



# MySQL Performance CookBook

Peter Zaitsev, co-founder, Percona Ltd

April 23-26 2007

Presented by



O'REILLY

# About Speaker

- Peter Zaitsev
- Co-Founder at Percona Ltd,
  - Consulting company specializing in MySQL and LAMP Performance
- Founder <http://www.mysqlperformanceblog.com>
- Spent 4.5 years with MySQL AB,
  - High Performance Group, Manager
    - Performance related Support, Development, Consulting etc
- Have designed and implemented numerous MySQL based projects

Presented by



O'REILLY

# About Presentation

- Selected Problems based on Development and Consulting experience
- Problems frequently met in real world applications
- Something people have trouble with
- Multiple solutions possible for many problems, take it as example adapt and for your needs
- Some may be ugly, you've been warned
- Your alternative solutions proposals are very welcome  
Just raise your hand to comment
- I frequently Blog about such cases as I pop into them

<http://www.mysqlperformanceblog.com>

Presented by



O'REILLY

# Performance Management

Presented by



O'REILLY

# How to kill Run Away Queries ?

- Interactive query is running for 30 seconds already, it is better to tell user it did not work than to have timeout
- Ask Dmitry Lenev to finally push his SQL\_TIMEOUT patch
- Or mark queries  

```
SELECT /*TIMEOUT:30*/ ....
```
- And have script which runs SHOW PROCESS LIST every second and kills them
- In MySQL 5.1 with Event Scheduler should be possible to implement entirely in MySQL

## How to Limit concurrency ?

- You've identified optimal concurrency for your search query is at 10 concurrent queries, how to avoid running more ?
- Assume you have 10 “slots” ,  
Select random of them, for example 5  
Use `SELECT GET_LOCK('search5',1)` to try to lock the slot
- Try the next slot if this one is busy
- If you can't find free slot fast enough/went a full cycle report error to the user

# How to implement real time notifications ?

- You want process to be informed as soon as possible when table is changed and you do not want it to pool all the time

- Updater:

```
SELECT GET_LOCK('queue')
```

.... update data...

```
SELECT RELEASE_LOCK('queue')
```

```
SELECT GET_LOCK('queue')
```

- Reader

```
SELECT GET_LOCK('queue') in the loop
```

Will be informed as soon as updated is done with the round and releases lock.

# How to check which queries load server

- Using slow query log ?
  - IT may be 1000s/sec of 1ms long queries are giving the load
- Pooling processlist ?
  - Very short queries are hard to catch because of OS scheduling
- Use microsecond resolution patch
  - <http://www.mysqlperformanceblog.com/2006/09/06/slow-query-log-a>
- Or use MySQL Proxy for timing queries on client size
  - <http://jan.kneschke.de/projects/mysql/mysql-proxy>
- Aggregate queries with mysqlsla
- Application level query profiling is also good



# How to profile web site for slow pages

- These can be very specific to the user/date
  - user has 50.000 images loaded while there are 10 in average
  - Someone happened to get to page 1000 in search results
- Have dynamic web page views logged with cpu time, wall clock time, sql time stored and analyze it regularly
- So you will know 95% requests are answered within 0.3 sec
- Often you would find some rare requests taking many minutes or hours

# How to avoid running out of connections

- Typical configuration - 5 web nodes
  - Apache with MaxClients=500
  - Single MySQL Server, 2500 connections may be required
- Are these 500 Clients really serving dynamic content
  - no, they are used for keep-alive, spoon feeding, static content most of the time
- Serve static content separately, Get lighttpd or another apache server in front to do keep alive and spoon feeding
  - or use FastCGI
- You rarely will need more than 20-30 of dynamic content generation scripts running at the same time.
- This saves a lot of memory in addition to connections

# Schema Design & Queries

Presented by



O'REILLY

# Design Schema for your Queries

- Or at least think about queries when designing schema
- Purely Object-Relationship based design can give very poor plans
- Learn MySQL Optimizer features and Limits
- Think how your query is executed by MySQL
- Consider Disk IO in particular

Presented by



O'REILLY

## Beware Distinct and Group By

- DISTINCT and GROUP BY often requires a lot of work even if used with limit
  - Some GROUP BY ... LIMIT may be well optimized
- `SELECT SUM(traffic) t, user u FROM users GROUP BY user, ORDER BY cnt DESC LIMIT 10`
- Create and maintain summary table
  - Often periodic-builds are good enough
  - Does not work if you have dynamic where clause
    - Using SphinxSearch can be real help in certain cases
- Cache result separately

## **COUNT(\*) is equally evil**

- `SELECT COUNT(*) FROM profiles WHERE gender='M' and age>18;`
- Often used together with displaying search results
- `SQL_CALC_FOUND_ROWS` barely helps
  - Full result query may contain number of joins
  - It will need to traverse all result set
- Multiple solutions to suite application
- Having count stored in table is most simple one
- Consider “users” and “pictures” user can have `user_pictures` field to avoid counting.

