

Oracle DBA & Developer Days 2011

日本オラクル、今年最大の技術トレーニングイベント

2011年11月9日(水)～11月11日(金) シェラトン都ホテル東京



ORACLE®

シバタツ流！チューニングの極意
「パフォーマンス・チューニングの勘どころ」

日本オラクル株式会社 テクノロジー製品事業統括本部 技術本部 Exadata技術部
プリンシパルエンジニア 柴田竜典

以下の事項は、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。

OracleとJavaは、Oracle Corporation 及びその子会社、関連会社の米国及びその他の国における登録商標です。文中の社名、商品名等は各社の商標または登録商標である場合があります。

シバタツって誰？

シバタツって誰？

- ・ 日本オラクル株式会社
テクノロジー製品事業統括本部 技術本部 Exadata技術部
プリンシパルエンジニア 柴田竜典
- ・ oracletech.jpにて『シバタツ流！ DWHチューニングの極意』を
全5回で連載
 - ・ これを読めばDWHデザインを間違わない！
 - ・ すべての回に似顔絵がついている！

シバタツ DWH

検索



もう少しプロフィール

- Oracle Exadata リリース当初から、お客様のSQLやデータを使用したPoC (Proof of Concept) を実施
 - 本番稼働しているたくさんのシステムのパフォーマンス・チューニングを経験
- 2010年には米オラクルの開発部門に所属
 - 米国のお客様のPoCを実施しつつ、そこから見えてきたOracle Database のパフォーマンス課題の解決に取り組む
- Oracle Exadata を担当するまではOracle TimesTen In-Memory Database を担当
 - RDBMSパフォーマンスに関わり続けている



セッション目次

- SQL
 - SQL概要
 - 性能問題を引き起こす5大SQL
 - SQL分析ツール
- DWH
 - DWH概要
 - 索引 / パーティショニング / 圧縮
- OLTP
 - OLTP概要
 - メモリー / ネットワーク / ストレージ
 - ボトルネックの監視と特定



SQL

SQLがすべて

- RDBMSデザインで重要な項目
 - スキーマ・デザイン
 - SQLデザイン
 - 実行戦略
 - 最適化テクニック
- このセクションではSQLにフォーカスします
 - OLTP向けSQLではありません
 - DWH向けSQLではありません
 - SQLワークロードではありません
 - SQLです

オプティマイザに正しく伝える

- ・ 正しい統計情報
 - ・ 予想された取得行数と実際の取得行数に乖離はないか？
 - ・ データの最新状況を反映しているか？
 - ・ サンプル数は正しいか？
 - ・ データに偏りはないか？ → ヒストグラム
 - ・ 列と列に相関性はないか？ → 拡張統計
- ・ 必要十分な制約
 - ・ NOT NULL
 - ・ NOT IN / NOT EXISTS が高速になる可能性
 - ・ 外部キー
 - ・ 結合を省ける可能性

シバタツ 統計情報

検索

性能問題を引き起こす 5大SQL

性能問題を引き起こす5大SQL

- ・ パース・エラー
- ・ 結合条件不足
- ・ 暗黙のデータ型変換
- ・ 関数 / ワイルドカード
- ・ 非等価結合（ファジー結合）

パース・エラー

- ・ パース・エラーのSQLは非常にコストが高くレポートからの調査が困難
 - ・ 例外処理やエラー生成は Oracle Database 内で高コスト
 - ・ 共有プールに残らない / SQL ID がない / トレースが取れない
 - ・ 自動ワークロード・リポジトリ (AWR) レポートの `parse count (failures)` で発生回数は分かる
- ・ ベテラン開発者の一部が、パース・エラーを利用してスキーマの状態確認やバージョン確認、機能確認などを行なうことがあるので気をつけること

パース・エラー

HISTORIES表が存在するかどうか確認する

-- 問題のあるSQL

```
SQL> SELECT 1 FROM histories;
```

```
SELECT 1 FROM histories
```

*

```
ERROR at line 1:
```

```
ORA-00942: 表またはビューが存在しません。
```

-- 正しい方法

```
SELECT COUNT(*) FROM user_tables
```

```
WHERE table_name = 'HISTORIES';
```

結合条件不足

- ・ 表の数を n とすると、結合条件は $n - 1$ 必要だが結合条件が不足すると直積結合が行われる
- ・ 新人開発者がデータ量が少ないテスト環境で DISTINCT を含む SELECT 文を書くときに起きやすいので気をつけること
- ・ データ量が増えてから性能問題を引き起こす

結合条件不足

LOCNO=100の場所で働く従業員を一覧する

-- 問題のあるSQL

```
SELECT DISTINCT ename FROM emp, dept
WHERE dept.locno = 100;
```

-- 2表の結合には1つの結合条件が必要

```
SELECT DISTINCT ename FROM emp, dept
WHERE dept.locno = 100
AND emp.deptno = dept.deptno;
```

暗黙のデータ型変換

- ・ フィルター条件の左辺と右辺のデータ型が違った場合
Oracle Database は文法エラーにせず
暗黙的にデータ型の変換を可能な限り行う
- ・ エラーにならない代わりに、以下のような問題が発生
 - ・ データ変換に必要なリソースが毎回無駄遣いされる
 - ・ 索引が使用されない / パーティション・プルーニングされないなどの適切でない実行計画が立てられる
 - ・ 暗黙のデータ変換が常に正しいとは限らない
例: 日付フォーマット 09/10/11 ←何年何月何日？

暗黙のデータ型変換

コーディング規約で統一する

- ・ 数字が入るが、演算に使うことはない列は NUMBER型ではなくVARCHAR2型で定義する
- ・ ルールが変わると先頭が0（ゼロ）になる可能性のある列は NUMBER型ではなくVARCHAR2型で定義する
- ・ 例
 - ・ 電話番号
 - ・ 郵便番号
 - ・ クレジットカード番号

暗黙のデータ型変換

運転免許証番号が1234567890の人物を表示する

-- 索引が使用されない

-- VARCHAR2型に対する問題のあるSQL

```
SELECT name FROM customers
WHERE driver_licence_no = 1234567890;
           VARCHAR2           NUMBER
```

-- 正しいSQL

```
SELECT name FROM customers
WHERE driver_licence_no = '1234567890';
           VARCHAR2           VARCHAR2
```

暗黙のデータ型変換

2011年11月9日の合計転送量を表示する

-- パーティション・プルーニングされない

-- DATE型に対する問題のあるSQL

```
SELECT SUM(byte) FROM transfers
WHERE created_date = '2011-11-09';
           DATE           VARCHAR2
```

-- 正しいSQL

```
SELECT SUM(byte) FROM transfers
WHERE created_date =
TO_DATE('2011-11-09', 'YYYY-MM-DD');
           DATE           DATE
```



DATE

関数とワイルドカード

- ・ 関数やワイルドカード（LIKE検索）を使用すると
オプティマイザが正しいカーディナリティを予測できない
- ・ `DYNAMIC_SAMPLING`ヒントを使用するか
拡張統計を使用して対応する

シバタツ 拡張統計

検索

関数とワイルドカード

郵便番号上3けたが107の顧客数を数える

-- × そのままでは問題のあるSQL

```
SELECT COUNT(*) FROM customers
WHERE SUBSTR(zipcode, 1, 3) = '107';
```

-- ○ 動的サンプリングを使用する

```
SELECT /*+ DYNAMIC_SAMPLING(customers 1) */
COUNT(*) FROM customers
WHERE SUBSTR(zipcode, 1, 3) = '107';
```

-- ○ 拡張統計を取得する

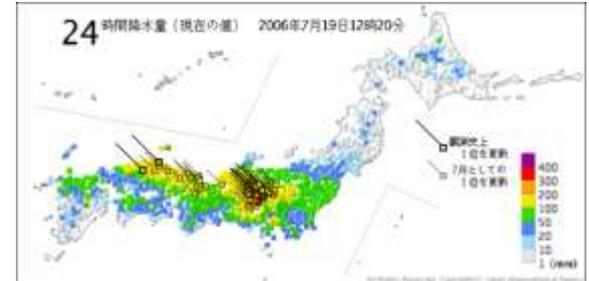
```
SELECT DBMS_STATS.CREATE_EXTENDED_STATS (
    'user1', 'customers',
    '(SUBSTR(zipcode, 1, 3))'
) FROM dual;
```

非等価結合（ファジー結合）

- ・ 結合条件は等価（=）でなければならない
- ・ 非等価結合は以下のような様々な課題がある
 - ・ データが膨れ上がる
 - ・ よりリソースが必要になる
 - ・ 遅い
 - ・ 正しいカーディナリティが予測されない
 - ・ パラレル・クエリー / ハッシュ結合が使われない
- ・ 非等価結合を使用しないようなスキーマ・デザインに変更する
- ・ 関数を使って等価結合に変更する
- ・ 最適な対応方法は状況により異なる

