Top Most Overlooked MySQL Performance Optimizations

Muhammad Irfan
Support Engineer

PERCONA

Our topics today

- WEBSCALE
- BIG DATA
- SHARDING
- READ-WRITE SPLITTING
- NAME YOUR BUZZWORD

Our topics today

- ~~WEBSCALE~~
- ~~BIG DATA~~
- ~~SHARDING~~
- ~~READ-WRITE SPLITTING~~
- ~~NAME YOUR BUZZWORD~~

Our topics today

- ~~WEBSCALE~~
- ~~BIG DATA~~
- ~~SHARDING~~
- ~~READ-WRITE SPLITTING~~
- ~~NAME YOUR BUZZWORD~~

Sorry, buzzwords are not included in this talk.

We’ll discuss boring stuff, but we hope it’ll be informative!

Agenda

- Choosing hardware
- Configuration
- Indexing
- Queries
- Table design
- Architecture

What matters - 1

Memory

- Try to have the working set in memory

Disks

- Throughput or latency?
- For most apps, latency is the major factor
- So beware of SANs and prefer SSDs!!

What matters - 2

CPU

- Faster is better
- Number of cores should match concurrency (Threads_running)

Network

- Usually not an issue
- Except if you’re using Galera or lots of BLOBs

Agenda

- Choosing hardware
- Configuration
- Indexing
- Queries
- Table design
- Architecture

What to do with configuration?

- More than 400 settings
  - Useless to look at all of them
- Many default settings are pretty good
- A few of them need to be changed though

What is an optimal configuration?

- The best configuration file doesn’t exist
- Look for a configuration where the server
  - is stable
  - has good performance
  - only a good balance between performance and data safety

InnoDB - 1

innodb_buffer_pool_size: critical for good general performance

- Rule of thumb: as large as you can

innodb_log_file_size: critical for good write performance

- Rule of thumb: monitor of much is written to the redo logs during 1 hour at peak time

InnoDB - 2

innodb_flush_log_at_trx_commit: D in ACID

- Rule of thumb: 1 for a master (data safety), 2 for slaves (performance)

innodb_file_per_table: allows you to reclaim space if you truncate/drop a table

- Set it to 1 except if many tables (10k+)

Replication

- *log_bin*: enables binary logging
  - Set it even if you only have 1 server for PITR
- *binlog_format*: in general ROW is better than STATEMENT (performance and reliability)
- *expire_logs_days*: MySQL will remove old binlog files after N days. Avoid running out of space :)

Other settings

Query cache: in general disable it

- But if the workload is read-mostly and concurrency is low, may be very beneficial

MyISAM settings

- Do you really need MyISAM?

Agenda

- Choosing hardware
- Configuration
- Indexing
- Queries
- Table design
- Architecture

Duplicate Indexes

- Indexes add costs to writes.
- Duplicate indexes are bad specially in write intensive workload.
- Duplicate indexes can hurt performance.
- Requires more disk storage.

Duplicate Indexes

How to identify duplicate indexes ?

pt-duplicate-key-checker from Percona toolkit comes to rescue to find duplicate indexes.

```
mysql> SHOW CREATE TABLE test\G
[...]
  PRIMARY KEY (`ID`),
  KEY `UID_IDX` (`UID`),
  KEY `name_idx` (`NAME`),
  KEY `name_phone_idx` (`NAME`,`PHONE`)
```

PERCONA

Duplicate Indexes

```
[root@centos ~]# pt-duplicate-key-checker --database=test
# ########################################################################
# test.test
# ########################################################################
# name_idx is a left-prefix of name_phone_idx
# Key definitions:
#   KEY `name_idx` (`NAME`),
#   KEY `name_phone_idx` (`NAME`,`PHONE`)
# Column types:
#     `name` varchar(20) default null
#     `phone` int(10) unsigned default '0'
# To remove this duplicate index, execute:
ALTER TABLE `test`.`test` DROP INDEX `name_idx`;
# ########################################################################
# Summary of indexes
# ########################################################################
```

