MySQL and PHP – State of the Union

Connectors, Best Practices, Performance, and the “Cloud”

New York City MySQL Meetup
December 9th, 2008

CommunityOne East

Sun Microsystems

March 18th, 2009

Web Performance for PHP Developers

Web Performance Meetup/Etsy.com

June 16th, 2009

Scaling & Optimizing MySQL

Sun Microsystems

December 16th, 2008

Hans Zaunere, Managing Member

Overview

Introduction

AMP Needs No Introduction

- Apache/MySQL/PHP
- Other variations on the theme
  - But the fundamentals remain

PHP glues together high-speed database with high-speed external libraries

LAMP

BAMP

OSAMP

SAMP

WAMP

XAMP

DAMP

Connectors…

I'm a php

- mysql – The Classic
- mysqli – Improving a Classic
- PDO – Abstracted Abstraction
- mysqlnd – A PHP Native

Pick the Right Connector

Connectors…

mysql Extension - The Classic

- http://www.php.net/mysql
- Still one of the fastest and quickest to implement
- Results are always strings
- Very mature and proven code
- Strictly functional
- No binary protocol support
- Always be real
  - mysql_real_escape_string() EVERYTHING

Connectors…

mysqli Extension – Improving a Classic

- http://www.php.net/mysqli
- Recommended for all new development
- Supports new and old MySQL features
- Prepared statements and binary protocol available
  - Result sets are returned as PHP native types
  - Some advantages/disadvantages
- Choice between functional or object-based code
- Just got persistent connections
  - Now “idiot proof” and yields almost 7x connections/second gain

Connectors…

PDO – Abstracted Abstraction

- http://www.php.net/pdo
  http://www.php.net/pdo-mysql
  http://dev.mysql.com/tech-resources/articles/mysql-pdo.html
- **P**HP **D**ata **O**bject
- Provides consistent OO interface to multiple database
  - May deviously hide differences between databases
    Know thy configuration and available database features
- Segmentation fault issues early on – still gun shy
- Still being developed, including support for mysqlnd
- Recommended only if abstraction is an immediate requirement

Connectors…

mysqlnd Library – A PHP Native

- http://blog.ulf-wendel.de/ (current) http://blog.felho.hu/what-is-new-in-php-53-part-3-mysqlnd.html
- Optional backend library for mysql/mysqli/PDO
- Takes advantage of PHP/Zend Engine internal structure
- Supports async queries
  - Deprecates --enable-mysqlnd-threading
- Long time in coming but still *apparently* developed
- Available in PHP 5.3 (alpha)
- Not recommended for prime time quite yet

Connectors

…Survey Says

- Stick with - or upgrade to - **mysqli**

Best Practices…

- Modern applications should use mysqli
- Know your hardware architecture
  - Use x64 where possible for RAM
  - Avoid mixed 32/x64 environments/libs
  - PHP needs --with-libdir=lib64
  - In production environments strive to have a pure lib64 system
  - Install and link against the right libraries for your architecture
- Know your compile options
  - Don’t let ./configure guess
  - --with-mysqli=/usr/bin/mysql_config - **specify file!**
  - --with-mysqli=mysqlnd (alpha)
- Know your client/server versions
  - Make sure they match as closely as possible

…and Techniques…

- mysqli objects or functions? Your choice but…
  - Objects may provide a more flexible, elegant and fool-proof implementation

- Be explicit
  - Specify the MySQL resource or result in **every** function
  - mysqli's OO implementation enforces better practices

```
$MYDB = mysql_connect('www.nyphp.com','root','=ro0t--');

// BAD
$R = mysql_query('SELECT * FROM BB.Account');

// GOOD
$R = mysql_query('SELECT * FROM BB.Account',$MYDB);
```

…and Techniques

- Be prepared with prepared statements?
  - Consider what the actual benefits are for your application
  - Not all applications may benefit

- What’s wrong with this picture?

```
$R = mysql_query('SELECT * FROM BB.Account',$MYDB);

$R2 = array();

while( ($R2[] = mysql_fetch_assoc($R)) !== FALSE );
```

Performance and Scaling…

www.mysqlperformanceblog.com

- Consider use case
  - Need to support varied queries in a framework environment?
  - … or highly repetitive queries in a batch environment?
- Consider highest-cost resources for your application
  - RAM? CPU? Database round-trips?
- Consider your data's flavor
  - Big blobs? Small strings and ints?
  - Lots of rows? Small result sets?
  - Complexity and relationships?

Performance and Scaling…

- Consider memory usage vs CPU
  - Storing results as an array of arrays (non-columnar)
    - mysqli_result_fetch_all(), mysqli_result_fetch_row(), etc.
  - Storing results as an columnar array
    - mysql_stmt_fetch_column() available in C but not in ext/mysqli (**NEEDED**)

```
// 1663 rows
$R = mysql_query('SELECT * FROM BB.Account',$MYDB);
$R2 = array();

// Non-columnar      0.034s            3.99mb
while( ($T = mysql_fetch_assoc($R)) !== FALSE )
    $R2[] = $T;

// Columnar          0.048s 41.18%     3.37mb -15.54%
while( ($T = mysql_fetch_assoc($R)) !== FALSE )
{
    foreach( $T as $Col => $Val )
            $R2[$Col][] = $Val;
}
```

