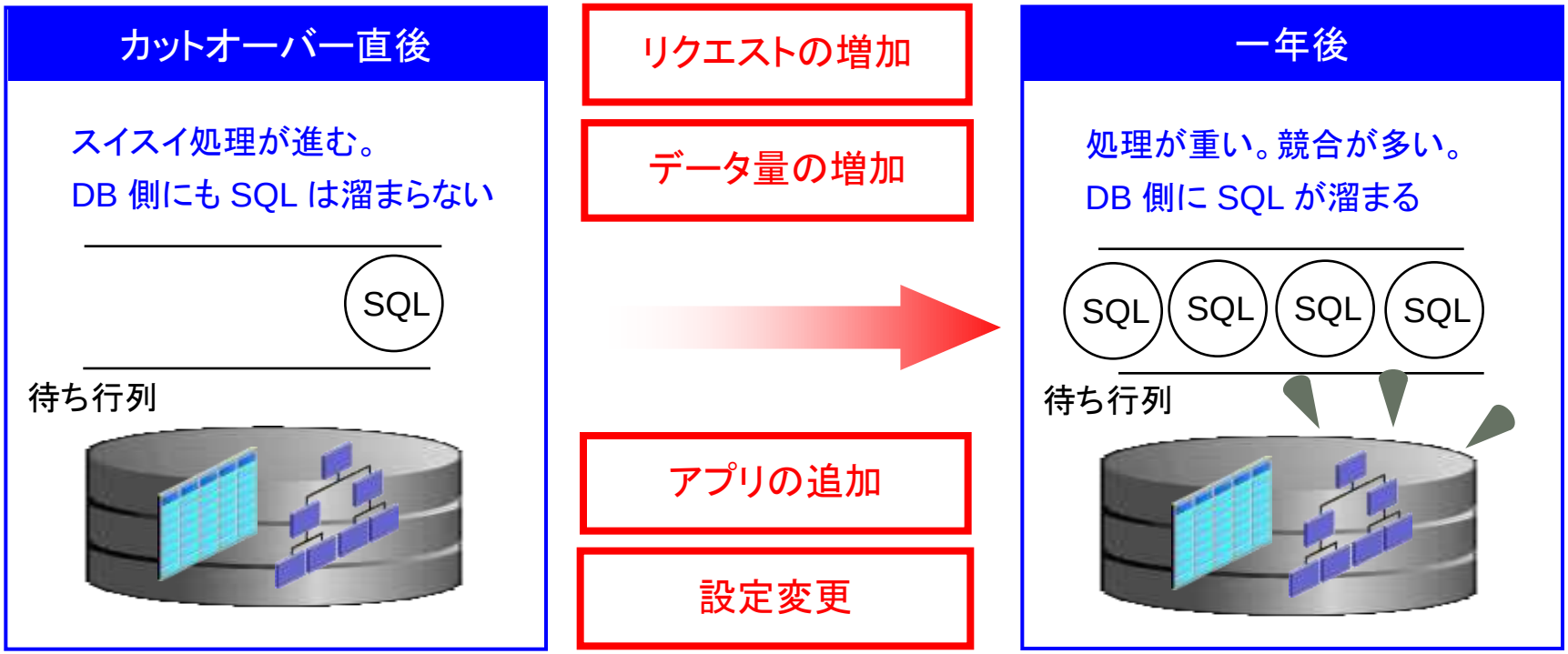
Oracle Directの無償技術サービス

- SQL Serverからの移行アセスメント
- MySQLからの移行相談
- PostgreSQLからの移行相談
- Accessからの移行アセスメント
- Application Server 移行相談
- Oracle Database バージョンアップ支援
- Oracle Developer/2000 Webアップグレード相談
- パフォーマンス・クリニック
- Oracle Database 構成相談
- Oracle Database 高可用性診断
- システム連携アセスメント

http://www.oracle.com/lang/jp/direct/services.html

そもそも、なぜ監視が必要なのか？

- システムの負荷は日々変化するため、監視をしていないと大トラブルにつながる可能性もあります



監視の目的

- パフォーマンス劣化をできるかぎり早期に検出し、問題が深刻化する前に対策が取れるようにします
- システム全体をスローダウンさせかねないリソースの圧迫を、できる限り事前に検知します

性能監視の現状と問題 (一般論)

- 即時にアプリケーションに返されるエラーは検知できるものの、遅延の検知は一般的に難しいです
- パフォーマンス・トラブルやスローダウンも検知できないのが普通です
- 多くの場合、お客様や業務チームからのクレームで気づかされています
  - 日々のワークロードやボトルネックの変化に気づけない
  - 大トラブルになって初めて気づく

性能監視の考え方

個々のボトルネックを監視するより、
遅延で生じる現象を捉えて監視した方が
管理が楽、かつ、監視漏れを防止できます

遅延を引き起こす要因

Oracle 以外のボトルネック
- CPU ネック
- I/O ネック
- ネットワーク待ち

Oracle 内部のボトルネック
- ロック競合
- ブロック競合
- 処理量増加に伴う遅延
- キャッシュ・フュージョンの遅延

Oracle の遅延を引き起こす

遅延により生じる現象

レスポンス時間の増加
リクエストのレスポンス時間が増える

待機時間の増加
待機イベントの待ち時間が増える

アクティブ・セッション数の増加
DB 側で SQL 実行中のシャドウが積み上がる

Agenda

1. なぜモニタリングが必要か
2. モニタリングを行う方法紹介
3. パフォーマンスの分析方法
4. GUIによるパフォーマンス監視・チューニング

Oracle Directの無償技術サービス

- SQL Serverからの移行アセスメント
- MySQLからの移行相談
- PostgreSQLからの移行相談
- Accessからの移行アセスメント
- Application Server 移行相談
- Oracle Database バージョンアップ支援
- Oracle Developer/2000 Webアップグレード相談
- パフォーマンス・クリニック
- Oracle Database 構成相談
- Oracle Database 高可用性診断
- システム連携アセスメント

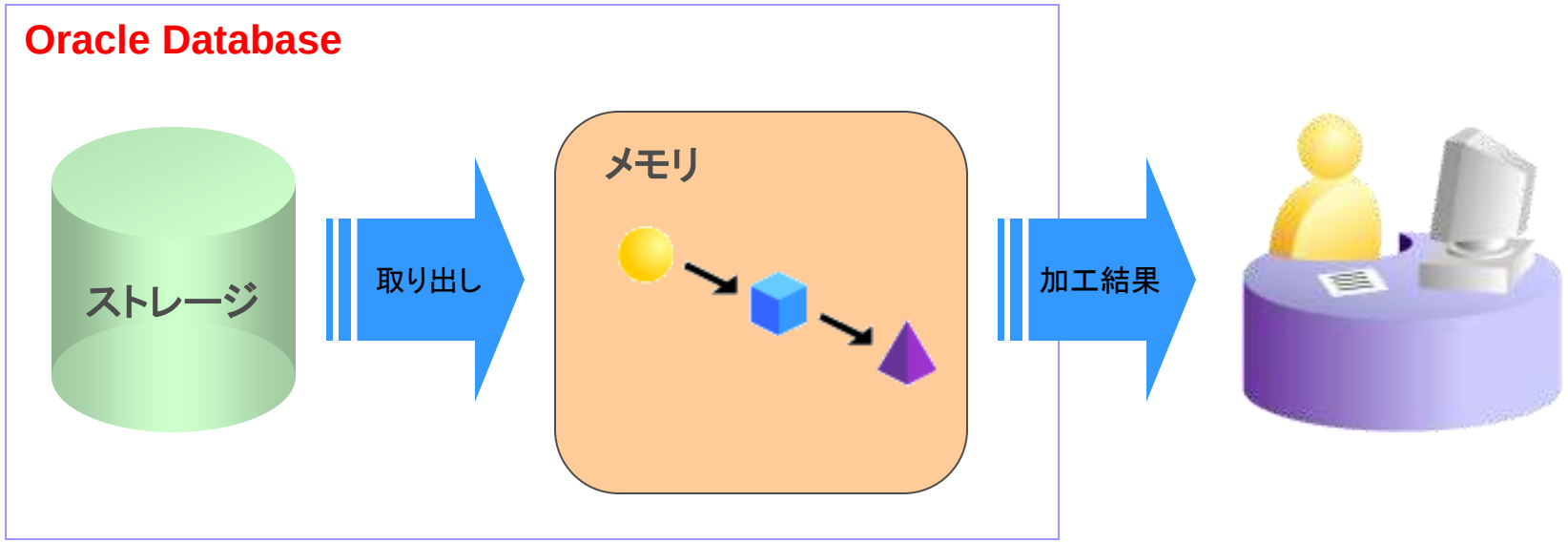
http://www.oracle.com/lang/jp/direct/services.html

