# Oracle Database経験者が
# MySQLの設計思想を知っていろいろ考える会

Oracle OpenWorld Unconference presented by JPOUG

2012/04/06 平塚 貞夫

# 自己紹介

- DBエンジニアやってます。専門はOracleとMySQL。
  - システムインテグレータで主にRDBMSのトラブル対応をしています。
  - 仕事の割合はOracle：MySQL＝8：2ぐらいです。

- Twitter：@sh2nd
- はてな：id:sh2

- 写真は実家で飼っているミニチュアダックスのオス、アトムです。

# 本日のお題

- 簡単な負荷テストツールを使って、Oracle DatabaseとMySQLの性能特性の違いを説明していきます。
- Oracle Database経験者の方に、「えっ、MySQLってそんなふうになってるんだ…」と気づいていただければ幸いです。
- MySQL経験者の方に、「えっ、Oracle Databaseってそんなことしてるんだ…」と気づいていただければ幸いです。

# MySQLについて

# MySQLの歴史

- 2001/01 MySQL 3.23
  RHEL 2.1/3に付属するバージョンです。

- 2003/03 MySQL 4.0
  UNIONに対応。

- 2004/10 MySQL 4.1
  サブクエリに対応。国際化対応、特に4.1.12からWindowsのShift_JIS拡張(Microsoft Code Page 932)に対応。RHEL 4に付属するバージョンです。

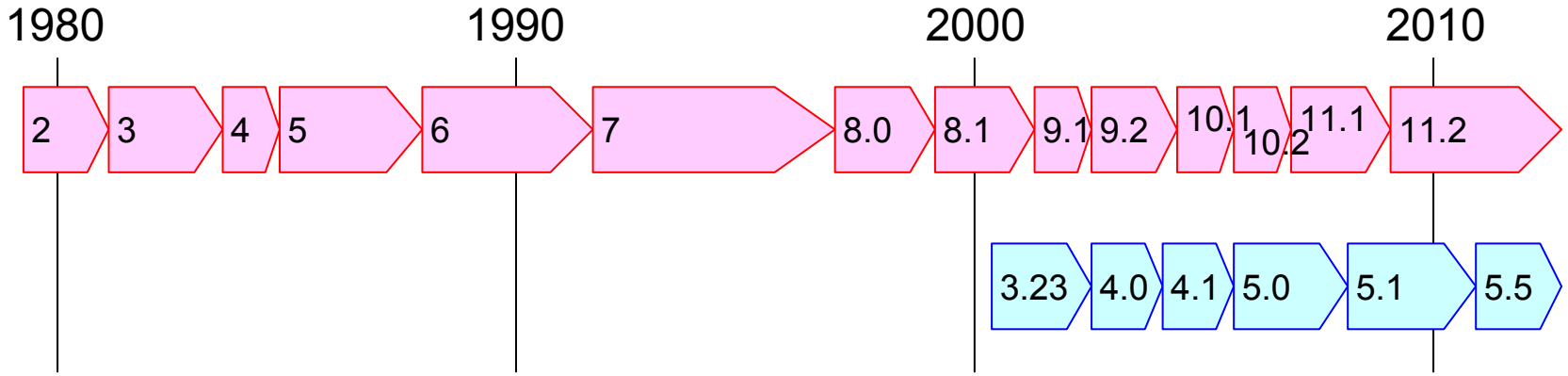
- 2005/10 MySQL 5.0
  ストアドプロシージャ、ビュー、トリガに対応。RHEL 5に付属するバージョンです。

- 2008/11 MySQL 5.1
  パーティショニング、イベントスケジューラ、ストレージエンジンプラグインに対応。RHEL 6に付属するバージョンです。

- 2010/12 MySQL 5.5
  準同期レプリケーション、4バイトUTF-8に対応。デフォルトストレージエンジンがInnoDBに変更、スケーラビリティが向上。現在の最新バージョンです。

# Oracle Databaseとの機能比較

- 機能的にはOracle 7～8に相当します。




- Oracle Database

3 トランザクション
4 読み取り一貫性
5 クライアントサーバ
6 行レベルロック、オンラインバックアップ、パラレルサーバ
7 ハッシュ結合、ストアドプロシージャ、トリガ、ジョブスケジューラ、CBO、ビットマップ索引、ヒストグラム統計
8.0 パーティショニング、索引構成表、逆キー索引
8.1 ローカル管理表領域、ファンクション索引、Statspack
9.1 自動セグメント領域管理、バインド・ピーク、AL32UTF8
9.2 ローカル管理SYSTEM表領域
10.1 RBO非サポート、自動オプティマイザ統計収集、AWR
10.2 透過的データ暗号化
11.1 適応カーソル共有、データベース常駐接続プーリング
11.2 オプティマイザ・フィードバック

- MySQL

4.0 UNION
4.1 サブクエリ
5.0 ストアドプロシージャ、ビュー、トリガ
5.1 パーティショニング、イベントスケジューラ
5.5 4バイトUTF-8

# パラメータのバインド機構について

# パラメータのバインド機構

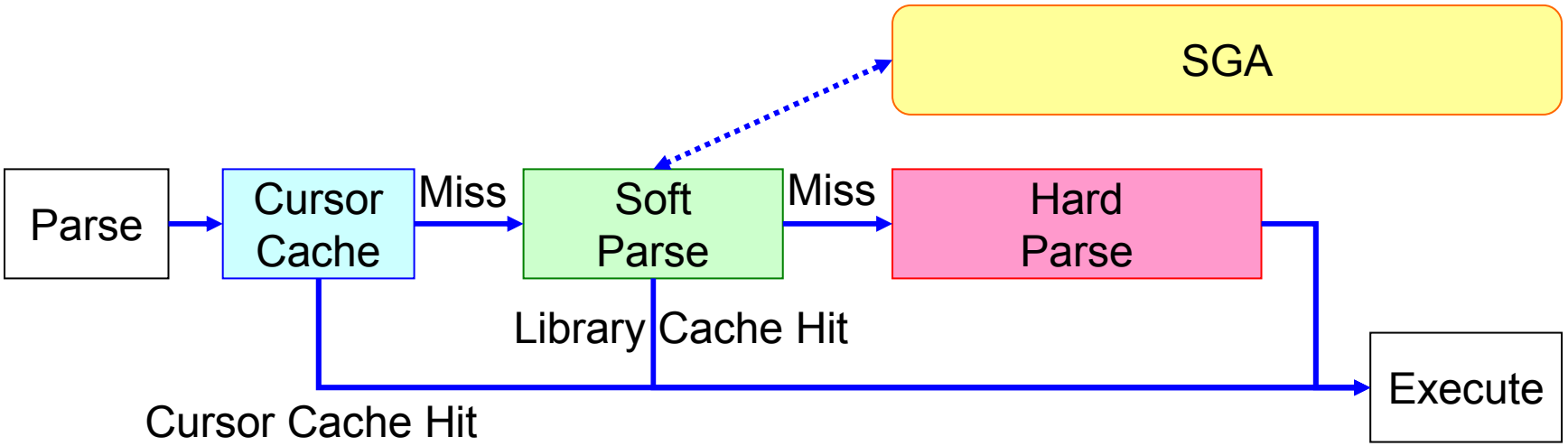
- OLTPでSQLを実行する際はパラメータのバインド機構を利用するべきという話は、みなさんよくご存知だと思います。
- パラメータのバインド機構を利用することで、SQLインジェクションの防止、およびSQL解析コストの低減が見込まれます。
- というか利用しないとORA-4031が発生してひどい目に遭います。
- Javaの場合は、PreparedStatementクラスを用いて以下のように記述します。

```
■パラメータのバインド機構を利用する
preparedStatement = connection.prepareStatement("SELECT c FROM sbtest WHERE id = ?");
preparedStatement.setInt(1, getRandomId()); // パラメータをバインドする
resultSet = preparedStatement.executeQuery();

■パラメータのバインド機構を利用しない
statement = connection.createStatement();
resultSet = statement.executeQuery("SELECT c FROM sbtest WHERE id = " + getRandomId());
```

