MySQL Scaling Pains

Jeremy Zawodny
Yahoo! Inc.

O'Reilly Open Source Convention
Portland, Oregon
Friday, July 11th, 2003

Jeremy@Zawodny.com
http://jeremy.zawodny.com/mysql/
About Me

• Long-time MySQL user
  – Using for 5-6 years
• Active on MySQL mailing lists
• Yahoo's resident “MySQL Geek”
  – Part of the platform engineering group
  – MySQL support, planning, tuning, etc.
• Write about LAMP for Linux Magazine
• Random MySQL consulting
• E-Mail: Jeremy@Zawodny.com
MySQL at Yahoo!

- 200-400 servers world-wide
- Mission-critical
- FreeBSD and Linux on commodity hardware
- Replaces home-grown DB systems, sometimes Oracle
- Used both “live” and in batch processing
- Replication and load-balancing to scale
Yahoo! Is Hiring

It's not just Amazon and Ticketmaster. We need good, open source-loving hackers to help build some of the world's most popular web destinations.

We use FreeBSD, C/C++ (gcc), Apache, Perl, PHP, MySQL, vi(m), Emacs, and even some Java here and there.

I'm collecting resumes...
Outline

- What this is About
- The Need to Scale
- Back to Basics
- Query Optimization
- Too Many Cooks
- Security
- Size Limits
- Lock Contention
- ALTER TABLE

- Disk I/O
- Replication
- Many Connections
- Table Partitioning
- Server Partitioning
- Full-Text Search
- Book Plug
What this is about

- MySQL is deceptively simple, but it is not a silver bullet
- This is about problem, pitfalls, gotchas, surprises, and annoyances that result from trying to grow or scale MySQL
- The goals are awareness and understanding
  - You can plan around most problems if you know
- There are not solutions to every problem
The Need to Scale

- Increasing demand (queries/sec)
- Lots of front-end clients
- Growing data size
- Application complexity and features
- More developers/DBAs
- All of the above at the same time!
Step #1: Back to Basics

• Benchmark & Analyze
  - Understand what's slow and why
  - Figure out how expensive queries are

• Optimize
  - Fix what you can fix
  - Design with scale in mind

• Tune
  - Adjust MySQL settings

• Monitor
Query Optimization

• Enable the slow query log
• Use a low long-query-time threshold
• Don't assume that logged queries really are slow
  - Test them, benchmark if necessary
  - Use EXPLAIN to verify
• Test alternative indexes
  - Fewer rows isn't always better
  - Consider disk seeks as well
Too Many Cooks

• You need documentation
  – What databases are used for what
  – Who owns them
  – Conventions (naming, etc.), procedures
  – Comment the my.cnf file
    • Use version control

• Areas of responsibility
  – Who does backups?
  – Who runs upgrades?
Security

• Don't share the root account
  - Force admins to either use personally named accounts
  - Or ssh to the server and then connect locally as root

• Carefully setup and re-evaluate application-level permissions
  - Don't give apps ALL PRIVILEGES ON *.*
Size Limits

• MyISAM tables have the mysterious 4GB limit
  - Set MAX_ROWS attribute in CREATE TABLE
  - Or use ALTER TABLE later

• InnoDB tablespace file sizes are pre-set
  - Autoextend helps but only applies to the last file
  - No control over which tables reside in which files
    • To be fixed later this year
  - Watch the COUNT(*) queries
Lock Contention

- MyISAM tables use table locks
  - Many readers or one writer
- Watch your read/write ratio
  - Many reads, few writes: good
  - Many writes, few reads: good
  - Many of both: bad
  - SHOW STATUS
    - Com_select, Com_insert
    - Com_update, Com_delete
More Lock Contention

- Watch the lock counters
  - SHOW STATUS
    - Table_locks_immediate
    - Table_locks_waited
    - I like to see 100:1 ratio

- Optimize those queries
- Double-check indexes
- Use caching when possible
- InnoDB deadlocks
  - Transaction overuse
Alter Table is Slooooodwww

• Very disk I/O intensive
• MySQL rewrites all indexes every time!
  – Make several index changes at once
  – Make sure your key_buffer is big
  – Same for myisam_sort_buffer
• Think through changes carefully
• Plan ahead
• Alter off-line if possible
• Consider a progressive migration
Disk I/O Issues

- Disks are very often the bottleneck
- If the CPU isn't nearly saturated, check disk I/O stats: vmstat, iostat, sar, etc.
- Low seek times are your friends
- SCSI disks use tagged command queuing
  - Better for multi-threaded random I/O
  - IDE disks don't, but the kernel may try
- RAID-0 or manual I/O balancing help
More Disk I/O

- RAID-10 better on master(s)
- Controller with RAM cache is a plus
- If using IDE disks, check DMA settings
- Benchmark several disk combinations
  - Use MySQL to do it
  - Try Bonnie++
  - Two slower disks may be better than one faster disk
Load Balancing

• First tune your server(s)
• Replication
• Monitoring
  – Track critical performance numbers
  – Watch for trends over time
• Load-balancers
  – Use the right algorithm
  – Develop reasonable health checks
Handling Many Connections

• Understand MySQL's memory usage
  – Global Variables
    • MyISAM key buffer, InnoDB buffer pool
  – Per-thread variables
    • Sort buffer, record buffer, join buffer, etc.

• Number of connections vs. Active connections

• The wait_timeout is your friend

• Connection pooling may be helpful

• Some thread libraries use limits ~1024
Data Partitioning: Tables

- Goal: Maximize cache utilization
- Replication + a hashing or “bucketing” algorithm
- Can split into several “range tables”
- Master gets all updates
- Your might filter updates on the slaves
- Also reduces table lock contention
  - Smaller sized tables
Divide users based on the first letter of username.
Data Partitioning: Servers

• Several independent MySQL instances
• When logical separation isn't enough
• Can tune and scale independently
  – Users on one cluster
  – Messages on another
  – Pictures on a third
• Common on large sites
Full-Text Search

• Upgrade from 3.23 to 4.0.xx
• Full-Text is a real pig
  – Indexes are large
  – Memory and disk use significant
• Adjust stopword list
  – ft_stopword_file
• Adjust min/max word length
  – ft_min_word_len
  – ft_max_word_len
• Don't overuse it
More Full-Text Search

• Full-Text is not very flexible
  – The stop word list is per server
  – The min word length is per server
  – The word boundary rules are per server
    • They're compile-time, not easy to configure

• You may need multiple instances for various projects with different needs

• This sucks, but...

• MySQL 4.1 should fix some of the issues
Questions?

• I'd like to know what problems you've faced and any novel (or not so novel) solutions that worked well.

• What would you like to know?