非等価結合（ファジー結合）

降水量ごとに地区を色分けする



-- × BETWEENを使用した非等価結合

```
SELECT l.city_id, g.color_id
FROM rainfall_data l, rainfall_grades g
WHERE l.rainfall BETWEEN g.low AND g.high;
```

-- ○ grade_idを返す関数を作成する

```
SELECT l.city_id, g.color_id
FROM rainfall_data l, rainfall_grades g
WHERE get_rainfall_grade(l.rainfall) =
       g.grade_id;
```

非等価結合（ファジー結合）

アクセスしたURLの提供会社をドメインを元に一覧する

-- LIKE検索を使用した非等価結合

```
SELECT DISTINCT p.provider_id
FROM access_logs l, provider_master p
WHERE l.url LIKE ('http://' || p.domain || '%')
AND l.user_id = 1234;
```

-- SUBSTRとINSTRでURLからドメイン名を抜き出す

```
SELECT DISTINCT p.provider_id
FROM access_logs l, provider_master p
WHERE p.domain =
      SUBSTR(l.url, 8, INSTR(l.url, '/', 1, 3) - 8)
AND l.user_id = 1234;
```

SQL分析ツール

SQL分析ツール

- ・ 静的分析
 - ・ EXPLAIN文 + `DBMS_XPLAN.DISPLAY`
 - ・ SQL*Plusの**AUTOTRACE**
- ・ 動的分析
 - ・ `DBMS_XPLAN.DISPLAY_*`
 - ・ リアルタイムSQL監視
- ・ トレース
 - ・ 10046 (SQLトレース)
 - ・ アクティブ・セッション履歴 (ASH)
後ほどOLTPセクションで

EXPLAIN文

- 可能性のある実行計画を表示する
 - DDL (CREATE TABLE AS SELECT)
 - DML
 - SELECT
- 長所
 - SQLの実行を必要としない
- 短所
 - 予測でしかなく、実際と異なる可能性がある
 - バインド・ピークが使用されない (バインド変数の値を考慮しない)

EXPLAIN文

```
-----  
-----  
| Id | Operation | Name | Rows | Bytes | Cost (%CPU)| Time |  
-----  
-----  
| 0 | SELECT STATEMENT | | 3 | 114 | 2 (0)| 00:00:01 |  
| 1 | TABLE ACCESS BY INDEX ROWID| EMP | 3 | 114 | 2 (0)| 00:00:01 |  
|* 2 | INDEX RANGE SCAN | EMP_N1 | 3 | | 1 (0)| 00:00:01 |  
-----  
-----
```

読めない……

EXPLAIN文

- ・ フォーマット
 - **SET TAB OFF**
 - ・ 等幅フォントを使用する(MS ゴシック / Courier New)
 - ・ スペース文字を維持
 - ・ 長い行が尻切れになったり折り返されることをできるだけ避ける

```
SQL> SET FEEDBACK OFF TAB OFF
```

```
SQL> SET LINESIZE 1000 PAGESIZE 1000
```

```
SQL> EXPLAIN PLAN
```

```
2> SELECT * FROM emp WHERE deptno = 10;
```

```
SQL> SELECT *
```

```
2> FROM table(DBMS_XPLAN.DISPLAY);
```

EXPLAIN文

Plan hash value: 3324114979

```
-----  
| Id  | Operation                | Name    | Rows  | Bytes | Cost (%CPU)| Time      |  
-----  
|  0  | SELECT STATEMENT         |         |    3  |  114  |    2   (0)| 00:00:01 |  
|  1  | TABLE ACCESS BY INDEX ROWID| EMP     |    3  |  114  |    2   (0)| 00:00:01 |  
|*  2  | INDEX RANGE SCAN        | EMP_N1  |    3  |       |    1   (0)| 00:00:01 |  
-----
```

Predicate Information (identified by operation id):

```
-----  
  
2 - access("DEPTNO"=10)
```

EXPLAIN文

- Predicate Information
 - フィルター
 - トランスフォーメーション
 - ブルーム・フィルター
 - オフロード候補
- Notes
 - 自動パラレル度 (Auto DoP)
 - パラレル度の根拠
 - 動的サンプリング
 - カーディナリティ・フィードバック

SQL*PlusのAUTOTRACE

- ・ 可能性のある実行計画を表示する
 - ・ DML
 - ・ SELECT
- ・ 長所
 - ・ SQLの実行を必要としないこともできる
- ・ 短所
 - ・ 実行計画は予測でしかなく、実際と異なる可能性がある
 - ・ バインド・ピークが使用されない（バインド変数の値を考慮しない）

SQL*PlusのAUTOTRACE

- **SET AUTOTRACE OFF**
 - ・ レポートを生成しない。デフォルト
- **SET AUTOTRACE ON**
 - ・ SQLを実行して結果を出力し、実行計画と実行統計を表示する
- **SET AUTOTRACE TRACEONLY**
 - ・ SQLを実行するが結果を出力せず、実行計画と実行統計を表示する
- **SET AUTOTRACE TRACEONLY EXPLAIN**
 - ・ SQLを実行せずに、実行計画のみ表示する
 - ・ EXPLAIN文と同じ
- **SET AUTOTRACE TRACEONLY STATISTICS**
 - ・ SQLを実行するが結果を出力せず、実行統計のみを表示する

SQL*PlusのAUTOTRACE

```
SQL> SET AUTOTRACE EXP
```

```
SQL> SELECT * FROM emp WHERE deptno = 10;
```

Execution Plan

Plan hash value: 3324114979

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		3	114	2 (0)	00:00:01
1	TABLE ACCESS BY INDEX ROWID	EMP	3	114	2 (0)	00:00:01
* 2	INDEX RANGE SCAN	EMP_N1	3		1 (0)	00:00:01

Predicate Information (identified by operation id):

2 - access("DEPTNO"=10)

SQL*PlusのAUTOTRACE

```
SQL> SET AUTOTRACE STAT
```

```
SQL> SELECT * FROM emp WHERE deptno = 10;
```

Statistics

```
-----  
0 recursive calls  
0 db block gets  
4 consistent gets  
0 physical reads  
0 redo size  
1159 bytes sent via SQL*Net to client  
525 bytes received via SQL*Net from client  
2 SQL*Net roundtrips to/from client  
0 sorts (memory)  
0 sorts (disk)  
3 rows processed
```

DBMS_XPLAN.DISPLAY_*

- ・ 実際に表示された実行計画を表示する
- **DBMS_XPLAN.DISPLAY_CURSOR**
 - ・ カーソル・キャッシュから
- **DBMS_XPLAN.DISPLAY_AWR**
 - ・ AWRから
- **DBMS_XPLAN.DISPLAY_SQLSET**
 - ・ SQLチューニング・セットから

DBMS_XPLAN.DISPLAY_*

```
SQL> SELECT * FROM emp WHERE deptno = :n;
```

```
SQL> SELECT * FROM
```

```
  2> table(DBMS_XPLAN.DISPLAY_CURSOR());
```

```
SQL_ID 5xt3urx81f9th, child number 0
```

```
-----  
SELECT * FROM EMP WHERE DEPTNO = :n
```

```
Plan hash value: 3324114979
```

```
-----  
| Id | Operation | Name | Rows | Bytes | Cost (%CPU) | Time |  
-----  
| 0 | SELECT STATEMENT | | | | 2 (100) | |  
| 1 | TABLE ACCESS BY INDEX ROWID | EMP | 3 | 114 | 2 (0) | 00:00:01 |  
|* 2 | INDEX RANGE SCAN | EMP_N1 | 3 | | 1 (0) | 00:00:01 |  
-----
```

```
Predicate Information (identified by operation id):  
-----
```

```
2 - access("DEPTNO"=:N)
```

リアルタイムSQL監視

- ・ 実際に表示された実行計画、待機イベントなどを表示する
- ・ 表示フォーマットにはText / HTML / Activeのいずれか
- ・ Oracle Enterprise Manager から表示するか
`DBMS_SQLTUNE.REPORT_SQL_MONITOR`関数から取得
- ・ Oracle Tuning Pack オプションが必要

