ORACLE®

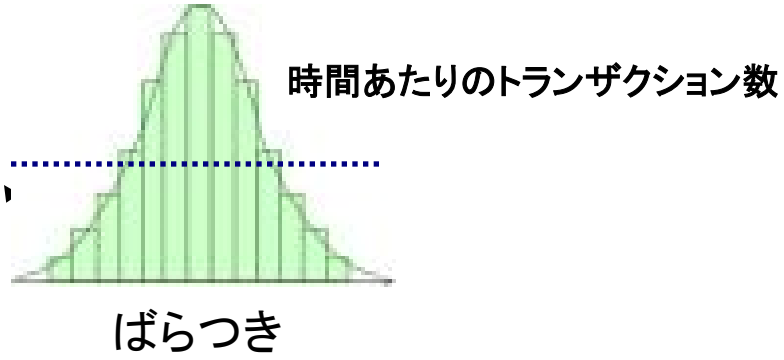
MySQLパフォーマンスチューニング概要

日本オラクル MySQL Global Business Unit

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。

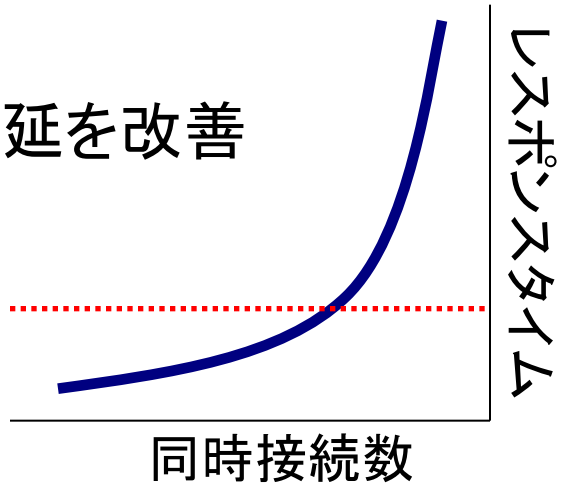
パフォーマンスとは？

- 単純な言葉ではあるが多くの意味が。。。
- チューニングの目的:
  - ユーザを満足させるため→直接的にor間接的に
- パフォーマンスの指標の例
  - スループット
  - レスポンスタイム / レイテンシ
  - スケーラビリティ
  - 上記の組合せ



キューイング

- 複数のユーザ/リクエストがある場合に発生
- 実行される前に「キュー」で待たさせる
- レスポンスタイム = キューイングによる遅延 + 実行時間
  - 実社会での例 – サポートセンターへの電話
- “ホッケースティック” - システムが飽和状態に近づくと、キューイングによる遅延が急激に増大する
- パフォーマンスの改善時に、キューイングによる遅延と実行時間のどちらを改善するかが重要
- 実行時間の短縮がキューイングによる遅延を改善



実行時間: Key to the hotspot

- 確認事項 – どこで実行時間が使われているか？
  - ネットワーク, CPU, ディスク, ロック...
- 直接的な計測方法
  - 例）Webのページを表示するためのクエリ実行時間の合計
- 間接的な計測方法
  - CPU利用率
  - ディスクIOのレイテンシ
  - ネットワークトラフィック
  - 処理待ちプロセス数
  - 同時実行中のクエリ数
  - etc.

ベンチマークテスト

- 重要なツール:
  - アプリケーションのパフォーマンスの計測
  - 変更点のパフォーマンスに与える影響の確認
  - スケーラビリティの評価
- ただし。。。
  - 実行方法を間違えると誤った指針となりうる
  - 結果を正しく読み取ることができる必要がある
- ありがちな間違い
  - データサイズが本番環境では100GBなのに1GBでテスト
  - データやリクエストのばらつきを考慮しないでテスト
  - 1ユーザだけでテスト
  - 実際のアプリケーションの特性と異なるベンチマークテストの結果から、全力でパフォーマンスチューニング

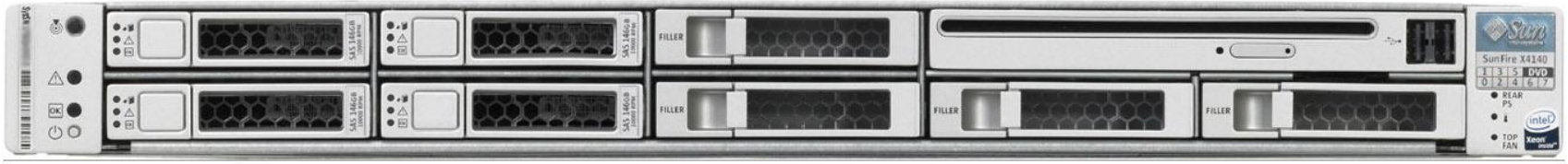
MySQL用ベンチマークツール

- 独自のベンチマークツールを作成
  - 一般ログの内容からSQL文を抽出
  - MySQL ProxyやTCP DumpでSQL文を取得
- DBT2
  - http://osdldbt.sourceforge.net/
  - http://nippondanji.blogspot.com/2009/03/dbt-2.html
- mysqlslap MySQL 5.1 +
  - http://dev.mysql.com/doc/refman/5.1/ja/mysqlslap.html
- SysBench
  - http://sysbench.sourceforge.net/
- supersmack
  - http://vegan.net/tony/supersmack/
- mybench
  - http://jeremy.zawodny.com/mysql/mybench/

ビジネス面からの考慮

- どのようなパフォーマンスチューニングでもコストがかかる
- 他の可能性の検討
  - ハードウェアの交換の方が、アプリの修正より安いことも
- そのパフォーマンス、スケーラビリティ、信頼性は本当に必要？
  - 99.999% の可用性は 99.9% よりも格段にコストがかかる
  - ピーク時の性能は平常時の何倍必要なのか？
- 常に全体像を把握しておくこと
  - 「これが最大のリスク/ボトルネックなのか？」
- どのチューニングがビジネスにとって重要か確認すること
  - “全て”をチューニングすることは多くの場合に無駄
  - 次善策を取ったときのコストとビジネスへのインパクトは？

Hardware: The Perfect MySQL Server



- より多くのCPUコア (特に5.5以降)
- x86_64 - より多くのメモリを活用できる64bit対応が重要
- LinuxまたはSolarisがベスト、WindowsやUnixでも可
- 高速なハードディスク or NAS/SAN……
  - RAID 10が一般的に推奨、RAID 5は参照比率が高い場合のみOK
  - ハードウェアRAIDのキャッシュバックアップバッテリーは重要
  - より多くのディスクでより高い性能 - 4ディスク以上を推奨
- …もしくはSSDなどを活用 (より高いスループット)
  - さらに高い性能が求められる場合にはFusion-IOなども選択肢
- 冗長性のため最低2つのNIC
- レプリケーションのスレーブはマスタと同等以上のスペックに

基本その1: システム変数の確認

- 稼働中のMySQLサーバの設定パラメタを知る

```
mysql> show variables like 'auto%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| auto_increment_increment | 1     |
| auto_increment_offset    | 1     |
| autocommit               | ON    |
| automatic_sp_privileges  | ON    |
+--------------------------+-------+
4 rows in set (0.00 sec)
```

```
shell> mysqladmin -uroot -S /tmp/mysql.sock variables | grep auto
| auto_increment_increment                  | 1
| auto_increment_offset                     | 1
| autocommit                                | ON
| automatic_sp_privileges                   | ON
```

