MySQL, PHP, Stuff

PHPCon East 2003

Jeremy Zawodny

Yahoo!

April 24th, 2003

New York

http://jeremy.zawodny.com/mysql/

About Me

- Engineer in Y! Search (prev. Y! Finance)
- MySQL user for over 5 years
- Active in MySQL community
- Write about LAMP for Linux Magazine
- MySQL advocacy & support at Yahoo!

`Mail: Jeremy@Zawodny.com`

`http://jeremy.zawodny.com/mysql/`

Outline

- MySQL
  - Versions
    - Features
    - Recommendations
  - Performance Tips
- PHP
  - Advice w/MySQL
  - New Stuff
- Other Stuff
- Q&A

MySQL at Yahoo!

- Roughly 200-400 servers world-wide
- FreeBSD and Linux
- Commodity hardware
- Replaces home-grown “database” systems
- Replaces Oracle in a few cases
- Typical install uses between 1-20GB
- Used both “live” and in batch processing
- Replication and load-balancing

Starting Questions

- What version of MySQL are you using?
- What languages are being used?
- Which operating systems?
- Familiarity with other RDBMS servers?
- Role? DBA? Developer? SysAdmin?
- MySQL dedicated or shared servers?
- How fast is your growth?
  - Transaction rates
  - Data volume

MySQL 3.23

- Stable
- Reliable
- Fast
- Standard on all Linux distributions today
- “Standard” and “Max” versions
- Max features InnoDB
  - Transactions
  - Row-level locking
  - Foreign keys

MySQL 3.23

- Introduced MyISAM to replace ISAM
- Full-text search support
- Handles very large data
- Built-in replication
  - Scaling is easy for read-intensive apps
- Only critical bugs will be fixed in 3.23
- Recommendations
  - Use 3.23 is you’re conservative
  - Think about when you can upgrade

MySQL 4.0

- "Production ready" as of 1 month ago
- InnoDB is standard
- Full-text search is much improved
  - Indexing is faster
  - Boolean searching
    - (+"microsoft windows" –"rocks")
  - Stop word list customization
- Replication re-worked
  - Dual threaded process
  - De-couple relay and execution

MySQL 4.0

- Query optimizer improvements
- Text mactching is faster
- Query cache
- SQL UNIONs
- On-the-fly tuning
- Bug fixes and minor improvements for 4.0
- Recommendations
  - Use 4.0 for new applications
  - Think about migrating to 4.0

MySQL 4.1

- Sub-queries!
- Internationalization
  - Per server/database/table/column character set selection
- Spatial data types
  - 2-D shapes (point, line, polygon, etc.)
  - GIS/mapping applications
  - PostgreSQL has had this for a while
- First alpha releast roughly 1 month ago
- Most new development going into 4.1

MySQL 4.1

- New “binary” protocol
  - Prepared statements
  - Big performance boost
- Recommendations
  - Look at MySQL 4.1 for applications you’ll build later this year
  - Consider the new mysqli PHP extension

MySQL 5.0

- Stored procedures!
  - Technically SQL-99 PSMs (persistent storage modules)
- Being developed in paralell with 4.1
- More full-text improvements
  - Per-table or per-index stop words, lengths
- Recommendations
  - It will be at least a year before you'd think about building production applications on 5.0
  - But it's still fun to play with and to track development

MySQL Performance Tips

- Query optimization
  - Enable the slow query log
  - Learn to use and read EXPLAIN output
  - Understand how indexes help
    - The “leftmost prefix” rule
  - Don’t ask for unnecessary data
    - SELECT * syndrome
  - Use the query cache (4.0+)
  - Try re-phrasing queries

MySQL Performance Tips

- Application Design
  - Use the right column types
  - Use the right table types
    - Concurrency/Locking
    - Features: full-text, foreign keys, etc.
  - Cache infrequently changed data
    - Or use HEAP (in-memory) tables
  - Don’t over-use sessions
  - Plan for growth, possibly using replication
  - Use transactions where they make sense

MySQL Performance Tips

- Server Tuning
  - Read and understand SHOW STATUS output
    - Bytes in/out per second
    - Queries per second
    - Active vs. idle vs. max connections
  - Understand critical resources
    - Memory
    - CPU
    - Disk I/O
- Customize your configuration file
  - Defaults are *very* conservative!