シバタツ SQL監視

検索

リアルタイムSQL監視

DWHに対するパラレル・クエリー時の計画統計

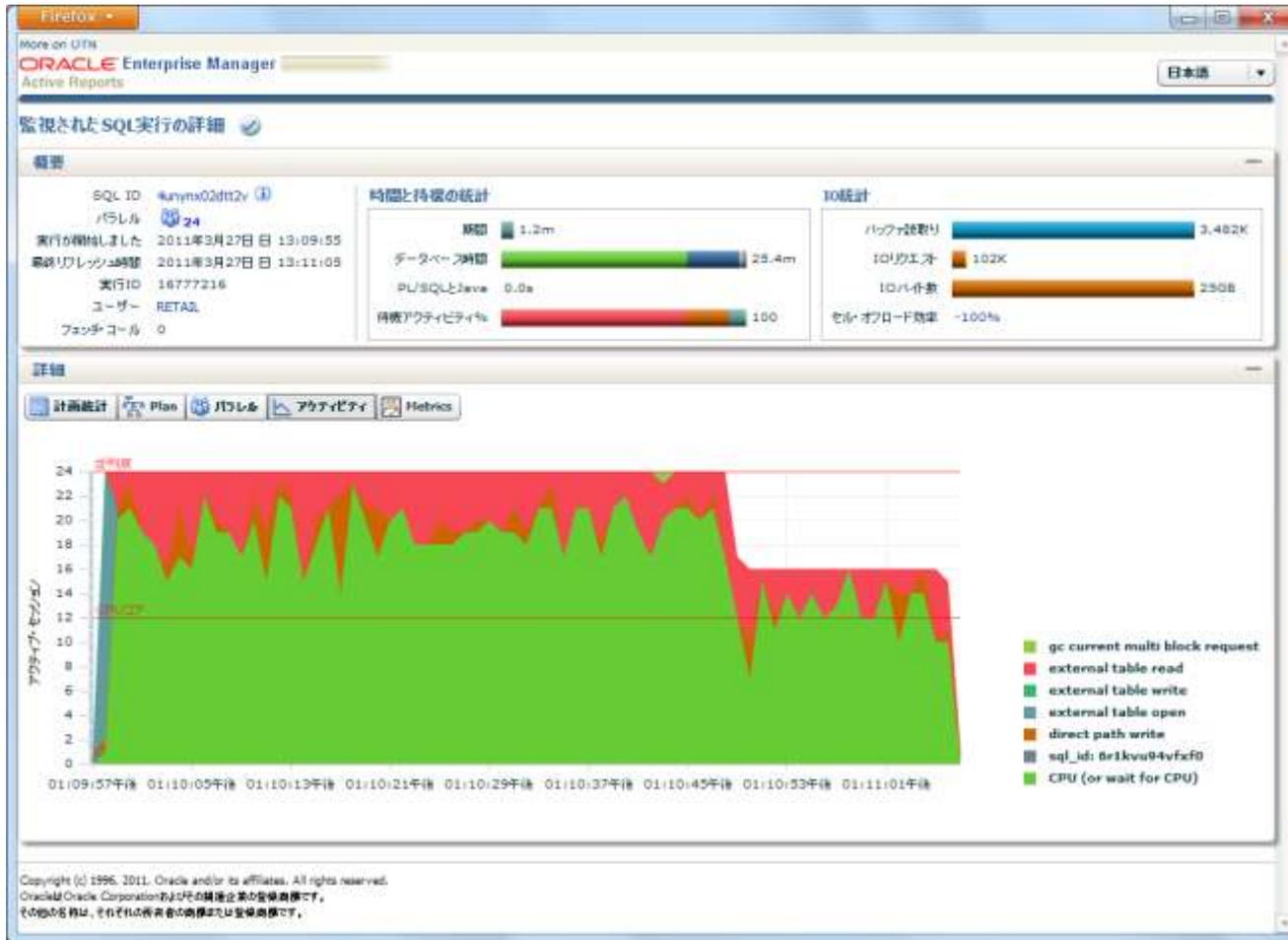


⋮



リアルタイムSQL監視

外部表を利用したローディング時のアクティビティ



リアルタイムSQL監視

```
SQL> SET TRIMSPPOOL ON TRIM ON
SQL> SET PAGESIZE 0 LINESIZE 1000
SQL> SET LONG 1000000 SET LONGCHUNKSIZE 1000000
SQL> SPOOL sqlmon.html
SQL> SELECT DBMS_SQLTUNE.REPORT_SQL_MONITOR(
  2>   sql_id => 'brd3nyszkb4v3',
  3>   type => 'ACTIVE'
  4> ) FROM dual;
SQL> SPOOL OFF
```

リアルタイムSQL監視

- ・ 一時表領域への読取り / 書込みが多くて遅い
 - ・ 一時表領域をチューニングする？
- ・ 一時表領域への読取り / 書込みの原因はハッシュ結合
 - ・ メモリー割当てをチューニングする？
- ・ ハッシュ結合が大量にPGAを使用している原因はBROADCASTしている行数が予想よりはるかに多いこと
 - ・ BROADCASTを無効にする？
- ・ BROADCASTしている行数が予想よりはるかに多い原因はフィルターしている2つの列に相関性があることにオプティマイザが気づいてないから
 - ・ 根本原因を発見

10046 (SQLトレース)

- ・ 各コールごとのトレースを取得
オプションによって待機イベントやバインド変数も取得可能
- ・ トレース・ファイルはtkprofであとで成型するのが一般的

- ・ 長所
 - ・ 特定期間内のすべてのSQLの情報が取得できる
 - ・ その他の方法で取得できない情報が取得できる
- ・ 短所
 - ・ トレースを有効にしないとイケない
 - ・ パラレル実行時には複数のトレース・ファイルができてしまい分析が困難
 - ・ 実行時間にオーバーヘッド

10046 (SQLトレース)

- ・ トレース・ファイルのファイル名に識別子をつける
 - ・ `ALTER SESSION SET TRACEFILE_IDENTIFIER = 'my10046';`
- ・ トレースを有効にする
 - ・ `ALTER SESSION SET EVENTS = '10046 trace name context forever, level 1';`
 - ・ Level 1 は `SET SQL_TRACE = TRUE` と同じ
 - ・ `EXECUTE DBMS_MONITOR.SESSION_TRACE_ENABLE;`
- ・ トレースを無効にする
 - ・ `ALTER SESSION SET EVENTS = '10046 trace name context off';`
 - ・ `EXECUTE DBMS_MONITOR.SESSION_TRACE_DISABLE;`

10046 (SQLトレース)

```
select * from emp where deptno = :n
```

call	count	cpu	elapsed	disk	query	current	rows
Parse	2	0.00	0.00	0	0	0	0
Execute	2	0.00	0.00	0	0	0	0
Fetch	4	0.00	0.00	0	8	0	8
total	8	0.00	0.00	0	8	0	8

Misses in library cache during parse: 1

Optimizer mode: ALL_ROWS

Parsing user id: 88 (TEACHER)

Number of plan statistics captured: 2

Rows (1st)	Rows (avg)	Rows (max)	Row Source	Operation
3	4	5	TABLE ACCESS BY INDEX ROWID EMP	(cr=4 pr=0 pw=0 time=152 us cost=2 size=114 card=3)
3	4	5	INDEX RANGE SCAN EMP_N1	(cr=2 pr=0 pw=0 time=120 us cost=1 size=0 card=3) (object id 64363)

10046 (SQLトレース)

```
Rows      Execution Plan
-----  -----
0  SELECT STATEMENT  MODE: ALL_ROWS
3  TABLE ACCESS    MODE: ANALYZED (BY INDEX ROWID) OF 'EMP' (TABLE)
3  INDEX            MODE: ANALYZED (RANGE SCAN) OF 'EMP_N1' (INDEX)
```

Elapsed times include waiting on following events:

Event waited on	Times	Max. Wait	Total Waited
-----	Waited	-----	-----
Disk file operations I/O	1	0.00	0.00
SQL*Net message to client	4	0.00	0.00
SQL*Net message from client	4	0.88	0.89

DWH

DWHで間違えるポイント

- ・ 間違ったハードウェア構成
 - ・ CPUとバランスの取れないディスクIO性能（ディスク本数）
 - ・ 間違ったネットワーク構成
- ・ クエリー
 - ・ 間違ったクエリー戦略
- ・ ETL / データ・ローディング
 - ・ 間違った手法
 - ・ 不必要な索引 / データ・マート
- ・ テスト環境から本番環境へ
 - ・ 予期せぬ大量の同時実行クエリー

シバタツ DWH

検索

DWHチューニングの心得

- SQL経由でできるだけデータベース内で処理する
 - ETLツールを使用している場合はそれがどのように処理しているかに注意すること
- OLTP用DML (UPDATE / DELETE / MERGE) をできるだけ使わない
 - 並列化されたハードウェアを最大限活用する
 - フラグメントを最小化し、圧縮効率を最大化する
 - REDOログ生成量を最小化する
- CPUとIOをできるだけ使い切ることを意識する