- 設定方法
  - 設定ファイル: my.cnf / my.ini
  - 一時的に変更する場合: SET [GLOBAL] <変数名>=<値>
    - 接続ごとの値(LOCAL)かサーバ全体での値(GLOBAL)か

基本その2: ステータス変数の確認

- MySQLサーバの稼働状況を知る

```
mysql> show status like 'innodb_buf%';
+---------------------------------------+--------+
| Variable_name                         | Value  |
+---------------------------------------+--------+
| Innodb_buffer_pool_pages_data         | 142    |
| Innodb_buffer_pool_pages_dirty        | 0      |
```

```
shell> mysqladmin -uroot -S /tmp/mysql.sock extended
+-----------------------------------------+-------------+
| Variable_name                           | Value       |
+-----------------------------------------+-------------+
| Aborted_clients                         | 0           |
| Aborted_connects                        | 0           |
```

- 個別のクエリが行っている処理内容の確認

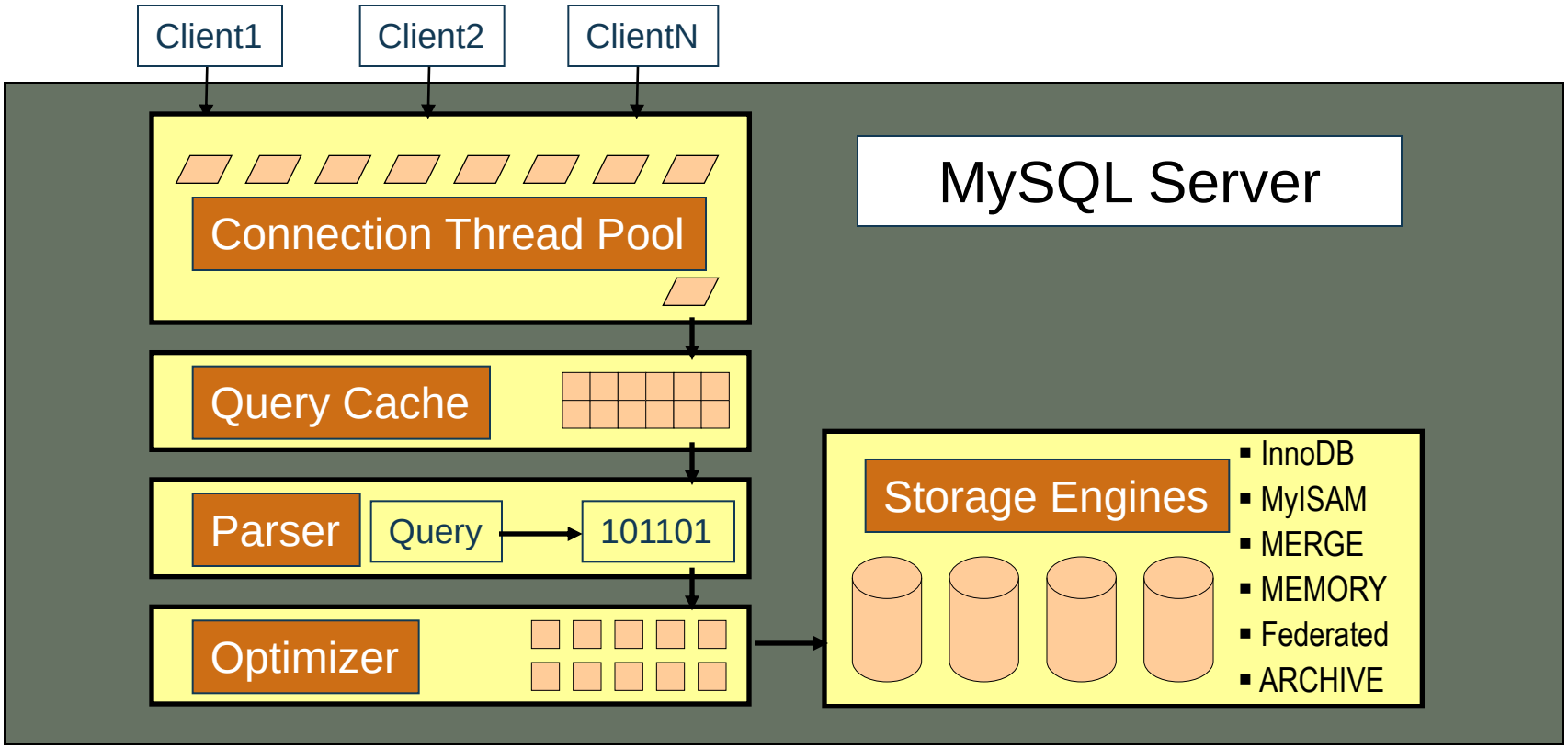
```
Mysql> FLUSH STATUS; <run query>; SHOW STATUS;
```

- 定期的にステータス変数を確認

```
shell> mysqladmin -u -p ... ex -i 15 -r | grep -v ' 0 '
```

http://dev.mysql.com/doc/refman/5.1/ja/server-status-variables.html

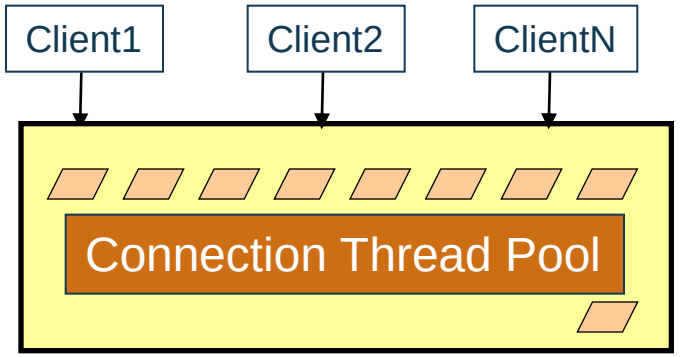
MySQL Serverのアーキテクチャ



サーバのコネクション＆スレッド

設定ファイル my.cnf :

- **max_connections (100)**
  - サーバが許容可能なコネクション数。多すぎるとメモリを消費しすぎる可能性あり



- **thread_cache_size (8)**
  - スレッドをコネクションの切断後にもキャッシュしておく数
  - 一般的には **max_connections/3**

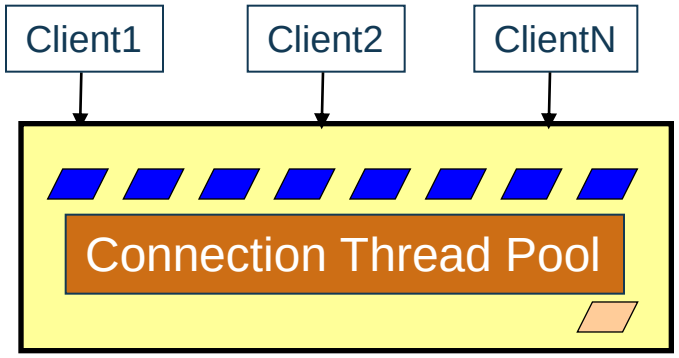
mysql> show status;

- **Max_used_connections**
  - **max_connections とあわせてチェック**
- **Threads_created**
  - thread_cache ミス
  - 低い数値であるべき

コネクションスレッド毎のバッファ

設定ファイル my.cnf:

