PostgreSQLとMySQL比較

JPUG四国支部 Yasuo Ohgaki
yohgaki@ohgaki.net / yohgaki@postgresql.jp

POSTGRESQL パフォーマンス

自己紹介

- 大垣　靖男
- PostgreSQL 四国支部 担当理事
- http://blog.ohgaki.net/

アジェンダ

- 主にWebアプリケーションにRDBMSを利用する場合を想定（していたが...）
- 既存のツールを使って比較
  - pgbench / mysqlbench

PostgreSQLのおさらい

- トランザクションサポート
- ストアドプロシージャ
- プリペアードステートメント
- オブジェクト指向型
- 拡張可能なデータ型
- SQL標準準拠
- 追記型アーキテクチャ

PostgreSQL

- 誤ったイメージ
  - 遅い
  - 高度な機能が必要なアプリケーションの場合に利用する
    - トランザクションやストアドプロシージャなど
  - VACUUMが必要で管理が煩雑
    - 追記型アーキテクチャ

知っておくべき基礎知識

- デフォルト設定は利用しない
  - パフォーマンスが1/30以下になることも！
- Auto Vacuumを利用する
  - ほぼメンテナンスフリー
- 他のデータベースでもガーベッジコレクションが必要
  - REINDEX
  - MySQL： OPTIMIZE TABLE
  - SQLite： VACUUM

ベンチマークの基礎知識

- pgbenchの結果が安定しない
  - チェックポイント
  - Auto Vacuum
- 大きなデータの場合、極端にパフォーマンスが低下
  - 全てのDBMSはディスクアクセスが発生するとディスク性能までパフォーマンスが低下
  - ディスクアクセスの取り扱いが同等なら性能も同等
  - 基本的にメモリに収まるデータを取り扱う
- トランザクション数はある程度は必要
  - 結果のブレが大きくなる

ベンチマーク環境

- ＤＢサーバ
  - Core2Quad Q6600（2.4GHZ　4コア）
  - 8GB PC6400
  - 400GB SATA-II
  - OS： MomongaLinux4 (Custom) 2.6.23-31m.mo4PAE　i686
  - FS： xfs

ベンチマーク環境

- ディスクとバッファ性能

```
[yohgaki@dev dbbench]$ sudo hdparm -tT /dev/sdc

/dev/sdc:
 Timing cached reads:   7412 MB in  2.00 seconds = 3712.05 MB/sec
 Timing buffered disk reads:  204 MB in  3.03 seconds =  67.38 MB/sec
```

ベンチマーク環境

- PostgreSQL
  - ソースからビルド。オプションは無し
  - UTF-8エンコーディングを使用
  - 7.4、8.0、8.1、8.2、8.3のそれぞれ最新版
- MySQL
  - MomongaLinux trunkのパッケージ
  - 5.0.51-3m

ベンチマーク環境

- クライアント
  - PentinumD820（2.8GHz DualCore）
  - 3GB PC4200

ベンチマーク環境

- ベンチマークツール
  - PostgreSQL： pgbench
  - MySQL：mysqlbench
- mysqlbench はpgbenchのポート
  - スレッド型だがパフォーマンスに違いはないはず
  - 試しにpgbenchのpthread版を作ったことがあるが目立った違いはなし
    - 私のwikiにソースあり

ベンチマーク環境

- スケーリングファクタ: 50
  - pgbench，mysqlbenchのオプション
  - 1 = 10万レコード（accounts）
  - 50 = 500万レコード
    - PostgreSQL： およそ2ＧＢ
    - MySQL： およそ900ＭＢ

