MySQL Performance

for DevOps

PERCONA

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

PERCONA

Table of Contents

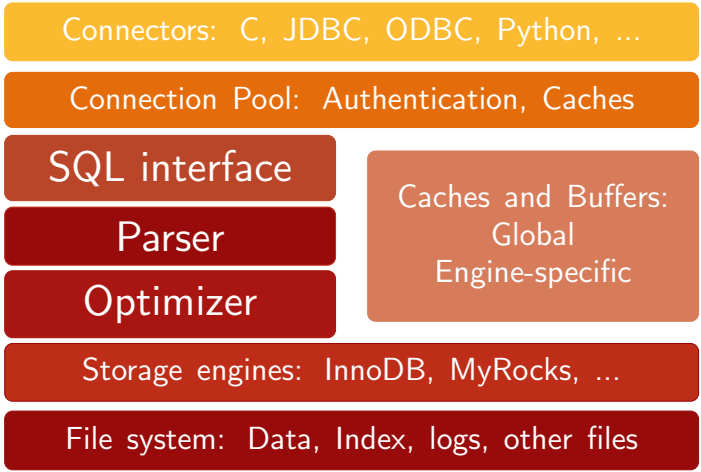
Introduction

What is MySQL?

- Database server
- More than 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



PERCONA

Where MySQL Installed in 2021?

- Bare hardware
- Cloud
- Container-orchestration systems

MySQL Kubernetes Operators

Percona K8 Operator for PXC

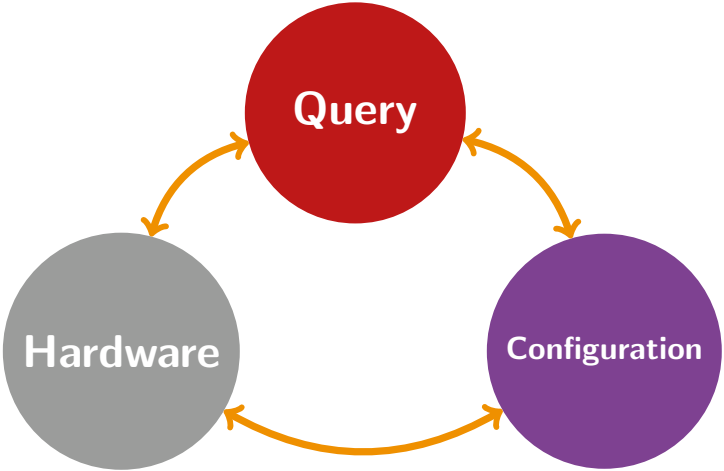
MySQL Operator

PERCONA

Is MySQL Simple?

- 620+ configuration variables
- Multiple storage engines
- Replication
  - Asynchronous
  - Synchronous
- Modern SQL support
- Built-in diagnostic
  - SQL standard
  - Specific for MySQL

What Affects Performance?



Introduction

Scalability

Write Scalability

- Not available
  - ~~Asynchronous Replication~~
    - ~~Single Source~~
    - ~~Multiple Source~~
  - ~~Synchronous Replication~~
    - ~~Galera Cluster~~
    - ~~PXC~~
    - ~~Group Replication~~
    - ~~InnoDB Cluster~~

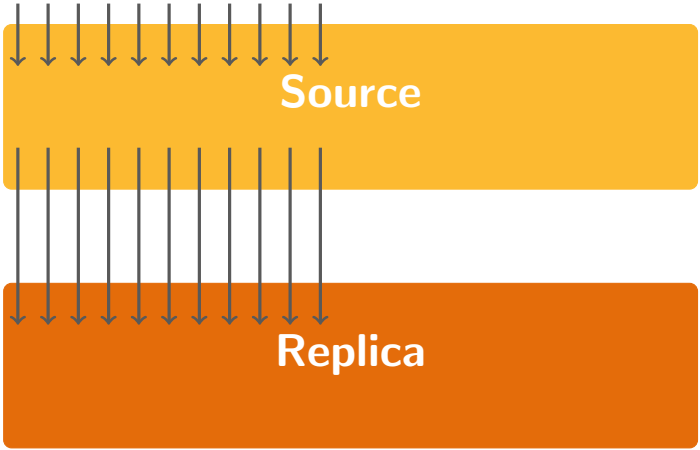
NDB Cluster*

- **Independent product!**

PERCONA

Write Scalability Does Not Exists:

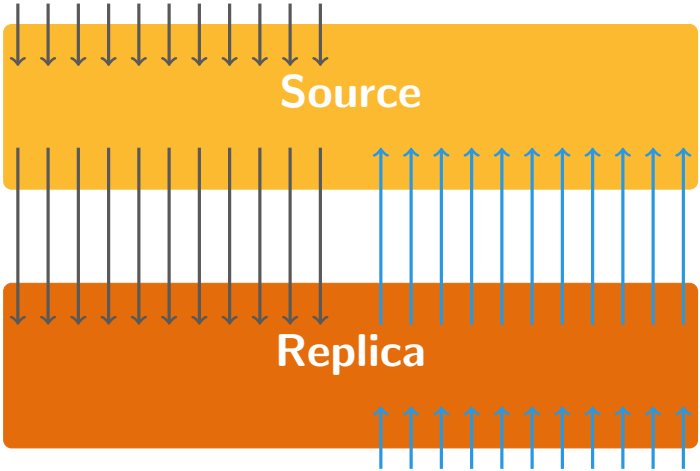
Asynchronous Replication Single Source



PERCONA

Write Scalability Does Not Exists:

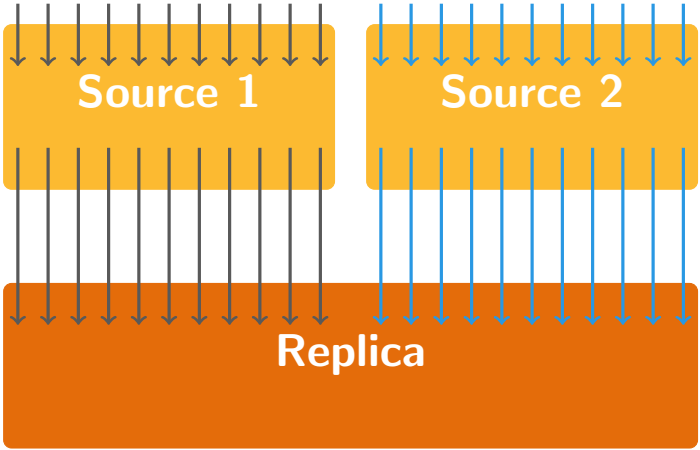
Asynchronous Replication Single Source



PERCONA

Write Scalability Does Not Exists:

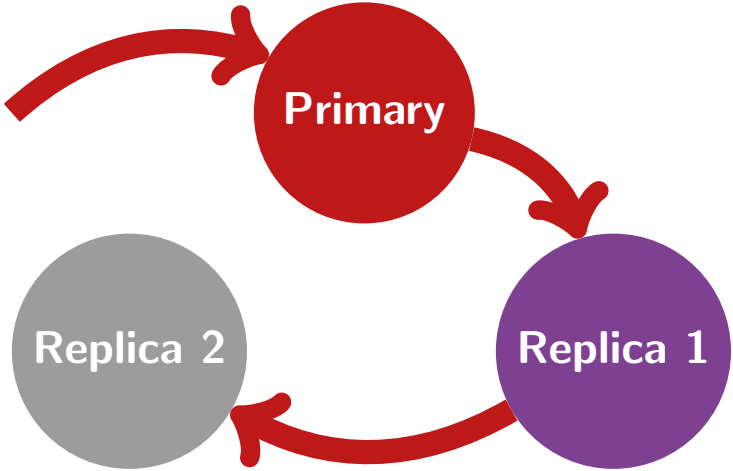
Asynchronous Replication Multiple Source



