



PERCONA
Performance Consulting Experts

MySQL Performance Basics

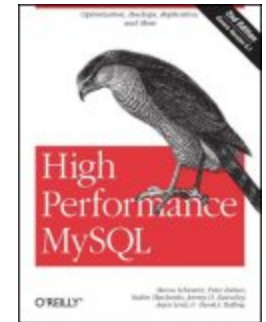
Baron Schwartz

beCamp 2008

May 3, 2008

Who Am I

- Lead Author, High Performance MySQL 2nd Edition
- Senior Consultant at Percona
- Author of Xaprb blog <http://www.xaprb.com/>
- Author of several FOSS tools
- These slides will be at <http://www.percona.com/>



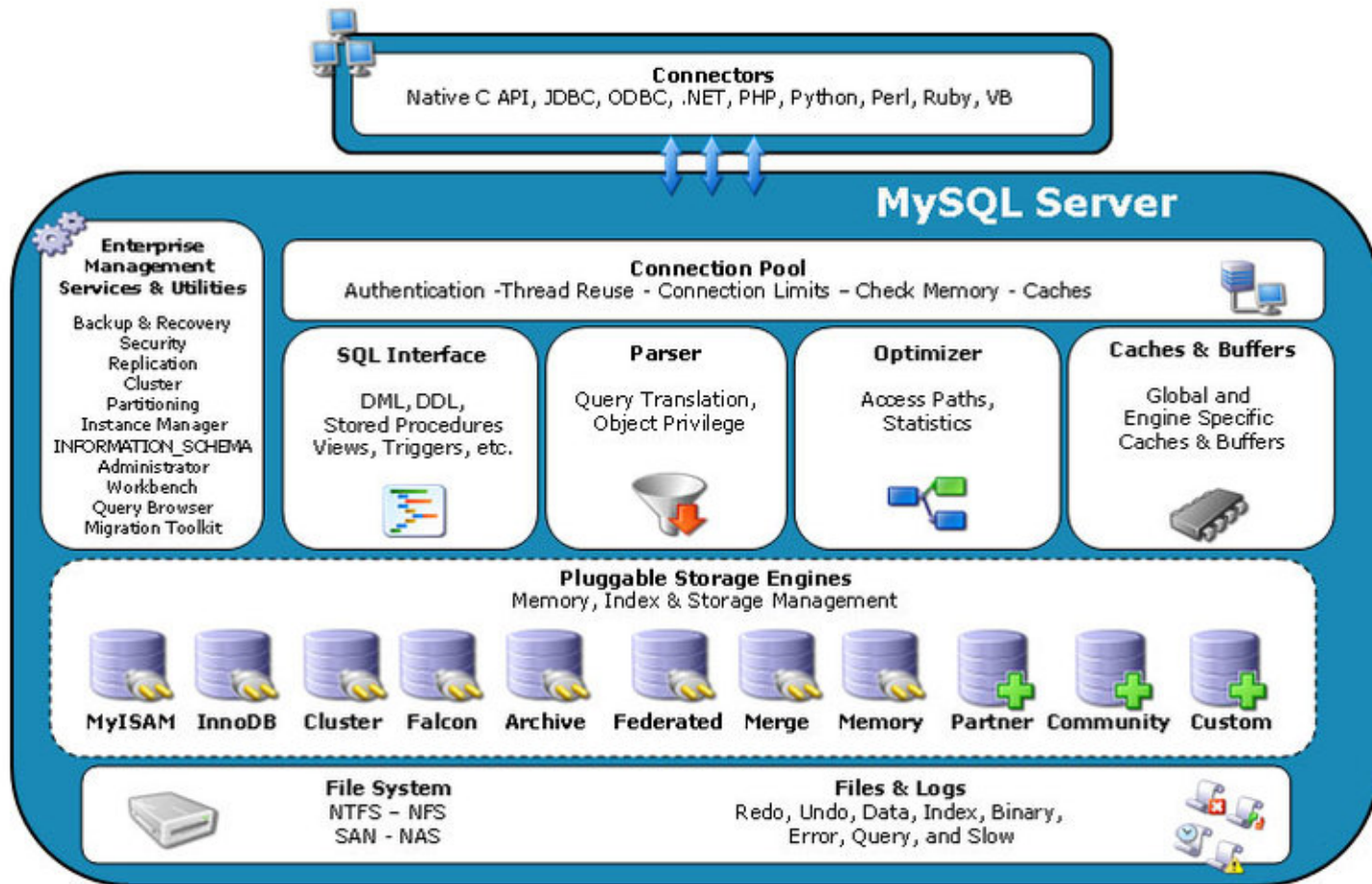
Brief Survey

- Do you or will you use MySQL?
- Do you have or expect performance problems?
- What do you know? What should I talk about?
- Quick thermometer: do you know about
 - indexes?
 - leftmost index prefix rule?

