Oracle Database経験者が MySQLの設計思想を知っていろいろ考える会 (プレビュー版)

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。

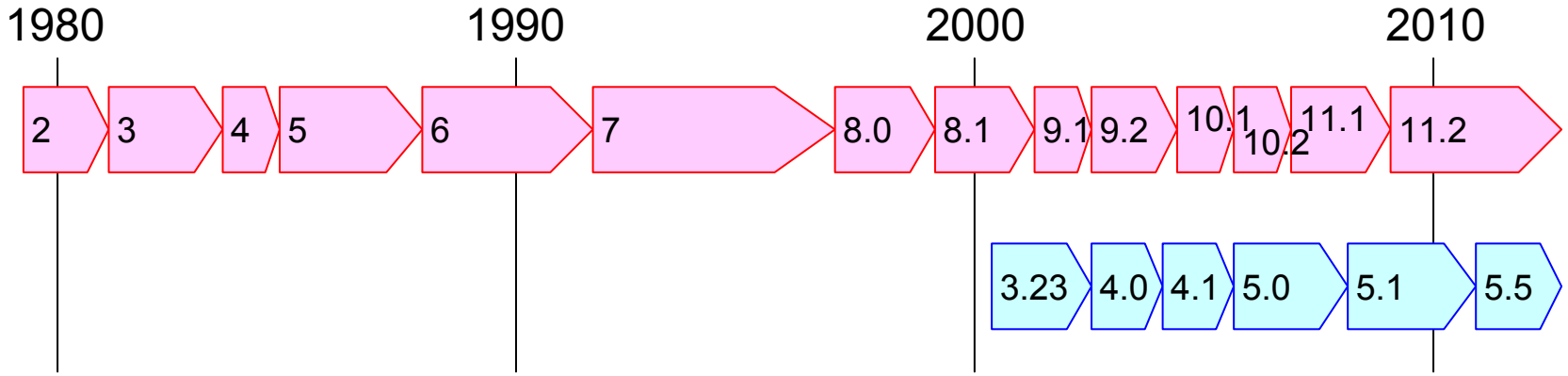
- 2005/10 MySQL 5.0
  ストアドプロシージャ、ビュー、トリガに対応。RHEL 5に付属するバージョンです。

- 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。

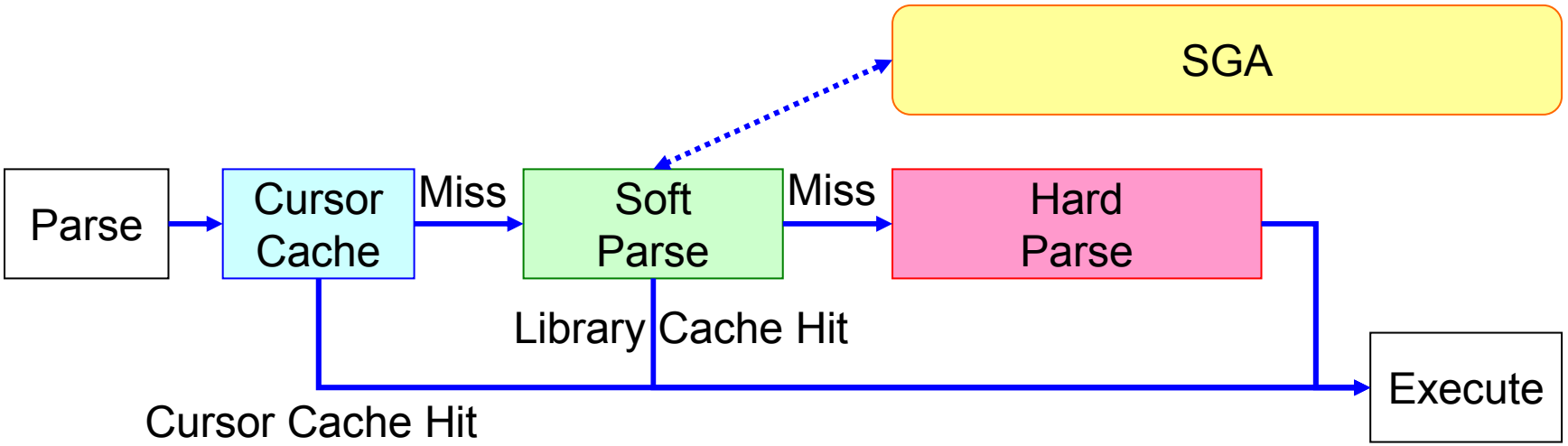
■パラメータのバインド機構を利用する
```
preparedStatement = connection.prepareStatement("SELECT c FROM sbtest WHERE id = ?");
preparedStatement.setInt(1, getRandomId()); // パラメータをバインドする
resultSet = preparedStatement.executeQuery();
```