## COUNT(\*) on Steroids

- So COUNT needs to traverse all rows but how to make it faster ?
- Get rid of JOINS which do not affect COUNT(\*)
- Make sure count uses covering index
  - “Using Index” in Where Clause
- Can give 5-10 times performance boost
  - Even more for IO bound workloads
- If that is not enough you can use distributed partitioning
  - Sphinx is good tool for some applications
    - 100.000.000+ rows counted in fraction of a second
- Applies to other aggregate functions as well

## Do you need count(\*) at all

- You may be simply browsing results page by page  
WordPress does it for example
- Can you have estimated count as Google does ?
- If you're just need it to draw "pager" you can do it while fetching data

```
SELECT * FROM ... LIMIT 1000
```

If you get less than 1000 you know the exact number

if you get 1000 you know it is 1000+ which is good enough

Can be used to cache the result set

May simple use query with no joins to fetch ids and get count at the same time



# Counter Table Tricks

- Counter table limits concurrency
  - It will serialize transactions updating same rows if you're using them
- Solution:
  - Keep more than one row for each value
- Instead of: `Ip_stats (ip,sum_traffic,num_packets)`
- `(ip,slot,sum_traffic,num_packets)`
  - With 10 or 100 slots
  - Random Slot is updated by transaction
  - `SELECT SUM(num_packets) where ip='123.45.67.89'`

## Dealing with High limit values

- `SELECT * FROM users ORDER BY last_online DESC limit 100000,10`  
MySQL traverses first 100000 rows and throws them away
- They can be Rare but can be welcoming door for DOS attack
- Search engine bots can go to high page numbers
- If you can't make limit run fast at least restrict it  
Even Google limits you to first 1000 results  
LiveJournal gives you calendar if you browse your entries too far

## High Limits: Just pre-generate offsets

- Very useful for ratings and other static data  

```
SELECT * FROM sites ORDER BY visits DESC LIMIT 100,10
```

change to

```
SELECT * FROM sites_rating WHERE position BETWEEN 101 and 110 ORDER BY position
```
- Expensive to update for dynamic data
- Can be complex to maintain for dynamic where clause
- May give surprising results if data changes a lot

## High Limit: Link to the next page

- You can specify exact position instead of limit
- `SELECT * FROM stories ORDER BY rating DESC LIMIT 10,10`
- `SELECT * FROM stories WHERE rating<=1000 and story_id<400000 ORDER BY rating DESC, story_id DESC`

Assumes story with rating 1000 and story\_id 400000 was last on the first page
- Have to add sorting by story\_id to have sorting fully defined

# Making skipping rows fast

- Similar to COUNT(\*) case
- Hard to get covering index directly because you need a lot of data
- Use Derived Table trick
- `SELECT * FROM users WHERE id IN (SELECT id FROM users WHERE gender='m' ORDER BY rating DESC limit 100,10)`

Covering index would be (gender,rating,id)

## Do not use LIMIT for Batch Jobs

- `SELECT * FROM TASK LIMIT N,100000`  
Even if table does not change  
has to scan and throw away more and more rows  
Average query complexity will be half of table scan.
- `SELECT * FROM TASK WHERE ID BETWEEN N AND N+99999` is much better choice
- Applies to simple cases when you so not want to “mark jobs as done”

# Power of Derived Tables

- Great way to get control over how query is executed
- Can be performed in many cases to get better performance
- For example:

- To help using partially covered Index

```
SELECT * FROM tbl WHERE a=5 AND id IN (SELECT id  
FROM tbl WHERE b LIKE "%test%")
```

Can use index covered index(B,ID)

- Performing Join post Group-By or Lookup Query

- Beware – if you join two derived tables together MySQL have to do full join.

# Dealing With Temporary Tables

- If you store long rows it will consume a lot of memory or go on disk
- If you use BLOB/TEXT fields table must be created on disk
  - Placing such tables on tmpfs is good idea
- In Memory temporary tables make Dynamic length Rows Static

VARCHAR(255) becomes CHAR(255) which can take 750+ bytes with UNICODE
- Try to keep VARCHAR Max Length short
- Make only Ids to go to temporary table and perform required JOIN on the second stage (by use of derived tables)



# Selecting Random Objects

- `SELECT * FROM banners ORDER BY rand() limit 1;`  
Worse way you can do it.
- Check <http://jan.kneschke.de/projects/mysql/order-by-rand> for ideas
- Can generate ID from 1 to max on the application and  
`SELECT * FROM banners WHERE id=<computed random>`  
Can use `<=` `>=` with ORDER BY to deal with holes
- Create 100000 random rows in the single sweep and store them in separate tables
- Most applications do not require perfect random values anyway.

