MySQL Performance
for DevOps
Sveta Smirnova
MySQL Support engineer
Author of
- MySQL Troubleshooting
- JSON UDF functions
- FILTER clause for MySQL
Speaker
- Percona Live, OOW, Fosdem, DevConf, HighLoad...
Table of Contents

• Introduction
• Hardware
• MySQL Configuration
  Important Options
  How MySQL Uses CPU
  Important Options Continued
• Query Tuning
  Indexes
  Optimizer Configuration
Introduction
What is MySQL?

- Database server
- 25 years of history
- Popular forks
  - Percona Server for MySQL
  - MariaDB Server
- Replication support from the beginning
MySQL Architecture

- mysqld
- Connectors
- Optimizer
- Caches
- Storage Engines
- Management

Connectors: C, JDBC, ODBC, Python, ...

Connection Pool: Authentication, Caches

SQL interface

Parser

Optimizer

Caches and Buffers: Global Engine-specific

Storage engines: InnoDB, MyRocks, ...

File system: Data, Index, logs, other files
What Affects Performance?

- Query
- Hardware
- Configuration
Hardware
• No swapping
  • sysctl vm.swappiness=1
- No swapping
  - sysctl vm.swappiness=1
- NUMA interleave
  - Enable in BIOS

Memory Configuration
• No swapping
  • `sysctl vm.swappiness=1`
• NUMA interleave
  • Enable in BIOS
• More is better
- No swapping
  - `sysctl vm.swappiness=1`
- NUMA interleave
  - Enable in BIOS
- More is better
- Memory access is faster than disk
Memory Configuration

- No swapping
  - `sysctl vm.swappiness=1`
- NUMA interleave
  - Enable in BIOS
- More is better
- Memory access is faster than disk
- Frequently accessed data should be in memory
Faster is better
- SSD
- NVMe
- Spinning disk

Disk Configuration
Faster is better
- SSD
- NVMe
- Spinning disk

Parallel writes
Faster is better
- SSD
- NVMe
- Spinning disk

Parallel writes
Battery-backed cache
IO scheduler
  • [noop] or [deadline]
  • sudo echo noop > /sys/block/DISK/queue/scheduler
  or sudo echo deadline > /sys/block/DISK/queue/scheduler
CPU Configuration

- **IO scheduler**
  - `[noop]` or `[deadline]`
  - `sudo echo noop > /sys/block/DISK/queue/scheduler`
    or `sudo echo deadline > /sys/block/DISK/queue/scheduler`

- **CPU governor**
  - Set to `performance`
- **IO scheduler**
  - [noop] or [deadline]
  - `sudo echo noop > /sys/block/DISK/queue/scheduler`
  - or `sudo echo deadline > /sys/block/DISK/queue/scheduler`

- **CPU governor**
  - Set to `performance`

- More cores is better
As fast as possible
- Speed of the line
  - RTT
- Bandwidth
- Stability
  - To avoid TCP packet re-submission
As fast as possible
On the Internet connection
• Clients can work
• Asynchronous replica will delay
• Synchronous clusters will be not functional
  ▪ Node disconnects with default options
  ▪ Very slow response times with adjusted configuration
MySQL Configuration
System Variables:

How to Change

- \texttt{SET [GLOBAL] \textit{var} = NEW\_VALUE}
System Variables:

How to Change

- SET [GLOBAL] var = NEW_VALUE
- Command-line option
  - --var=new_value
System Variables:

How to Change

- SET [GLOBAL] var = NEW_VALUE
- Command-line option
  - --var=new_value
- Configuration file
  - In the default location
  - Specified by option --defaults-file
    [mysqld]
    var=new_value
MySQL Configuration

Important Options
- `innodb_buffer_pool_size`
  - Ideally should hold active data set
• `innodb_buffer_pool_size`
• `innodb_log_file_size`
  • Should hold changes for an hour
- **innodb_buffer_pool_size**
- **innodb_log_file_size**
  - Should hold changes for an hour
  - Too low
- `innodb_buffer_pool_size`
- `innodb_log_file_size`
  - Should hold changes for an hour
  - Good
- **innodb_buffer_pool_size**
- **innodb_log_file_size**
- **innodb_io_capacity**
  - Default is too small for fast disks
  - Up to number of IOPS your disk can handle
  - **Do not set too high!**
• `innodb_buffer_pool_size`
• `innodb_log_file_size`
• `innodb_io_capacity`
• `innodb_flush_method`
  • In most cases: 0DIRECT
  • **Test on your filesystem!**
InnoDB