シバタツ UPDATE

検索

索引 パーティショニング 圧縮

DWHで使うべきチューニング手法

- ・ 使うべきでないチューニング手法
 - ・ 索引

- ・ 使うべきチューニング手法
 - ・ パーティション
 - ・ 圧縮

索引の課題と現状

- ・ 索引はOLTP向けのテクニックである
 - ・ 索引は取得行が少ない場合は有効だが
一般的なDWHでは取得行が多いクエリーが多い
- ・ 索引メンテナンスによって大規模データベースやOracle Real Application Clusters (Oracle RAC) がスケールしなくなる
- ・ 索引によってローディングやデータ変換が遅くなる
- ・ 「索引がなくても遅くならない」と言っても信じてもらえない
- ・ もしIO帯域が70MB/sから70GB/sに変わっても
実行計画や実行方法を変えない

索引で速くなるのか？

- ・ 結果が100万行の検索
 - ・ 索引はキャッシュされているがデータはされていないとする
 - ・ 5ms per I/O の環境で100万ランダムIOPSする
 - ・ 1回の検索に5000秒かかる
 - ・ 1時間たっても検索が終わらない原因
 - ・ 順次読取りで14.4GB/sec出る環境で5000秒あったら何GBスキャンできる？
 - ・ 72TB
- ・ ディスクのスループット性能は並列化により上げられるがレスポンス時間は長年進化がない
 - ・ 今までの意識を変える時期

とある現場での一日

```
SELECT t1.ename, t2.ename FROM emp t1, emp t2
WHERE t1.empid = t2.empid AND t1.empid = 'WMSYS'
```

Rows	Row Source Operation
-----	-----
528384	HASH JOIN
8256	TABLE ACCESS FULL T
1833856	TABLE ACCESS FULL T

「あー、もー、ほんとコストベース・オプティマイザは腹立つわー。索引があるのに何で使わないんだ！ 索引イコール速いイコール正しい程度なのが、なんでオプティマイザは分からないかな……。やっぱ信頼できるのはルールベースだけだな。ルールベースに変更したらどうなるか見てみようか」

とある現場での一日

```
SELECT /*+ RULE */ t1.ename, t2.ename FROM t t1, t t2
WHERE t1.oid = t2.oid AND t1.owner = 'WMSYS'
```

Execution Plan

```
-----
0          SELECT STATEMENT Optimizer=HINT: RULE
1    0      TABLE ACCESS (BY INDEX ROWID) OF 'T'
2    1        NESTED LOOPS
3    2          TABLE ACCESS (FULL) OF 'T'
4    2          INDEX (RANGE SCAN) OF 'T_IDX' (NON-UNIQUE)
```

「ほらみる。ルールベースだとちゃんと索引を使うじゃないか。ルールベースが廃止だとかバカ言ってるんじゃないよ。一応、キャッシュヒット率も確認しておこうか」

とある現場での一日

```
SELECT phy.value, cur.value, con.value,  
       1 - (phy.value / (cur.value + con.value)) "Cache hit ratio"  
FROM v$sysstat cur, v$sysstat con, v$sysstat phy  
WHERE cur.name = 'db block gets'  
AND con.name = 'consistent gets'  
AND phy.name = 'physical reads'
```

VALUE	VALUE	VALUE	Cache hit ratio
1277377	58486	121661490	.989505609

「キャッシュ・ヒット率も99%。悪くないんじゃない？ 索引を使っていてキャッシュ・ヒット率も高い。文句なし」

とある現場での一日

表スキャン時のSQLトレース

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	35227	5.63	9.32	23380	59350	0	528384
total	35229	5.63	9.33	23380	59350	0	528384

Misses in library cache during parse: 1

Optimizer goal: CHOOSE

Parsing user id: 80

Rows Row Source Operation

```
-----  
528384 HASH JOIN  
      8256 TABLE ACCESS FULL T  
1833856 TABLE ACCESS FULL T
```

とある現場での一日

索引スキャン時のSQLトレース

call	count	cpu	elapsed	disk	query	current	rows
Parse	1	0.00	0.00	0	0	0	0
Execute	1	0.00	0.00	0	0	0	0
Fetch	35227	912.07	3440.70	1154555	121367981	0	528384
total	35229	912.07	3440.70	1154555	121367981	0	528384

Misses in library cache during parse: 0

Optimizer goal: RULE

Parsing user id: 80

Execution Plan

```
-----  
0      SELECT STATEMENT Optimizer=HINT: RULE  
1    0      TABLE ACCESS (BY INDEX ROWID) OF 'T'  
2    1      NESTED LOOPS  
3    2      TABLE ACCESS (FULL) OF 'T'  
4    2      INDEX (RANGE SCAN) OF 'T_IDX' (NON-UNIQUE)
```

パーティショニング・デザインのゴール

- ・ データ管理
 - ・ データ・ローディング / データ削除
 - ・ 管理オペレーションの極小化
- ・ クエリーの最適化
 - ・ パーティション・プルーニング
 - ・ ハッシュ・ワイズ結合

シバタツ パーティショニング

検索

DWHの一般的なパーティショニング戦略

- ・ イベント日時で大きな表をパーティショニングする
 - ・ 日付指定するクエリーで一番使われる日付列を選ぶ
 - ・ できればこの日付列は更新が行われずデータ・ローディングや削除で使われる列が良い
 - ・ この列の更新が行われる場合はスキーマ・デザインに問題がないか再確認すること
- ・ サブパーティション
 - ・ CPU負荷が高い（結合負荷が高い）場合はHASHでハッシュ・ワイズ結合する
 - ・ IO負荷が高い場合はRANGEまたはLISTでさらにパーティション・プルーニングする

パーティショニングでよくある間違い

- 日付のRANGEパーティションが大きすぎる / 小さすぎる
 - クエリー時の最小単位やロード / 削除時の最小単位に合わせる
 - 1回のIO単位 (例: 1MB) より小さくしない
- HASHパーティション数を2の累乗 (2, 4, 8, 16...) 以外にする
 - 累乗にしないとサイズに偏りが発生する
 - パーティション数はCPU数の2倍が基本だが、複合パーティションの場合はパーティション数が1万を超えないように注意する
- データに偏りがある列でパーティションする
 - パーティションのサイズに偏りが出る

圧縮デザイン

- ・ 圧縮デザインはDWHデザインにおいて2番目に大事なポイント
- ・ 圧縮による影響
 - ・ 良い点: 表スキンの性能向上 / ディスク容量減
バックアップ要件低下
 - ・ 悪い点: UPDATE / DELETEが遅くなる
- ・ 理想的なデザインがされたスキーマであれば追記のみのもっとも大きな表を圧縮すればよい
 - ・ しかし現実には理想的なデザインである場合は少ない

OLTP

OLTP理論

- ・ ジム・グレイ
アンドレアス・ロイター
『トランザクション処理』
(日経BP) を読む
- ・ でかい、重い、売ってない

トランザクション処理 上
概念と技法

内容紹介

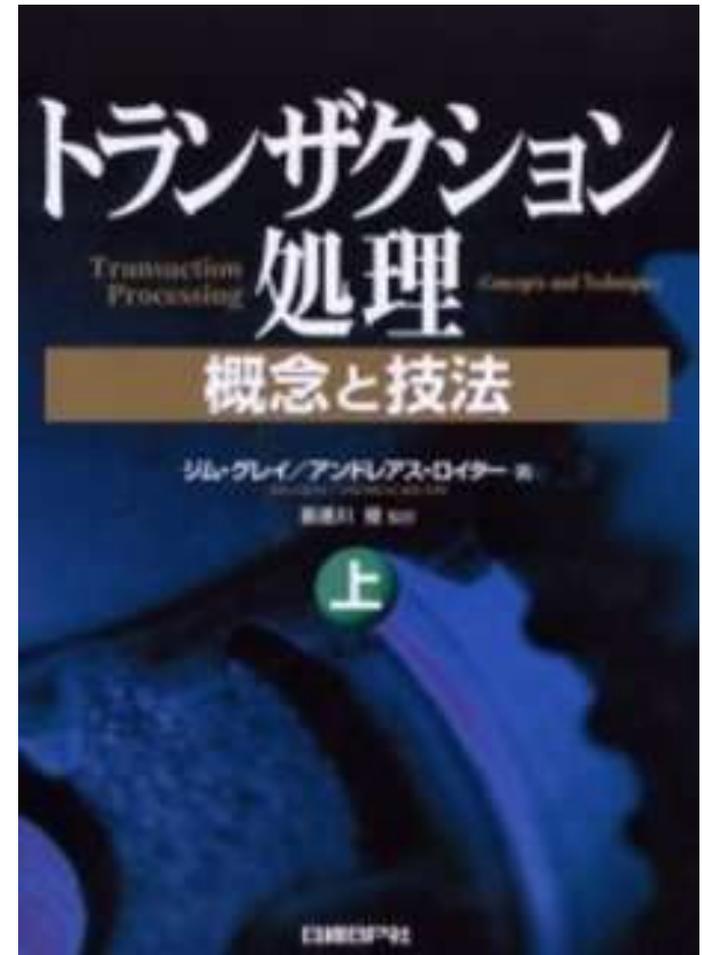
トランザクション処理の巨人、ジム・グレイ
在庫管理まで、さまざまな業務分野で用いら
れ、トランザクション処理の基礎、フォルトトレランス、ト
グ、コンカレンシー・コントロール、リカバ
ル、[『上巻』](#)・[『下巻』](#)、併せて1400ページの大著で