## Sorting – Beware Filesort

- Watch out for “filesort” - external sorting which MySQL has to use
- Does not scale with large amount of objects
- Use proper index if it can do the trick
- `SELECT * FROM TBL WHERE A IN(1,2) ORDER BY B LIMIT 5`

MySQL can't use index for sorting unless you have = for all previous columns

`(SELECT * FROM TBL WHERE A=1 ORDER BY B LIMIT 5`

`UNION SELECT * FROM TBL WHERE A=2 ORDER BY B LIMIT 5)`

`ORDER BY B LIMIT 5`

# Sorting and Join

- If sorting is done by column from other than first table MySQL can't do index based sort

```
SELECT * FROM USES,PICTURE WHERE  
    USER.ID=PICTURE.USER_ID AND USER.NAME="john"  
ORDER BY PICTURE.ADDED DESC LIMIT 10;
```

If name is unique pre-fetching often helps

```
SELECT * FROM PICTURE WHERE USER_ID=1234 ORDER  
    BY ADDED DESC LIMIT 10
```

- Caching columns is other helpful technique

```
SELECT * FROM PICTURE WHERE USER_COUNTRY=123  
ORDER BY ADDED DESC LIMIT 10;
```

# Sorting and Indexing

- MySQL May have hard time selecting index for sorting VS lookup
- `SELECT * FROM T WHERE A=5 AND B=6 ORDER BY C LIMIT 10`
  - Having index on (A,C) and (A,B) MySQL Will likely chose (A,B)
  - LIMIT is nor really well considered while optimizing
  - Have index (A,B,C) or use FORCE INDEX
- Can be tricky to set up indexes for all lookup/sort combinations
- Can use Sphinx for distributed sorting and filtering

# What is about giant sorts and group by ?

- Data analyses

```
SELECT COUNT(*) C FROM LOG GROUP BY PAGE ORDER BY  
C DESC LIMIT 1000;
```

- Make sure GROUP BY is done using sorting if number of pages is large (SQL\_BIG\_RESULT hint)  
and tmp table if it is small.
- Group by can do index scan if sorting by the same column
- **sort\_buffer\_size** set before the query
- **max\_length\_for\_sort\_data** may need to be tuned
- Watch out for sort file size – varchars, blobs.

# “Manual Joins”

- `SELECT * FROM A,B WHERE A.BID=B.ID AND A.C=5`
- Changing to  
`SELECT * FROM A WHERE A.C=5`  
`SELECT * FROM B WHERE B.ID IN (<List of retrieved values>)`
- Sounds silly, but what are the benefits ?
- Can be faster as each row from B is retrieved once  
And because IN list will be sorted for optimal retrieval
- Works if table A,B are located on different servers
- Can use caching to filter out list of retrieved values  
Get B values from memcache with multi-get and only get missed once from the database

# Beware subqueries

- MySQL Subqueries optimization is currently weak
  - Many subqueries working for other databases will run extremely slow with MySQL
- How can you workaroud ?
  - Convert to JOINS
  - Convert to derived table (subselect in FROM)
  - Use explicitly created temporary table (with index)
  - Change to IN list created by application.

# Using temp VIEW instead of Derived Table

- Derived tables are same as inline views  
Right but they happen to use different code bases in MySQL
- `SELECT ... FROM (SELECT .... FROM ...) ...`  
Will always materialize select in temporary table and no indexes
- `VIEWS` can use temporary table and “merge” execution method.
- So you can use view to get merge way of query execution  
`CREATE VIEW A ...`  
`SELECT * FROM A ...`  
`DROP VIEW A`



# Index Merge for Full Text Search

- MySQL Full Text Search index is only index which is used full text search
  - You can't use both FT index and index on category for example
- You can add extra column converting your where clause XCAT25, XUSER12345 for category=25 and user=12345
  - Unique keywords which are not met in the text itself
- Add appropriate keywords in full text search query if searching within given object
  - Boolean search
- Does not make Full Text Search to fly but can help in some cases

# Backup and Recovery

Presented by



O'REILLY

# Getting fast cross storage engine backups

- We want it binary for fast backup and recovery
- LVM or other snapshotting techniques is a great choice  
FLUSH TABLES WITH READLOCK  
<CREATE SNAPSHOT>  
SHOW MASTER STATUS; SHOW SLAVE STATUS;  
UNLOCK TABLES;
- Copy data from snapshot any way you like  
rsync, rdiff, tar via ssh etc
- Make sure you have enough space on the snapshot volume
- On LVM2 you can run Innodb recovery with separate MySQL instance before copying

# Making LVM Backup Warmer

- LVM Backup is not hot backup
  - Same applies to Innodb Hot backup
- FLUSH TABLES WITH READ LOCK may take very long to execute
  - Long queries
  - Many open tables
- Pre-Flush tables by running FLUSH TABLE TBL; for all tables in system (one by one)
- Make sure there are no long running queries when backup takes place

# Point in time recovery via Binary Log

- “Standard way to do point in time recovery”
- `Mysqlbinlog myhost-bin.000005 | mysql`
  - Slow, a lot of conversions
  - Can get in problems with mysqlbinlog bugs or version issues
  - No easy way to filter out the stuff
- Start dump master to read binary logs you want to recover
  - It can have empty database it does not matter
  - Use `CHANGE MASTER TO` to initiate recovery
  - `START SLAVE UNTIL...` can be used to stop where needed
  - `replicate-do-wild-table`** can be used to recover only tables you need

# Time for Questions

- More Performance Tips

<http://www.mysqlperformanceblog.com>

- Helping to find out what tips matter for you

[consulting@mysqlperformanceblog.com](mailto:consulting@mysqlperformanceblog.com)

Presented by



O'REILLY