- **sort_buffer_size (2M)**
  - ソート用のメモリサイズ。このサイズを超えるソートはディスクを利用。**512K**, **1M**程度で十分なケースも。
- その他の**read, read_rnd, etc…** バッファはデフォルトで問題ないケースも多い
- バッチ処理などの場合、処理実行前に動的にこれらのパラメタを変更する



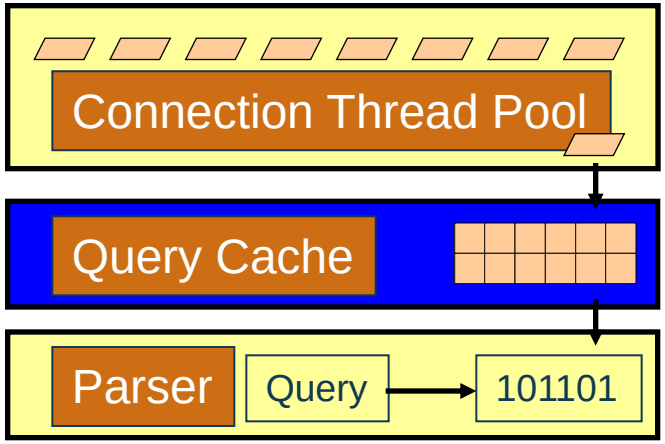
mysql> show status;

- **Sort_merge_passes -**
  - ファイルを利用したマージソートのパス数
  - ソートがメモリ上だけで収まらない場合には要確認
  - インデックスの利用を検討

クエリキャッシュ

設定ファイル my.cnf:

- **query_cache_size (0)**
  - クエリキャッシュに割り当てるメモリサイズ
  - 一般的には**32M**で**OK**
- **query_cache_type (ON)**
  - 最悪のケースではパフォーマンスのオーバーヘッドが約15%
  - SELECTの比率が高いサーバで有効



mysql> show status;

- **Qcache_hits, Qcache_inserts**
  - キャッシュヒット/登録件数、ヒット率が小さければクエリキャッシュを無効にすることも検討
- **Qcache_lowmem_prunes**
  - メモリ不足のためにキャッシュが削除された回数。

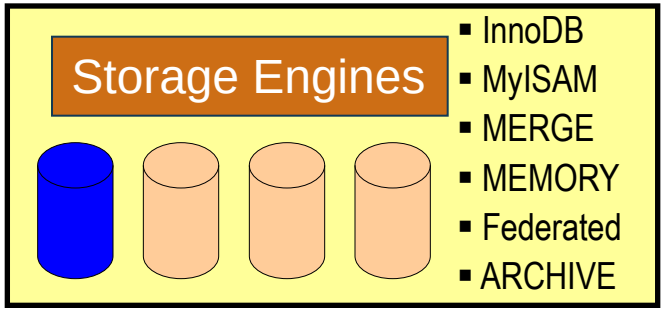
「ストレージエンジンでそんなに違いが出るの？」

秒間のINSERT件数で比較

ユーザ数	MyISAM	InnoDB	Archive
1	3,203.00	2,670.00	3,576.00
4	9,123.00	5,280.00	11,038.00
8	9,361.00	5,044.00	13,202.00
16	8,957.00	4,424.00	13,066.00
32	8,470.00	3,934.00	12,921.00
64	8,382.00	3,541.00	12,571.00

InnoDB ストレージエンジン

- InnoDBはACIDトランザクション対応：
  - MVCC
  - 分離レベルを設定可能
  - 行レベルロック



メモリ上のデータ

- データとインデックスをキャッシュ
- データの変更はメモリ上で
- ROLLBACKに備えUNDOログを用意
- チェックポイント毎にデータとUNDOログをディスクに書き込み

メモリ上のログ

- 全てのトランザクションの処理内容を記録

ディスク上のデータ

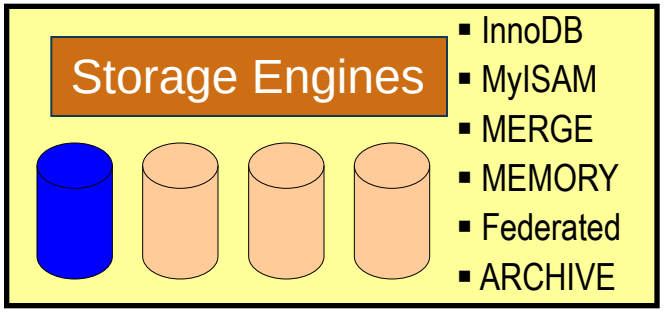
- データファイルへの書き込みはトランザクション中または後
- トランザクション中にデータが書かれた場合、UNDOログでROLLBACKに備える

ディスク上のログ

- トランザクションがコミット完了前に、REDOログに処理内容を記録
- REDOログ記録直後にコミットが完了
- バックグラウンドスレッドがメモリ上のREDOログをディスクに書き込み

InnoDB パフォーマンス Tips

- **innodb_buffer_pool_size**
  - MySQL&InnoDBのみを利用していれば、メインメモリの80%程度を割り当てる
  - データとインデックスの両方をキャッシュ
- **innodb_log_file_size**
  - innodb_buffer_pool_sizeの25%～100%
  - ログファイルがどの程度頻繁に切り替わっているかをチェック
  - 値を大きくするとクラッシュ後のリカバリ時間が長くなる
- **innodb_flush_log_at_trx_commit**
  - 1 (遅い) コミット時にログをフラッシュ。**真のACID**
  - 2 (速い) コミット時にはOSのキャッシュにログをフラッシュ、ディスクとのシンクは毎秒1回
  - 0 (最速) ログを毎秒1回(またはそれ以下)フラッシュ



mysql> SHOW INNODB STATUS;

InnoDBの内部での稼働情報

- ファイル IO
- バッファプール
- ログ情報
- 行/ロック情報

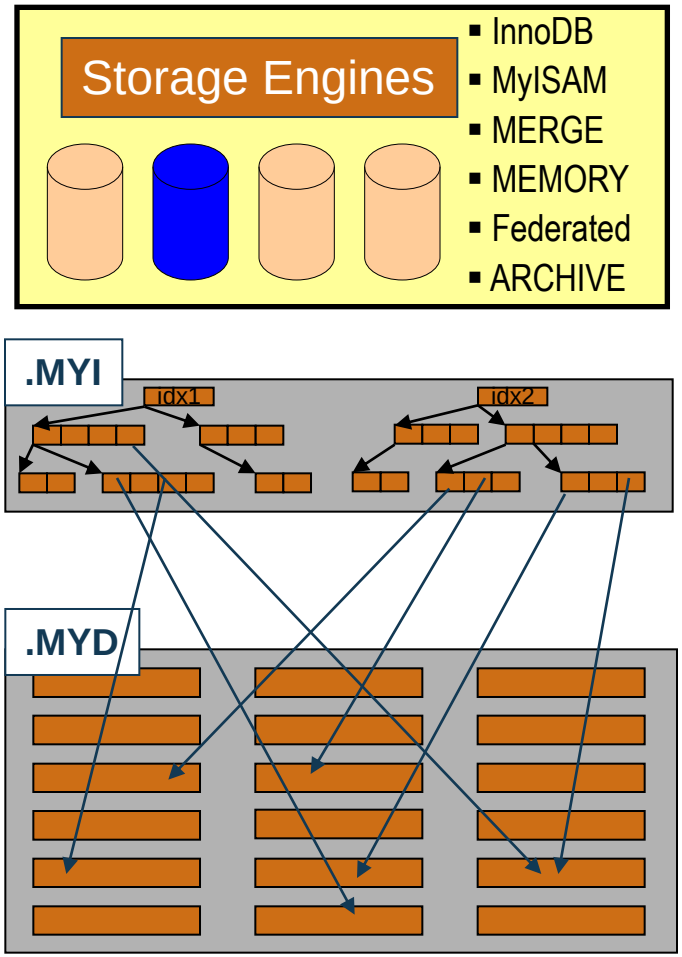
アクセス頻度の高いデータをメモリに格納

DBT-2 (W200)	Transactions per Minute	%user	%iowait
Buffer pool 1G	1125.44	2%	30%
Buffer pool 2G	1863.19	3%	28%
Buffer pool 5G	4385.18	5.5%	33%
Buffer pool 30G (All data in cache)	36784.76	36%	8%