この商品は完売いたしました

[目次を見る](#)

[返品・解約について](#)

<http://ec.nikkeibp.co.jp/item/books/P81020.html>

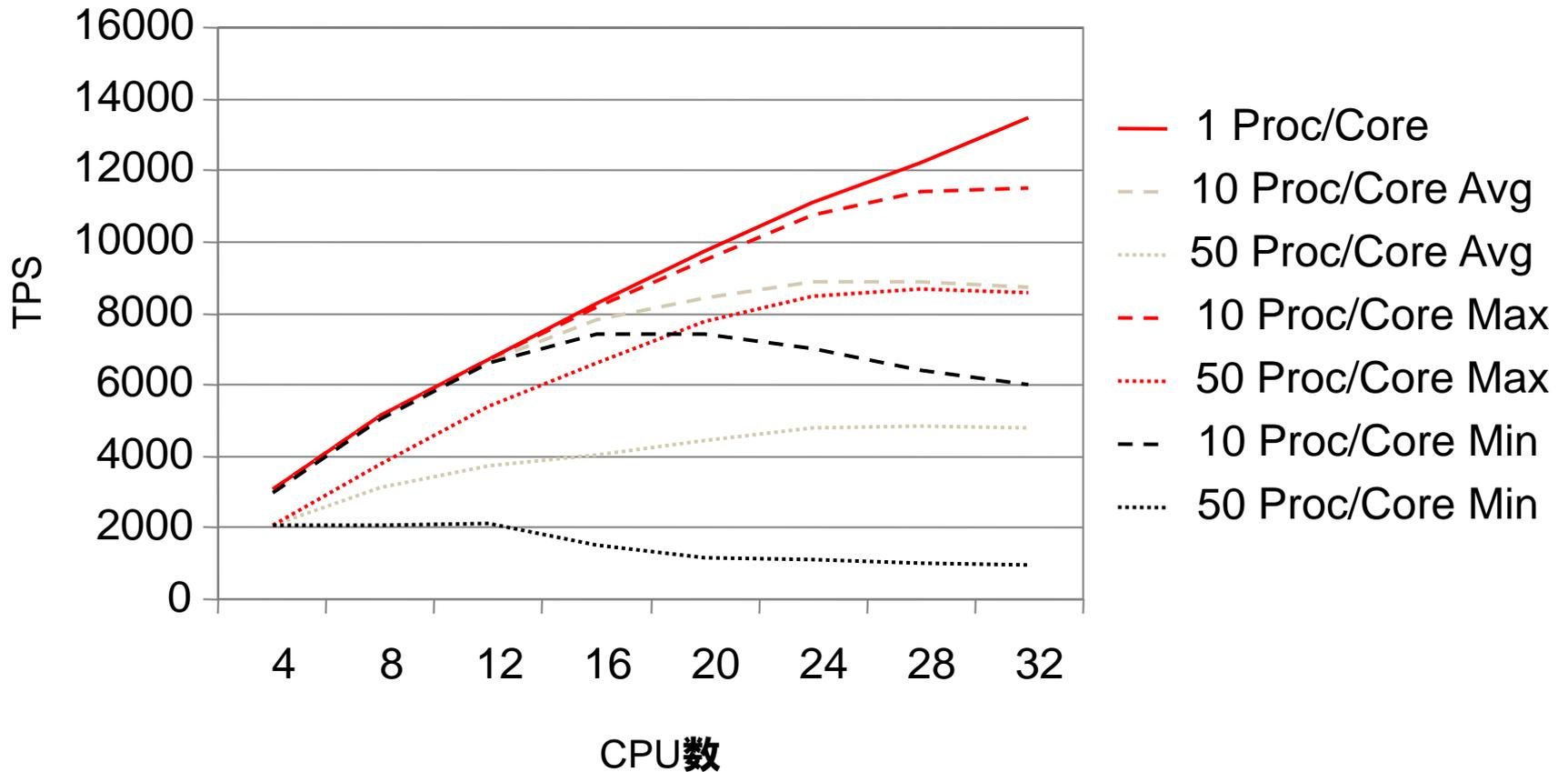


データベース・パフォーマンスの基本原則

- ・ Oracle Database はプロセス・ベース・アーキテクチャ
 - ・ 効率的にSQLを実行するためには
各プロセスがSQLが完了するまで
OSに効率的にスケジューリングされる必要がある
 - ・ プロセス数が多くてスケジューラされづらかったり
効率的でないSQLによって長時間スケジューラを押しえられたり
長時間なんらかにブロックされたりすると
データベース・パフォーマンスは劣化する
- ・ データベース・パフォーマンス・エンジニアは
CPU消費量の多いプロセスを探し
ブロッキング・イベントを排除することが仕事

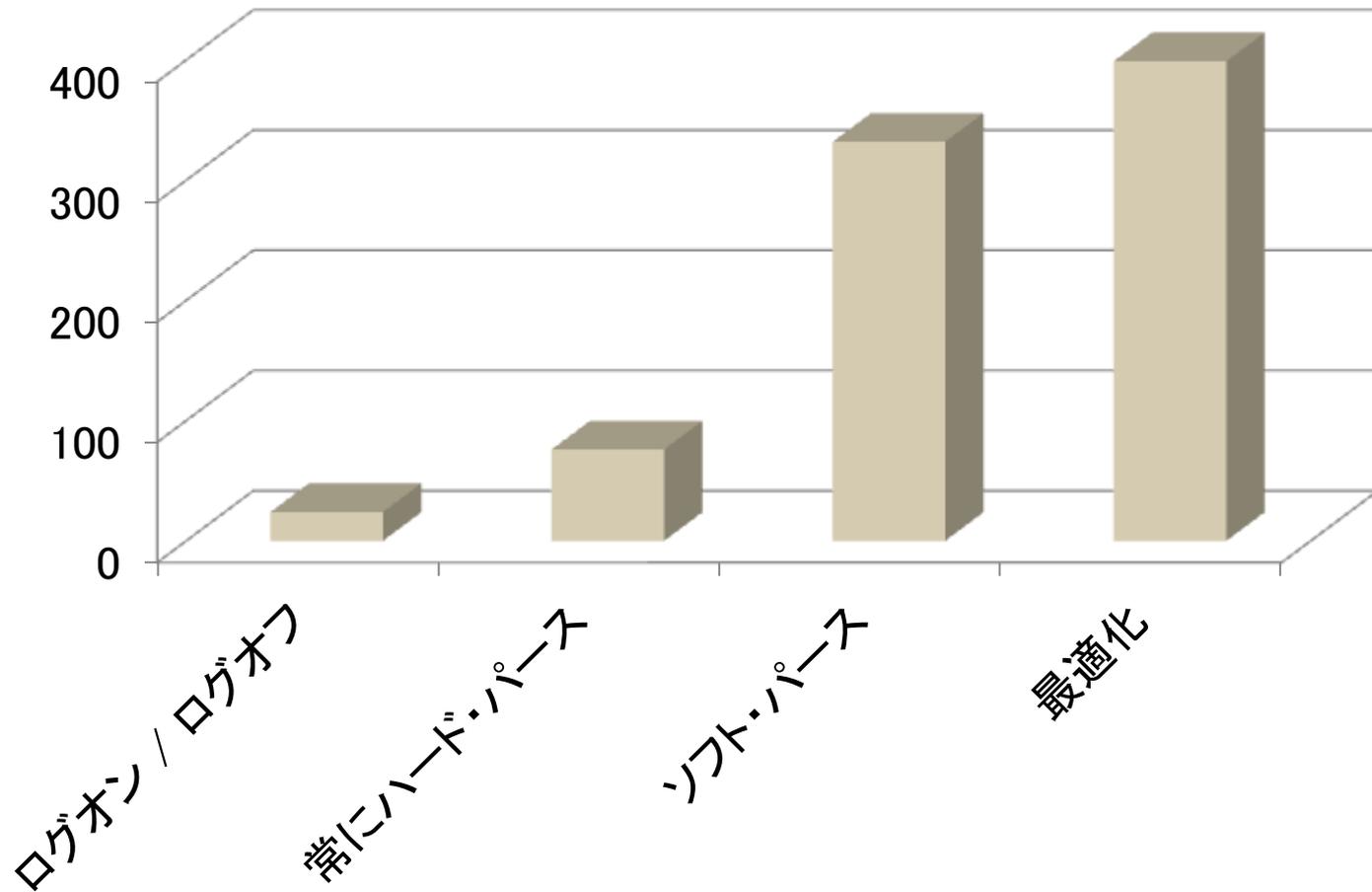
データベース・パフォーマンスの基本原則

コネクション数の影響



データベース・パフォーマンスの基本原則

パースのオーバーヘッド



メモリー ネットワーク ストレージ

Linuxでのメモリー管理

- ・ 物理メモリー
 - ・ お金を出して買ったアレ
- ・ 仮想メモリー
 - ・ OSが提供するメモリー空間
- ・ ページ・テーブル
 - ・ 物理メモリーと仮想メモリーをマッピングする
- ・ Hugepage
 - ・ ページ・テーブルをより効率的に管理できる