The Basics

- 50,000 Foot View
- Data Storage
- Table Design
- Indexing
- Queries
- Scaling and HA

The Grand Unified View



Data Storage

- Storage engines are really, really important
 - Each storage engine is very different
 - This is the single biggest difference from other DBs
- Storage engines determine:
 - index types
 - transaction capability
 - workload-specific characteristics
- I recommend InnoDB for general real-world usage
 - Other engines have their places too

Storage Engine Overview

- InnoDB
 - Transactional, ACID compliant, fast, proven, stable
 - Clustered primary key! You must design with this in mind
- MyISAM (default storage engine)
 - Fast for certain types of queries
 - Special features (full-text, geospatial, packed indexes)
 - Zero concurrency
- Others
 - Memory, PBXT, Blackhole, MANY more

Table Design

- Try to represent data “the right way”
 - Use the appropriate data type
- Store data as compactly as you can
 - MySQL offers many data types: INT, TINYINT etc
- Learn normal forms
 - Normalize first, then denormalize if necessary
 - Second or third normal form is usually all you need

Indexing

- Indexes help MySQL find rows fast
 - Use them!
- Example:
 - `SELECT ... WHERE order_date = '2008-05-03'`
 - You should have an index on `order_date`.
- Isolate the indexed columns
 - Bad:
 - `to_days(order_date) >= to_days(now()) - 30`
 - Good:
 - `order_date >= now() - interval 30 day`

Queries

- Make your queries access less data
 - Indexes are a great way to reduce data access
- Make your application access less data
 - Don't `SELECT * FROM tbl` unless you need all columns
- Issue as few queries as possible
 - Sometimes a single large query is better than many small
 - Sometimes many small queries are better!
- Learn about JOINS
 - Don't match data in the application code

Queries, Continued

- Every high-performance application sometimes breaks the rules on the last slide
- Sometimes joins in the application are better
- Sometimes fetching data you don't need is better
- Caching and other techniques can turn waste into gain
- But use best practices to start with, and break the rules when you know how

A Query to Avoid

- MySQL has some gotchas
- The most serious is a query of this form:
 - ```
SELECT .. FROM invoices
WHERE order_id [NOT] IN (
 SELECT order_id FROM orders);
```
- Rewrite these queries as JOINS
- There are other gotchas, but this is the one I see most

# Scaling and HA

- This is a basic talk, so let's clarify:
  - Performance: how fast the car is
  - Capacity: the speed limit \* number of lanes
  - Availability: how often a lane or highway is drivable
  - Scalability: the ability to add more cars and more highways without slowing traffic
- The techniques for achieving this are well-known and extensively used
  - You are not the first person to do this

# Vertical Scaling

- Buying bigger, faster hardware
- More CPUs, memory, disk drives
- Does not work very well with MySQL
  - No intra-query parallelization, so single-CPU-bound
  - Critical parts of the architecture are single-threaded
  - Some parts are not happy with high concurrency
- Today's typical hardware that works well
  - 4 cores, 32-64GB RAM, up to 20 disk drives

# A Note on Hardware

- Some generic advice:
  - Use 64-bit, and double-check that you are using 64-bit OS
  - If you can afford it, buy enough RAM to fit your working set
  - For high performance, fast disk drives are important
    - 2.5" 15K RPM SAS drives are nice – you can fit more drives, and the smaller size gives faster access time
  - RAID 10 is the best RAID configuration
    - RAID 5 is cheaper but not as good for write-heavy
  - Make sure the RAID card has a battery-backed write cache

# Horizontal Scaling

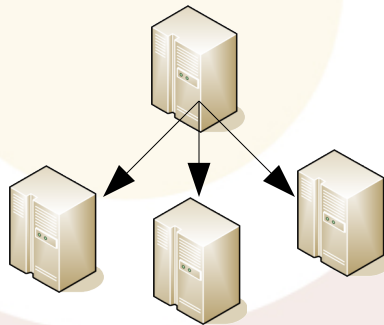
- If your data is larger than a single server, you must partition it
  - Functional partitioning: separate servers for different kinds of data
  - Data sharding: separate servers for separate partitions of data
  - Data archiving: move away unused data (time-based partitioning)
- Each approach makes some things harder
  - Each approach has limits

# What about MySQL Cluster?

- I do not think that word means what you think it means
- MySQL Cluster is not like other types of clusters
  - In-memory, distributed, shared-nothing architecture
  - Performs well for single-table queries by primary key
  - Does not perform well for joins or table scans
- It is not a general-purpose solution
  - It is for telcos, basically, but has other uses too

# Replication

- MySQL has built-in replication
- Very easy to set up
- Generally works very well, but has gotchas
- Asynchronous
- Extremely, very, highly useful



# High Availability + Scalability

- My favorite architecture: master-master replication
  - Two servers that are co-masters and co-slaves
  - But only one is active (writable) at a time
- Useful for all kinds of things
  - Offline maintenance tasks
  - Failover when there's a crash... much more
- Master-master replication is only for HA
  - But if each pair is a shard, you can build scalable architectures with it too