- DBT-2 ベンチマーク (更新処理が多数)
- 20-25GBの”ホット”なデータ (200 warehouses, 1時間実行)
- Nehalem 2.93GHz x 8 cores, MySQL 5.5.2, 4 RAID1+0 HDDs
- メモリのサイズはSELECTだけではなくINSERT/UPDATE/DELETEにも影響
  - INSERT: インデックスにランダムに値と追加する際の参照更新
  - UPDATE/DELETE: 値を変更する際のランダムな参照と更新

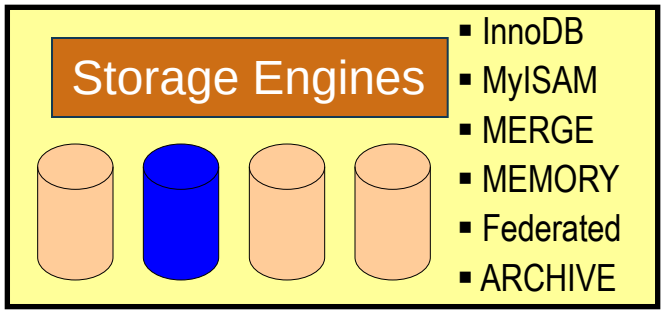
MyISAM ストレージエンジン

- トランザクション非対応、電源障害時などにデータ破損のおそれ有り
- ディスクやメモリのフットプリントは小さい
  他のRDBMSの 75%-50%程度
- テーブルロック、並列INSERT
- 読み取り専用にもできる
- インデックスのみがMySQLによってキャッシュ、データはOS任せ
- データロード性能、インデックス作成は高速
- 表はOSレベルでコピー可能



MyISAM パフォーマンス Tips

- **key_buffer_size (8M)**
  - MySQL&MyISAMのみを利用していれば、メインメモリの25-33%を割り当てる
- **myisam_sort_buffer_size (8M)**
  - インデックス作成時には大きくしておく
- **myisam_recover=FORCE,BACKUP**
  - テーブルアクセス時に破損をチェック。
- **delay_key_write=ALL**
  - インデックスをディスクに反映させるのをテーブルが閉じられたタイミングのみにする
  - クラッシュ時にデータが破損する可能性が向上するため注意



mysql> show status like 'key%'

- **Key_read_requests, Key_reads**
  - Key_reads/Key_read_requests キャッシュミス率
- **Key_write_requests, Key_writes**
  - キャッシュミス率は高いことが多い
- **Key_blocks_unused**
  - 値が大きい場合はキャッシュサイズが大きすぎる可能性大

スキーマのデザイン

- 正規化
  - OLTPや書き込み多い環境
  - データの冗長性を削減
  - JOINのオーバーヘッド
  - トータルのデータサイズは小さくなる
  - E/R図とスムーズに連携
- 非正規化
  - OLAPやレポーティング
  - データの冗長性が高い
  - JOINを削減できる
- データ型
  - tinyint, smallint, mediumint, などを使いデータ量を削減
  - JOINに使う列は同じデータ型に
  - char(64)ではなくvarchar(64)
  - 可能なところはNOT NULLを宣言
  - varchar(255)ではなくvarchar(64)
  - DECIMAL(9)ではなくINT
- インデックス
  - 複数列
  - プレフィックス
  - “covering index”

インデックス

- インデックスは参照時の性能は向上するが、更新時はオーバーヘッド
  - 小さなインデックス、プレフィックス index(name(8))
- MySQLはインデックス内で順序が先の列のみ利用可能
  - key (a,b)　.... where b=5 はインデックスを使わない
- インデックスは必要最小限に留めること
  - 例）性別に対するインデックス
- ユニークなインデクスにはUNIQUE キーワードをつける
- 重複するようなインデックスの利用は避けるべき
  - key(a, b) があるなら key(a) は削除
- BTREE インデックスはソートされた結果を返す
  - select * from t where b=5 order by c　… key(b,c) optimal
- “covering indexes”は高速、行のデータのフェッチが不要
  - select c from t where b=5 …の場合、key(b,c) が良い
- OPTIMIZE TABLE … でインデックスのソートと最適化

SQL オプティマイザの制御

- SELECT STRAIGHT_JOIN * from tbl1,tbl2 ...
  - SQL文に書かれたテーブルの順に処理を行う
- USE INDEX / FORCE INDEX / IGNORE INDEX
  - SELECT * FROM Country USE INDEX(PRIMARY)
  - ヒント句、MySQLでの利用ケースはあまり多くない
  - インデックスを強制的に使わせるケースはある
- ANALYZE TABLE …
  - 通常はあまり必要とされないが、大量にデータの更新があった後などに実行することも

データ型選択時における注意点

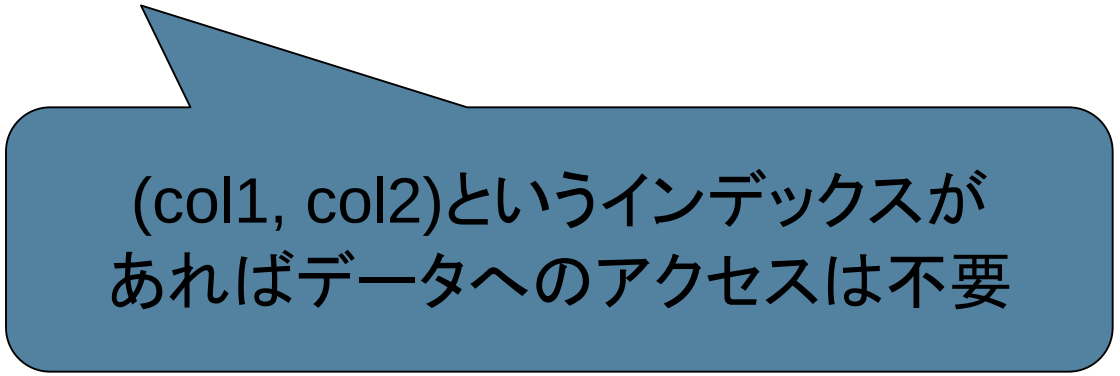
- 出来るだけ小さいサイズの型を使う。
  - 本当にその文字列長は必要か？
  - 文字コードを工夫（日本語が不要ならlatin1やascii、binaryを利用する）
  - ENUMの活用。文字列だが数値として格納される。例) ENUM('YES','NO')
- PROCEDURE ANALYSE()の活用
  - 格納されているデータを分析
  - 最適なデータ型を提案