pgbenchコマンド

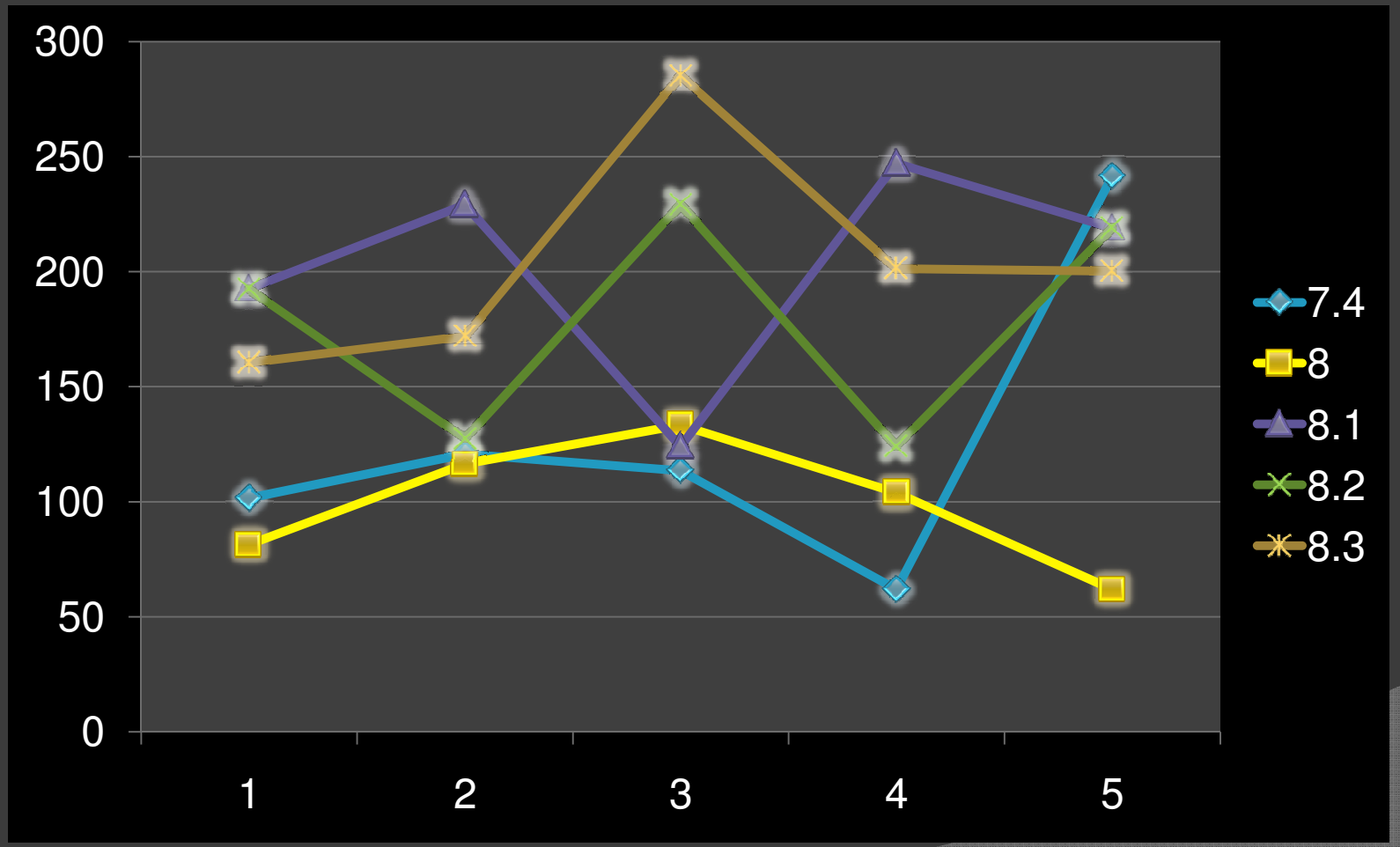
- c はクライアント数
- t はトランザクション数

- 例：
  - /usr/bin/pgbench -h 192.168.100.50 -U yohgaki -v -c 20 -t 150 pgbench

PostgreSQLの設定

- デフォルトのpostgresql.conf
  - 共有バッファ：8～32MB
  - ソートバッファ：2ＭＢ
  - キャッシュ：8MB
  - fsync： on
  - チェックポイントセグメント： 48MB

デフォルト設定
c:20 t:250



考察

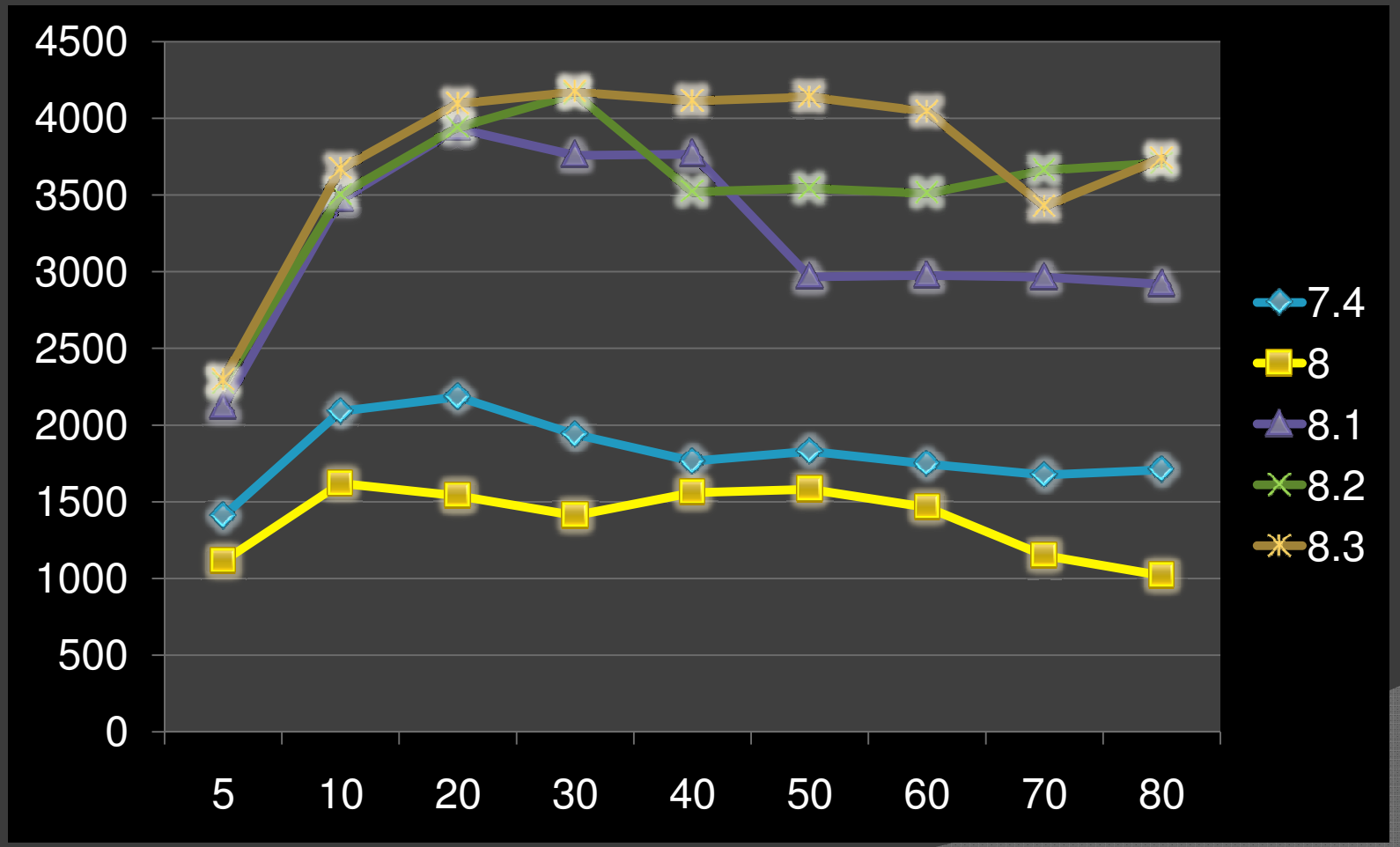
- クライアント数、トランザクションに関係なく結果の幅が大きい
- 多少の最適化でも数十倍の性能向上が可能（次のベンチマーク結果を参照）
- デフォルト設定でのベンチマークは意味がない