PERCONA

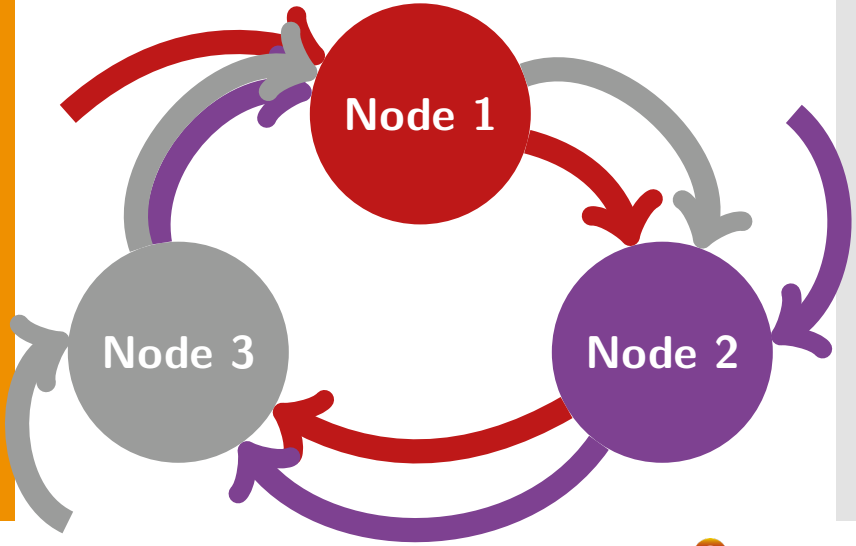
Write Scalability Does Not Exists:

Synchronous Replication Single Primary



Write Scalability Does Not Exists:

Synchronous Replication
Multiple Primary



PERCONA

Read Scalability:

Asynchronous Replication

- Updates from the source could be delayed
  - Not possible to predict delay
  - **Not real time!**

Read Scalability:

Asynchronous Replication

- Updates from the source could be delayed
  - Not possible to predict delay
  - **Not real time!**
- Reads may affect replication performance
  - Updates **would** be delayed

Read Scalability:

Synchronous Replication

- Higher resource usage may delay replicated transactions
- As a result all writes on the cluster could be stopped

PXC Flow control
GR Throttling

PERCONA

Hardware

How to Tune?

- Bare hardware
  - Operating system options

PERCONA

How to Tune?

- Bare hardware
  - Operating system options
- Clouds
  - Administrator console

How to Tune?

- Bare hardware
  - Operating system options
- Clouds
  - Administrator console
- Percona Kubernetes Operator for PXC

  cr.yaml

  - `pxc.resources.requests|limits.memory`
  - `pxc.resources.requests|limits.cpu`
  - `pxc.resources.requests.ephemeral-storage`
  - `pxc.resources.limits.ephemeral-storage`
  - `pxc.volumeSpec.resources.requests.storage`

What to Tune?

- Memory
- Disk
- CPU
- Network

Memory Configuration

- No swapping
  - `sysctl vm.swappiness=1`

PERCONA

Memory Configuration

- 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

Memory Configuration

- 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**

Memory vs Disk Access

Why no Swap?

- MySQL has memory buffers for frequently accessed data

Why no Swap?

- MySQL has memory buffers for frequently accessed data
- OS does not necessarily know which data is frequently accessed
- OS may use algorithms that do not fit for MySQL performance

Why no Swap?

- MySQL has memory buffers for frequently accessed data
- OS does not necessarily know which data is frequently accessed
- OS may use algorithms that do not fit for MySQL performance
- **Swap should never be used with MySQL!**

How MySQL Uses Memory

- Global buffers
  - Allocated up to the limit and freed at shutdown
    - InnoDB Buffer Pool Size
    - Performance Schema tables

How MySQL Uses Memory

- Global buffers
- Session buffers
  - Allocated when session starts
  - Freed at the disconnect
    - Connection and result buffers: from `net_buffer_length` up to `max_allowed_packet`

How MySQL Uses Memory

- Global buffers
- Session buffers
- Operation-specific buffers
  - Allocated for the operation life time
  - Can be allocated multiple times
    - `join_buffer_size`: for each tables pair in `JOIN`
    - `tmp_table_size`: for each temporary table

Disk Configuration

- Faster is better
  - SSD
  - NVMe
  - ~~Spinning disk~~

PERCONA

Disk Configuration

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

Disk Configuration

- Faster is better
  - SSD
  - NVMe
  - ~~Spinning disk~~
- Parallel writes
- Battery-backed cache

How MySQL Uses Disk

- Tables data

PERCONA

How MySQL Uses Disk

- Tables data
- Log files
  - Binary
  - Storage engine
    - InnoDB redo log file
  - Error, general, audit, ...