PERCONA

Duplicate Indexes

- Redundant index on Name column
- To take benefit of composite index leftmost column is enough. name_phone_idx is suffice

https://www.percona.com/blog/2012/06/20/find-and-remove-duplicate-indexes/

Duplicate Indexes

- MySQL 5.6 produces warnings on duplicate key
- MySQL 5.7 produces error on duplicate key in strict mode
- Does it mean pt-duplicate-key-checker is useless in MySQL 5.6/5.7 ? **NO**
- MySQL 5.6/5.7 produces warning/error on duplicate index but not on redundant indexes

Duplicate

```
mysql> ALTER TABLE test ADD INDEX name_idx (NAME);
Query OK, 0 rows affected, 0 warning (0.77 sec)
Records: 0  Duplicates: 0  Warnings: 0


mysql> ALTER TABLE test ADD INDEX name_idx2 (NAME); -- Duplicate Index

Query OK, 0 rows affected, 1 warning (0.77 sec)

Records: 0  Duplicates: 0  Warnings: 1

mysql> SHOW WARNINGS;

+-------+------
+---------------------------------------------------------------------------------------------------
----------+

| Level | Code | Message
      |

+-------+------
+---------------------------------------------------------------------------------------------------
----------+

| Note  | 1831 | Duplicate index 'name_idx2' defined on the table 'test.test'. This is deprecated and will be
disallowed in a future release. |

+-------+------
+---------------------------------------------------------------------------------------------------
----------+
```

PERCONA

Redundant Indexes

```
mysql> ALTER TABLE test ADD INDEX name_uid_idx (NAME,UID);

Query OK, 0 rows affected (1.11 sec)

Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE test ADD INDEX name_idx (NAME); -- Redundant Index
Query OK, 0 rows affected, 0 warning (0.77 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

- No warnings/errors on redundant key
- pt-duplicate-key-checker identify redundant index

https://www.percona.com/blog/2013/05/31/the-small-improvements-of-mysql-5-6-duplicate-index-detection/

Identify Missing Indexes

- Log queries with --log-queries-not-using-indexes
- EXPLAIN queries and pay attention to Key column.

```
mysql> EXPLAIN SELECT * FROM City\G
*************************** 1. row ***************************
             id: 1
  select_type: SIMPLE
        table: City
        type: ALL
possible_keys: NULL
        key: NULL
        key_len: NULL
        ref: NULL
        rows: 4188
        Extra: NULL
```

Identify Missing Indexes

- Percona Server supports log_slow_filter to log queries only matching filter e.g. full_scan, full_join
- full_scan: The query performed a full table scan.
- full_join: The query performed a full join (a join without indexes).

https://www.percona.com/doc/percona-server/5.5/diagnostics slow_extended_55.html

Agenda

- Choosing hardware
- Configuration
- Indexing
- Queries
- Table design
- Architecture

PERCONA

Identify

- Enable slow query log with long_query_time=0 to log all queries.
- Enable slow query log for specific time period
- Enable slow query log on peak traffic time.
- Use pt-query-digest tool on slow query log to get aggregate report.

Identify Worst Queries

- Save Previous settings.

```
mysql> SELECT @@global.log_slow_verbosity INTO @__log_slow_verbosity;
mysql> SELECT @@global.long_query_time INTO @__long_query_time;
mysql> SELECT @@global.slow_query_log INTO @__slow_query_log;
mysql> SELECT @@global.log_slow_slave_statements INTO @__log_slow_slave_statements;
```

- Keep this one for pt-query-digest

```
mysql> SELECT NOW() AS "Time Since";
```

- Set values to enable query collection

```
mysql> SET GLOBAL slow_query_log_use_global_control='log_slow_verbosity,long_query_time';
mysql> SET GLOBAL log_slow_verbosity='full';
mysql> SET GLOBAL slow_query_log=1;
mysql> SET GLOBAL long_query_time=0;
mysql> SET GLOBAL log_slow_slave_statements=1;
```

PERCONA

Identify Worst Queries

- Verify settings are OK

```
mysql> SELECT @@global.log_slow_verbosity, @@global.long_query_time, @@global.slow_query_log, @@global.
log_slow_slave_statements;
```

- wait for 30 - 60 minutes, keep this one too for pt-query-digest.