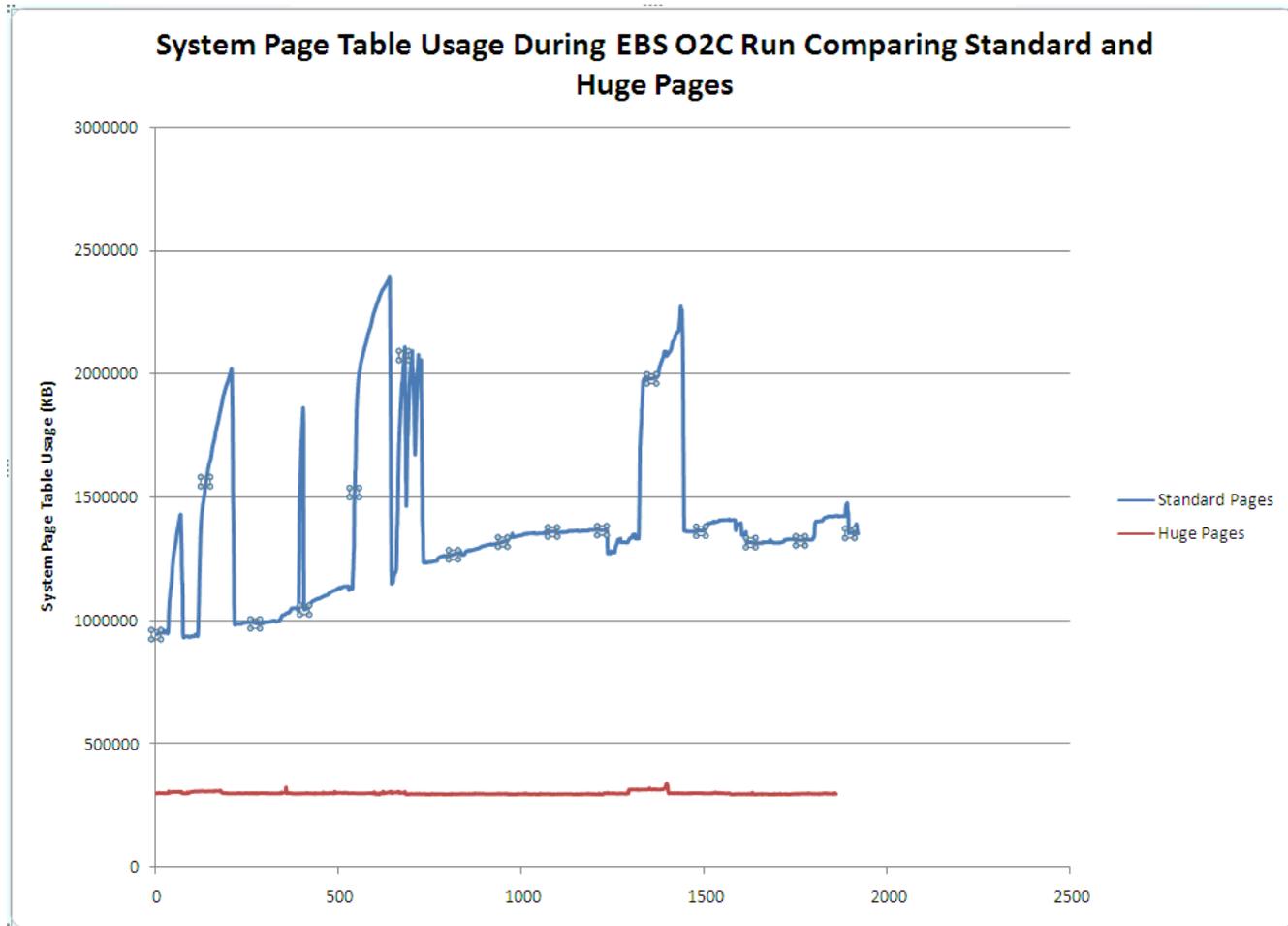
LinuxでのHugepage

- Linuxの仮想メモリーはページ単位で管理されている
- デフォルトのページ・サイズは4KB
- Hugepageは2MBに設定できる
 - コストが $4 \div 2048 = 512$ 分の1に減る
- Hugepageはメモリーにロックされ、ページアウトされない
 - 確保したHugepageを使うならメリット
 - 確保したHugepageを使わないならデメリット

Hugepageをいつ使うべきか

- Hugepageが適するシステム
 - OLTPシステム
 - キャッシュ・ヒット率がほぼ100%
 - 平均buffer_gets/executeが小さい
 - ソート量が少ない
 - 多くの場合に索引スキャンされる
 - SGAが32GB以上のシステム
- Hugepageが適さないシステム
 - データウェアハウス・システム

Hugepageの効果



LinuxでのHugepage

Hugepageを有効にするOSの設定

- Oracle Database 管理者リファレンス 11g リリース2(11.2) for Linux and UNIX-Based Operating Systems 参照
- 必要なページ数の見積もり
- `/etc/sysctl.conf` に `vm.nr_hugepages` の値を設定
- `sysctl -p`

LinuxでのHugepage

Hugepageを有効にするOSの設定

- `cat /proc/sys/vm/nr_hugepages`
50000
- `cat /proc/meminfo | grep -i hugepage`
HugePages_Total: 50000
HugePages_Free: 40990
HugePages_Rsvd: 38296
HugePages_Surp: 0
Hugepagesize: 2048 kB

LinuxでのHugepage

Hugepageを有効にする Oracle Database 11.2.0.2 の設定

- **USE_LARGE_PAGES**初期化パラメータ
 - **TRUE** (デフォルト)
 - ・ Hugepageが使用可能であれば使用する
 - **FALSE**
 - ・ Hugepageを使用しない
 - **ONLY**
 - ・ Hugepageを必ず使用する
使用できない場合はインスタンス起動がエラーになる
- 自動メモリー管理 (AMM) の**MEMORY_TARGET** / **MEMOERY_MAX_TARGET**初期化パラメーターとは併用できない

LinuxでのHugepage

Hugepageを有効にする Oracle Database 11.2.0.2 の設定

・ アラートログ

```
Starting ORACLE instance (normal)
***** Huge Pages Information *****
Huge Pages memory pool detected (total: 50000 free: 47397)
DFLT Huge Pages allocation successful (allocated: 17152)
NUMA Huge Pages allocation on node (7) (allocated: 13568)
NUMA Huge Pages allocation on node (6) (allocated: 13568)
Huge Pages allocation failed (free: 2754 required: 14336)
Allocation will continue with default/smaller page size
*****
```

ネットワーク

データベース・エンジニア vs ネットワーク・エンジニア

- ・ 同じ言語で話しているかに気をつけること
 - ・ Byte vs Bit / MB vs Mb
 - ・ 8b/10b
- ・ ネットワーク性能とは？
 - ・ ファイル / データ転送
 - ・ バンド幅
 - ・ レイテンシ
 - ・ Pingレスポンス時間

