



PERCONA
Performance Consulting Experts

Taking Advantage of

MySQL Performance Improvements

Baron Schwartz, Percona Inc.

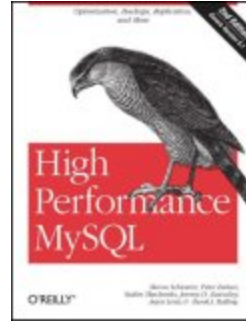
Introduction

- About Me (Baron Schwartz)
 - Author of High Performance MySQL 2nd Edition
 - Creator of Maatkit, innotop, and so on
- About Percona
 - We make MySQL more valuable to users & clients
 - Services: Consulting, Support, Training, Development
- About Percona Server with XtraDB
 - 100% free and open-source
 - Improved performance and features

This is me!



This is my book!



<http://tinyurl.com/highperfmysql>



This is the book you are looking for.

About ODTUG Kaleidoscope

- Long Beach, California, June 26-30 2011
- 250+ Sessions with 1 Full Track for MySQL
- <http://www.kscope11.com>



How do you build a
high-performance web app on MySQL?

How do you build a high-performance app, period?



Recent Trends in Web Apps

- Lots of memory
- Cloud computing
- Database sharding
- Agile development techniques
- Many-CPU hardware
- Flash and hybrid storage (SSD, FusionIO)
- Non-ACID databases (NoSQL)
- Sophisticated ORMs and frameworks

How can we make sense of all this?

Recent Hardware Changes

- Bigger
- Faster
- Cheaper
- More cores

Hardware Cost Factors

- Power and cooling
- Rack space
- Procurement and sticker price
- Operations: app complexity, failures

Recent Software Changes

- So-called “NoSQL” databases
 - Are they really a solution?
 - Sometimes, yes.
- Application frameworks (Ruby On Rails)
- Agile development
 - Continuous integration
 - Constant testing
 - Frequent small deploys
- Sharding has become really popular

What about Cloud Computing?

Databases in the Cloud

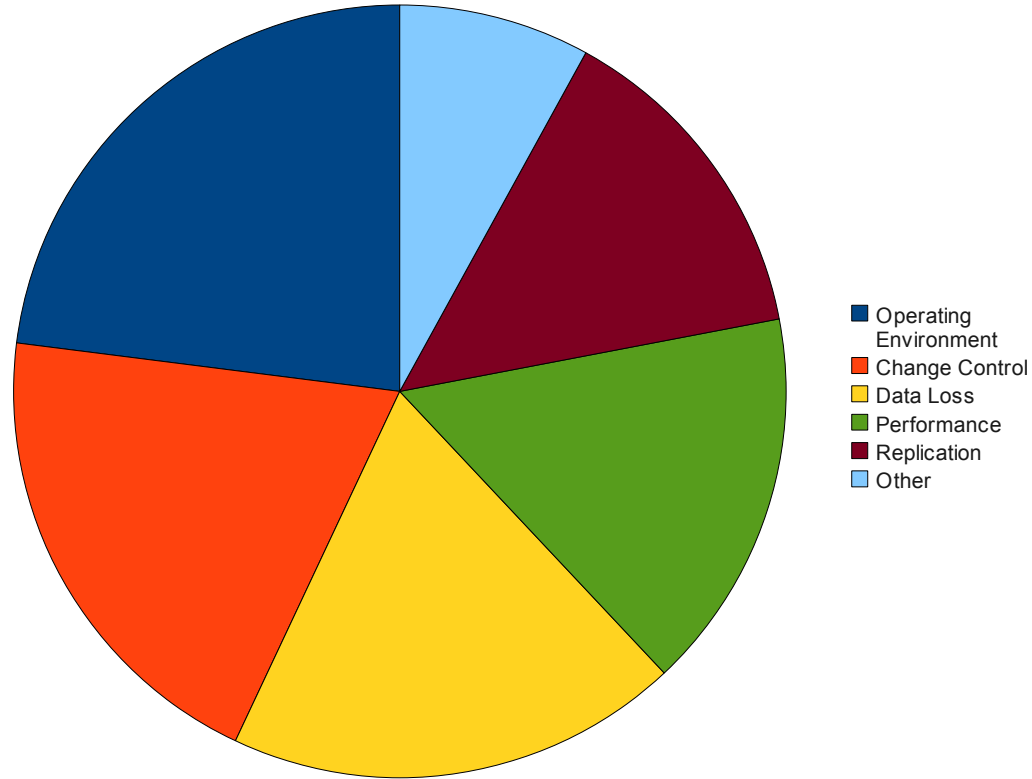
- What does a database need?
 - Memory
 - CPU
 - Storage
 - Network
- Which of those work well in the cloud?
- What happens if those don't work well?
- How can you work around that?
 - ... yeah, that's right, sharding.