カスタマイズド設定

- 変更箇所：8.3.0

```
+shared_buffers = 1600MB                # min 128kB or
max_connections*16kB
+temp_buffers = 80MB                    # min 800kB
+work_mem = 100MB                               # min 64kB
+bgwriter_delay = 1000ms                # 10-10000ms between rounds
+fsync = off                            # turns forced synchronization on or
off
+wal_buffers = 512kB                    # min 32kB
-#wal_writer_delay = 200ms              # 1-10000 milliseconds
+wal_writer_delay = 1000ms              # 1-10000 milliseconds
+commit_delay = 10000                   # range 0-100000, in microseconds
+checkpoint_segments = 32               # in logfile segments, min 1, 16MB
each
+random_page_cost = 3.0                 # same scale as above
+autovacuum = off                       # Enable autovacuum subprocess?
'on'
```

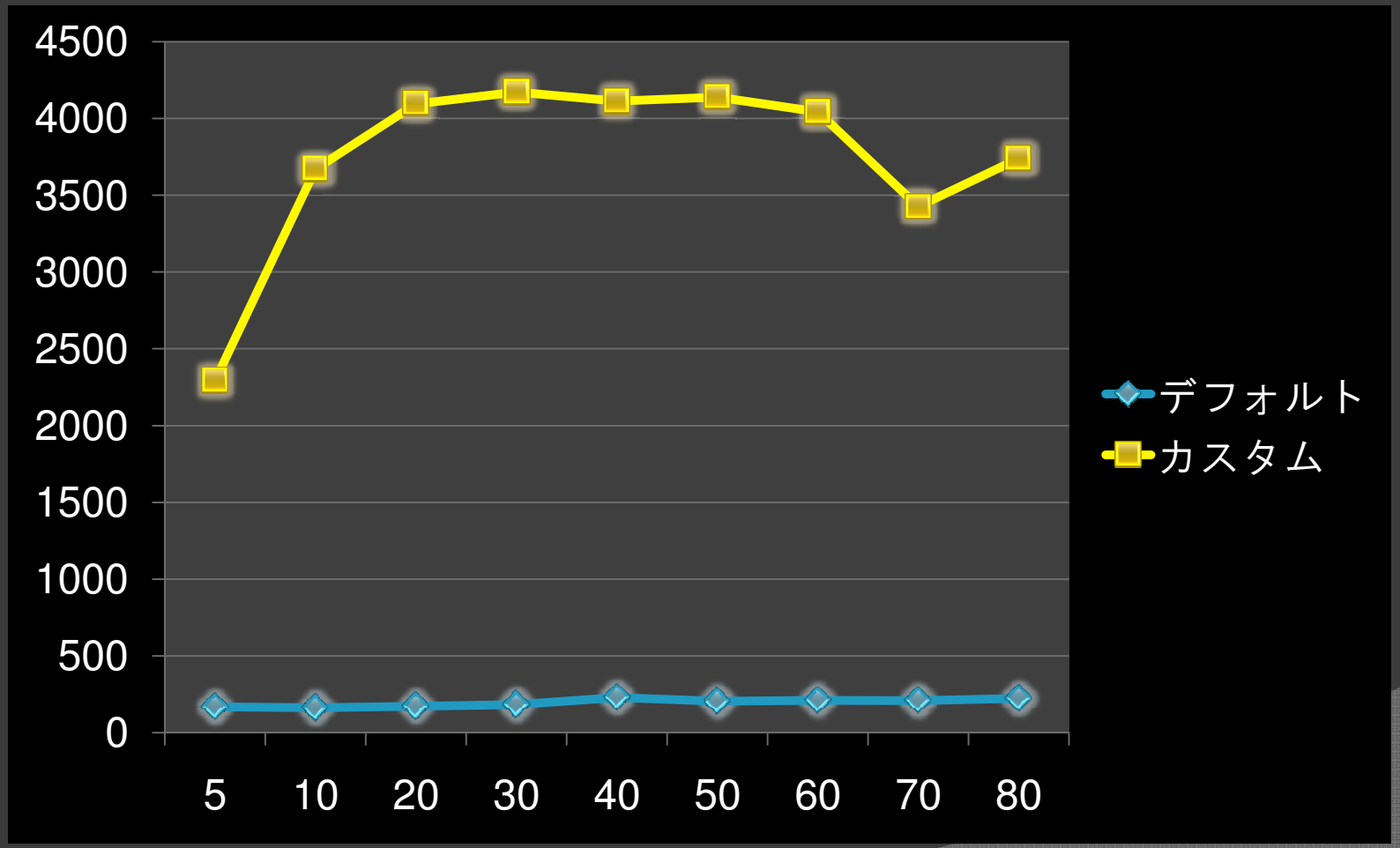
カスタマイズド設定
PostgreSQL



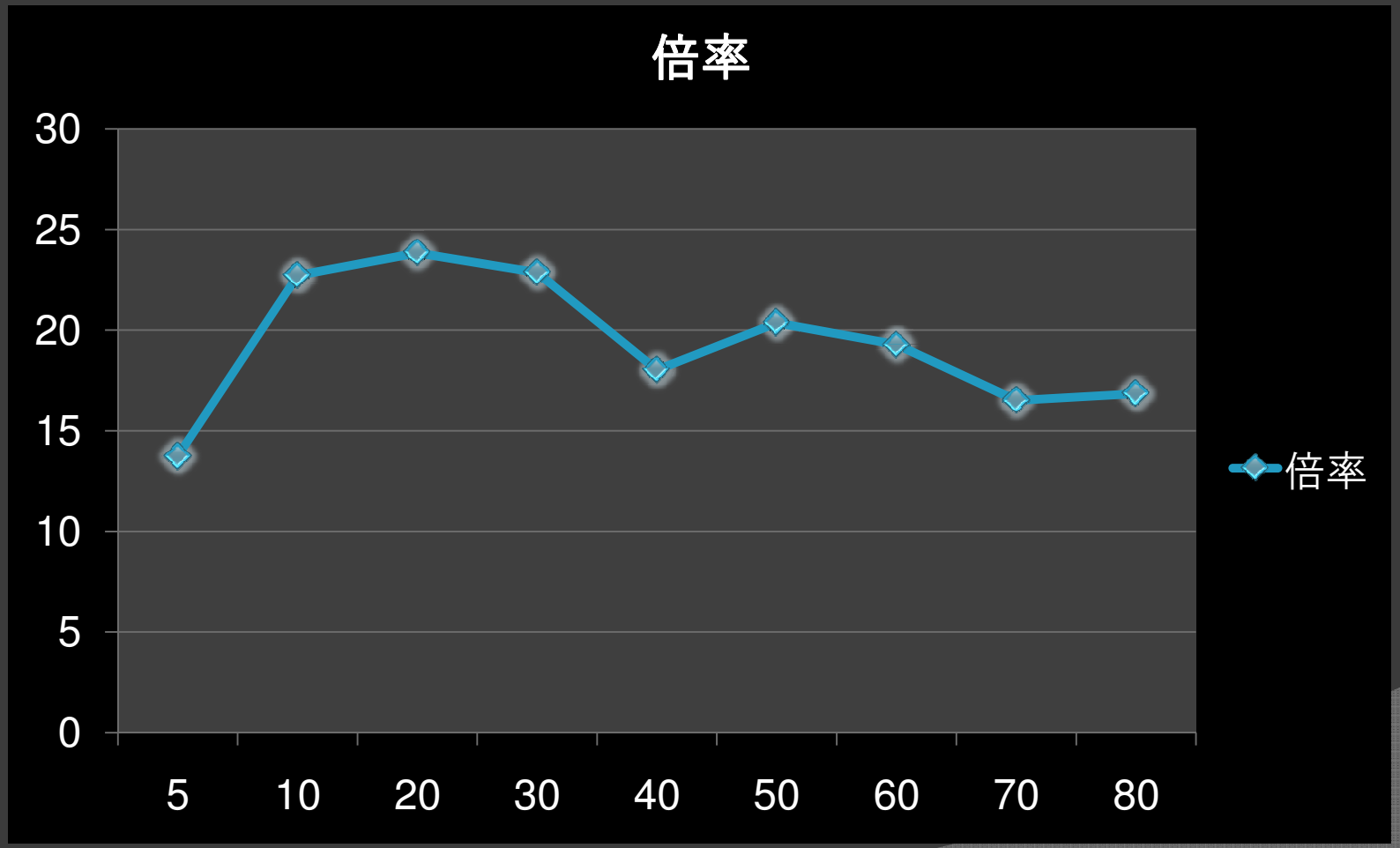
結果

- PostgreSQL 7.4と8.0では結果が逆転
  - 8.0に追加されて機能・コードがオーバーヘッドとなっている
- PostgreSQL　8.2と8.3では性能が近い
  - トランザクションがそれほど性能差がでないクエリ
- PostgreSQL 8.2、8.3共に非常に良くスケールする
  - クライアントが増えてもスループットが劣化しない