確認したい情報と診断ツール

確認したい情報	診断ツール
SQL文の実行計画	EXPLAIN PLAN SQL*Plus AUTOTRACE SQLトレース Oracle Enterprise Manager
SQL文のパフォーマンス統計情報	SQL*Plus AUTOTRACE SQLトレース STATSPACK Oracle Enterprise Manager
ある期間のパフォーマンス統計情報 （メモリヒット率、待機イベント情報、etc.）	STATSPACK Oracle Enterprise Manager

SQLの実行計画とは？

- SQLを実行するには、多数のステップが必要となります。
  - データベースからデータを物理的に取り出すステップ
  - ユーザーが発行する文に返すデータ行の準備ステップ
- 文を実行するためにOracleが使用するステップの組み合せ
  ⇒ 『実行計画』



実行計画の例

『従業員表(EMP表)と部署表(DEPT表)から、各部署ごとの従業員リストを作成するSQL』

```
SQL> set autotrace on
SQL> select d.dname,e.empno,e.ename,e.job
       from emp e,dept d
       where e.deptno=d.deptno;
```

```
Execution Plan
----------------------------------------------------------
 0      SELECT STATEMENT Optimizer=CHOOSE (Cost=5 Card=14 Bytes=392)
   1    0   HASH JOIN (Cost=5 Card=14 Bytes=392)
   2    1     TABLE ACCESS (FULL) OF 'DEPT' (Cost=2 Card=4 Bytes=44)
   3    1     TABLE ACCESS (FULL) OF 'EMP' (Cost=2 Card=14 Bytes=238)
```

結合方法

従業員表(EMP)へのアクセス方法

部署表(DEPT)へのアクセス方法

実行計画の確認方法

1. **Explain plan for <SQL>**
   - 実際には**SQL**は実行されない
   - **plan_table**が必要
2. **SQL*PLUSのAUTOTRACEコマンド**
   - **set autotrace traceonly explain**以外は実際に**SQL**を実行
   - **plan_table**が必要
3. **SQLトレース**
   - **SQL**のトレースを取得
   - **Tkprof**コマンドによりトレースファイルからプランを取得
4. **V$SQL及びV$SQL_PLAN(9*i*～)**
   - 共有プールの**SQL** 文の実行計画を**V$SQL_PLAN**ビューを使用して検索
5. **Enterprise Manager（10g～）**

EXPLAIN PLAN：実行計画の確認

入力されたSQL文の実行計画を確認可能

EXPLAIN PLAN FOR <調べたいSQL文>

utlxplan.sqlで作成する

PLAN_TABLE表に実行計画が格納される

PLAN_TABLEを検索すれば、実行計画がわかる

EXPLAIN PLAN 出力例

```
EXPLAIN PLAN
  FOR SELECT * FROM EMP
      WHERE JOB = 'ANALYST' ;
```

PLAN_TABLE表の検索

@$ORACLE_HOME/rdbms/admin/utlxpls.sql <Linux/Unix>
@%ORACLE_HOME%¥rdbms¥admin¥utlxpls <Windows>

```
| Id  | Operation                    | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |     1 |    37 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| EMP        |     1 |    37 |     2   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | JOB_INDEX  |     3 |       |     1   (0)| 00:00:01 |
```

SQL*PlusのAUTOTRACE機能

実際に行われたSQL文に対する実行計画・パフォーマンス統計の確認可能

< 実行の手順 >

① オプティマイザの実行計画を保存するための表(PLAN_TABLE)を作成

SQL> @$ORACLE_HOME/rdbms/admin/utlxplan <Linux/Unix>

SQL> @%ORACLE_HOME%¥rdbms¥admin¥utlxplan <Windows>

② SYSユーザでPLUSTRACEロールや動的表（V$表）を作成

SQL> @$ORACLE_HOME/sqlplus/admin/plustrce <Linux/Unix>

SQL> @%ORACLE_HOME%¥rdbms¥admin¥plustrce <Windows>

③ SQLを実行するユーザにPLUSTRCEロールを付与

SQL> grant plustrace to scott

④ SQLを実行するユーザでログインし、autotrace機能をON

SQL> set autotrace on

SQL*PlusのAUTOTRACE機能

⑤ SQLを実行すると実行結果の後に実行計画と統計情報が出力

```
Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE (Cost=12 Card=1 Bytes=55)
   1    0   SORT (AGGREGATE)
   2    1     HASH JOIN (Cost=12 Card=1 Bytes=55)
   3    2       TABLE ACCESS (FULL) OF 'ORDERS' (Cost=7 Card=12 Bytes= 300)
   4    2       TABLE ACCESS (FULL) OF 'LINEITEM' (Cost=4 Card=17 Bytes= 510)
Statistics
----------------------------------------------------------
       1460  recursive calls
         23  db block gets
        291  consistent gets
        157  physical reads
          0  redo size
        185  bytes sent via SQL*Net to client
        516  bytes received via SQL*Net from client
          3  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

SQLトレース機能

効果的なSQL文か?
パフォーマンス劣化の原因は?

SQLトレース により以下の情報を知ることが可能

- SQL文の解析/実行フェッチ回数
- CPU時間/経過時間
- 物理的な読込み回数/論理的な読込み回数
- 処理された回数
- ライブラリ・キャッシュ・ミス

SQLトレースの取得手順

①初期化パラメータの設定

TIMED_STATISTICS = True
USER_DUMP_DEST = ディレクトリ名
（トレース・ファイルが書き込まれるディレクトリ名）

②トレースの開始

・セッション単位：ALTER SESSION SET SQL_TRACE = TRUE ;
・インスタンス単位：初期化パラメータ　SQL_TRACE = TRUE

問題のあるSQL文の実行

③トレースファイルの翻訳

tkprofユーティリティ：
TKPROF　取得したトレースファイル　出力ファイル

TKPROF出力結果例

disk … データファイルから読込んだブロック数

query + current …
アクセスしたデータブロックの合計