Where am I going with all of this?

Bringing it back to performance

- A system that's down isn't high-performance.
- Therefore,
 - Strive for uptime first.
 - Then strive for performance next.

Causes of Downtime



Causes of the Causes

- System complexity
- Inexperienced staff
- Poor leadership
- Accepting defaults
- Wrong application architecture
- Many different technologies

Causes of the Causes of the Causes

Complexity adds cost.

Complexity adds points of failure.

Complexity makes people malfunction.

Complexity kills.

Performance & Reliability Tip #1

Make things as simple as possible, but no simpler.

- Einstein

Add exactly as much redundancy as needed.

- Your Humble Servant

Ideal Application Scaling Solution

- Buy powerful commodity hardware
- Use proven solutions such as a good RAID card
- Make everything redundant, except...
 - Run on a single database server if you can!
- Take and verify backups often
- Monitor, measure, and do capacity planning
- Do not shard unless you have to

Why is this ideal?

Because it's simple.

- Simple is cheaper.
 - Less hardware.
- Simple is more reliable.
 - Fewer points of failure.
- You can hire for simple.
- You can troubleshoot and analyze simple.
- Simple is cascading.
 - Your entire app will be simpler if your DB is simpler.

Let's Talk About the New MySQL!

(It makes simple possible.)

Old Performance Truths

- Early 2008, with 5.0 stable and 5.1 beta:
 - MySQL (InnoDB) couldn't use many CPUs
 - MySQL (InnoDB) couldn't use much I/O
 - MySQL (InnoDB) couldn't use much memory
- The key bottleneck was InnoDB
 - Mutex contention
 - Hard-coded defaults from the olden days
 - Inefficient internal algorithms

Even Worse...

- You couldn't measure what MySQL was doing.
- You cannot improve what you cannot measure!