# SQL解析フェーズの流れ

- Oracle DatabaseのSQL解析フェーズにおいて、パラメータのバインド機構がどのような影響を与えるのかを復習しておきます。

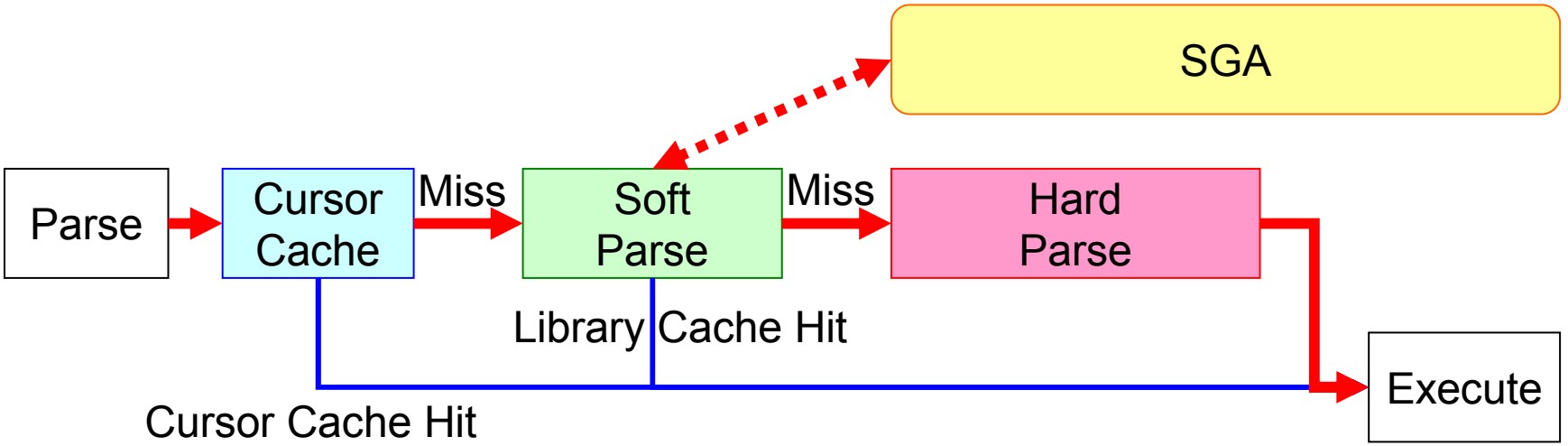
- 以下はOracle DatabaseにおけるSQL解析フェーズの流れを図示したものです。




SQL解析フェーズの流れ

# パラメータのバインド機構を利用しない場合

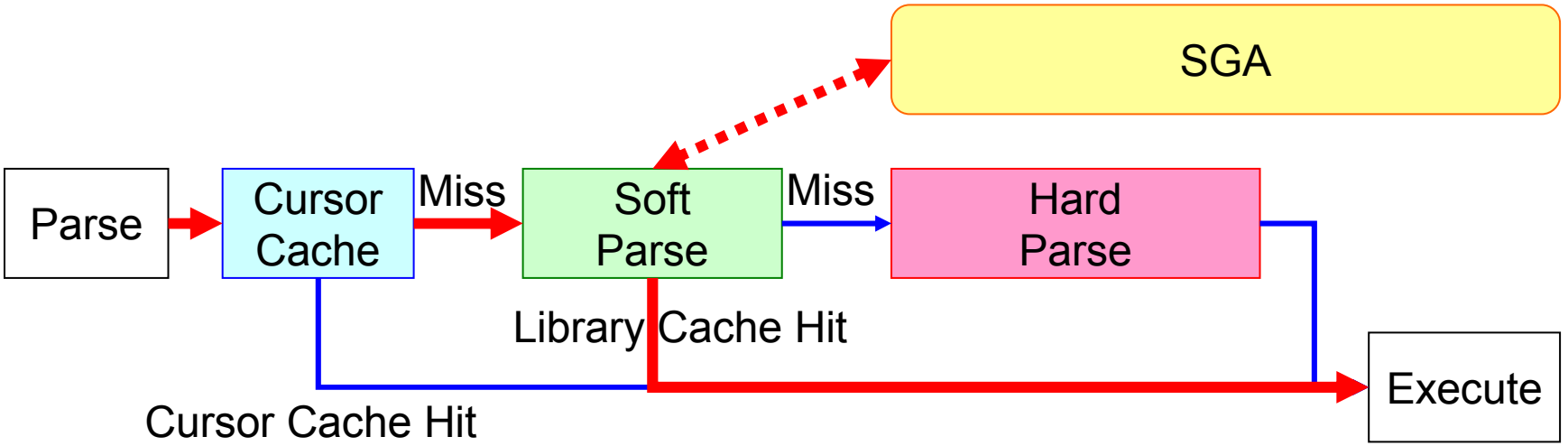
- パラメータのバインド機構を利用しない場合は、ライブラリキャッシュに同一のSQL文がキャッシュされている確率が著しく低下するため、ほとんどのSQLに対してHard Parseが行われます。




```
■Load Profile (Per Second)
Parses:                2,131.6
Hard parses:           2,089.3

■Instance Efficiency Percentages
Library Hit %:                 43.30     Soft Parse %:          1.98
Execute to Parse %:             0.15     Latch Hit %:          98.95
Parse CPU to Parse Elapsd %:   48.35     % Non-Parse CPU:      35.32
```