■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。

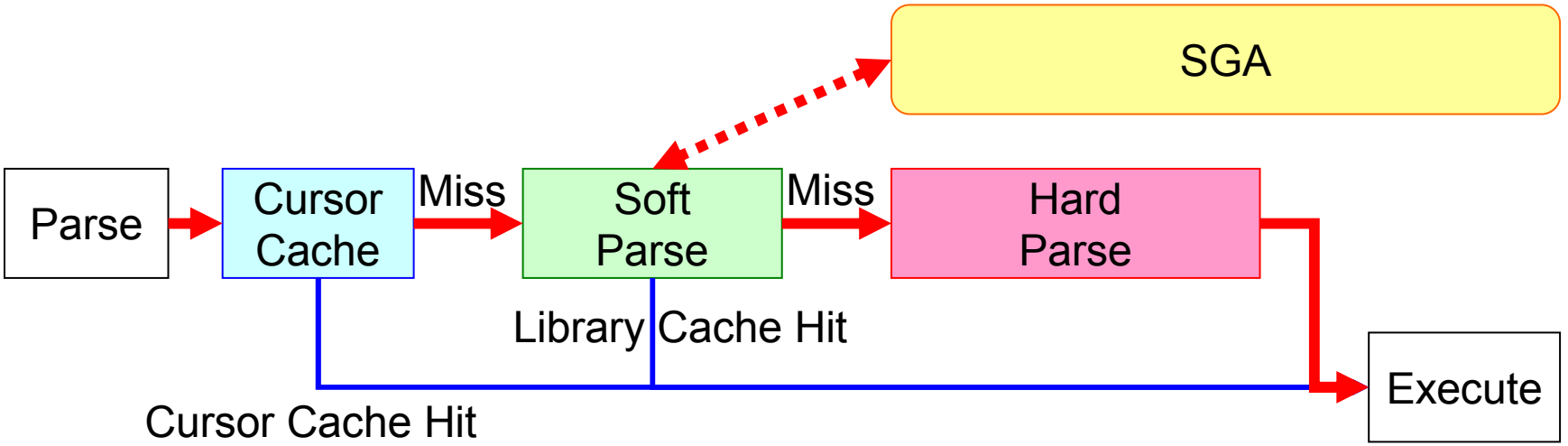
- 以下はOracle DatabaseにおけるSQL解析フェーズの流れを図示したものです。