マルチカラムインデックスの活用

- MySQLのオプティマイザが同時に利用出来るインデックスはひとつだけ
  - インデックスマージが効く場面は限定的
  - インデックスマージより遙かに高速
- 複数の検索条件、ソート条件がある場合はマルチカラムインデックスが役立つ！！

Covering Index

- インデックスだけでクエリを解決出来るようにすること
  - データへアクセスしないのでとても高速
  - EXPLAINコマンドのExtraフィールドは「Using Index」
  - これを目的として多くのインデックスを作成しすぎると
    更新性能は劣化するため注意が必要
- SELECT col1, col2 WHERE col1 = x
  ORDER BY col2;



MySQLにおけるパーティショニング

- パーティショニングとは？
  - データを特定のカラムの値によって分類
  - それぞれにデータ、インデックスを持つ
- 5.1から導入
  - 現行は同一のストレージエンジンを使用したもののみをサポート
  - 同じテーブル内に違うストレージエンジンのパーティションを持つことは出来ない。
- パーティションのタイプは4つ
  - Range ... カラム値の範囲を指定。
  - List ... カラム値をリストアップ。
  - Hash ... 数値カラムのハッシュ値で分ける。
  - Key ... 文字列カラムのハッシュ値で分ける。

ORACLE

MySQLサーバのログファイル

- MySQLサーバは4種類のログを出力
  - エラーログ (Error log)
    - syslog: mysqld_safe のオプションで出力可能
      - --syslog[=tag]
      - --skip-syslog (デフォルト)
  - 一般クエリログ (General Query log)
  - スロークエリログ (Slow Query log)
  - バイナリログ (Binary log)

クエリ監視 – スロークエリログ

```
#Time: 08073101 16:25:24
#User@Host: root[root] @ localhost [127.0.0.1]
#Query_time: 8 Lock_time: 0 Rows_sent: 20 Rows_examined: 243661
SELECT part_num FROM `inventory`.`parts` WHERE
(`ven` = "foo") ORDER BY `delivery_datetime` DESC LIMIT 100;
```

メリット
- 指定した時間を超えたクエリを記録
- インデックスを使わないクエリも記録可能 (5.0以降)
- 問題のあるクエリの追跡に必要なデータを追加

MySQLサーバ運用のポイント
- ログが大容量になると、FLUSH LOGS を使用して管理。
- エントリを解析/ソートして、関連性を調べる。
- mysqldumpslowでログ内容を集計

クエリ監視 – SHOW PROCESSLIST;

メリット
- 現在のプロセスを表示
- クエリ実行状況を表示
- 問題のあるクエリを追跡するために必要なデータを追加

MySQLサーバ運用のポイント
スクリプト作成:
- 自動化
- スロークエリログと統合
- 結果を集約して解析
- 問題発生時、DBAに通知

```
mysql> SHOW FULL PROCESSLIST¥G
******** 1. row ******************
Id: 1
User: MyUser
Host: localhost
db: inventory
Command: Query
Time: 1030455
State: Sending Data
Info: SELECT part_num from 'inv';
   …..
2 rows in set (0.00 sec)
```

問題のあるクエリを解決 – EXPLAIN;

```
EXPLAIN SELECT part_num
FROM `inventory`.`parts`
WHERE (`ven` = "foo")
ORDER BY `delivery_datetime`
DESC LIMIT 100;¥G

******** 1. row **************
           ID: 1
  select_type: SIMPLE
        table: parts
         type: ref
possible_keys: ven, part#
          key: ven
      key_len: 3
          ref: null
         rows: 872
        Extra: Using WHERE
1 row in set (0.00 sec)
```

解析
- インデックスがどのように使用されているか?
- ファイルソートが必要だったか?
- どのテーブル、カラムがクエリで使用されているか?

修正/チューニング – 以下の作業を繰り返し行う:
- インデックスを追加/変更
- テーブル定義、データ型など変更
- クエリ構造を変更

EXPLAINの各項目

- テーブルごとに1行出力される。
  - id... クエリのID（テーブルのIDではないので注意）
  - select_type... クエリの種類を表す
  - table... 対象のテーブル
  - type... レコードアクセスタイプ。どのようにテーブルにアクセスされるかを示す。
  - possible_keys... 利用可能なキー。
  - key/key_len... 選択されたキーとその長さ。
  - rows... 行数の概算見積もり。
  - Extra... オプティマイザヒント。

MySQL 5.5

InnoDBがデフォルトのストレージエンジンに

- ACIDトランザクション、外部キー、クラッシュリカバリ
- 性能/CPUスケーラビリティの向上、データ圧縮

高可用性の向上

- 準同期型(Semi-synchronous)レプリケーション
- レプリケーション・ハートビート

ユーザビリティの向上

- SIGNAL/RESIGNAL
- パーティショニングオプション追加
- PERFORMANCE_SCHEMA

MySQL 5.5 - 性能の向上

- InnoDBの性能改善点
  - Multiple Buffer Pool Instances
  - Multiple Rollback Segments
  - Extended Change Buffering
    (with delete buffering, purge buffering)
  - Improved Purge Scheduling
  - Improved Log Sys mutex
  - Separate Flush List mutex
- MySQLサーバの性能改善点
  - Better Metadata Locking within Transactions
  - Split LOCK_open mutex
  - Eliminated LOCK_alarm mutex as bottleneck
  - Eliminated LOCK_thread_count as bottleneck
  - Improved Performance/Scale on Win32, 64
- クラッシュリカバリの性能が10倍以上向上