# パラメータのバインド機構を利用する場合

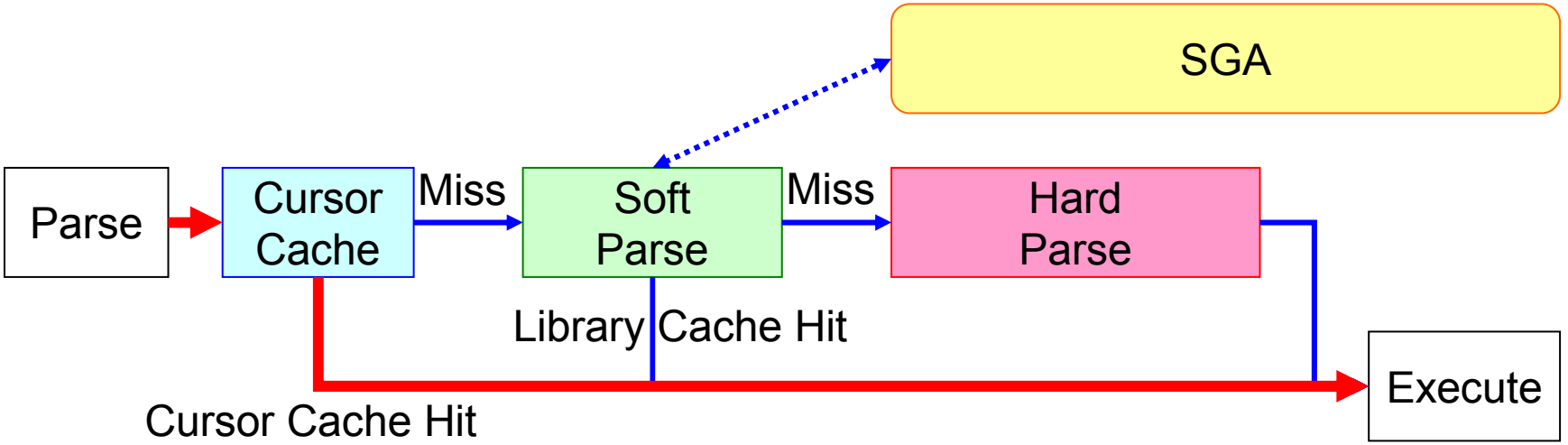
- パラメータのバインド機構を利用する場合は、ライブラリキャッシュに同一のSQL文がキャッシュされている確率が高くなるため、多くのSQLはSoft Parseまでで解析が完了します。




```
■Load Profile (Per Second)
Parses:                6,350.8
Hard parses:               0.1

■Instance Efficiency Percentages
Library Hit %:                 100.00    Soft Parse %:          100.00
Execute to Parse %:              0.01    Latch Hit %:            99.94
Parse CPU to Parse Elapsd %:    82.69    % Non-Parse CPU:        93.37
```

# PreparedStatementプールを利用する場合

- コネクションプールが提供するPreparedStatementプール機能、あるいはセッション・カーソル・キャッシュを利用すると、一度使用したカーソルを再利用することができます。この場合はSQLの解析自体がスキップされます。




```
■Load Profile (Per Second)
Parses:                  11.0
Hard parses:              0.4

■Instance Efficiency Percentages
Library Hit %:                  99.96     Soft Parse %:          96.49
Execute to Parse %:             99.85     Latch Hit %:           99.93
Parse CPU to Parse Elapsd %:    35.40     % Non-Parse CPU:       99.96
```

# 負荷テスト

# 負荷テストの構成

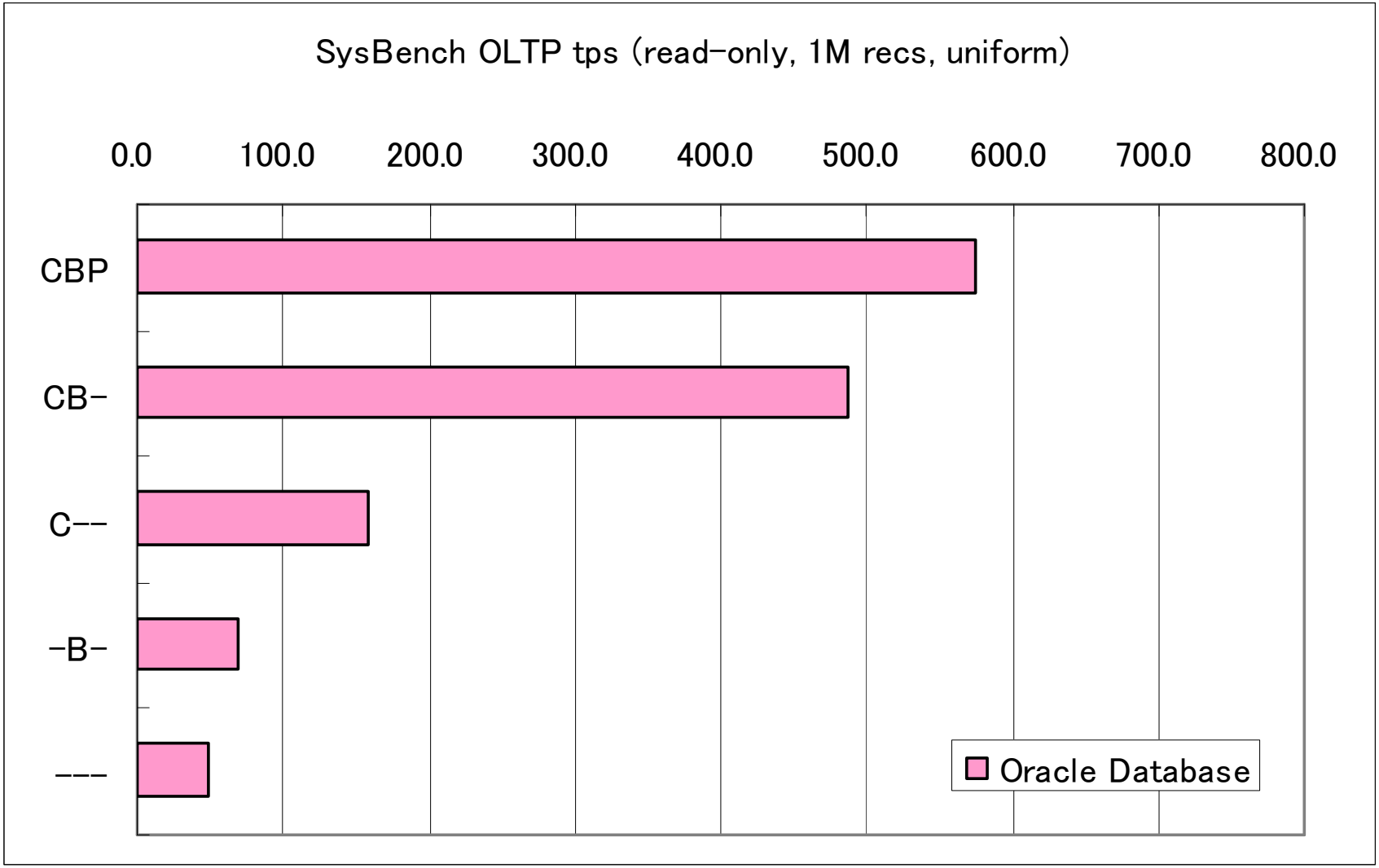
- SysBench OLTPテストをJavaに移植して、機能追加をしました。
  - 単一テーブルに対する読み取りをマルチスレッドで実行します。
  - Oracle DatabaseとMySQLでまったく同じコードが動作します。
  - コネクションプールの利用有無を指定できます。(Apache Commons DBCP)
  - パラメータのバインド機構の利用有無を指定できます。
  - PreparedStatementプール機能の利用有無を指定できます。