MySQL Performance Tips

- Memory use is very important
  - Global caches/buffers
    - key_buffer
    - innodb_buffer_pool
    - table_cache
    - thread_cache
  - Per-thread caches/buffers
    - sort_buffer
    - record_buffer
    - join_buffer
- Leave some memory for the OS

PHP and MySQL

- Benchmarking
  - PHP: ab (apache bench)
  - MySQL: mysql-super-smack
  - Many problems appear only under load!

PHP and MySQL

- Persistent connections
  - MySQL connection overhead is pretty small
    - Server-side resources are minimal
    - The protocol is light
  - To help even more
    - Disable DNS lookups
    - Set a reasonable thread_cache value

PHP and MySQL

- Sessions
  - Be careful with MySQL-based session data
  - It's easy to over-use
  - Cookie-based sessions are often sufficient
  - Can be problematic w/replication and load-balancing setups

PHP's mysqli extension

- Using PHP4+ and MySQL 4.1+
- Written by Georg Richter <georg@php.net>
  - 70+ functions
- Improve performance of
  - repetitive non-SELECT queries
  - non-cacheable SELECT queries
- Send the server a query to parse & cache
- You get back a statement handle
- Execute the statement many times
- May not benefit all web apps
- Can be a big help to batch processing
- Application servers and middleware

PHP’s mysqli extension

- Classified as “experimental” right now
- Requires the MySQL 4.1 client library
  - Will be bunded in the future
- Can make replication-aware apps easier

The mysqli API

```
<?php
// normal query
$link = mysqli_connect("localhost", $user, $passwd);
$rc = mysqli_query($link, $sql);

// prepare select, bind, execute, fetch, close
$stmt = mysqli_prepare($link, "SELECT col1, col2 from my_table");
mysqli_bind_result($stmt, &$c1, &$c2);
mysqli_execute($stmt);
mysqli_fetch($stmt);
$test = array($c1,$c2);
mysqli_stmt_close($stmt);
mysqli_close($link);
?>
```

The mysqli API

```
<?php
// connect, prepare insert, bind, execute, close

$link = mysqli_connect("localhost", $user, $passwd);
$stmt = mysqli_prepare($link, "INSERT INTO my_table VALUES (?,?)");
mysqli_bind_param($stmt, &$d1, MYSQLI_BIND_STRING,
                         &$d2, MYSQLI_BIND_STRING);
$d1 = ‘MySQL';
$d2 = ‘PHP';

// the execute could be in a loop to insert many values
mysqli_execute($stmt);
mysqli_stmt_close($stmt);

// for replication setups
mysqli_slave_query($link, $sql);
mysqli_master_query($link, $sql)
?>
```

Stupid Query Tricks

- Use SQL_CALC_ROWS and FOUND_ROWS() rather than double-queries:
  - SELECT … LIMIT N, M
  - SELECT COUNT(*)
- Instead:
  - SELECT … LIMIT N, M
  - SELECT FOUND_ROWS()
- Requires far less overhead on MySQL

Stupid Query Tricks

- Use a UNION to re-write a slow OR query

```
SELECT * FROM mytable
WHERE col1 = ‘foo’ OR col2 = ‘bar’
```

```
(SELECT * FROM mytable
WHERE col1 = ‘foo’)
UNION
(SELECT * FROM mytable
WHERE col2 = ‘bar’)
```

Final Advice

- Read
- Learn
- Test
- Ask
- Monitor
- Benchmark

For More Info…

- MySQL mailing lists
  - Visit lists.mysql.com
- Books
  - MySQL Manual
  - MySQL (Paul’s Book)
  - Managing & Using MySQL
- Web searching

Questions and Answers