ネットワーク指標

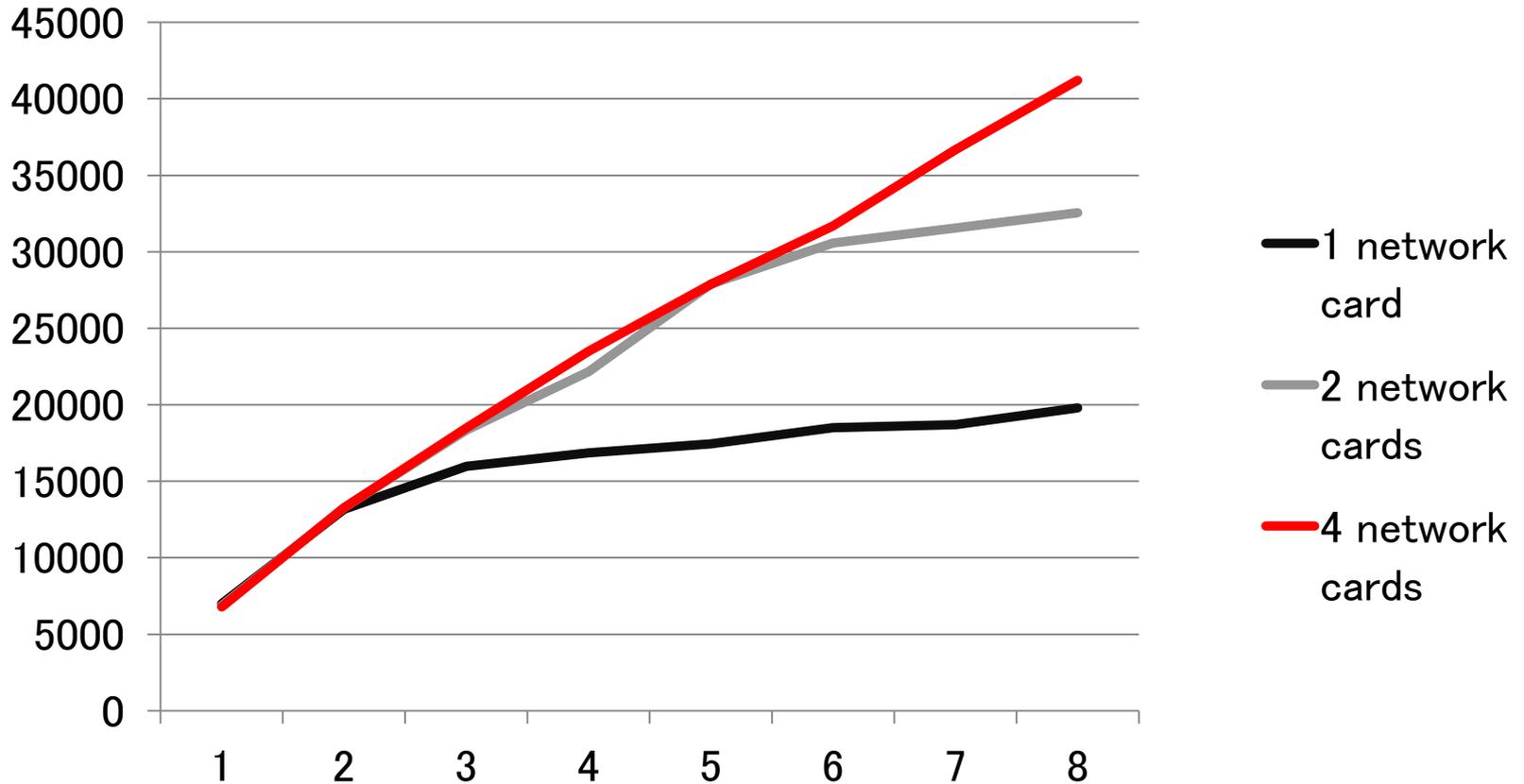
- バンド幅
 - ネットワークが転送できるバイト数
 - データ・コピーやマイグレーションでバンド幅がボトルネックになることがある
 - Oracle Net (SQL*Net) を使用したOLTPでバンド幅がボトルネックになることはほとんどない
- パケット数
 - ネットワークが耐えられる小さなメッセージ数
 - Chattyなアプリケーションでパケット数がボトルネックになることがある
 - SOA / ERP / ネットワーク経由で1行ごとに処理するバッチなど
 - 1GbEか10GbEかは処理できるパケット数に関係ない
関係するのはネットワーク・カードの性能

ネットワーク・パケット

- Oracle Net とRACのキャッシュ・フュージョンはChatty
- 1つのパケットは1つのOS割込みを引き起こす
 - ソフトウェア割込み (Soft IRQ) はOSカーネルによって処理される
 - ソフトウェア割込みはCPUリソースを使用する
 - 大量のソフトウェア割込みはCPUを使い切る
- 一般的なLinuxでは、秒間6万 Oracle Net ラウンドトリップで1つのCPUが使い切られる
- CPUがボトルネックになり、スループットが飽和する
帯域は関係ない

ネットワーク・パケット

ネットワーク・カードの枚数によるスループット比較



ボトルネックの確認

- ・ AWRからははっきりとは分からない
- ・ topでは分からない
- ・ Linuxではmpstatを使用する

```
mpstat -P ALL
```

CPU	%usr	%nice	%sys	%iowait	%soft	%idle
all	61.21	0.00	14.54	0.00	8.43	15.82
0	0.00	0.00	0.50	0.00	99.50	0.00
1	80.50	0.00	15.50	0.00	4.00	0.00

- ・ CPUは16%の余力があるように見えるが
実際にはスループットはこれ以上伸びない

ネットワークのまとめ

ワークロード	バンド幅の問題？	パケット数の問題？
LOB / Secure Files	Yes	No
Oracle Net	たぶん違う	Yes
RACキャッシュ・フュージョン	たぶん違う	Yes
RACインターノード・パラレル実行	Yes	No
Data Guard (同期)	たぶん違う	Yes
Data Guard (非同期)	Yes	たぶん違う

ストレージ

意識すべきアクセス性能

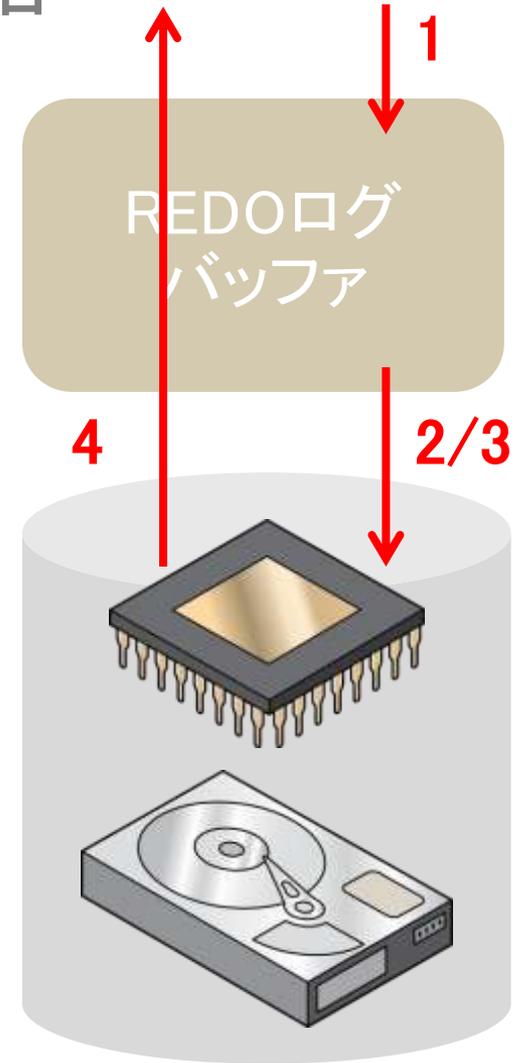
- ・ 理想的なブロック・アクセス性能

- ・ CPU2次キャッシュ 1ナノ秒
- ・ DRAMメモリー 1マイクロ秒
- ・ NANDフラッシュ (PCI接続) 10マイクロ秒
- ・ ディスク 5-10ミリ秒

REDOログおさらい

一般的なエンタープライズ・ストレージの場合

1. COMMITコール
 2. REDOログ・バッファから
REDOログ・ファイルに書き出される
 3. REDOエントリーは
ディスク・コントローラの
キャッシュに書き込まれる
ディスクのシリンダーではない
 4. COMMIT完了
Write Back
- ・ SSDにすれば
必ず速くなるわけではない



Exadataでのアプローチ

Exadata Smart Flash Log

1. COMMITコール
 2. REDOログ・バッファからREDOログ・ファイルに書き出される
 3. REDOエントリはディスク・コントローラのキャッシュとPCI接続フラッシュのそれぞれに書き込まれる
 4. いずれかが書き終わればCOMMIT完了
- ・ ディスク・コントローラとPCI接続フラッシュ
それぞれの異常値が排除され、レスポンス時間が安定する

ボトルネックの 監視と特定

ボトルネックと監視と特定

- ・ 自動ワークロード・リポジトリ (AWR) レポート
- ・ 自動データベース診断モニター (ADDM) レポート
- ・ アクティブ・セッション履歴 (ASH) レポート