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。

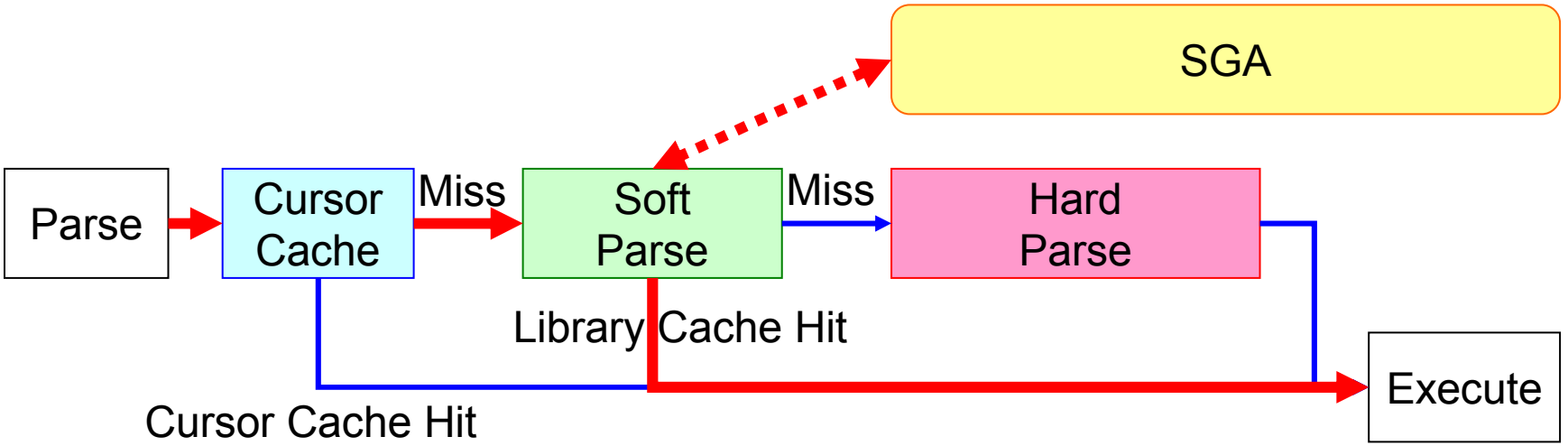



```
■Load Profile (Per Second)
Parses:                2,131.6
Hard parses:           2,089.3

■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。

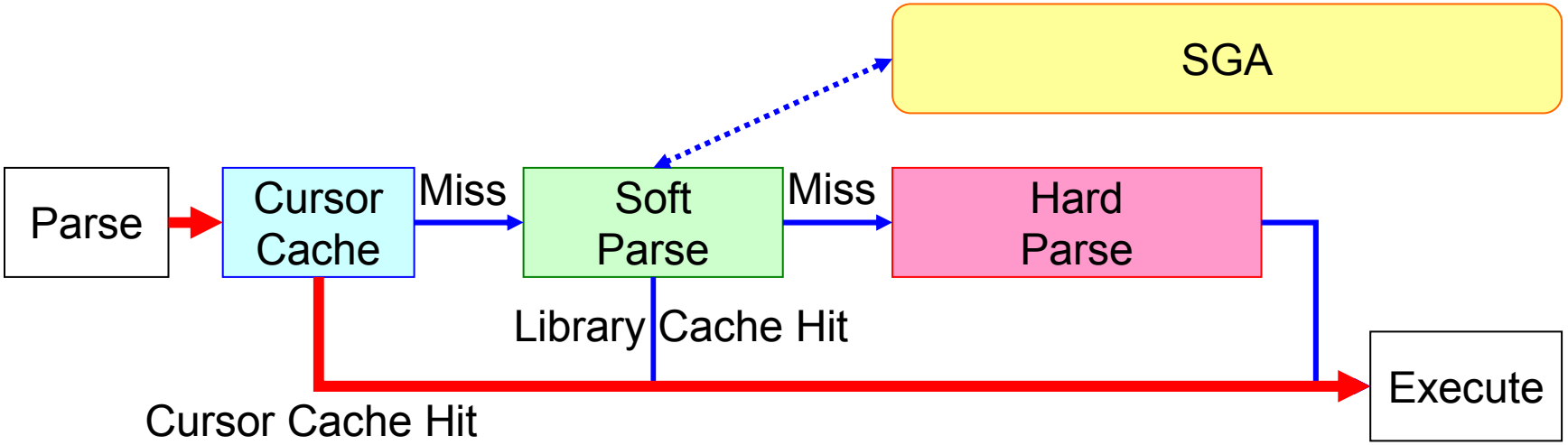



```
■Load Profile (Per Second)
Parses:                6,350.8
Hard parses:               0.1

■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

続きはUnconferenceで！