- テスト環境
  - Scientific Linux 6.2 64bit KVM上のCentOS 5.7 64bit
  - Core i5-2400S (Quad-Core、2.50GHz)から2コアを割り当て
  - Oracle Database 11g R2、SGA_TARGET = 4G
  - MySQL 5.5.22、innodb_buffer_pool_size = 1024M

- テスト項目の見方
  - C:コネクションプールの利用有無
  - B:パラメータのバインド機構の利用有無
  - P:PreparedStatementプール機能の利用有無

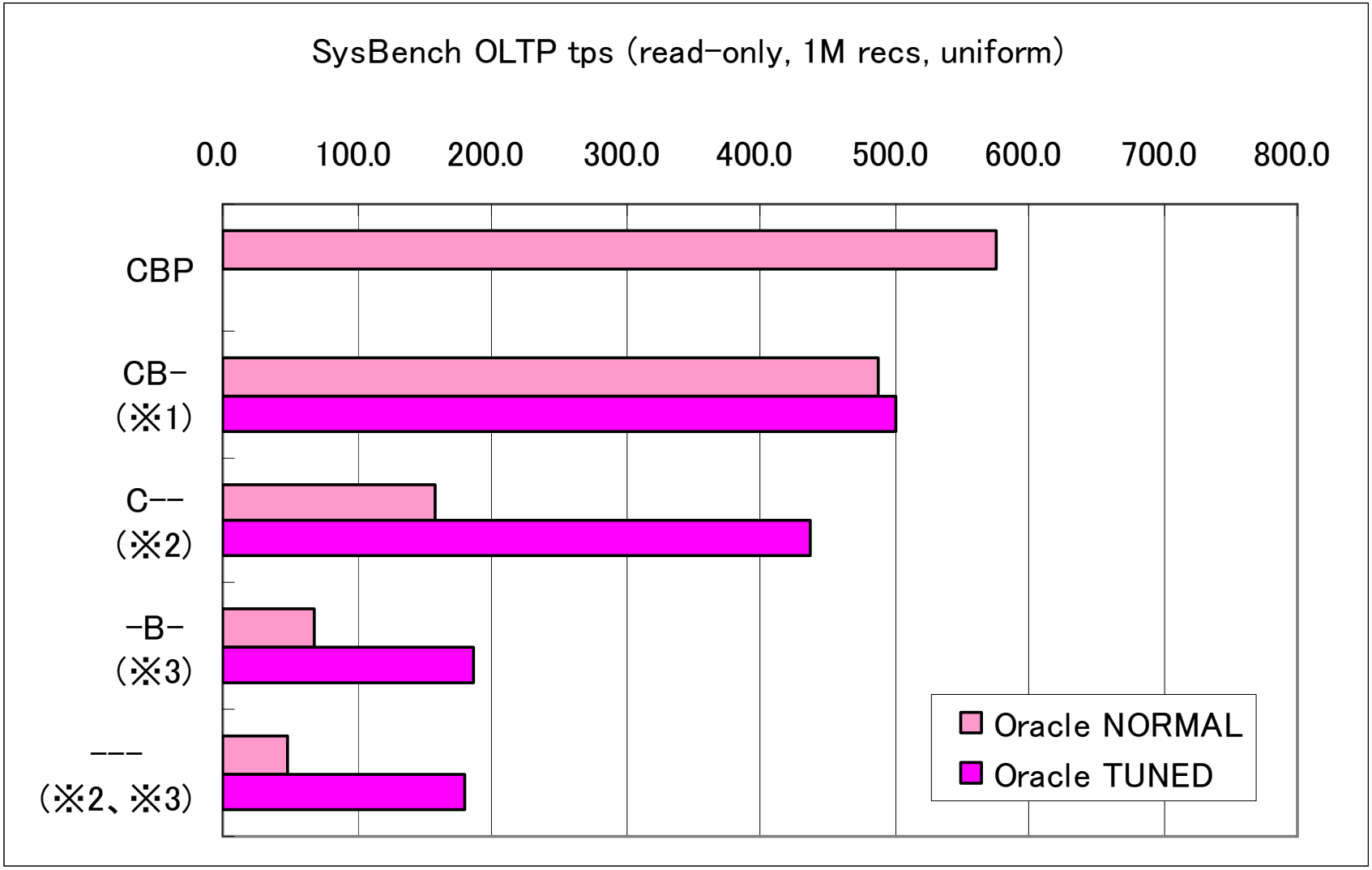
# Oracle Databaseの場合

- 結構ひどいですが、おおむね予想通りかと思います。



# Oracle Databaseの場合 チューニング

- ちょっとひどすぎるので、少しチューニングしておきます。



# Oracle Databaseのチューニング項目

- ※1 SESSION_CACHED_CURSORS = 50
  一度使用したカーソルをPGAにキャッシュする機能です。これによってCB-の結果がCBPにある程度近づきますが、あまり効果は大きくないようです。
  SESSION_CACHED_CURSORSは11g R1からデフォルト値が50になっています。比較のためにチューニング前は0に設定しています。