```
SELECT balance  FROM  accounts
WHERE acc_num = 49999

call     count   cpu  elapsed  disk  query current rows
-------- -----  ---- ------- ----- ------ ------- ----
Parse        1  0.43     0.54     3      3       0    0
Execute      1  0.00     0.00     0      0       0    0
Fetch        1 61.76    79.36  5883   5883       0    1
-------- -----  ---- ------- ----- ------ ------- ----
total        3 62.19    79.90  5886   5886       0    1
```

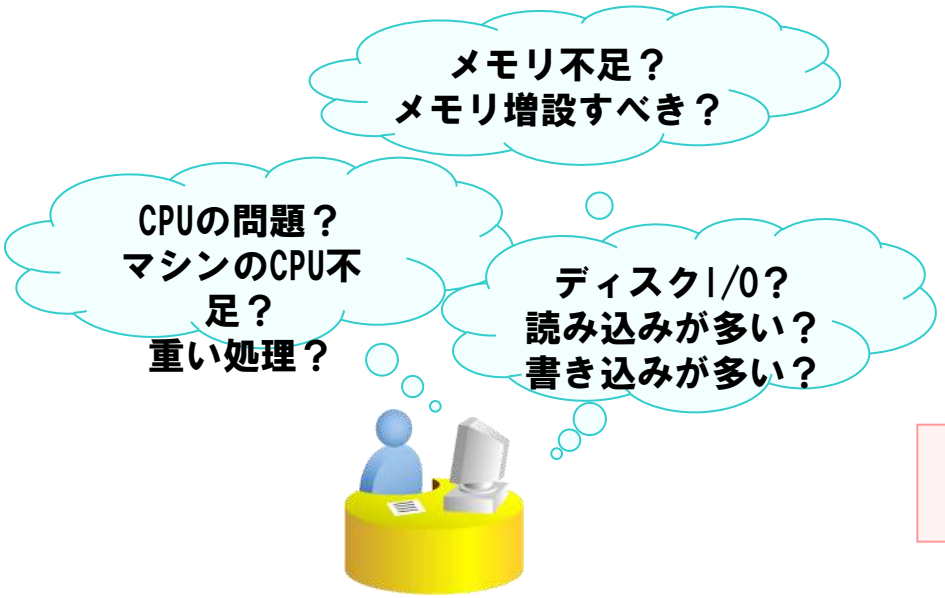
rows…処理された行数

SQL文の処理フェーズ毎の分析結果　cpu　…　cpu時間(秒)
elapsed　…　経過時間(秒)

Statspackの概要

Statspackとは

STATISTICS PACKAGE
パフォーマンス・チューニングに役立つ情報を
レポートという形で提供するツール

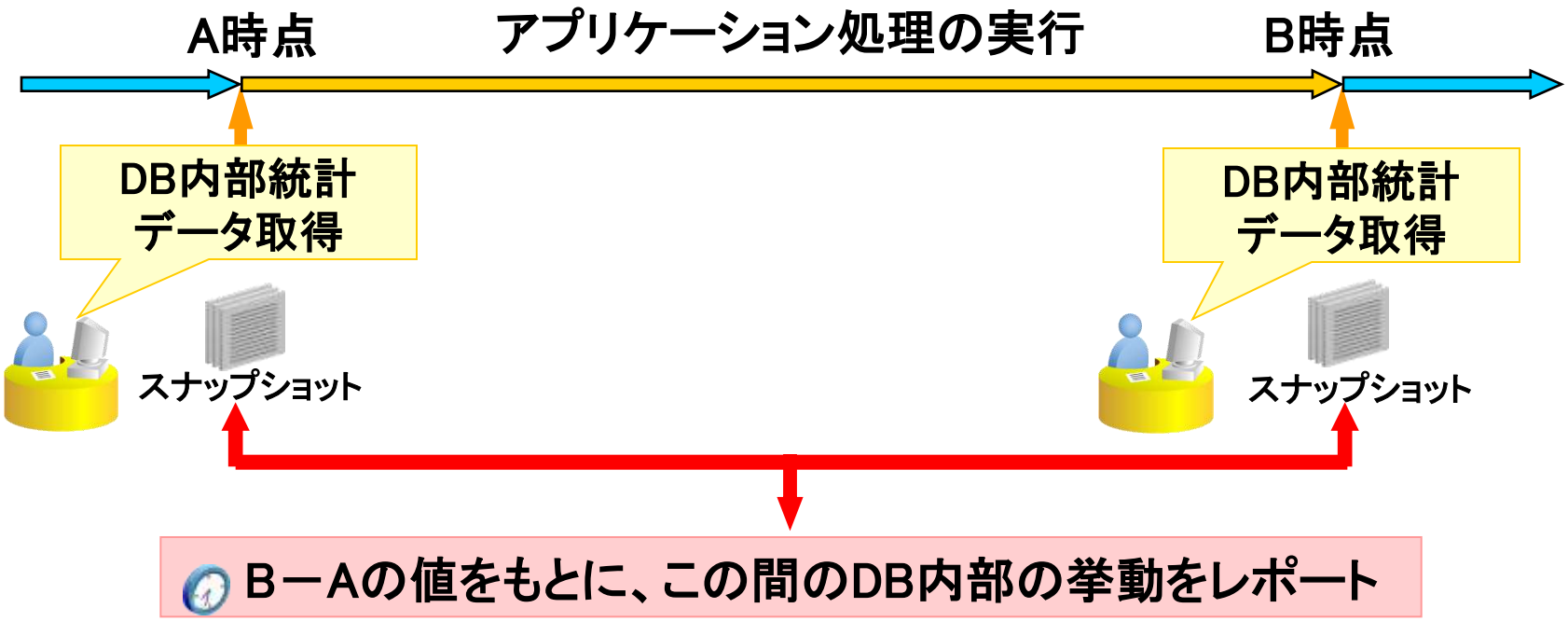


- ある期間で行われた処理の統計情報
  - メモリのヒット率
  - 待機情報の内訳と詳細
  - トランザクション傾向
  - 表領域への書き込み/読み込み

パフォーマンス劣化の原因を分析

Statspackの概要

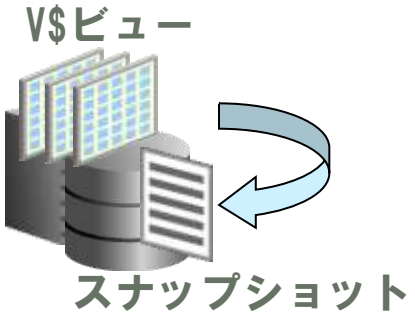
Statspackの実行イメージ



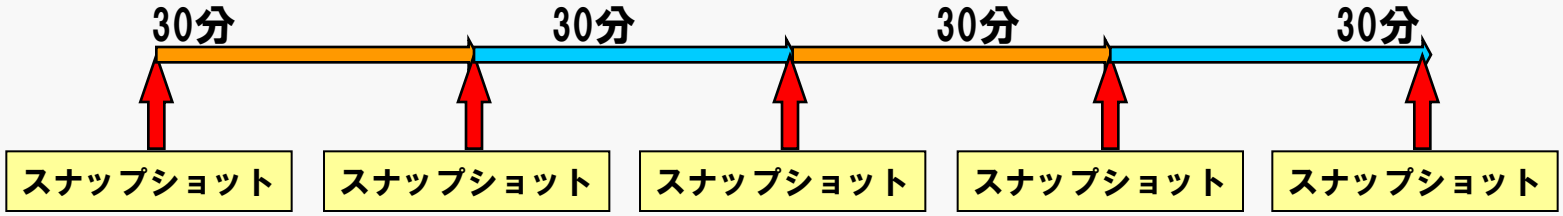
ある2時点で取得した内部統計データの差分を元に、
その間のパフォーマンス統計データを結果レポートに出力

Statspackの概要
スナップショットとは

- 「スナップショット」とは
  - ある時点に収集されたパフォーマンス統計データの集合
  - 内部表（V$ビュー）から情報を取得



スナップショットはどのくらいの間隔で取得するのが良いのでしょうか？



30分～1時間程度の間隔で、常時取得をするのがよいでしょう。
長すぎると、統計データが平均化されて、特定の問題が検知しにくくなります。
また、問題発生時のみ取得しても状況判断がしにくいため「通常の状態」も取得しておくことをおすすめします。
保存期間を決めて古くなったものは定期的に削除するようにしてください。

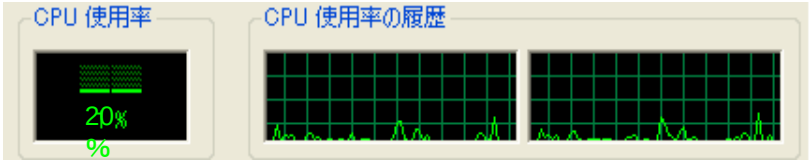
Statspackの概要

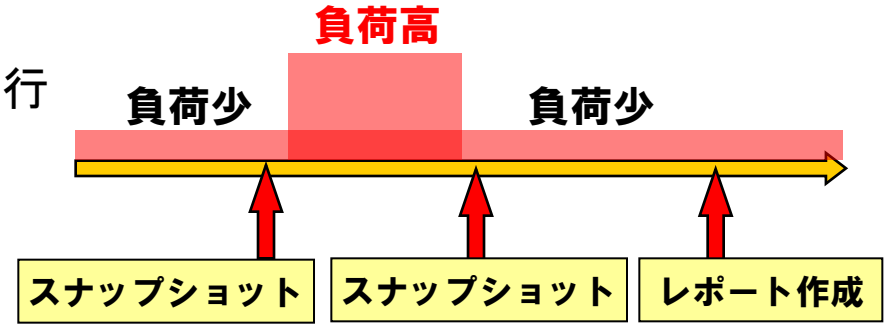
スナップショット取得時の内部動作

Statspack自体が負荷になることはないのでしょうか

Statspackの実体はプロシージャと実行スクリプトですので、
インストールしたのみで、 サーバーの負荷に影響を与えることはありません。
またスナップショットの取得も、通常はそれほど負荷がかかることはありません。
（Level10のスナップショットでは負荷がかかる可能性があります。）

- スナップショット取得時には、内部表（V$ビュー）から情報を取得
- 主にCPUリソースを使用
  - 特にCPU使用率の高い時間は避けたほうが無難
- CPU使用量の高い時間を避けるなどの工夫
  - 高負荷な時間をはさんで取得
  - レポートの作成は負荷が低いときに実行



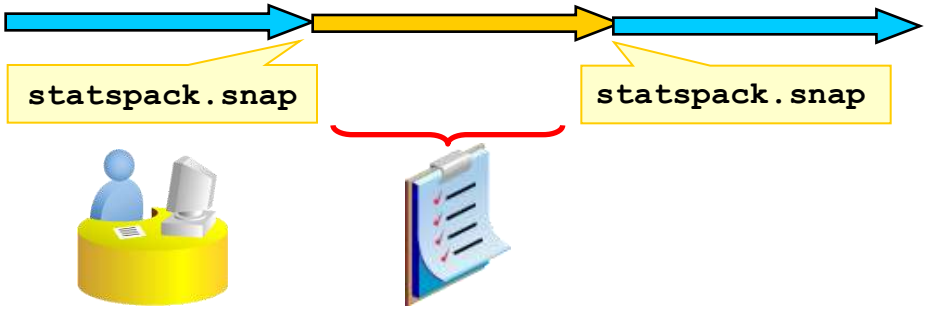


Statspackの実行
スナップショットの取得

- PERFSTATユーザーで、スナップショット取得プロシージャを実行
  - statspack.snapを実行