```
mysql> SELECT NOW() AS "Time Until";
```

- Revert to previous values

```
mysql> SET GLOBAL log_slow_verbosity=@__log_slow_verbosity;
mysql> SET GLOBAL slow_query_log=@__slow_query_log;
mysql> SET GLOBAL long_query_time=@__long_query_time;
mysql> SET GLOBAL log_slow_filter=@__log_slow_slave_statements;
```

- Verify settings are back to previous values

```
mysql> mysql> SELECT @@global.long_query_time, @@global.slow_query_log, @@global.log_slow_verbosity, @@global.
slow_query_log_file;
```

Identify Worst Queries

- pt-query-digest (replace values for time-since, time-until and log name)

```
$ pt-query-digest --since='<time-since>' --until='<time-until>' --limit=100% /path/to/slow_query_log_file.log >
/path/to/report.out
```

- Queries with highest no. of Calls and Response time per call are top candidates for optimization.

```
# Profile
# Rank Query ID           Response time   Calls  R/Call Apdx V/M   Item
# === ================== ============== ====== ====== ==== ===== =======
#   1 0x0DAFCB462BC6D560  5274.000 20.0% 3527   7.8492 0.00 66... SELECT UNION table1 table2
#   2 0x0557AA7C284F2249 32018.000 13.9% 4513   6.6335 0.21 3.95  SELECT table1 table2
#   3 0x28A9C50583F9A8D4 22174.000 9.6%  9141   1.7309 0.05 28... SELECT table3
#   4 0x5BEABDEDDD53395E 12150.000 5.0%  1574   1.3245 0.23 16.59 UPDATE table4
```

- Query with Rank 1 and 2 should be optimized first.

https://www.percona.com/blog/2014/03/14/mysql-slow-query-log-tools-and-tips/

Agenda

- Choosing hardware
- Configuration
- Indexing
- Queries
- Table design
- Architecture

Primary keys (InnoDB)

- The PK is a clustered index
  - Means all data “lives” in the primary key
- PK is appended to each secondary keys
- Consequences:
  - PK lookups are faster than SK lookups
  - PK should be short

Indexes - 1

- Only 1 index per table per query can be used
  - One exception: index_merge algorithm
- Leftmost prefixes of composite indexes can be used
  - Not rightmost prefix
- Consequence: prefer fewer multi-column indexes over many single-column indexes

Indexes - 2

- Beware of indexes on varchar columns
  - Maximum size is always used
- Example: indexed varchar(255) utf8
  - Each index record takes 3x255=765 bytes...

Foreign keys

- Use them at your own discretion
- Pros: referential integrity, relations between tables are human-readable
- Cons: adds hidden overhead, potential performance problems are hard to debug

Agenda

- Choosing hardware
- Configuration
- Indexing
- Queries
- Table design
- Architecture

PERCONA

Read scalability

- Read traffic can be sent to replicas
- Prerequisite: app is able to split reads and writes
- But remember that replication is asynchronous
- Usually you want to route critical reads to the master, non-critical reads to slaves

Write scalability

- MySQL replication can’t scale writes
  - All writes must go to the master first
- If replicas are not able to handle the write workload (but the master is)
  - Try multi-threaded replication in 5.6/5.7
- If the master can’t handle the write workload
  - Sharding will become mandatory

High availability

	Semi-automated failover	Automated failover	Notes
Basic master-slaves	No	No	Master-master repl. is sometimes used, quite dangerous though
Master-slaves with MHA	Yes	Yes	MHA: set of Perl scripts, external to MySQL
Master-slaves with GTID and MySQL Utilities	Yes	Yes	MySQL 5.6+ only
Galera-based options	Yes	Yes	Doesn’t use MySQL replication

Join us at Percona Live Europe

When: October 3-5, 2016

Where: Amsterdam, Netherlands

The Percona Live Open Source Database Conference is a great event for users of any level using open source database technologies.

- Get briefed on the hottest topics
- Learn about building and maintaining high-performing deployments
- Listen to technical experts and top industry leaders

Get the early bird rate now extended till Aug 8th! **Register now**

https://www.percona.com/live/plam16/registration

Sponsorship opportunities available as well here.

Q&A

Time for questions

PERCONA

Thank you

PERCONA