デフォルト　vs　カスタマイズド
PostgreSQL 8.3.0



何倍くらい速いか？



結果

- PostgreSQLのデフォルト設定は使ってはならない
- 簡単なチューニングで10〜20倍の性能

mysqlbenchコマンド

- pgbenchのポートなので同じ
- c：クライアント数
- t：トランザクション数

- 例：
  - ./mysqlbench -h 192.168.100.50 -U mysqlbench -P mysqlbench -v -c 5 -t 600 mysqlbench

MySQLの設定

- ソースに付属のmy-huge.cnfとカスタムの設定を使用
- MyISAM
- 文字エンコーディングはUTF-8

- その他に特別な設定なし

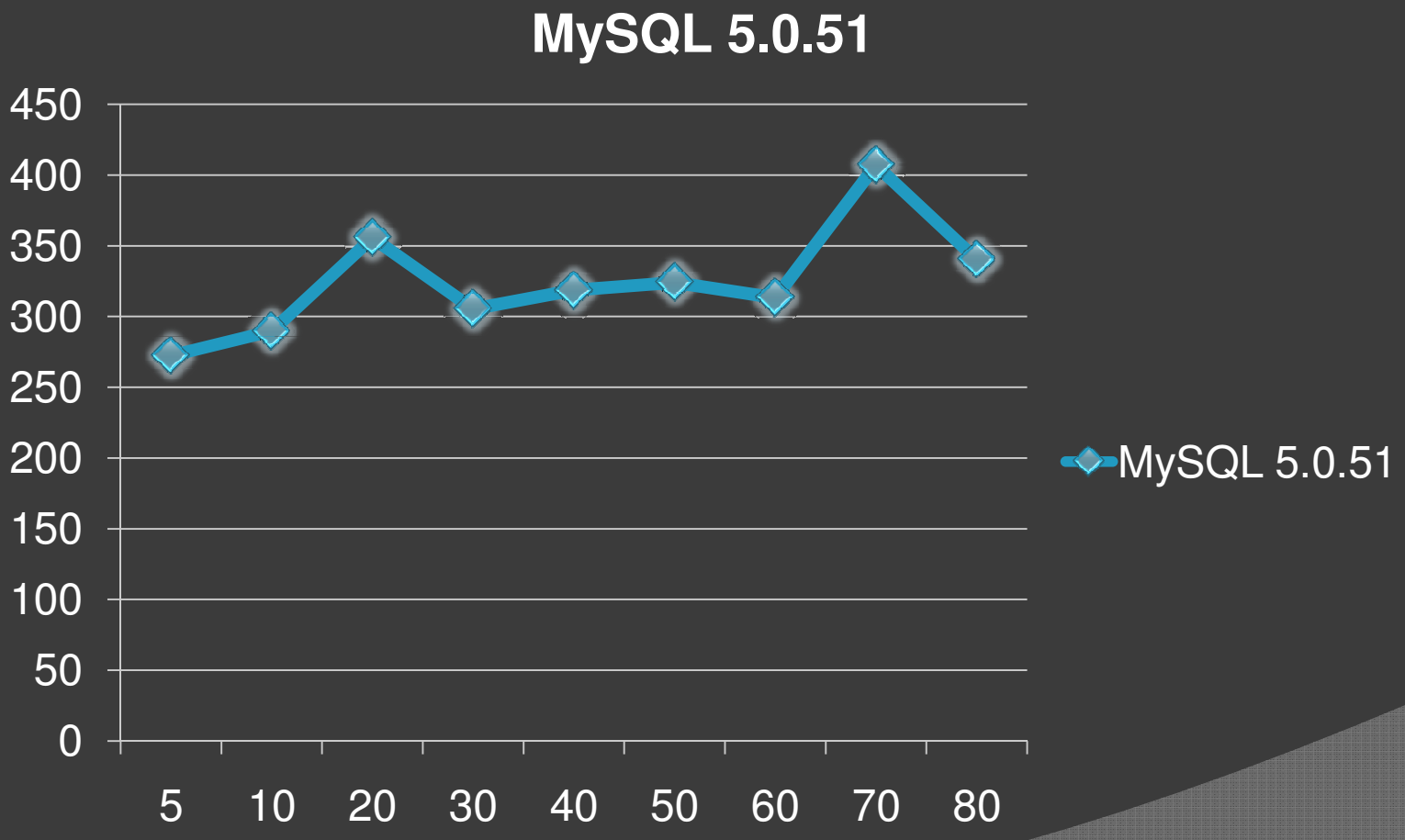
デフォルトの設定

- my-huge.cnfはhugeでもない

key_buffer = 384M
max_allowed_packet = 1M
table_cache = 512
sort_buffer_size = 2M
read_buffer_size = 2M
read_rnd_buffer_size = 8M
myisam_sort_buffer_size = 64M
thread_cache_size = 8
query_cache_size = 32M