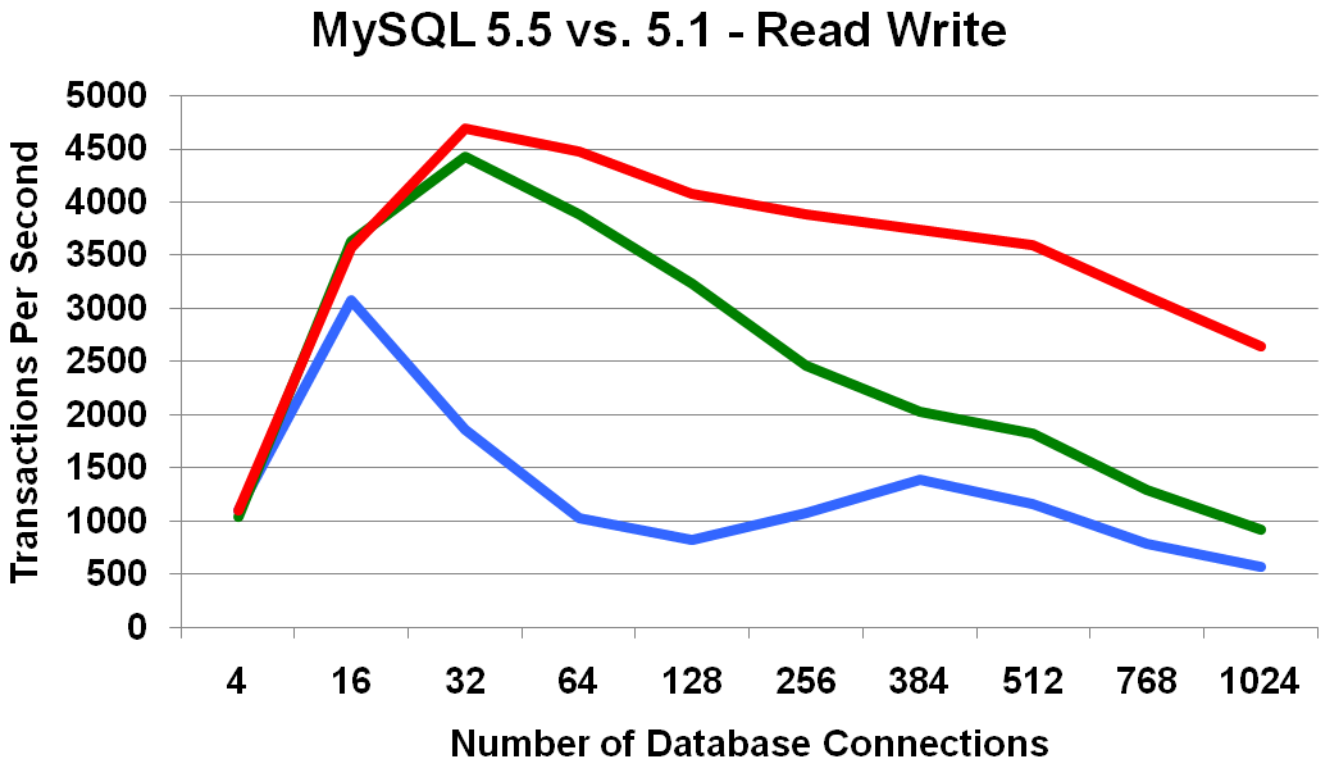
MySQL 5.5 - 性能の向上

- InnoDBの性能改善点
  - 複数バッファ・プール・インスタンス
  - 複数ロールバック・セグメント
  - 拡張変更バッファリング
    (削除バッファ、with delete buffering, purge buffering)
  - Improved Purge Scheduling
  - Improved Log Sys mutex
  - Separate Flush List mutex

- MySQLサーバの性能改善点
  - Better Metadata Locking within Transactions
  - Split LOCK_open mutex
  - Eliminated LOCK_alarm mutex as bottleneck
  - Eliminated LOCK_thread_count as bottleneck
  - Improved Performance/Scale on Win32, 64

- クラッシュリカバリの性能が10倍以上向上

MySQL 5.5 SysBench Benchmarks Linux



GA

MySQL 5.5.6
(New InnoDB)

MySQL 5.1.50
(InnoDB Plug-in)

MySQL 5.1.50
(InnoDB built-in)

370% performance gain

for MySQL 5.5 over 5.1.50; at scale

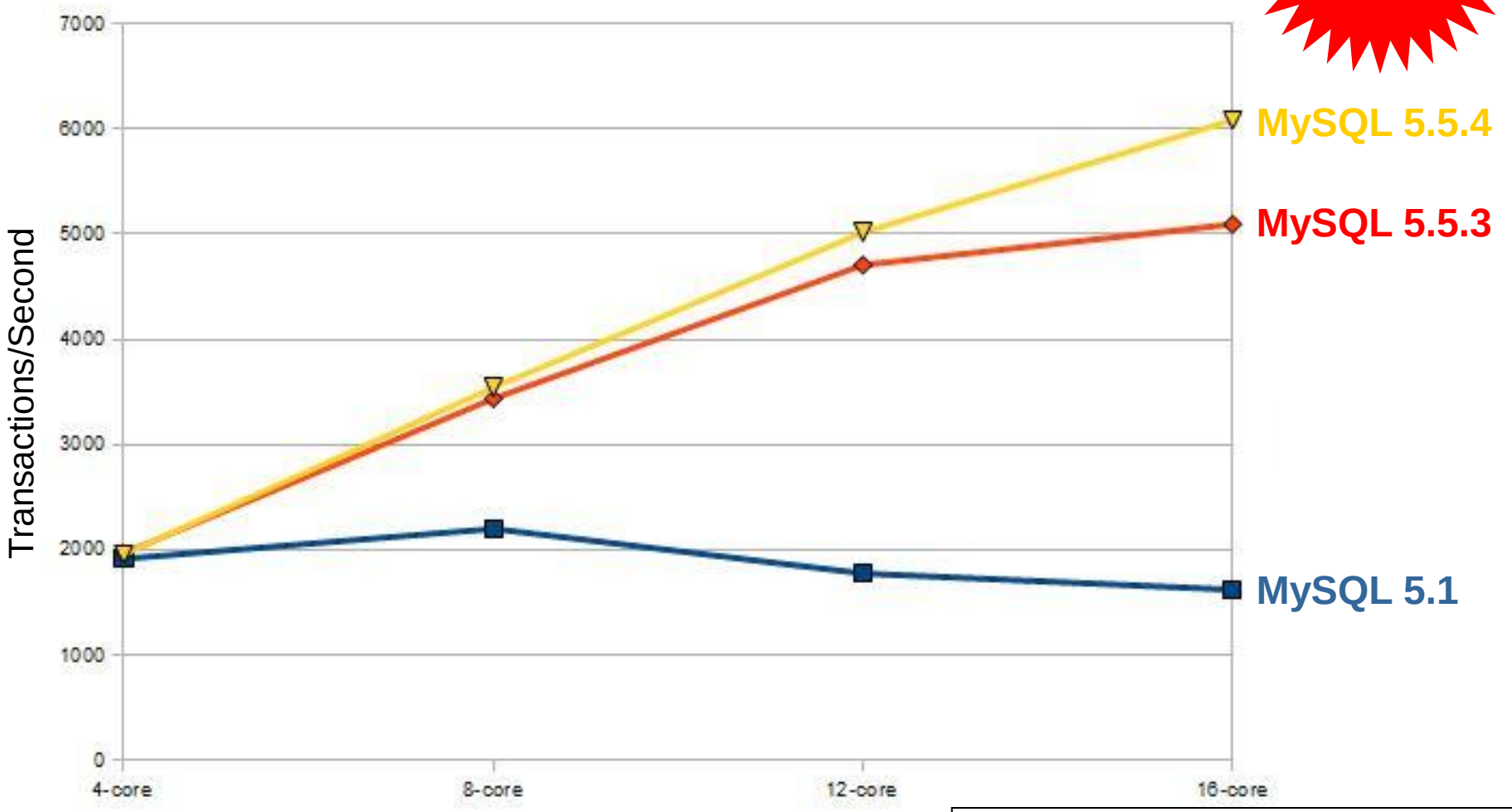
Intel Xeon X7460 x86_64
4 CPU x 6 Cores/CPU
2.66 GHz, 32GB RAM
Fedora 10

ORACLE

MySQL 5.5 Scales on multi core

SysBench Read Write

GA



AMD Opteron 7160 (Magny-Cours) @2100 MHz
64 GB memory
2 x Intel X25E SSD drives
OS is Oracle Enterprise Linux with the Enterprise Kernel
4 sockets with a total of 48 cores.