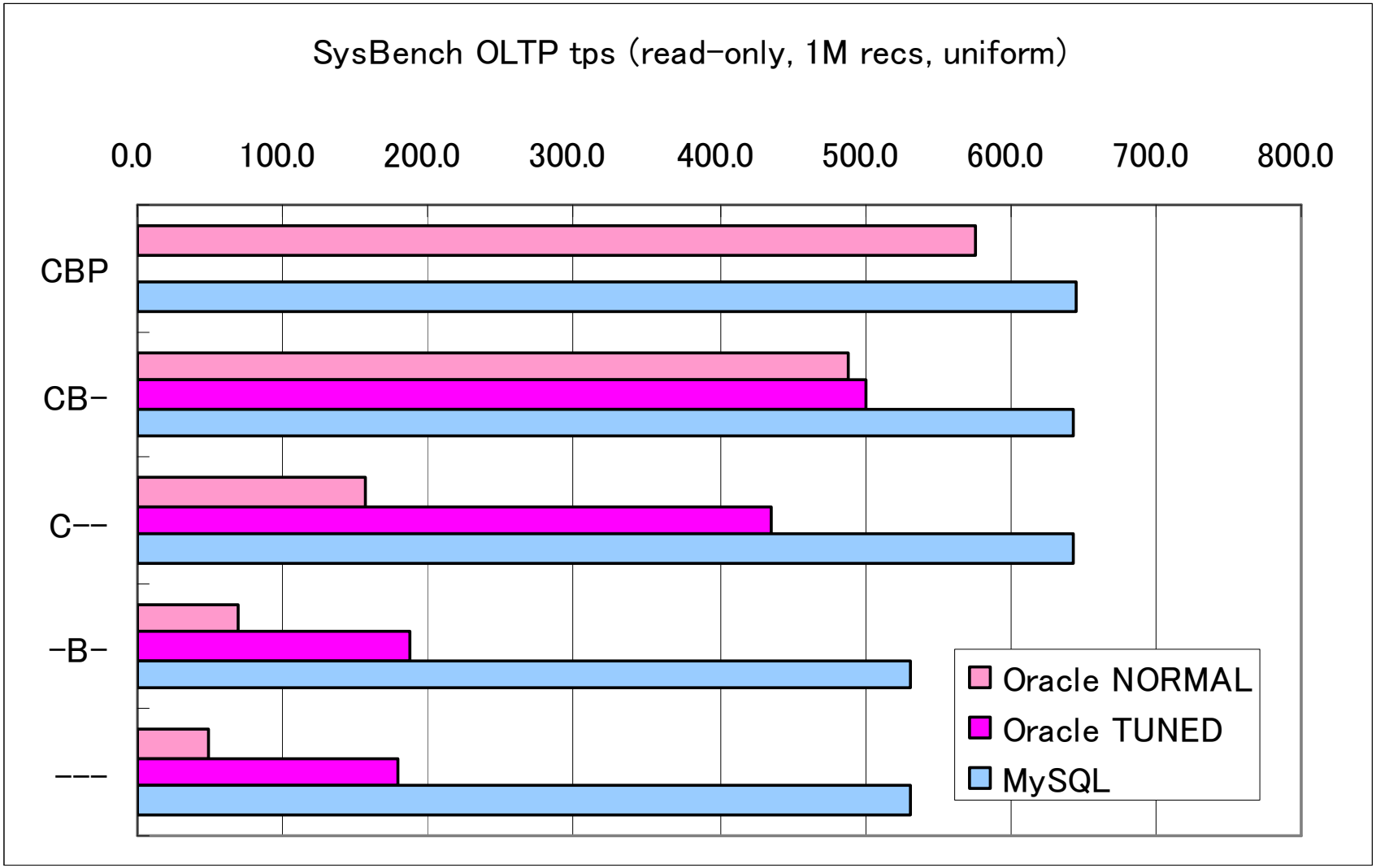
- ※2 CURSOR_SHARING = FORCE
  類似したSQL同士でカーソルを共有することでHard Parseを抑える機能です。劇的な効果があります。ただ類似判定をするためにある程度のオーバーヘッドが発生するので、CB-には追いつきません。昔から不具合が多いと言われている機能ですが、個人的にはそろそろFORCEがデフォルトになってほしいと思っています。

- ※3 共有サーバ構成
  Oracle Databaseの専用サーバ構成は接続に重量級のプロセス生成を伴うため、非常に遅いです。共有サーバ構成にすることである程度負荷を軽減できます。

- ※3については11g R1新機能のデータベース常駐接続プーリングが利用できる環境であれば、もう少し改善できると考えられます。(Javaからは利用できません)

# MySQLの場合

- 不思議な結果が出ました。



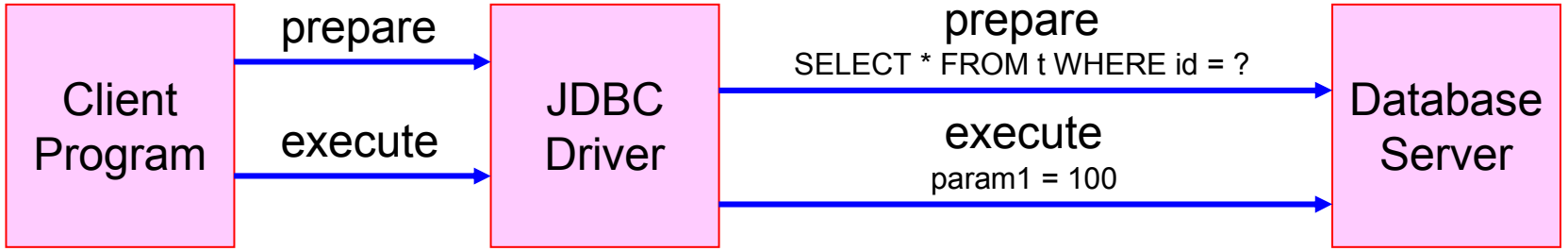
# MySQLの性能特性

- PreparedStatementプール機能の利用有無は、性能にまったく影響しません。
- パラメータのバインド機構の利用有無は、性能にまったく影響しません。
- コネクションプールの利用有無が性能に与える影響は、軽微です。

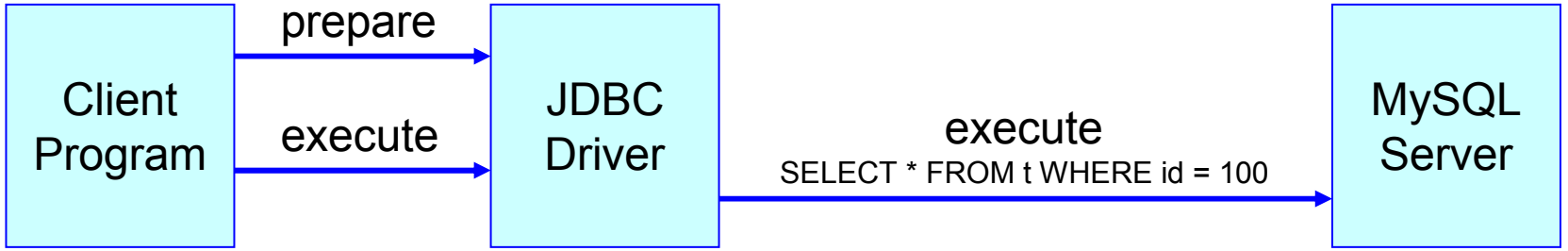
# 解説

# Client-Side PreparedStatement

- MySQLのJDBCドライバは、パラメータのバインド機構をクライアント側でエミュレーションしています。Client-Side PreparedStatementと呼ばれている仕組みです。




一般的なRDBMS




MySQL

- そのためパラメータのバインド機構を利用した場合でも、MySQLサーバにはJDBCドライバによってリテラル化されたSQLが送信されます。つまりパラメータのバインド機構はクライアント側のオーバーヘッドになるだけであって、サーバ側には何の変化ももたらしません。