- PostgreSQLの8〜32MBよりは随分大きい

デフォルト設定
my-huge.cnf



結果

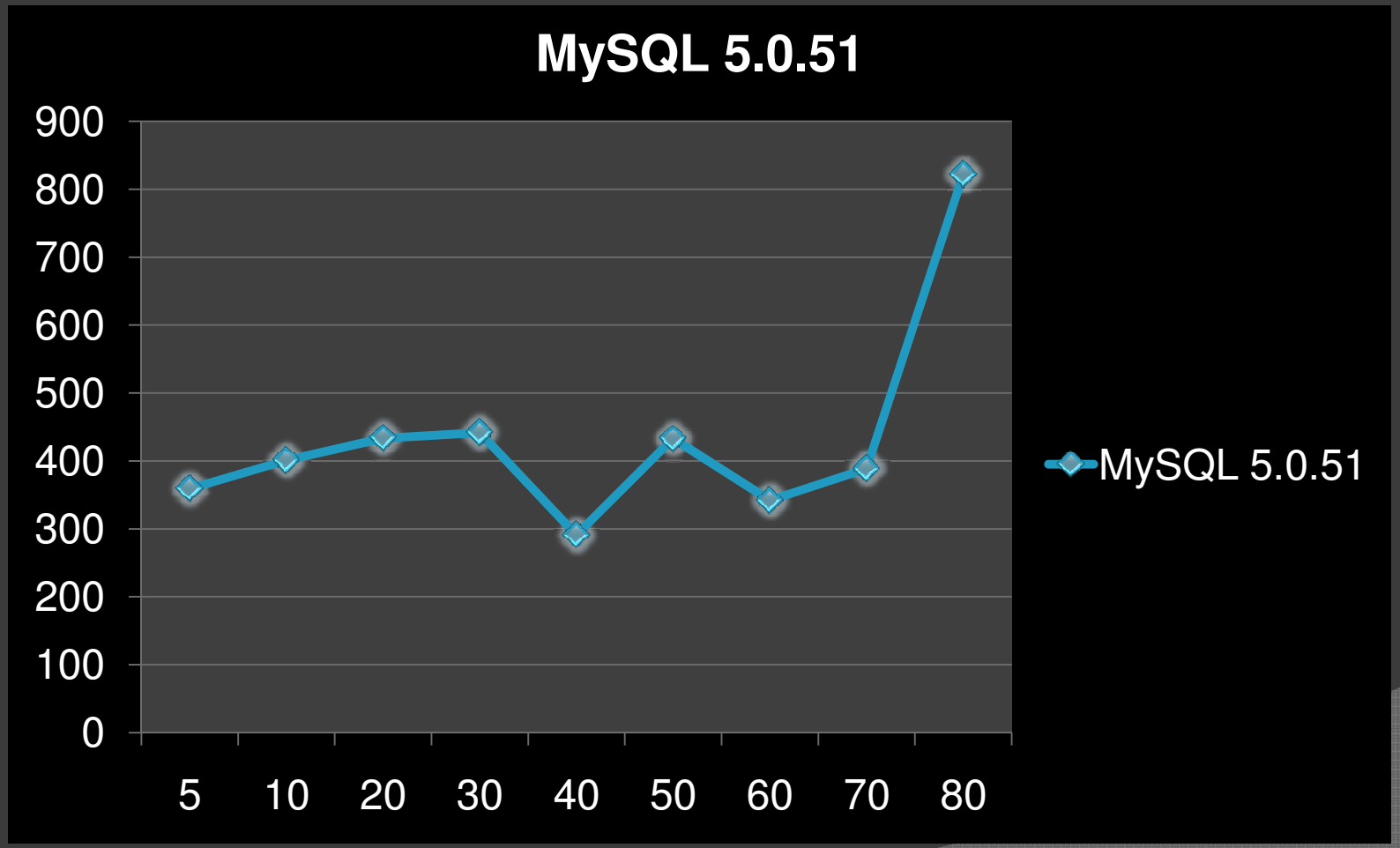
- MySQLもデフォルト設定は使えない
- PostgreSQLより多くのメモリを利用している分速い
- 結果のブレが大きい

カスタムの設定

- my-huge.cnfからの変更箇所

key_buffer = 3840M
sort_buffer_size = 100M
read_buffer_size = 100M
read_rnd_buffer_size = 80M
myisam_sort_buffer_size = 100M
thread_cache_size = 64
query_cache_size = 256M