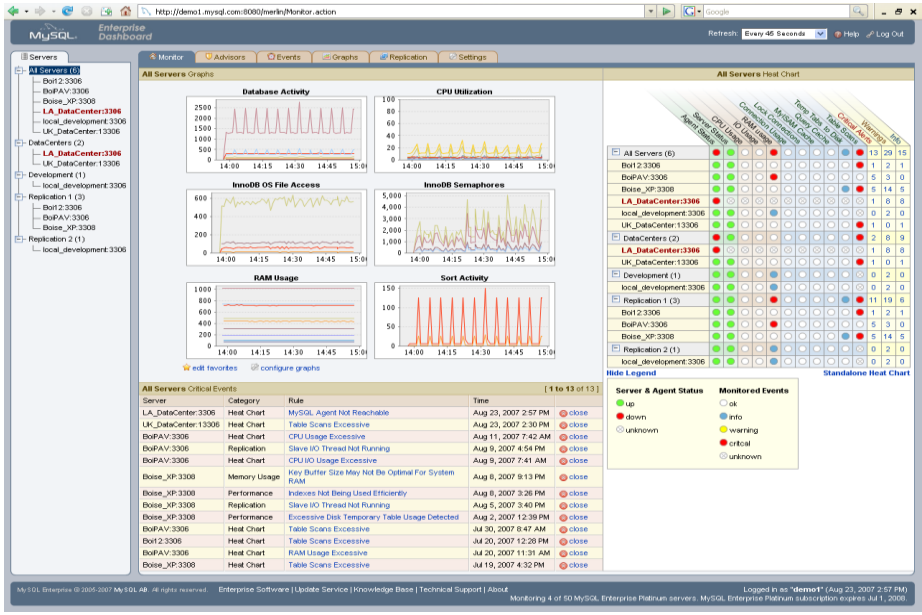
MySQL Enterprise Edition

MySQL Database 	• 高信頼性、高性能 • 運用の容易性
MySQL Enterprise Backup 	• **高速**オンラインホットバックアップ • ポイントインタイムリカバリ
MySQL Enterprise Monitor 	• 全MySQLサーバの一括監視 • **MySQL Query Analyzer**
MySQL Workbench 	• データベース設計 & アプリ開発 • 管理ツール MySQL Administration
MySQL Enterprise Security 	• **External Authentication 外部認証** • LDAP, Kerberos, Windows AD など
MySQL Enterprise Scalability 	• Thread Pooling • 持続可能な**高性能**
Oracle Premier Support 	• 24x7, 無制限インシデント • **コンサルテイティブサポート**

ORACLE

MySQL Enterprise Monitor

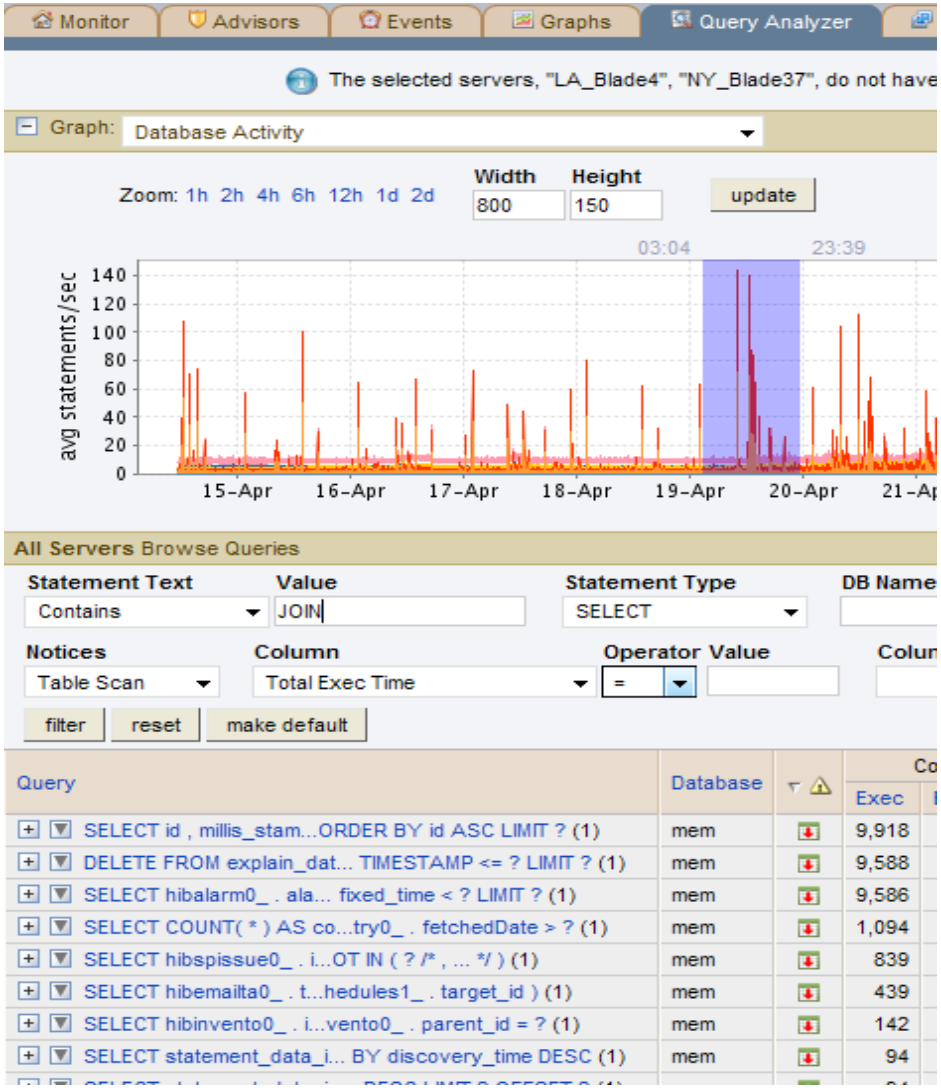
- 複数のMySQLサーバを一括監視可能なダッシュボード
- システム中のMySQLサーバやレプリケーション構成を自動的に検出し監視対象に追加
- ルールに基づく監視と警告
- **問題が発生する前に通知**
- 問題のあるSQL文の検出、統計情報の分析が可能なQuery Analyzer



“バーチャルなMySQL DBA”アシスタント

クエリ解析機能 - MySQL Query Analyzer

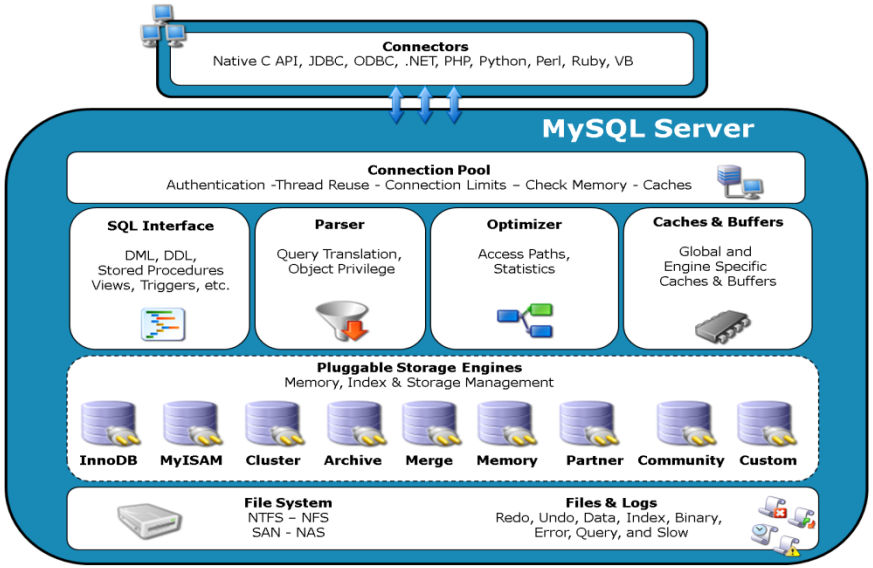
- 全てのMySQLサーバの全てのSQL文を一括監視
- vmstatなどのOSコマンドやMySQLのSHOWコマンドの実行、ログファイルの個別の監視は不要
- クエリの実行回数、エラー回数、実行時間、転送データ量などを一覧表示
- チューニングのための解析作業を省力化