# シンプルなSQLオプティマイザ

- MySQLがSQL実行のたびに実行計画生成を行っていてどうして速いのかというと、これはもうSQLオプティマイザがシンプルであるためという一言につきます。

- NESTED LOOPS JOINしかできません。
  MERGE JOINやHASH JOINはありません。そのため大きなテーブル同士を結合して大量のデータを返すクエリは、何日かかっても終わらないことがあります。今後MySQL 5.6においてBatched Key Accessという、MERGE JOINに近い効果をもたらすアルゴリズムが導入される予定です。またフォークプロジェクトのMariaDB 5.3では、すでにHASH JOINが導入されています。

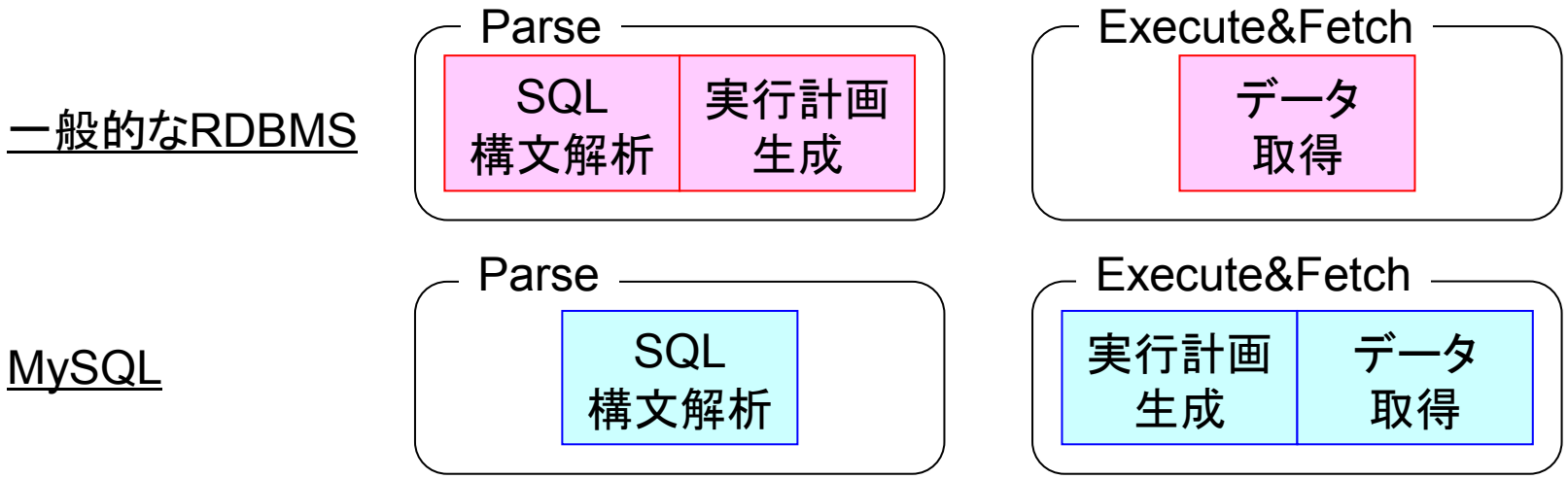
- クエリー・リライトはほとんどできません。
  Oracle Databaseで行うようなサブクエリのネスト解除などは、MySQLではできません。今後MySQL 5.6においてTable Pulloutという機能が導入され、以下のような例でOracle Databaseと同じクエリー・リライトができるようになる予定です。

```
SELECT * FROM sales WHERE cust_id IN (SELECT cust_id FROM customers);

⇒ MySQLはsalesテーブルから1件取得するごとにcustomerテーブルをフルスキャンします。

⇒ Oracle Databaseはこのクエリを以下のようにリライトします。
   SELECT sales.* FROM sales s INNER JOIN customers c ON s.cust_id = c.cust_id;
```

# Server-Side PreparedStatement

- MySQLサーバにはパラメータのバインド機構がないというわけではなくて、MySQL 4.1以降であればバインド機構はあります。JDBCドライバにおいても接続プロパティuseServerPrepStmtsにtrueを設定すれば利用することは可能です。

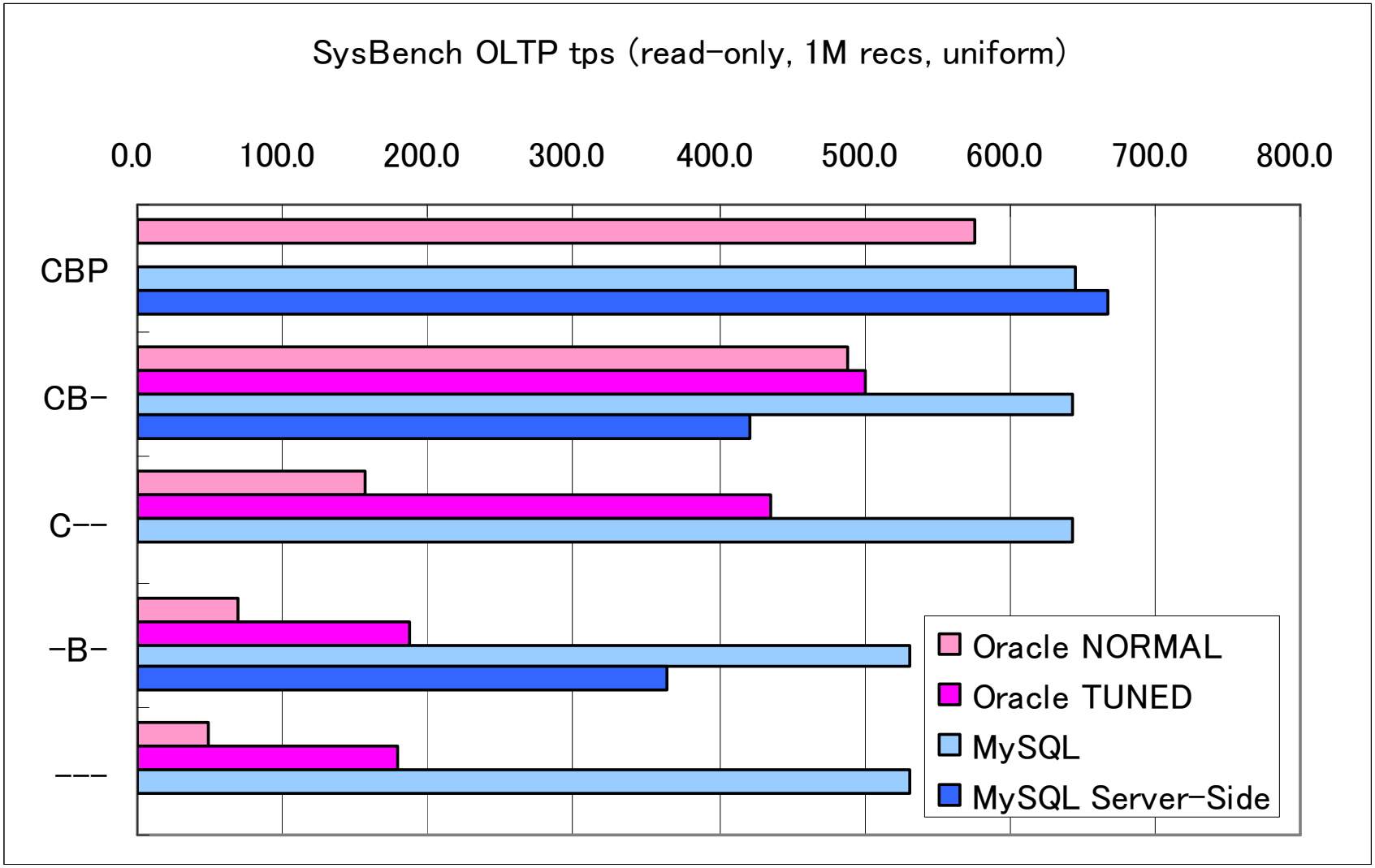
- しかしMySQLはSQL解析フェーズにおいて構文解析しか行わず、SQL実行フェーズで都度実行計画生成を行うアーキテクチャになっています。そのため事前にSQL解析フェーズを済ませておくことによる性能上のメリットはごくわずかで、MySQLサーバとの通信が2往復することのデメリットがメリットを上回っています。