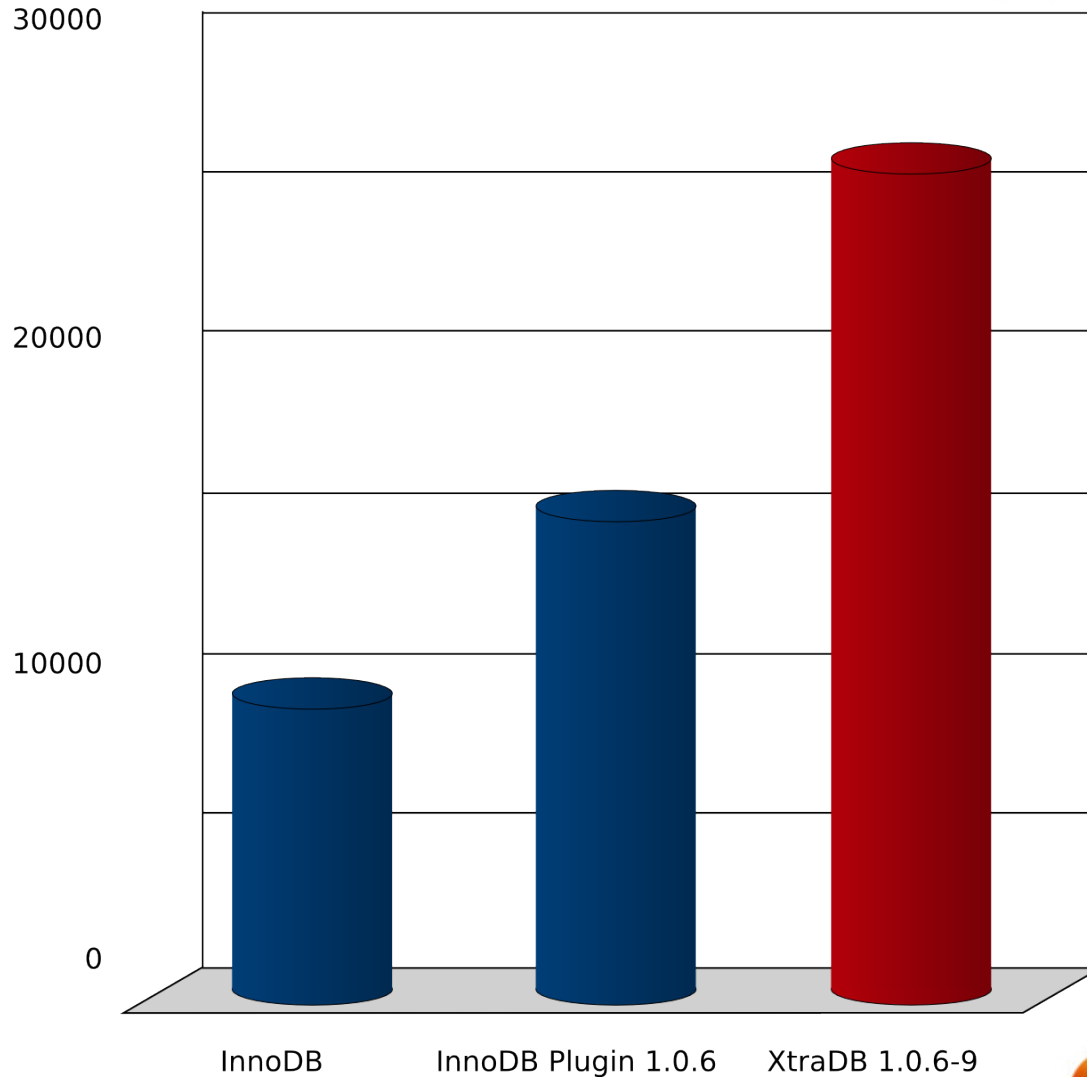
Three Major Problems

- Scalability limitations
- Single-node performance limits
- Lack of instrumentation

What's Changed?

- InnoDB Plugin for MySQL 5.1 (now GA)
- Improvements in MySQL 5.5 (not yet GA)
- Google's enhancements
- Percona's enhancements

XtraDB (in January, outdated!)



Better Performance in a Nutshell

- Use InnoDB Plugin or XtraDB
- Set the buffer pool BIG
 - Reserve memory for the OS, don't use a ratio of RAM
- Set `innodb_flush_method=O_DIRECT`
 - You must have a good RAID controller with BBU
 - Also set `swappiness` to 0 and block scheduler to deadline
- Enable multiple read and write IO threads
 - `innodb_[read|write]_io_threads = <number of disks>`
- Increase `innodb_io_capacity` to $\sim 150 * \text{num disks}$

Better Performance, Cont'd

- Set the `innodb_log_file_size` to at least an hour's worth of writes
 - Watch `Innodb_os_log_written`
- Use a separate purge thread (XtraDB only)
 - Set `innodb_use_purge_thread = 1`
- Use adaptive checkpointing
 - Set `innodb_adaptive_checkpoint = estimate`
- Set `skip_name_resolve`
- Disable the query cache
 - Set `query_cache_type = 0`, set `query_cache_size = 0`

Performance Analysis

- Performance = Response Time.
 - The key is to understand, measure, and optimize this
- Use the “slow query log”
 - Set `long_query_time=0`
 - Analyze the results with `mk-query-digest` from Maatkit
- Advanced techniques
 - Use after application optimization is exhausted
 - Learn to use `strace`, `oprofile`, and `gdb`

Current Scaling Possibilities

- Today we can scale *much* bigger than in 2008
 - Hundreds of GB of RAM
 - Dozens of CPUs
 - Tens of thousands of IOPS

* This is as far as we've pushed with commodity hardware. We can probably go a lot further.

What's Still Not Great?

- Friends don't let friends use the query cache
 - It is not scalable
 - It is a square peg in a round hole for web apps anyway
- Replication
 - It is single-threaded on the slave
 - A busy master can easily outpace a powerful slave
- Bottlenecks are moving

New in MySQL 5.1

- Partitioning
 - Really just N tables under the hood
 - This changes things in interesting ways
 - Poor server instrumentation: hard to analyze & tune
- Row-based replication
 - Black-box implementation
 - When it is slow, it's hard to find out why
 - Poor server instrumentation: hard to analyze & tune

Summary

- Use InnoDB Plugin
- Consolidate and use more powerful hardware
- Don't just use defaults
- Simplify for better performance and reliability
- Be scientific
- Complain when something isn't instrumented

Thanks!

Slides will be online at percona.com

Questions and comments welcome!

<my name>@percona.com