Performance and Scaling…

Connectors - Connectors - Connectors

- Simple benchmarking of the current Connector scene
  - 500 executions of a simple SELECT
  - Retrieving 1663 rows each time

```
mysql> SELECT * FROM BB.Account

 `AccountGID`          bigint(20)
 `R_UserGID`           bigint(20)
 `InsertedTS`          timestamp
 `UpdatedTS`           timestamp
 `LastLoginTS`         timestamp
 `Status`              varchar(31)      collate utf8_unicode_ci
 `Type`                varchar(31)      collate utf8_unicode_ci
 `Admin`               tinyint(3)
 `Bucks`               decimal(15,2)
 `BID`                 varchar(11)      collate utf8_unicode_ci
 `CurrentAlias`        varchar(63)      collate utf8_unicode_ci
 `Description`         varchar(1023)    collate utf8_unicode_ci
```

Performance and Scaling…

Text Queries (Text Protocol)

	store_result (buffered)	**use_result (unbuffered)**	**Notes**

ext/**mysql**	18.2s	10.0s	mysql_query()
			mysql_unbuffered_query()

ext/**mysqli**	19.2s	10.0s	mysqli_result::free() required for unbuffered
ext/mysqli::**mysqlnd**	20.7s	10.0s	mysqli_result::fetch_all() not better than for() iteration mysqli_result::free() required for unbuffered

- All values available as PHP strings, regardless of column’s type
  - NULLs generally remain as NULLs in PHP
- Always *_free_result() as a best practice
  - Required for unbuffered queries

Performance and Scaling…

Prepared Statements (Binary Protocol)

	store_result (buffered)	**use_result (unbuffered)**	**Notes**
ext/**mysqli**	12.5s/12.3s	8.4/8.2s	prepare on execute / prepare once
ext/mysqli::**mysqlnd**	14.6s/14.2s	8.4s/8.2s	mysqli_result::free() required for unbuffered
ext/**mysqli**	26.6s	21.0s	additional PHP coding to mimic mysqli_result::fetch_assoc() behavior
ext/mysqli::**mysqlnd**	29.2s	22.4s	

- Each prepare requires an additional database roundtrip
  - http://forge.mysql.com/worklog/task.php?id=3359
- All values available as PHP types corresponding to column type
- *_free_result() and *_stmt_close() not always needed
  - But recommended as best practice

Performance and Scaling…

Observations and Upshots

- No significant difference between localhost vs remote
  - …as it relates to buffered vs. unbuffered
- mysqlnd has actually appeared slower overall
  - Slowdown in handling of PHP variables – **WHAT?**

- Framework operations
  - Handle a variety of queries
  - Typically pull rows for later processing
  - Unbuffered (**use_result**) is faster **but**..
    - Avoid lengthy per-row processing retrieval as the tables are locked until all rows are fetched (mostly a MyISAM problem)
    - Use buffered (**store_result**) in this case

Performance and Scaling…

Observations and Upshots

- Specific/batch operations
  - Prepared statements are better
  - Large data, highly repetitive
  - Not well implemented for general query backend
  - Complex logic required to mimic array/object row fetching – **SHAME**

- Generally - **or** - all things considered…
  - No **huge** difference between prepared and text queries
  - Prepared problematic depending on data structures required

YMIGTV – Your Mileage Is Guaranteed To Vary

Performance and Scaling…

Optimize Optimize Optimize

- Buffers and configuration
  - Tune buffers for storage engines and operations
- Queries
  - Indexes and correct usage absolutely critical
  - Avoid automatic query generation
  - **WRITE QUERIES CAREFULLY**
- Hardware
  - RAM is fast but disks can be too… pick your battles
  - Augment CPU for crunch-intensive applications

MySQL Responds Very Well To The Right TLC

Performance and Scaling…

I'll Say It Again

- Buffers and configuration
  - Architect the storage engines/schema from the outset
  - Tune the right buffers (key_buffer_size, innodb_*, heap, etc)
- Queries
  - Try to use numeric keys
  - Double check EXPLAIN – left/to/right/prefixed
  - Partitioning and disk layout
  - Be careful with sub-queries and temp. tables

Convenience Kills Performance
MySQL Will Kill You Without TLC

…Scaling and Performance

Scaling Performance…?

- Define scaling for **your** application and requirements
  - **Frequent** or **Complex** functional changes vs. **moderate** traffic
    - Functional Scaling
  - **Stagnant** or **Simple** functional changes vs. **huge** traffic
    - Traffic Scaling

- Sharding i.e. **Application Managed Partitioning**
  - Functional vs. Key vs. Combination
  - Be ready for complexity – Bring in expertise

- Add in **memcached** for shard recombination caching

Scaling and Performance

Scaling Performance…?

- Replication – Tried and True
  - Able to handle very heavy load if done correctly
  - Comfortable with both Functional and Traffic scaling
  - **Master-Master** is an option if application is aware
- MySQL Cluster
  - Can provide some of the best performance in the industry…
  - … but only in specific cases
  - Pairs well with **Sharding** as a replacement for memcached
- Keep tabs on your data's path, lifecycle and type
  - Know where it's come from, what it's doing, and where it's going

Know Thy Data – Love Thy Data

Clouds and Smog

Wait, what does that mean?

- Keep Your Feet On the Ground – And Your Head Out of the Cloud
  - Clouds mean a lot of different things right now
- Don’t put everything on one server – duh
- There’s no silver bullet – Don’t try to cheat
  - Give it time, progress is being made
- No cloud will simply scale a database – why…?
  - Cloud isn’t parallel processing (yet)
- Varying types of Clouds
  - Application/API - Azure, Google Apps
  - Virtualization - Sun, EC2/Xen, VZ, VMWare
  - Start-up marketing fluff – **BE WARY**

It’s All About Architecture and Optimization ALWAYS

Thank You

hans.zaunere@nyphp.com

For renowned online support, New York PHP Mailing Lists are free and available to anyone:

http://www.nyphp.org/mailinglists.php

MySQL SIG:

http://lists.nyphp.org/mailman/listinfo/mysql