カスタマイズド設定

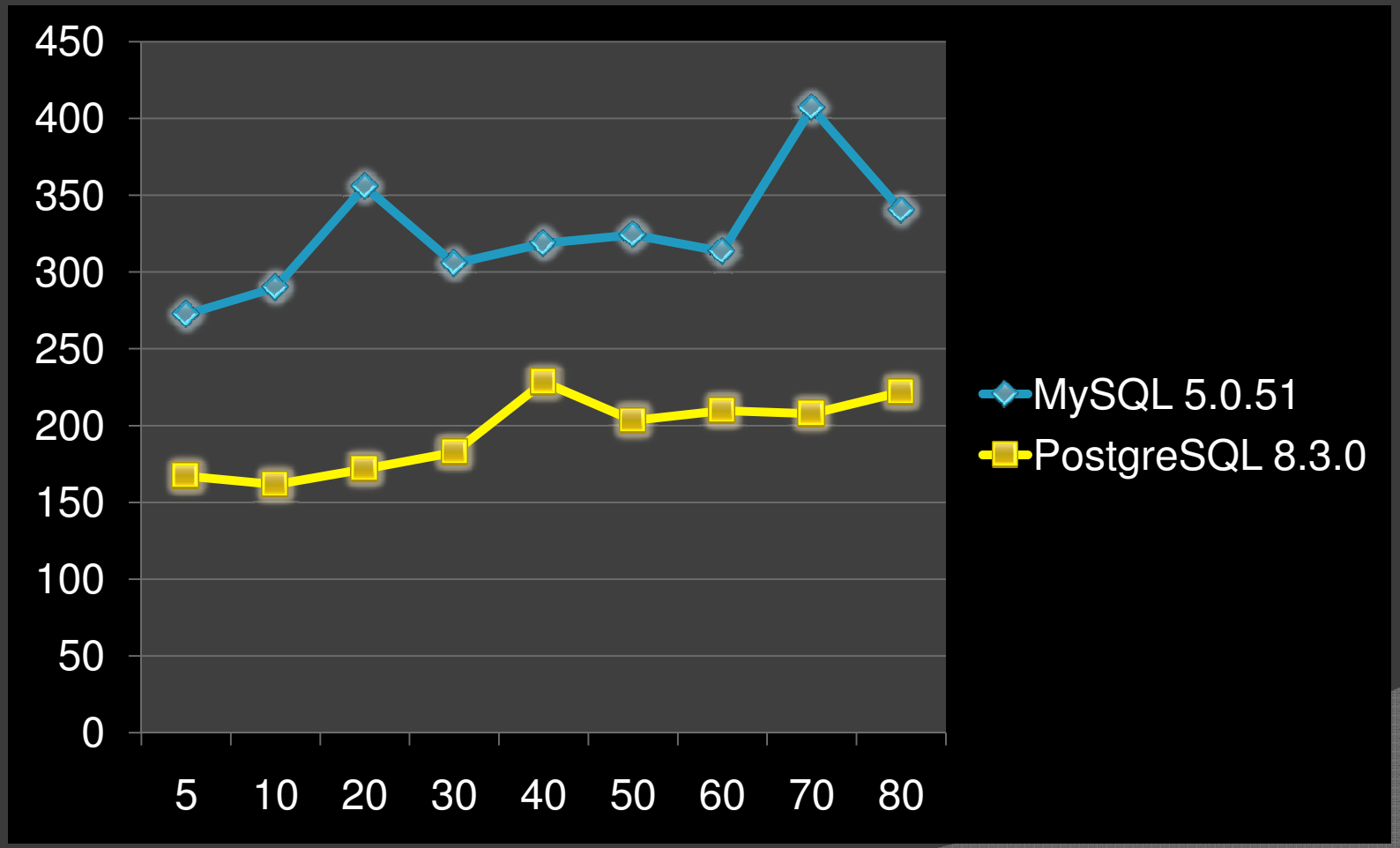

MySQL 5.0.51
900
800
700
600
500
400
300
200
100
0
5
10
20
30
40
50
60
70
80
MySQL 5.0.51