- `innodb_buffer_pool_size`
- `innodb_log_file_size`
- `innodb_io_capacity`
- `innodb_flush_method`
- `innodb_thread_concurrency`
  - 0 or number of CPU cores
MySQL Configuration

How MySQL Uses CPU
One thread per connection
- CPU used only for active threads
How MySQL Uses CPU

- One thread per connection
  - CPU used only for active threads
- Background work by storage engines
Connection and Engine Threads
What Happens with Threads

? ≤ CPU cores?
What Happens with Threads

? \( \leq \) CPU cores?
Yes Executed simultaneously
What Happens with Threads

? \(\leq\) CPU cores?
Yes Executed simultaneously
No Wait in a queue
What Happens with Threads

? ≤ CPU cores?
Yes Executed simultaneously
No Wait in a queue

? Does the disk support parallel write?
What Happens with Threads

- ? <= CPU cores?
  - Yes  Executed simultaneously
  - No   Wait in a queue
- ? Does the disk support parallel write?
  - Yes  Write happens
? \leq\ CPU cores?
Yes  Executed simultaneously
No   Wait in a queue

? Does the disk support parallel write?
Yes  Write happens
No   Wait in a queue
MySQL Configuration
Important Options Continued
Changing these compromise durability!
Changing these compromise durability!

- `innodb_flush_log_at_trx_commit`
  1: full ACID, default

Once per second not guaranteed for 0 and 2

- DDL can cause faster flushing
- Scheduling may delay flushing

- `sync_binlog`
  0: Synchronization handled by the system
  1: At each transaction commit, default
  N: After N binary log group commits

In case of power or OS crash not flushed transactions can be lost
-changing these compromise durability!

- **innodb_flush_log_at_trx_commit**
  1. full ACID, **default**
  2. logs written at each commit, flushed per second

- MySQL can handle up to **1M** INSERTs per second
- Safe with PXC, Galera and InnoDB Clusters
Changing these compromise durability!

- `innodb_flush_log_at_trx_commit`
  - 1: full ACID, `default`
  - 2: logs written at each commit, flushed per second
  - 0: logs are written and flushed once per second

- Once per second not guaranteed for 0 and 2
- DDL can cause faster flushing
- Scheduling may delay flushing

Synchronization

©2020 Percona
Changing these compromise durability!

- `innodb_flush_log_at_trx_commit`
  1: full ACID, default
  2: logs written at each commit, flushed per second
  0: logs are written and flushed once per second

Once per second not guaranteed for 0 and 2
- DDL can cause faster flushing
- Scheduling may delay flushing
• Changing these compromise durability!

• `innodb_flush_log_at_trx_commit`

• `sync_binlog`
  0: Synchronization handled by the system
  1: At each transaction commit, *default*
    - No transaction lost
  N: After N binary log group commits
    - In case of power or OS crash not flushed transactions can be lost
Table Handlers

- `table_open_cache`
  - The number of open tables for all threads
  - Increase when
    - Connections in the `PROCESSLIST` are waiting for opening a table
    - Value of global status variable `Opened_tables` is larger than `Open_tables`
Table Handlers

- `table_open_cache`
- `table_definition_cache`
  - Size of the cache for table definitions
  - Increase when the Value of `Opened_table_definitions` is larger than `Open_table_definitions`

Increase OS open files limit if needed.
Table Handlers

- `table_open_cache`
- `table_definition_cache`
- Increase OS open files limit if needed
Query Tuning
You communicate with database using queries

- Even via NoSQL interface
- They are not SQL queries, but still queries
You communicate with database using queries
  - Even via NoSQL interface
  - They are not SQL queries, but still queries

Data, that you request, matters
  - 1,000,000,000 rows vs 1 row
Query execution workflow

1. Query sent
2. Connection Pool: Authentication, Caches; SQL interface; Parser
3. Optimizer
4. Storage engines
5. Hardware
Query Tuning
Indexes
MySQL Indexes
- B-Tree Mostly
- LSM Tree
- Fractal Tree
- R-Tree Spatial
- Hash Memory SE
- Engine's
- WHERE the_column = a_value
- WHERE the_column IN(value1, value2, value3)
- WHERE the_column LIKE 'value%'
- WHERE the_column LIKE '%value'
When MySQL Uses Indexes:

- WHERE left_part = value1 AND right_part = value2
- WHERE left_part = value1 OR right_part = value2
- WHERE right_part = value1 AND left_part = value2
- WHERE right_part = value1 OR left_part = value2
When MySQL Uses Indexes:

Joins

- `table1 JOIN table2 ON table1.column1 = table2.column2`
When MySQL Uses Indexes:

Joins

- `table1 JOIN table2 ON table1.column1 = table2.column2`
- Same as `FROM table1, table2 WHERE table1.column1 = table2.column2`
GROUP BY the_column
GROUP BY left_part, right_part
GROUP BY right_part, left_part
GROUP BY the_index, another_index
**When MySQL Uses Indexes:**

- ORDER BY the_column
- ORDER BY left_part, right_part
- ORDER BY right_part, left_part
- ORDER BY the_index, another_index
When MySQL Uses Indexes:

ORDER BY

5.7 `ORDER BY left_part DESC, right_part ASC`

8.0 `ORDER BY left_part DESC, right_part ASC`

- `left_part` must be descending
- `right_part` must be ascending
- `the_index(left_part DESC, right_part ASC)`
Deterministic, **built-in**
- Return same value for the same argument
- WHERE the_column = FLOOR(123.45)

Non-deterministic
- Return different values for different calls
- WHERE the_column = RAND() * 100

**Indexes are not used**

MySQL: Use indexes on generated columns
MariaDB: Use indexes on generated columns

When MySQL Uses Indexes:
Expressions
Deterministic, **built-in**
- Return same value for the same argument
- \[ \text{WHERE the\_column} = \text{FLOOR}(123.45) \]

Non-deterministic
- Return different values for different calls
- \[ \text{WHERE the\_column} = \text{RAND()} \times 100 \]
Deterministic, **built-in**
- Return same value for the same argument
- \[ \text{WHERE the\_column} = \text{FLOOR}(123.45) \]

Non-deterministic
- Return different values for different calls
- \[ \text{WHERE the\_column} = \text{RAND()} \ast 100 \]

Stored functions and UDFs
- Indexes are not used

MySQL: Use indexes on generated columns
MariaDB: Use indexes on generated columns
Query Tuning
Optimizer Configuration
Temporary tables

- `tmp_table_size`
- `max_heap_table_size`
- `default_tmp_storage_engine`

Increase Size of Optimizer Temporary Objects
- Temporary tables
- Buffers for query execution
  - join_buffer_size
  - JOIN conditions, not using indexes
Increase Size of Optimizer Temporary Objects

- Temporary tables
- Buffers for query execution
  - `join_buffer_size`
  - `read_buffer_size`
    - Caching indexes for ORDER BY
    - Bulk insert into partitions
    - Caching result of nesting queries
Increase Size of Optimizer Temporary Objects

- Temporary tables
- Buffers for query execution
  - join_buffer_size
  - read_buffer_size
  - read_rnd_buffer_size
    - Multi-Range Read optimization
• Temporary tables
• Buffers for query execution
  • join_buffer_size
  • read_buffer_size
  • read_rnd_buffer_size
  • select_into_buffer_size
    - SELECT INTO OUTFILE
    - SELECT INTO DUMPFILE

Increase Size of Optimizer Temporary Objects

©2020 Percona
Increase Size of Optimizer Temporary Objects

- Temporary tables
- Buffers for query execution
  - `join_buffer_size`
  - `read_buffer_size`
  - `read_rnd_buffer_size`
  - `select_into_buffer_size`
  - `sort_buffer_size`
    - `ORDER BY`
    - `GROUP BY`
Increase Size of Optimizer Temporary Objects

- Temporary tables
- Buffers for query execution
  - `join_buffer_size`
  - `read_buffer_size`
  - `read_rnd_buffer_size`
  - `select_into_buffer_size`
  - `sort_buffer_size`
  - **Change only at the session level!**
Hardware

RAM: more is better
Disk: SSD or NVMe
CPU: more cores, better concurrency
Net: highest speed possible
Hardware

Configuration

● InnoDB
  - innodb_buffer_pool_size
  - innodb_log_file_size
  - innodb_thread_concurrency
  - innodb_io_capacity
  - innodb_flush_method
  - innodb_flush_log_at_trx_commit

● Server
  - sync_binlog
  - table_open_cache
  - table_definition_cache

Conclusion
Hardware
Configuration
Query Performance
- Add indexes
- Adjust Optimization buffers
  - tmp_table_size
  - join_buffer_size
  - read_buffer_size
  - read_rnd_buffer_size
  - select_into_buffer_size
  - sort_buffer_size
Troubleshooting hardware resources
Troubleshooting configuration issues
MySQL Query Tuning for DevOps
Percona Monitoring and Management
Percona Kubernetes Operators
Thank you!

www.slideshare.net/SvetaSmirnova
twitter.com/svetsmirnova
github.com/svetasmirnova