- なおSQL実行フェーズで都度実行計画生成を行うアーキテクチャになっていることから、MySQLにはOracle Databaseで言うバインド・ピークの問題は存在しません。そもそも、MySQLにはライブラリキャッシュに相当する機構がありません。

# MySQLの場合 Server-Side PreparedStatement

- PreparedStatementプール機能と組み合わせた場合にのみ、性能が向上します。



# 索引構成表

- MySQL(InnoDB)は全てのテーブルが索引構成表となります。仕様です。製品ごとの違いをまとめておきます。
  - Oracle Database：両方サポート。デフォルトはヒープ構成表。
  - IBM DB2：両方サポート。デフォルトはヒープ構成表。
  - Microsoft SQL Server：両方サポート。デフォルトは索引構成表。
  - PostgreSQL：ヒープ構成表のみサポート。
  - MySQL(InnoDB)：索引構成表のみサポート。

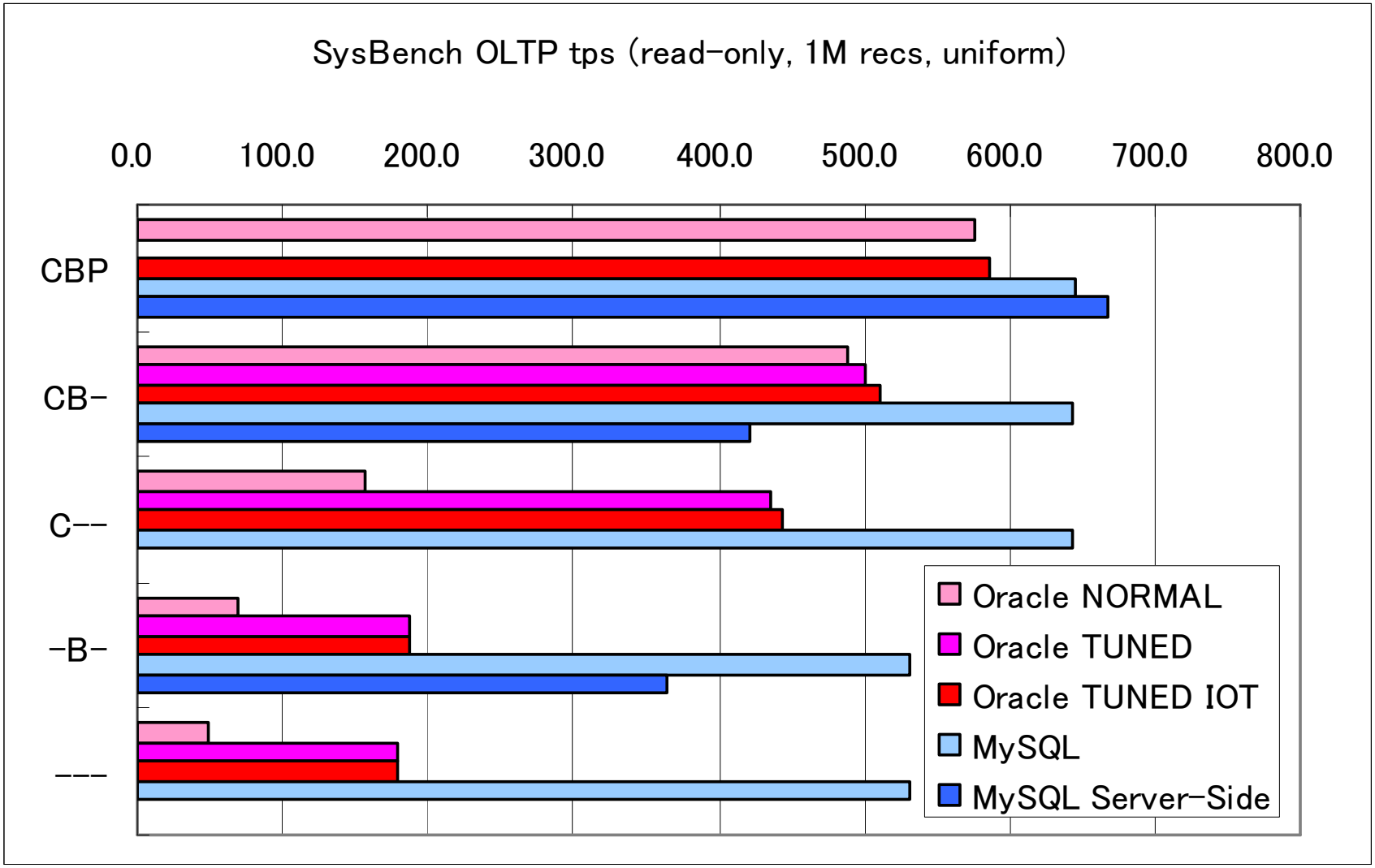
- 索引構成表のメリットは、主キーによるINDEX UNIQUE SCANやINDEX RANGE SCANが速くなるという点です。デメリットは、更新が遅くなるという点です。

- SysBench OLTPテストは以下のようにすべてのクエリが主キー検索となっているため、索引構成表の方が有利です。

```
SELECT c FROM sbtest WHERE id = ?
SELECT c FROM sbtest WHERE id BETWEEN ? AND ?
SELECT SUM(k) FROM sbtest WHERE id BETWEEN ? AND ?
SELECT c FROM sbtest WHERE id BETWEEN ? AND ? ORDER BY c
SELECT DISTINCT c FROM sbtest WHERE id BETWEEN ? AND ? ORDER BY c
```