```
SQL> connect PERFSTAT / ********
SQL> execute statspack.snap
```

- パフォーマンスを監視するために、定期的にスナップショットを取得
  - Statspackでは2つのスナップショットを比較分析するため、最低でも2つのスナップショットが必要



Statspackの実行
レポートの作成

- PERFSTATユーザーで、レポート作成スクリプトを実行
  - スクリプトはデータベースインストール時に配布済み
    ORACLE_HOME/rdbms/admin/spreport.sql

```
SQL> connect PERFSTAT / ********
SQL>@<ORACLE_HOME>/rdbms/admin/spreport.sql
```

  - 必要な項目を入力
    - 分析するスナップショットの指定(2つ)
    - レポート名の指定

Statspackの実行
結果レポート例

- 結果はテキスト形式で出力

report1_2.lst - ワードパッド

ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ヘルプ(H)

```
STATSPACK report for

Database    DB Id    Instance     Inst Num  Startup Time   Release     RAC
---------- ---------- ------------ -------- --------------- ----------- ---
          1214983497 orcl                1 18-May-09 18:01 11.1.0.6.0  NO

Host Name            Platform                       CPUs Cores Sockets   Memory (G)
---------- ---------- ------------------------------ ---- ----- ------- ----------
     edtkr0p11       Linux IA (32-bit)                 1     0       0          1.0

Snapshot       Snap Id     Snap Time      Sessions Curs/Sess Comment
---------- ---------- ------------------ -------- --------- -------------------
Begin Snap:          1 18-May-09 16:27:15       33       5.3
  End Snap:          2 18-May-09 16:48:19       34       5.5
   Elapsed:               19.07 (mins)
   DB time:               83.27 (mins)     DB CPU:       12.31 (mins)

Cache Sizes           Begin        End
~~~~~~~~~~~       ---------- ----------
    Buffer Cache:       92M              Std Block Size:         8K
     Shared Pool:      156M                  Log Buffer:     5,805K

Load Profile              Per Second    Per Transaction   Per Exec   Per Call
~~~~~~~~~~~~         ---------------  ---------------  ---------- ----------
      DB time(s):                4.4            185.0       0.05       0.88
       DB CPU(s):                0.7             27.4       0.01       1.43
       Redo size:          646,587.6     27,398,156.0
   Logical reads:            8,920.6        377,969.0
   Block changes:            4,159.3        176,230.6
  Physical reads:               24.7          1,045.7
 Physical writes:               62.9          2,664.4
      User calls:                0.5             18.1
          Parses:                6.7            284.4
     Hard parses:                0.2              6.2
W/A MB processed:                0.0              0.8
          Logons:                0.0              0.7
        Executes:               89.5          3,791.7
      Rollbacks:                0.0              0.0
   Transactions:                0.0
```

report1_2.lst - ワードパッド

ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ヘルプ(H)

```
Instance Efficiency Indicators
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

            Buffer Nowait %:  100.00       Redo NoWait %:  100.00
            Buffer  Hit   %:   99.72   Optimal W/A Exec %:  100.00
            Library Hit   %:   99.49        Soft Parse %:   97.84
         Execute to Parse %:   92.50         Latch Hit %:  100.00
Parse CPU to Parse Elapsd %:   42.98     % Non-Parse CPU:   99.46

 Shared Pool Statistics        Begin   End
                               ------  ------
             Memory Usage %:   91.44   92.05
    % SQL with executions>1:   74.42   77.14
  % Memory for SQL w/exec>1:   67.20   70.13

Top 5 Timed Events                                                    Avg %Total
~~~~~~~~~~~~~~~~~~                                                   wait   Call
Event                                            Waits    Time (s)   (ms)   Time
----------------------------------------- ------------ ----------- ------ ------
CPU time                                                        745          16.3
Data file init write                              681          111    164    2.4
control file parallel write                       890           91    102    2.0
db file sequential read                         8,651           74      9    1.6
log file parallel write                         1,227           58     48    1.3
          -------------------------------------------------------------

Host CPU  (CPUs: 1  Cores: 0  Sockets: 0)
                    Load Average
                    Begin     End      User   System    Idle     %IO    %CPU
                  ------- -------  ------- ------- ------- ------- --------
                     2.33    4.64    82.05   17.27    0.17    0.17

Note: There is a 12% discrepancy between the OS Stat total CPU time and
      the total CPU time estimated by Statspack
        OS Stat CPU time: 1004(s) (BUSY_TIME + IDLE_TIME)
      Statspack CPU time: 1144(s) (Elapsed time * num CPUs in end snap)
```

Agenda

1. なぜモニタリングが必要か
2. モニタリングを行う方法紹介
3. パフォーマンスの分析方法
4. GUIによるパフォーマンス監視・チューニング

Oracle Directの無償技術サービス

- SQL Serverからの移行アセスメント
- MySQLからの移行相談
- PostgreSQLからの移行相談
- Accessからの移行アセスメント
- Application Server 移行相談
- Oracle Database バージョンアップ支援
- Oracle Developer/2000 Webアップグレード相談
- パフォーマンス・クリニック
- Oracle Database 構成相談
- Oracle Database 高可用性診断
- システム連携アセスメント

http://www.oracle.com/lang/jp/direct/services.html

Statspackレポートを使用した分析
分析ポイント

- Statspackレポートの以下のような項目を確認
  - ①傾向の確認
  - ②メモリ使用状況の確認
  - ③待機状況の確認
  - ④効率の悪いSQL文の確認
  - ⑤I/O状況の確認

Statspackレポートに出力される項目は、バージョンによって若干異なります。このセミナーではOracle11gを使用しています。

レポートのどの部分を確認すればいいのでしょうか？全て見る必要はありますか？

レポートの最初に概要が表示されます。まず、測定時間は適切か、想定どおりの負荷は見られるかを確認してください。次に問題の切り分けをします。ボトルネックは、メモリですか？CPUですか？ディスクI/Oですか？大まかに問題を確認したうえで、問題箇所の詳細を調べてください。

Statspackレポートの分析

全体像の把握

実はデータベースの問題ではないことも・・・
全体的な負荷状況や、OSリソースの使用率などを確認し、
データベースに問題があるかどうかを判断します。

```
Snapshot     Snap Id     Snap Time           Sessions Curs/Sess Comment
~~~~~~~~     -------- ------------------ -------- --------- --------
Begin Snap:       12   20-May-09 17:04:06       34       4.6
  End Snap:       13   20-May-09 17:37:07       34       4.5
   Elapsed:        33.02 (mins)
   DB time:        80.29 (mins)
    DB CPU:        17.40 (mins)