How MySQL Uses Disk

- Tables data
- Log files
  - Binary
  - Storage engine
    - InnoDB redo log file
  - Error, general, audit, ...
- Disk-based temporary tables

How MySQL Uses Disk

- Tables data
- Log files
  - Binary
  - Storage engine
    - InnoDB redo log file
  - Error, general, audit, ...
- Disk-based temporary tables
- **You may put these on different disks**

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 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`

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`
- More cores is better

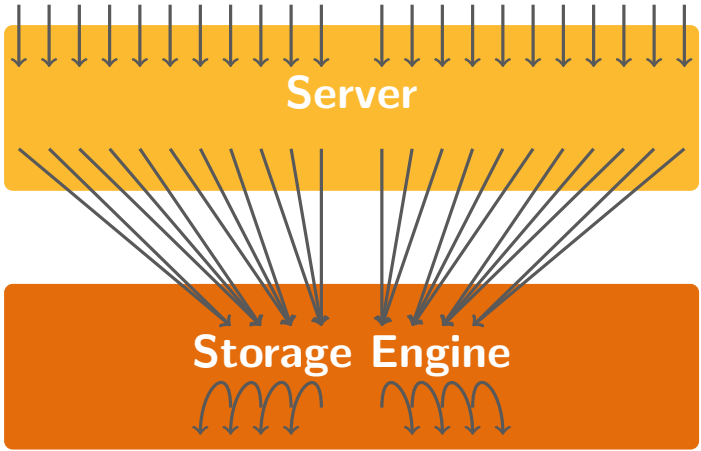
How MySQL Uses CPU

- 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?

PERCONA

What Happens with Threads

? <= CPU cores?

Yes Executed simultaneously

PERCONA

What Happens with Threads

?	$<=$ CPU cores?
Yes	Executed simultaneously
No	Wait in a queue

PERCONA

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

What Happens with Threads

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

Network Configuration

- As fast as possible
  - Speed of the line
    - RTT
  - Bandwidth
  - Stability
    - To avoid TCP packet re-submission

PERCONA

Network Configuration

- 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

How MySQL Uses Network

- Communication between server and client
  - Regular client
  - Application
  - Replication connection (IO) thread
  - Traffic between synchronous nodes

MySQL Configuration

What Can be Configured?

- Server options
- Components
  - Storage engines
    - InnoDB
  - Plugins
  - Server
    - Binary logging
    - Optimizer

System Variables and Options: Scope

- Global
  - Parameters, necessary for all server processes
    - Location of server files: datadir etc.
    - Shared buffers
    - More
- Session
  - Control connection-specific parameters

MySQL Option Tables

PERCONA

System Variables: How to Change

- `SET [GLOBAL|PERSIST] var = NEW_VALUE`

System Variables: How to Change

- `SET [GLOBAL|PERSIST] var = NEW_VALUE`
- Command-line option: `--var=new_value`

System Variables: How to Change

- `SET [GLOBAL|PERSIST] var = NEW_VALUE`
- Command-line option: `--var=new_value`
- Configuration file
  - In the default location
    - Specified by option `--defaults-file`

```
[mysqld]
var=new_value
```


PERCONA

System Variables: How to Change

- PXC Operator Configuration
  - In `cr.yaml`

```
spec:
  secretsName: my-cluster-secrets
  pxc:
    ...
      configuration: |
        [mysqld]
        innodb_log_file_size=8G
```


PERCONA

System Variables: How to Change

- PXC Operator Configuration
  - In ConfigMap
    - Create custom configuration file `my.cnf`

```
[mysqld]
innodb_log_file_size=8G
```


PERCONA

System Variables: How to Change

- PXC Operator Configuration
  - In ConfigMap
    - Create custom configuration file `my.cnf`

```
[mysqld]
innodb_log_file_size=8G
```