# Oracleの場合 チューニング＆索引構成表

- ごくわずかですが、性能が向上しています。



# 新規接続の速さ

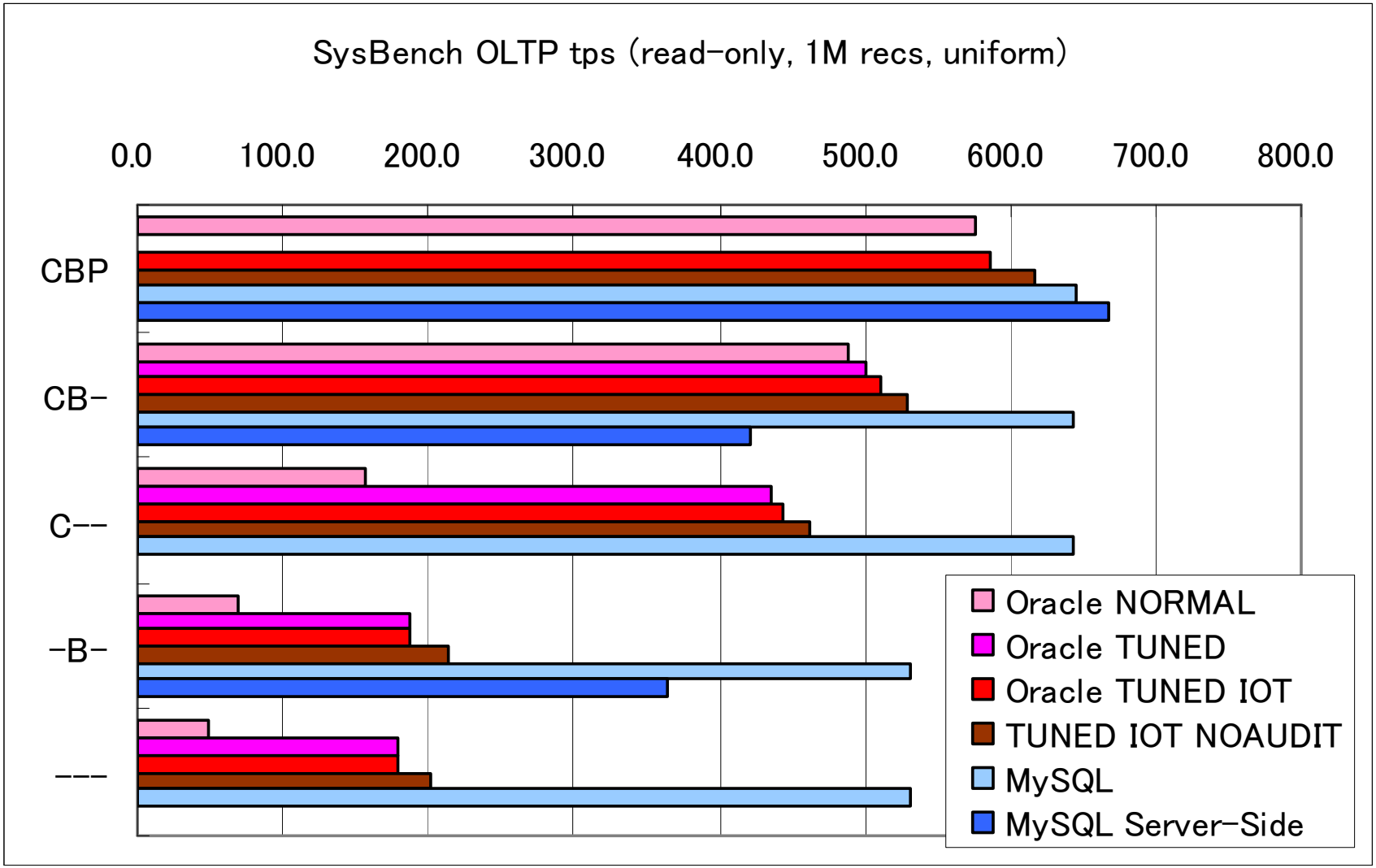
- MySQLではコネクションプールを利用しなくてもそれほど性能が低下しません。ただこれはMySQLの新規接続が速いというよりも、Oracle Databaseが遅すぎるというか…。

- Oracle Databaseの新規接続が遅い理由は以下のものが考えられます。
  - プロセスモデルアーキテクチャです。
    個人的にはApacheのようにpreforkしておけばいいのにと常々思っています。
  - 新規接続時に行う処理が多いです。
    新規接続時に7個程度のSQLが実行されているようです。
    特に、11g R1から標準監査がデフォルトで有効化されています。

- MySQLの新規接続が速い理由は以下のものが考えられます。
  - スレッドモデルアーキテクチャです。
    スレッドの生成コストは、プロセスの生成コストに比べると低いです。
  - スレッドをキャッシュできます。
    切断時にスレッドを保持しておくことで、スレッドの生成コストをゼロにできます。
  - 新規接続時に行う処理が少ないです。
    ほぼ何もしません。監査機能はありません。

# Oracleの場合 チューニング＆索引構成表＆標準監査なし

- コネクションプールを利用しない場合についても、性能が向上しています。



# まとめ

# いろいろ考える

- Oracle Database
  - 頑張れば良くなりますが、頑張らなかったときはひどい目に遭います。
  - 機能を増やせるだけ増やすとともに、随所にさまざまなキャッシュ機構やバイパス機構を組み込むことで性能を担保するアーキテクチャです。
  - ほかの製品にある機能はだいたいあるので、大差で負けることはないです。

- MySQL
  - 頑張っても頑張らなくても同じです。DBだけのエンジニアに仕事はありません。
  - 機能をシンプルに保つことで性能を担保するアーキテクチャです。
  - 足りない機能は自分で作ってしまえば、世界中の人に喜ばれます。

## さらにいろいろ考える

- MySQLは機能面において、Oracle Databaseを15年遅れで追いかけているといった状況です。ちょっとクセのある製品ではありますが、このまま開発が続けば遠くない将来にOracle 8i程度の機能は備えてしまうのではないかと考えています。

- そうなるとOLTPはもうMySQLで十分という状況になると思います。そのときOracle Databaseはどこで使われているのでしょうか。引き続き基幹系システムで大規模なバッチ処理を捌いているのでしょうか。それともExadataが切り開いたBusiness Intelligence、Data Warehouseの分野に特化した製品になっているのでしょうか。

- DBエンジニアがこの先生きのこるには…。

# 参考文献


- SysBench: a system performance benchmark
  http://sysbench.sourceforge.net/

- @IT：事例に学ぶWebシステム開発のワンポイント(11)
  JDBC接続を高速化する－PreparedStatementキャッシュの威力－
  http://www.atmarkit.co.jp/fjava/rensai2/webopt11/webopt11.html

# 宿題

1. 今回作成したプログラムは公開してありますので、ご自身の環境で負荷テストを行ってみてください。

2. 今回のようなワークロードに対して、PostgreSQLはどのような性能特性をもっているでしょうか。調べてみてください。

3. Oracle Databaseの次期バージョンにどのようなことを期待しますか？考えてみてください。

4. MySQLの次期バージョンにどのようなことを期待しますか？考えてみてください。