Host CPU  (CPUs: 1  Cores: 0  Sockets: 0)
~~~~~~~~          Load Average
                  Begin     End      User   System    Idle     WIO      WCPU
                -------- --------  ------- ------- ------- -------- --------
                   4.16     4.23    52.01   47.18    0.29     0.29

Memory Statistics                        Begin          End
~~~~~~~~~~~~~~~~~                    ---------- ------------
                    Host Mem (MB):      1,010.3      1,010.3
                     SGA use (MB):        399.1        399.1
                     PGA use (MB):        316.7        313.8
  % Host Mem used for SGA+PGA:           70.9         70.6
```

DB TimeはDB内の処理に要した時間の累計です。セッション数が増えると、複数処理が並列して行われるため、DBを使用する時間が増える傾向にあります。

CPU使用率やメモリ使用率からサーバーのリソースをどれくらい使用しているかを確認します。

処理傾向の分析
全体的な処理傾向の把握

Load Profile	Per Second	Per Transaction	Per Exec	Per Call
DB time(s):	2.4	5.2	0.02	0.33
DB CPU(s):	0.5	1.1	0.00	0.07
Redo size:	911,035.3	1,951,092.9		
Logical reads:	19,805.9	42,416.8		
Block changes:	6,253.8	13,393.2		
Physical reads:	34.9	74.8		
Physical writes:	85.3	182.6		
User calls:	7.3	15.7		
Parses:	11.6	24.7		
Hard parses:	0.6	1.3		

REDO生成量、ブロックが変更量、ディスクへの読み書きの量から処理傾向を確認します。

「Load Profile」から、この期間の処理の傾向が分かります。
複数のレポートを比較する場合、同じ処理傾向のレポートを比較することで、問題の特定がしやすくなります。
また、初期状態や平常時の処理傾向をベースラインとして保存しておくことで業務の変化を追跡しやすくなるでしょう。

メモリ効率の確認

メモリ使用状況の確認

Cache Sizes

	Begin	End			
Buffer Cache:	500M		Std Block Size:	8K	
Shared Pool:	300M		Log Buffer:	5,805K	

Instance Efficiency Indicators

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.83	Optimal W/A Exec %:	99.97
Library Hit %:	98.87	Soft Parse %:	94.59
Execute to Parse %:	90.36	Latch Hit %:	99.96
Parse CPU to Parse Elapsd %:	21.97	% Non-Parse CPU:	98.56

一般的に、ヒット率の理想は95%といわれています。
ヒット率が低い場合は、メモリが不足している可能性があります。

Shared Pool Statistics	Begin	End
Memory Usage %:	78.64	84.65