結果

- デフォルトよりは向上
- しかし、性能的には不十分

Postgresql とmysqlの比較

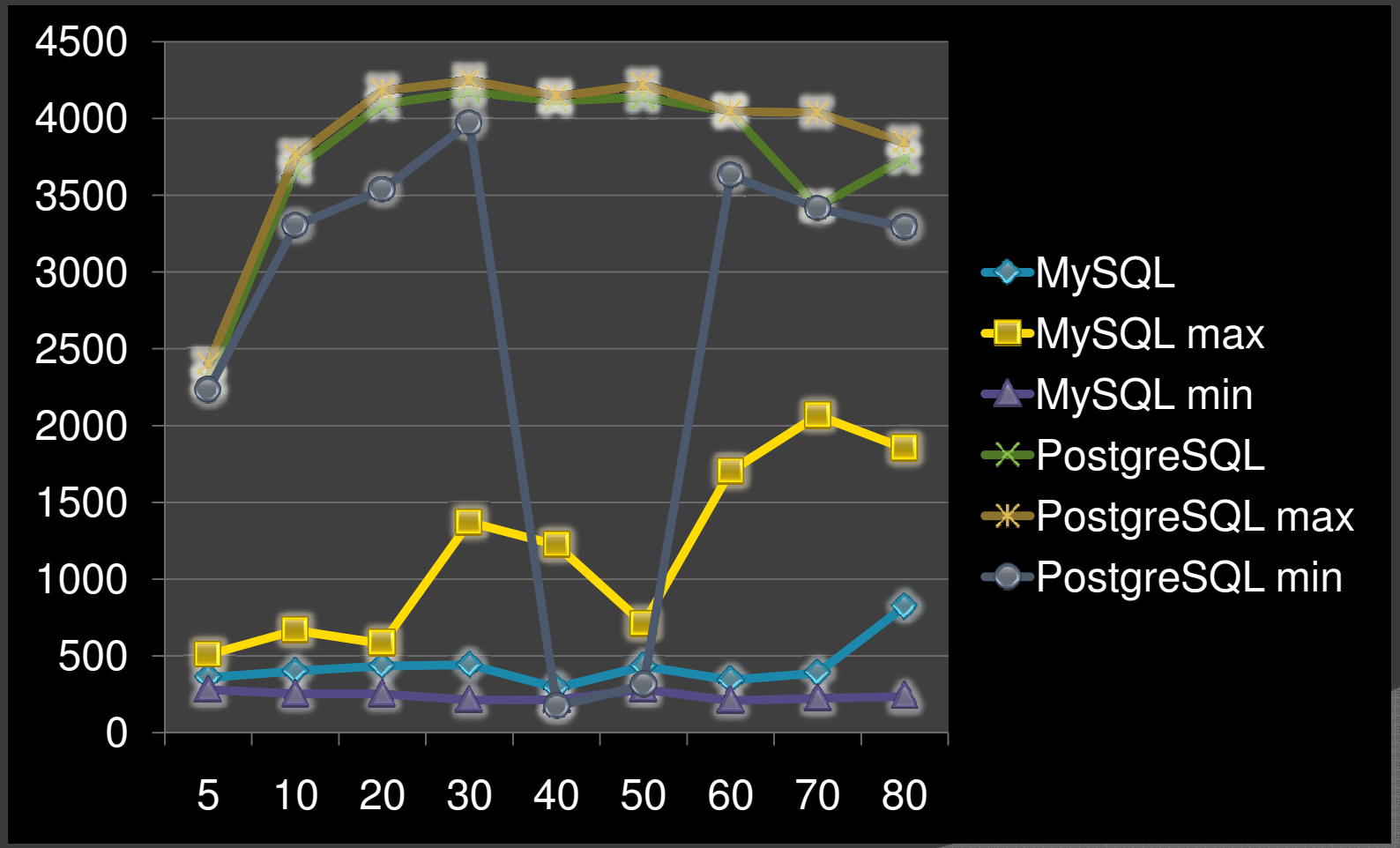
MySQL vs PostgreSQl
デフォルト設定



結果

- 予想通り（？）MySQLの方が速い
- しかし、MySQLの方が多いメモリ
  - 384MB　vs．32MB

MySQL vs. PostgreSQL
カスタマイズド設定



結果

- MySQLの結果に時々、異常に速い場合がある
- MySQLの性能は最大値でも半分にも満たない（メジアンは10分の1以下）
- PostgreSQLには異常なパフォーマンス低下がある
  - Check Point

PostgreSQLパフォーマンス低下理由

- チェックポイントセグメント
  - チェックポイントセグメントがいっぱいになると書き込みが開始される
  - ディスクヘッドの動きが大きくなり大幅なパフォーマンス低下となる

データサイズ

- pgbench/mysqlbenchともに500万レコード
  - MySQLの方が小さい
- データなどに依存
  - 例：RSSフィードのデータベース - MySQLからPostgreSQLに移行
    - 10GBのデータが3GBになった場合も
    - しかも、パフォーマンスは3倍

ベンチマークノート

ベンチマーク中のロード

- すべてのテストでCPUの性能を使い切ることはなし
- ロードアベレッジも多くない
- I/O待ち状態も（見かけ上は）あまりない

8.2と8.3では速度は違わないが。。

- ベンチマーク中のディスク書き込み
  - 8.2　＞　8.3
  - およそ2倍から3倍（vmstat 目視）
- 5000万トランザクションを行った後のデータベースクラスタサイズ
  - 8.2　＜　8.3
  - 3GB弱と2GB強

PostgreSQL 8.3のパフォーマンス向上

- HOT
  - 可能でメモリ上でのみ追記を行いディスクを使わない
- 同期読み込み
  - テーブルスキャンを同時に行う場合にパフォーマンス向上

HOT意味

- Webアプリケーションのセッション管理などに効果大
  - セッションIDは固定でインデックス付き
  - データや最終アクセス時間はインデックス無し
  - セッションデータは基本的に毎回書き込みが発生しているが、HOTのおかげでディスクへ追記型の書き込みが発生しない

HOT意味

- ディスクI/O減
  - 将来のパフォーマンス向上に余地
- Vacuum自体の必要性も軽減
  - Auto Vacuumは自動実行され、チェックポイントと同様にパフォーマンスが低下する

同期スキャンの意味

- クエリの作り方、インデックスの張り方、設定のチューニングの仕方が良くない場合でも速い

注意点

- PostgreSQLのベンチマークはチェックポイントの発生を抑制、Auto Vacuumは無効化している
  - 実運用ではチェックポイントは発生する。Auto Vacuumを有効にするべき。（特に8.3）
- **fsync=false**
  - **fsync=falseではデータの安全性は保障されないが、参照系のWebサイトではfsync=falseを検討する価値は十分にある**
- MySQLはデフォルトのMyISAMを利用している
  - 実はInnoDbの方が速い場合も多い
- データはメモリに全部載る

おススメのベンチマーク

- サブクエリのパフォーマンス
- INクエリのパフォーマンス
- JOINのパフォーマンス
- 巨大なテーブルのスキャン
- MySQLのInnoDBパフォーマンス
- 単純ルックアップ

- ご清聴ありがとうございました