MySQL Enterprise Scalability

MySQL Thread Pool

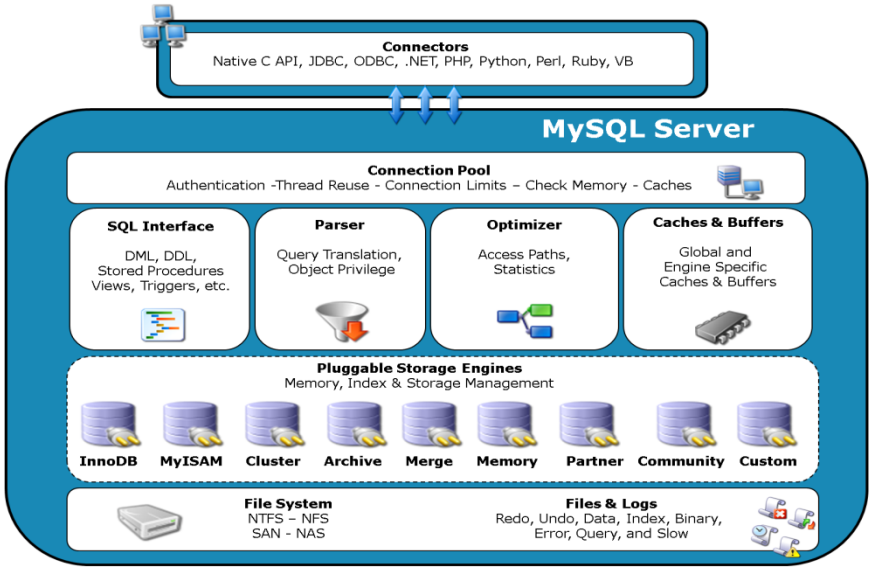
- MySQL default thread-handling – excellent performance, can limit scalability as connections grow
- MySQL Thread Pool improves sustained performance/scale as user connections grow
- Thread Pool API



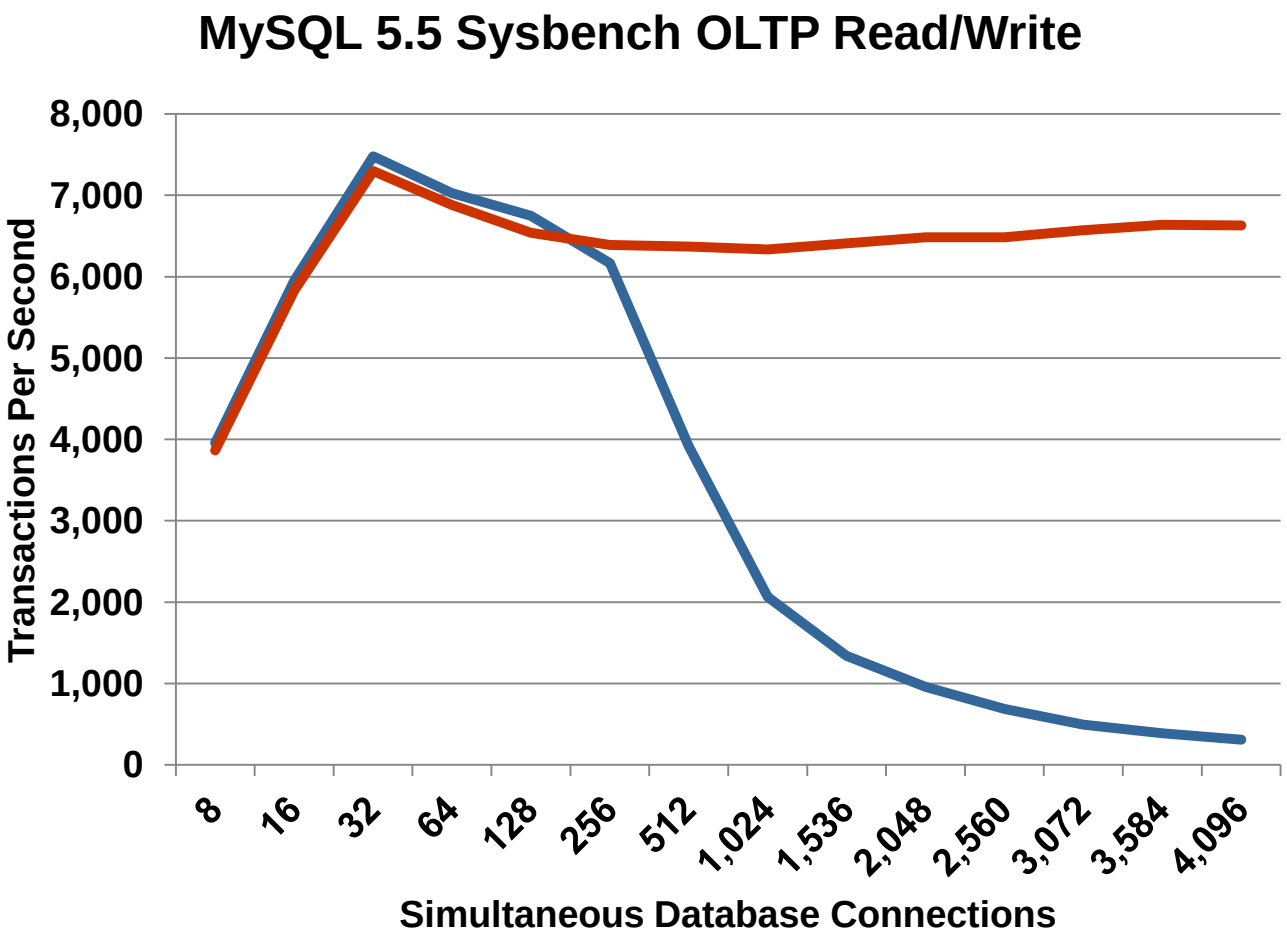
MySQL Enterprise Scalability
MySQL スレッド・プール

New!

- MySQL デフォルト・スレッド処理
  優れたパフォーマンスであるが、接続の増加に対する
  スケーラビリティに限界
- MySQL スレッド・プール
  接続の増加に対し、パフォーマンス/スケーラビリティが持続
- Thread Pool API



MySQL Enterprise Scalability



MySQL Enterprise Edition

With Thread Pool

MySQL Community Server

Without Thread Pool

MySQL 5.5.16
Oracle Linux 6.1, Unbreakable Kernel 2.6.32
2 sockets, 24 cores, 2 X 12-core
Intel(R) Xeon(R) X5670 2.93GHz CPUs
72GB DDR3 RAM
2 X LSI SCSI Disk (MR9261-8i) (597GB)

20x Better Scalability with Thread Pool

Hardware and Software

ORACLE®

Engineered to Work Together


ORACLE®