% SQL with executions>1:	75.87	74.98
% Memory for SQL w/exec>1:	78.18	74.10

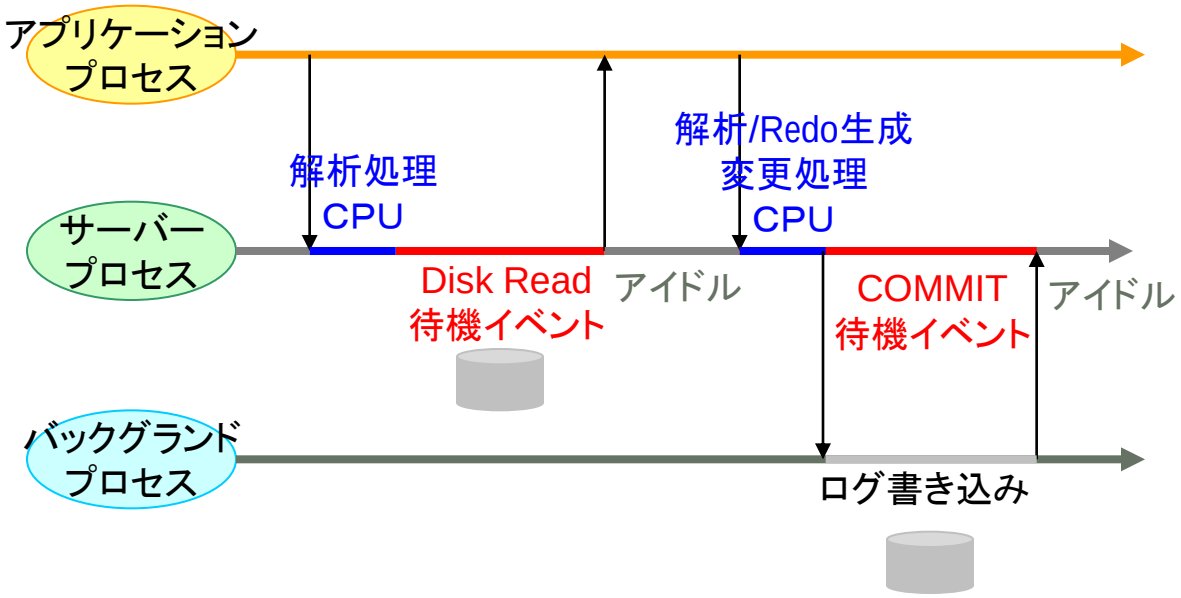
同一SQLが実行されている割合が分かります。同一SQLが多く実行されている環境では、ヒット率は高くなる傾向にあります。

ヒット率など理想値は処理傾向によっても変わります。
一般的にOLTP処理よりも、分析処理やバッチ処理のほうが値が低くなる傾向にあります。

待機イベントの確認
待機イベントとは

- 待機イベント:プロセスが何らかの作業を待機している状態
  - ディスクからのデータの読み込み
  - ディスクへのデータの書き込み
  - メモリ領域の開放
  - 他のユーザーの処理（ロック）

待機イベントとＣＰＵ使用時間をチューニングすることで、レスポンスを早くすることができます。



待機イベントの確認

待機状況の確認

Waits　　:イベントのために待機した合計回数
Time(s)　　:イベントの合計待機時間および合計CPU時間(秒)
Avg wait(ms)　:イベントの平均待機時間

Top 5 Timed Events

Event	Waits	Time (s)	Avg wait (ms)	%Total Call Time
CPU time		1,062		55.6
db file sequential read	23,823	222	9	11.6
log file parallel write	3,040	117	38	6.1
db file scattered read	2,714	116	43	6.1
control file paral	118	101	860	5.3

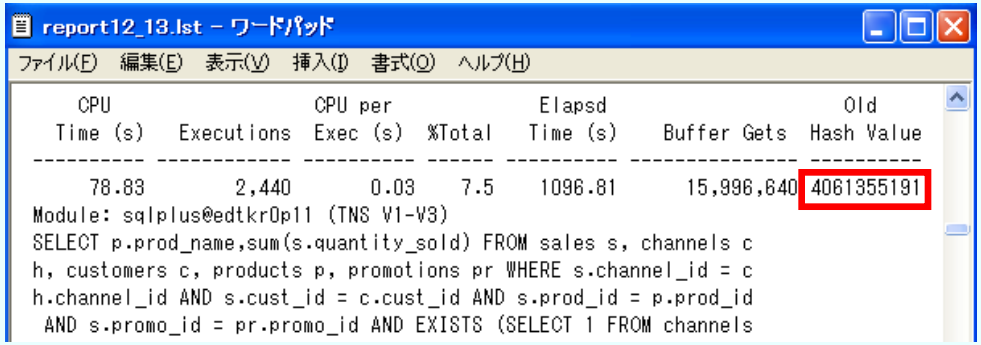
「CPU time」は待機ではなく、CPUを使った処理時間を表すため、「CPU time」が最上位にリストされるのが理想です。（ただし、純粋なCPU不足の可能性もあるのであわせてCPU使用率を確認してください。）

「CPU time」より上位にリストされているイベントがあれば、その項目がボトルネックであると考えられます。上位の待機イベントから詳細を確認します。

問題のあるSQL文の確認

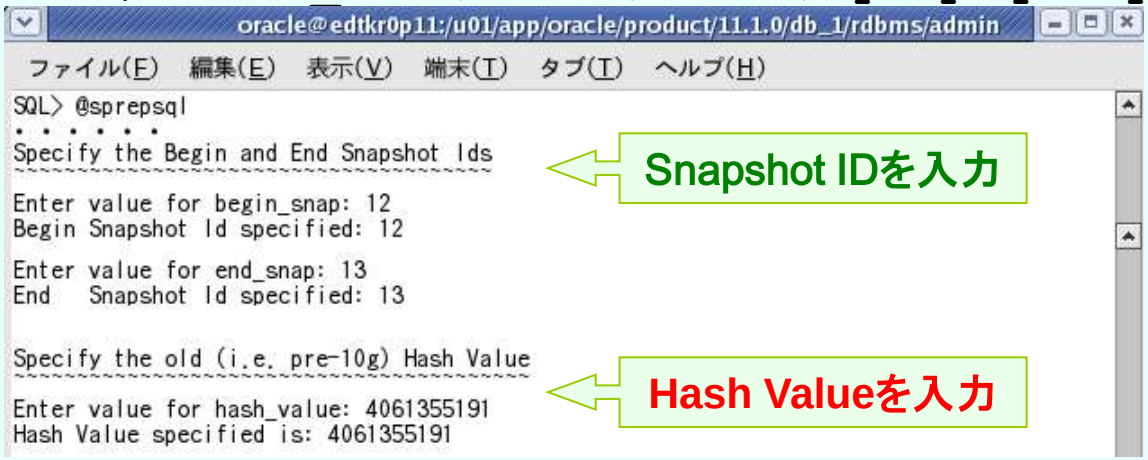
SQL全文と実行計画の確認　実行例

1. Statspackレポートで、実行計画を確認したいSQL文のHash Valueを確認



2. sprepsqlスクリプトの実行

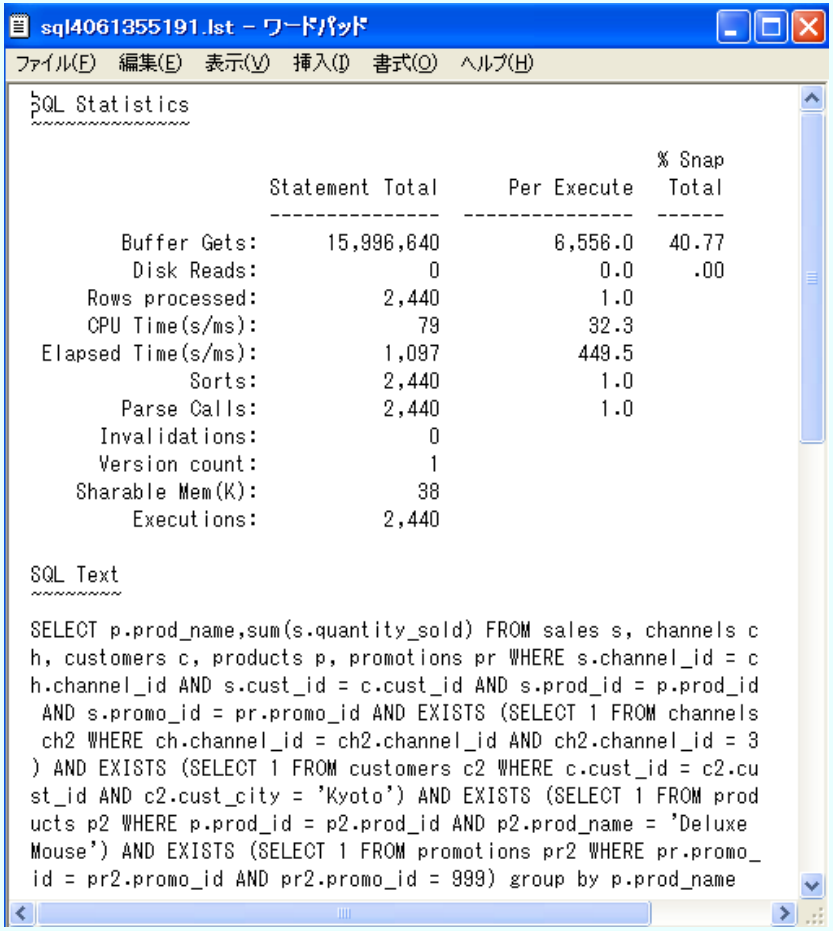
```
SQL> @$ORACLE_HOME/rdbms/admin/sprepsql.sql
```

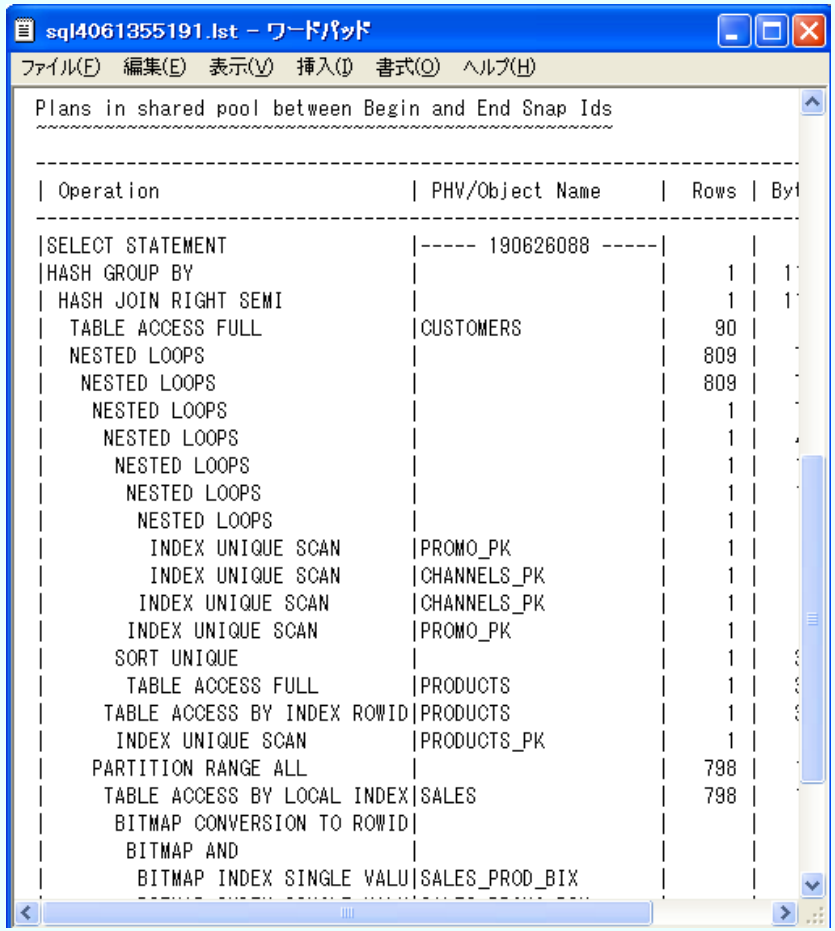


問題のあるSQL文の確認

SQL全文と実行計画の確認　結果例

3. SQL統計、SQL全文、SQLの実行計画が出力





I/O状況の確認

表領域単位のI/O量の確認

Tablespace IO Stats

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
UNDOTBS1	8,808	4	6.3	1.0	12,364	6	6	361.7
USERS	11,939	6	10.7	3.6	8,455	4	0	0.0
SYSAUX	2,343	1	29.6	1.3	2,085	1	0	0.0
SYSTEM	1,923	1	19.9	2.2	254	0	0	0.0
EXAMPLE	1,679	1	28.6	5.7	35	0	0	0.0
TEMP	69	0	5.1	9.9	66	0	0	0.0

どの表領域へのアクセスが多いかを確認することができます。
特定のディスクの表領域にアクセスが集中している場合は、ディスクを追加しI/Oを分散させることを検討してください。
また、複数のディスクにデータをストライピングすることも効果的です。

I/O状況の確認

セグメント単位のI/Oの確認

Segments by Logical Reads