- ・ Oracle Diagnostic Pack オプションが必要
- ・ Oracle Enterprise Manager から取得 / 管理が可能

自動ワークロード・リポジトリ (AWR) レポート

- ・ 特定期間内のデータベース・アクティビティ統計
- ・ HTML形式がオススメ
- ・ SQL*Plusから取得する場合

```
@?/rdbms/admin/awrrpt.sql
```

```
-- RAC全体での概要レポート
```

```
@?/rdbms/admin/awrgrpt.sql
```

```
-- 特定SQLのみのレポート
```

```
@?/rdbms/admin/awrsqrpt.sql
```

自動ワークロード・リポジトリ (AWR) レポート

Firefox

WORKLOAD REPOSITORY report for

DB Name	DB Id	Instance	Inst num	Startup Time	Release	RAC
DEMO	3489592566	demo1	1	24-Mar-11 12:03	11.2.0.2.0	YES

Host Name	Platform	CPUs	Cores	Sockets	Memory (GB)
awa01.jp.oracle.com	Linux x86 64-bit	16	8	2	47.15

	Snap Id	Snap Time	Sessions	Cursors/Session
Begin Snap:	425	24-Mar-11 18:10:14	54	.8
End Snap:	426	24-Mar-11 18:10:52	54	.8
Elapsed:		0.63 (mins)		
DB Time:		10.66 (mins)		

Report Summary

Cache Sizes

	Begin	End		
Buffer Cache:	6,224M	6,224M	Std Block Size:	8K
Shared Pool Size:	1,344M	1,344M	Log Buffer:	9,500K

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	16.9	25.6	0.54	1.39
DB CPU(s):	0.9	1.3	0.03	0.07
Redo size:	40,155.1	60,960.3		
Logical reads:	6,736.2	14,021.0		

自動データベース診断モニター (ADDM) レポート

- ・ パフォーマンス問題の原因分析と問題を修正するための推奨案の提供
- ・ ADDMレポートはAWRLレポートと同時に取得すべき (デフォルト)
- ・ SQL*Plusから取得する場合

```
@?/rdbms/admin/addmrpt.sql
```

自動データベース診断モニター (ADDM) レポート

ADDM Report for Task 'TASK_42'

Analysis Period

AWR snapshot range from 1 to 9.
Time period starts at 11-10-26 13:00:14
Time period ends at 11-10-26 21:00:20

Analysis Target

Database 'EXAMPLE' with DB ID 2472155583.
Database version 11.2.0.2.0.
ADDM performed an analysis of instance EXAMPLE1, numbered 1 and hosted at
foo01.jp.oracle.com.

Activity During the Analysis Period

Total database time was 1378 seconds.
The average number of active sessions was .05.

Summary of Findings

Description	Active Sessions Percent of Activity	Recommendations
1 Top SQL Statements	.05 96.43	1
2 Undersized PGA	.01 29.72	0

アクティブ・セッション履歴 (ASH) レポート

- ・ 短い期間で発生する一時的な問題の分析にAWRより有効
- ・ セッション単位 / モジュール単位での分析が可能
- ・ SQL*Plusから取得する場合

```
@?/rdbms/admin/ashrpt.sql
```

アクティブ・セッション履歴 (ASH) レポート

Firefox

ASH Report For EXAMPLE/EXAMPLE1

DB Name	DB Id	Instance	Inst num	Release	RAC	Host
EXAMPLE	2472155583	ExAMPLE1	1	11.2.0.2.0	YES	foo01.jp.oracle.com

CPUs	SGA Size	Buffer Cache	Shared Pool	ASH Buffer Size
24	14,336M (100%)	12,320M (85.9%)	1,920M (13.4%)	48.0M (0.3%)

	Sample Time	Data Source
Analysis Begin Time:	26-Oct-11 21:51:48	V\$ACTIVE_SESSION_HISTORY
Analysis End Time:	26-Oct-11 22:06:49	V\$ACTIVE_SESSION_HISTORY
Elapsed Time:	15.0 (mins)	
Sample Count:	2,726	
Average Active Sessions:	3.03	
Avg. Active Session per CPU:	0.13	
Report Target:	None specified	

ASH Report

- Top Events
- Load Profile
- Top SQL
- Top PL/SQL
- Top Java
- Top Call Types
- Top Sessions
- Top Objects/Files/Latches
- Activity Over Time

最後に

パフォーマンス・エンジニアとは

- ニセモノのパフォーマンス・エンジニア
 - Google検索してチューニング
 - 他システムでうまくいった方法そのままにチューニング
 - 言い伝えに基づいたチューニング
- 本物のパフォーマンス・エンジニア
 - パフォーマンスとは、すべてのワークロードが期間内に終わることであることを理解し、なんらかの対処をする前にどこで時間がかかっているかを論理的に調査する
 - チューニングとは、時間がかかっている場所を論理的に見つけだし、かかっている時間を減らす作業

チューニングのライフサイクル

1. パフォーマンス問題の発見
2. 問題の範囲を特定
3. ゴールの設定
4. 適切なデータの取得
5. データベース時間の配分の調査
 - ・ 最も性能向上しそうな部分の特定
6. 最も性能向上するチューニング方法でシステムを変更
7. ゴールが達成できたかどうかの検証
 - ・ 達成できていなかったら4から繰り返す



OTNセミナーオンデマンド

コンテンツに対する
ご意見・ご感想を是非お寄せください。

OTNオンデマンド 感想



http://blogs.oracle.com/oracle4engineer/entry/otn_ondemand_questionnaire

上記に簡単なアンケート入力フォームをご用意しております。

セミナー講師/資料作成者にフィードバックし、
コンテンツのより一層の改善に役立てさせていただきます。

是非ご協力をよろしくお願いいたします。

OTNセミナーオンデマンド

日本オラクルのエンジニアが作成したセミナー資料・動画ダウンロードサイト

掲載コンテンツカテゴリ(一部抜粋)

Database 基礎

Database 現場テクニック

Database スペシャリストが語る

Java

WebLogic Server/アプリケーション・グリッド

EPM/BI 技術情報

サーバー

ストレージ



超入門! Oracle データベースって何
再生時間: 60分

100以上のコンテンツをログイン不要でダウンロードし放題

データベースからハードウェアまで充実のラインナップ

毎月、旬なトピックの新作コンテンツが続々登場

例えばこんな使い方

- 製品概要を効率的につかむ
- 基礎を体系的に学ぶ/学ばせる
- 時間や場所を選ばず(オンデマンド)に受講
- スマートフォンで通勤中にも受講可能



毎月チェック!



コンテンツ一覧 はこちら

<http://www.oracle.com/technetwork/jp/ondemand/index.html>

新作&おすすめコンテンツ情報 はこちら

<http://oracletech.jp/seminar/recommended/000073.html>

OTNオンデマンド



オラクルエンジニア通信

オラクル製品に関わるエンジニアの方のための技術情報サイト

オラクルエンジニア通信 - 技術資料、マニュアル、セミナー

Oracleエンジニアのための技術情報サイト by Oracle Japan

新着情報を知りたい

技術資料を探したい

セミナーを受けたい

About

Oracleエンジニアの方がスキルアップしていただくために、厳選した情報をお届けしています

技術資料



インストールガイド・設定チュートリアルetc. 欲しい資料への最短ルート

アクセスランキング



他のエンジニアは何を見ているのか？人気資料のランキングは毎月更新

特集テーマ Pick UP



性能管理やチューニングなど月間テーマを掘り下げて詳細にご説明

技術コラム



SQLスクリプト、索引メンテナンスetc. 当たり前運用/機能が見違える!?

<http://blogs.oracle.com/oracle4engineer/>

オラクルエンジニア通信



The screenshot shows the top section of the oracletech.jp website. On the left is the 'oracletech.jp' logo with the tagline '好奇心が、エンジニア人生を豊かにする。'. On the right is the 'ORACLE' logo, a search bar, and social media icons for Twitter, Facebook, LinkedIn, YouTube, and RSS. Below these is a red navigation bar with five buttons: '製品/技術情報', 'スキルアップ', 'セミナー', 'キャンペーン', and 'ちょっと一息'.

製品/技術
情報



Oracle Databaseっていくら？オプション機能も見積れる簡単ツールが大活躍

セミナー



基礎から最新技術までお勧めセミナーで自分にあった学習方法が見つかる

スキルアップ



ORACLE MASTER ! 試験頻出分野の模擬問題と解説を好評連載中

Viva!
Developer



全国で活躍しているエンジニアにスポットライト。きらりと輝くスキルと視点を盗もう

<http://oracletech.jp/>

oracletech



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



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
ご登録の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

Hardware and Software Engineered to Work Together

ORACLE®