Owner	Tablespace	Object Name	Subobject Name	Obj. Type	Logical Reads	Pct Total
SH	EXAMPLE	CUSTOMERS_PK		INDEX	12,006,240	33.5
SYSMAN	SYSAUX	MGMT_JOB_EXEC_SUMMAR		TABLE	9,746,144	27.2
SH	USERS	SALES_COPY		TABLE	3,903,200	10.9
SYSMAN	SYSAUX	MGMT_JOB_EXEC_SUMM_I		INDEX	3,781,568	10.6
SH	EXAMPLE	CUSTOMERS		TABLE	3,560,240	9.9

Segments by Physical Reads

Owner	Tablespace	Object Name	Subobject Name	Obj. Type	Physical Reads	Pct Total
SH	USERS	SALES_COPY		TABLE	30,885	50.2
SH	EXAMPLE	SALES	ES_Q3_2001	TABLE	1,245	2.0
SH	EXAMPLE	SALES	ES_Q1_2000	TABLE	1,210	2.0
SH	EXAMPLE	SALES	ES_Q1_1999	TABLE	1,207	2.0

どの表や索引へのアクセスが多いかを確認することができます。
ここで上位に上がった表に対して行われている処理（SQL文）を
チューニング対象にすると良いでしょう。

Agenda

1. なぜモニタリングが必要か
2. モニタリングを行う方法紹介
3. パフォーマンスの分析方法
4. GUIによるパフォーマンス監視・チューニング

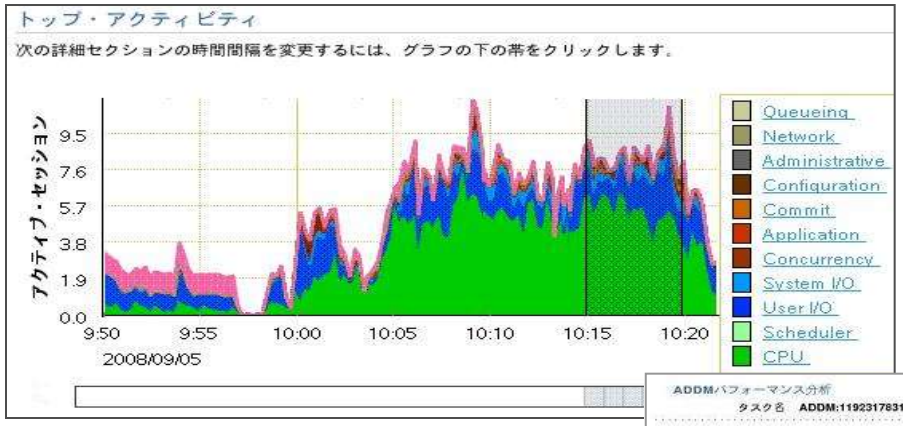
Oracle Directの無償技術サービス

- SQL Serverからの移行アセスメント
- MySQLからの移行相談
- PostgreSQLからの移行相談
- Accessからの移行アセスメント
- Application Server 移行相談
- Oracle Database バージョンアップ支援
- Oracle Developer/2000 Webアップグレード相談
- パフォーマンス・クリニック
- Oracle Database 構成相談
- Oracle Database 高可用性診断
- システム連携アセスメント

http://www.oracle.com/lang/jp/direct/services.html

Enterprise ManagerによるGUIによるリアルタイムモニタリング

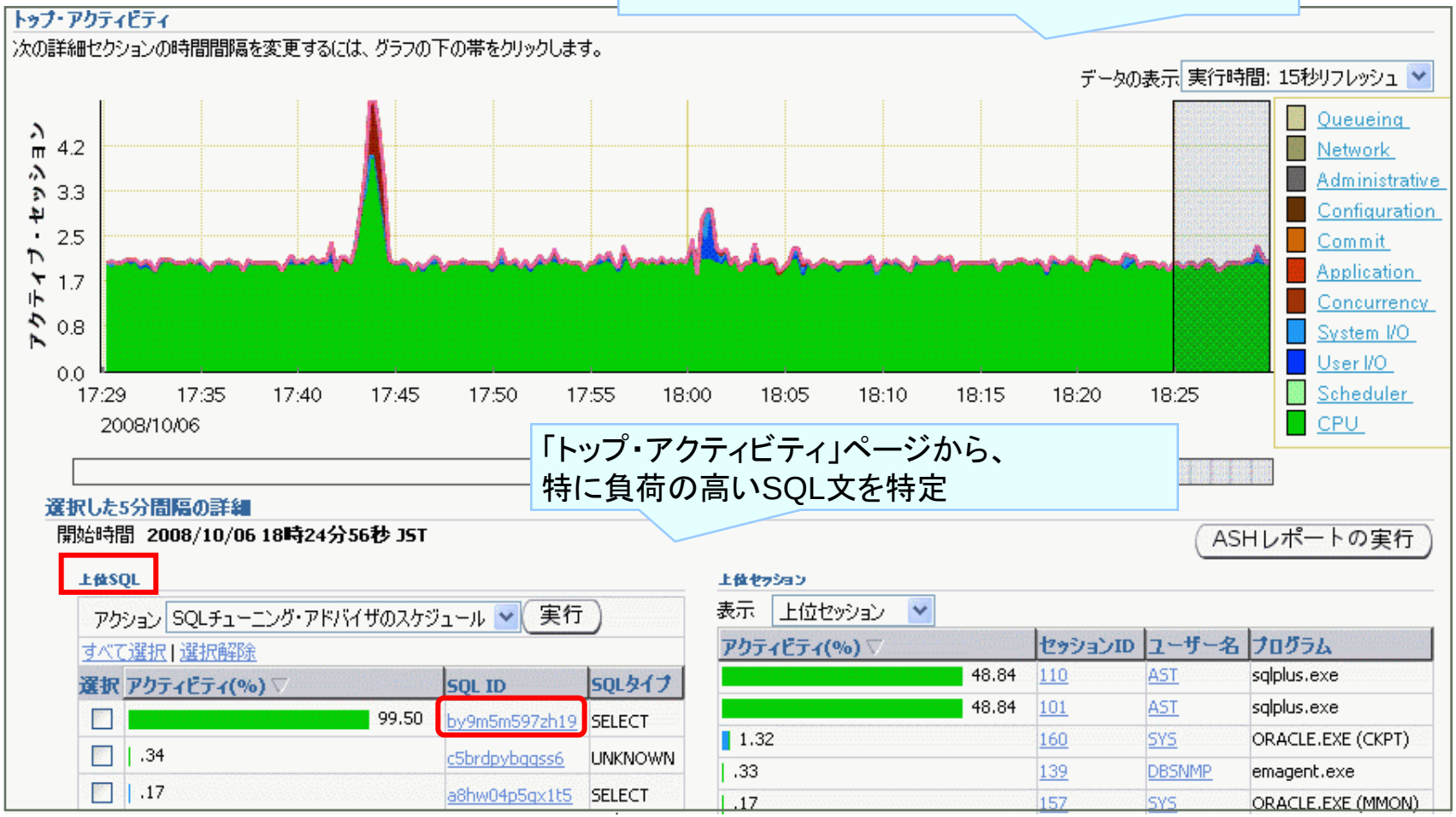
- Enterprise Manager
  - ブラウザからアクセスするデータベース管理ツール
  - GUIの画面から負荷状況をグラフィカルに表示
  - ボトルネック項目をリスト



SQLチューニング・アドバイザの実行例

EE Diag Tun

Enterprise Managerの「パフォーマンス・ページ」からデータベースの負荷状況を確認



「トップ・アクティビティ」ページから、特に負荷の高いSQL文を特定

EE Diag Tun

SQLチューニング・アドバイザによる自動チューニング

- Oracle Database10gから実装されたアドバイス機能
- 高負荷で問題となるSQL文や実行計画を診断し、アドバイスを提示
  - 統計の再取得
  - SQL文の問題点を探し、SQL文の修正方法
  - 必要な索引の作成をアドバイス
  - SQLプロファイルの作成

高負荷のSQL文

SQLチューニング・アドバイザ

Enterprise Managerが負荷を軽減する最適な対処方法を提示

失効・欠落している統計の収集

Index の作成

SQL文の再構成

SQLプロファイルの作成

SQLチューニング・アドバイザの実行例

EE Diag Tun

SQLの詳細: by9m5m597zh19

SQL IDに切替え [ ] 実行 データの表示 [実行時間: 手動リフレッシュ] リフレッシュ SQLワークシート SQLチューニング・アドバイザ

テキスト

```
select /*+ USE_NL(s c) FULL(s) FULL(c) AST */ c.cust_id, sum(s.quantity_sold)
from sh.sales s, sh.customers c
where s.cust_id = c.cust_id and c.cust_id < 2 group by c.cust_id
```

「上位SQL」から、負荷の高いSQL文を特定
→このSQL文の実行計画を確認

チューニング対象のSQL文を選び、「SQLチューニング・アドバイザのスケジュール」から実行

詳細

次の詳細を参照するには計画ハッシュ値を選択してください。 計画ハッシュ値 4005616876

統計 アクティビティ プラン 計画管理 チューニング履歴

データソース カーソル・キャッシュ 取得時間 2008/10/06 18:32:22 (UTC+09:00) 解析スキーマ AST オプティマイザ・モード ALL_ROWS

追加情報

すべて開く | すべて閉じる

操作	オブジェクト	順序	行	バイト	コスト	CPU (%)	時間	パーティションの開始	パーティションの停止	問合せブロック名/オブジェクトの別名	フィルタ	予測
SELECT STATEMENT		6			9,398	100						
HASH GROUP BY		5	1	13	9,398	1	0:1:53			SEL$1		"C"."CUST_ID"[NUMBER,22], SUM(...
NESTED LOOPS		4	1	13	9,397	1	0:1:53					"C"."CUST_ID"[NUMBER,22], "S"....
TABLE ACCESS FULL	SH.CUSTOMERS	1	1	5	405	0	0:0:5			SEL$1 / C@SEL$1	"C"."CUST_ID"<2	"C"."CUST_ID"[NUMBER,22]
PARTITION RANGE ALL		3	1	8	8,992	1	0:1:48	1	28			"S"."QUANTITY_SOLD"[NUMBER,22]
TABLE ACCESS FULL	SH.SALES	2	1	8	8,992	1	0:1:48	1	28	SEL$1 / S@SEL$1	("S"."CUST_ID"<2 AND "S"."CUST...	"S"."QUANTITY_SOLD"[NUMBER,22]

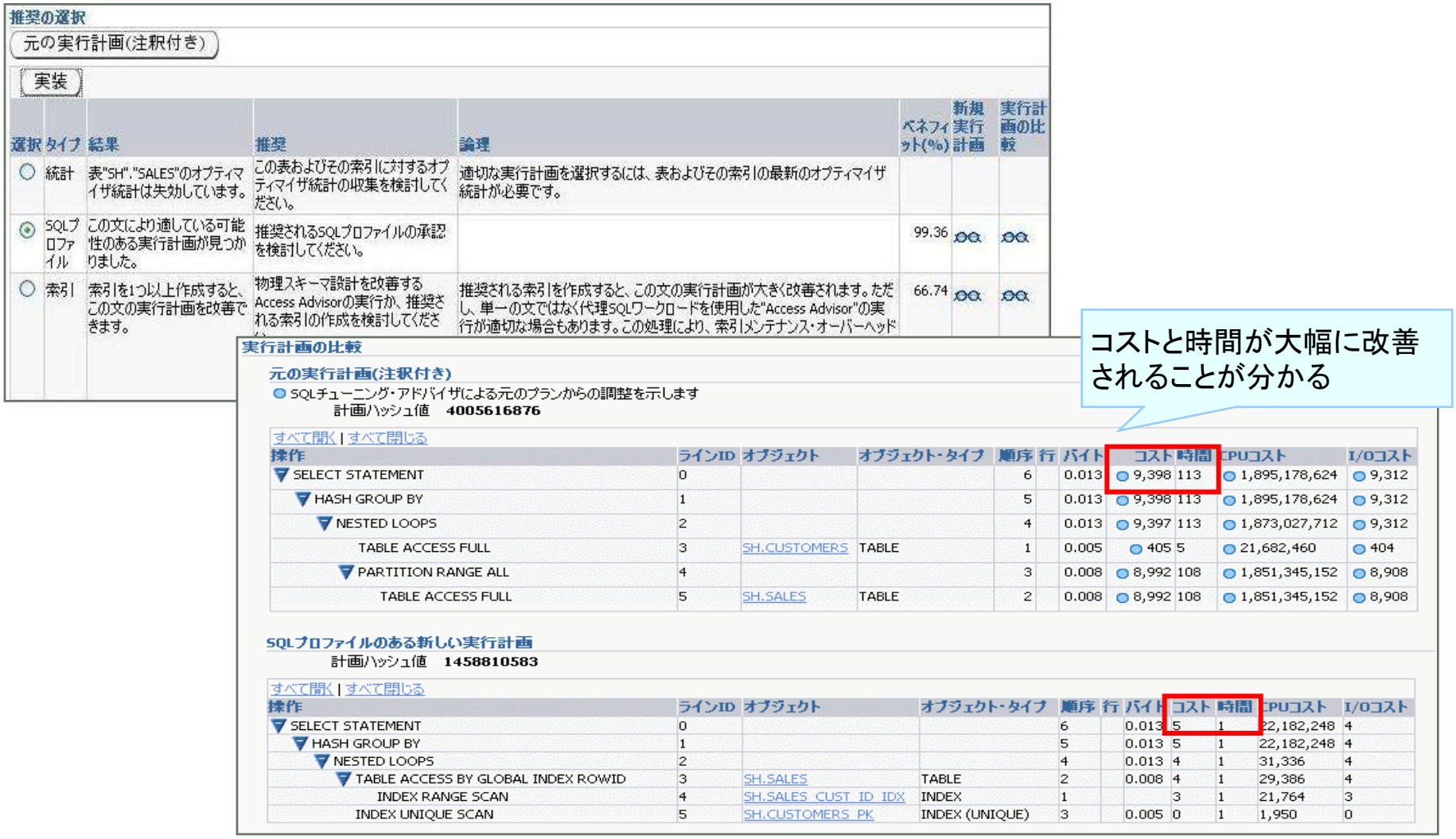
リライトの説明の表示

統計 アクティビティ プラン 計画管理 チューニング履歴

SQLワークシート SQLチューニング・アドバイザのスケジュール

SQLチューニング・アドバイザの実行例

EE Diag Tun



推奨の選択

元の実行計画(注釈付き)

実装

選択	タイプ	結果	推奨	論理	ベネフィット(%)	新規実行計画	実行計画の比較
○	統計	表"SH"."SALES"のオプティマイザ統計は失効しています。	この表およびその索引に対するオプティマイザ統計の収集を検討してください。	適切な実行計画を選択するには、表およびその索引の最新のオプティマイザ統計が必要です。			
◉	SQLプロファイル	この文により適している可能性のある実行計画が見つかりました。	推奨されるSQLプロファイルの承認を検討してください。		99.36	oo	oo
○	索引	索引を1つ以上作成すると、この文の実行計画を改善できます。	物理スキーマ設計を改善するAccess Advisorの実行か、推奨される索引の作成を検討してくださ	推奨される索引を作成すると、この文の実行計画が大きく改善されます。ただし、単一の文ではなく代理SQLワークロードを使用した"Access Advisor"の実行が適切な場合もあります。この処理により、索引メンテナンス・オーバーヘッド	66.74	oo	oo

実行計画の比較

元の実行計画(注釈付き)

SQLチューニング・アドバイザによる元のプランからの調整を示します

計画ハッシュ値 4005616876

すべて開く | すべて閉じる

操作	ラインID	オブジェクト	オブジェクト・タイプ	順序	行	バイト	コスト	時間	CPUコスト	I/Oコスト
SELECT STATEMENT	0			6		0.013	9,398	113	1,895,178,624	9,312
HASH GROUP BY	1			5		0.013	9,398	113	1,895,178,624	9,312
NESTED LOOPS	2			4		0.013	9,397	113	1,873,027,712	9,312
TABLE ACCESS FULL	3	SH.CUSTOMERS	TABLE	1		0.005	405	5	21,682,460	404
PARTITION RANGE ALL	4			3		0.008	8,992	108	1,851,345,152	8,908
TABLE ACCESS FULL	5	SH.SALES	TABLE	2		0.008	8,992	108	1,851,345,152	8,908

SQLプロファイルのある新しい実行計画

計画ハッシュ値 1458810583

すべて開く | すべて閉じる

操作	ラインID	オブジェクト	オブジェクト・タイプ	順序	行	バイト	コスト	時間	CPUコスト	I/Oコスト
SELECT STATEMENT	0			6		0.013	5	1	22,182,248	4
HASH GROUP BY	1			5		0.013	5	1	22,182,248	4
NESTED LOOPS	2			4		0.013	4	1	31,336	4
TABLE ACCESS BY GLOBAL INDEX ROWID	3	SH.SALES	TABLE	2		0.008	4	1	29,386	4
INDEX RANGE SCAN	4	SH.SALES_CUST_ID_IDX	INDEX	1			3	1	21,764	3
INDEX UNIQUE SCAN	5	SH.CUSTOMERS_PK	INDEX (UNIQUE)	3		0.005	0	1	1,950	0

コストと時間が大幅に改善されることが分かる

まとめ

- サービスレベルを保つためにも、日々のモニタリングは重要
- Oracleでは以下の方法でモニタリングを行えます
  - Explain Plan
  - AUTO TRACE
  - SQLトレース
  - Statspack
  - Enterprise Manager

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。

システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。
http://www.oracle.co.jp/inq_pl/INQUIRY/quest?rid=28

※フォームの入力には、Oracle Direct Seminar申込時と同じ
　ログインが必要となります。
※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録されている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096

※月曜~金曜　9:00~12:00、13:00~18:00
（祝日および年末年始除く）

ORACLE

以上の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。