    - Create ConfigMap:

```
$ kubectl create configmap \
> cluster1-pxc --from-file=my.cnf
```


PERCONA

System Variables: How to Change

- PXC Operator Configuration
  - Via a Secret Object
    - Decode `my.cnf`:

```
$cat my.cnf | base64
W215c3FsZF0KaW5ub2RiX2xvZ19maWxlX3NpemU9OEcK
```


PERCONA

System Variables: How to Change

- PXC Operator Configuration
  - Via a Secret Object
    - Decode `my.cnf`:
      `$cat my.cnf | base64`
      `W215c3FsZF0KaW5ub2RiX2xvZ19maWxlX3NpemU9OEcK`
    - Create a `my-secret.yaml`:

```
apiVersion: v1
kind: Secret
metadata:
  name: cluster1-pxc
data:
  my.cnf: "W215c3FsZF0KaW5ub2RiX2x...
```

System Variables: How to Change

- PXC Operator Configuration
  - Via a Secret Object
    - Decode `my.cnf`:

      ```
      $cat my.cnf | base64
      W215c3FsZF0KaW5ub2RiX2xvZ19maWxlX3NpemU9OEcK
      ```
    - Create a `my-secret.yaml`:

      ```
      apiVersion: v1
      kind: Secret
      metadata:
        name: cluster1-pxc
      data:
        my.cnf: "W215c3FsZF0KaW5ub2RiX2x...
      ```
    - Apply the secret

      ```
      $ kubectl create -f my-secret.yaml
      ```


PERCONA

System Variables: How to Change

- PXC Operator Configuration
  - In `cr.yaml`
  - In ConfigMap
  - Via a Secret Object
  - **Restart PXC**

Dynamic Variables: Who Can Change

- Global options and few session options

< 8.0	A user with privilege `SUPER`
$8.0 +$	A user with privilege `SYSTEM_VARIABLES_ADMIN`

Dynamic Variables: Who Can Change

- Global options and few session options

  < 8.0 A user with privilege `SUPER`

  $8.0 +$ A user with privilege `SYSTEM_VARIABLES_ADMIN`

- Session options

  < 8.0 Anybody

  $8.0 +$ Restricted privileges
    - `SYSTEM_VARIABLES_ADMIN`
    - `SESSION_VARIABLES_ADMIN`

  $8.0 +$ Not restricted privileges: anybody

Dynamic Variables: Who Can Change

- Global options and few session options

  < 8.0 A user with privilege `SUPER`

  $8.0 +$ A user with privilege `SYSTEM_VARIABLES_ADMIN`

- Session options

  < 8.0 Anybody

  $8.0 +$ Restricted privileges
    - `SYSTEM_VARIABLES_ADMIN`
    - `SESSION_VARIABLES_ADMIN`

  $8.0 +$ Not restricted privileges: anybody

- **There are no limits!**

Buffers: When Allocated

- Those which control behavior of whole server
  - Once at server startup
  - Can start with low values, then grow to specified

Buffers: When Allocated

- Those which control behavior of whole server
  - Once at server startup
  - Can start with low values, then grow to specified
- Connection options
  - For every session when the connection opens

Buffers: When Allocated

- Those which control behavior of whole server
  - Once at server startup
  - Can start with low values, then grow to specified
- Connection options
  - For every session when the connection opens
- Operation-specific
  - For every operation when needed
  - Can be allocated more than once

MySQL Configuration

Important Options

InnoDB

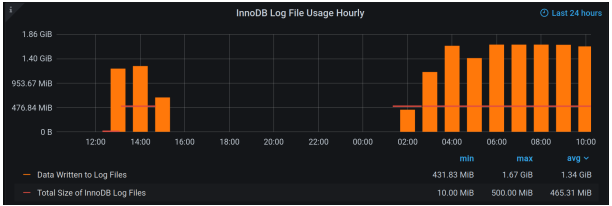
- `innodb_buffer_pool_size`
  - Ideally should hold active data set

InnoDB

- innodb_buffer_pool_size
- innodb_log_file_size
  - Should hold changes for an hour

InnoDB

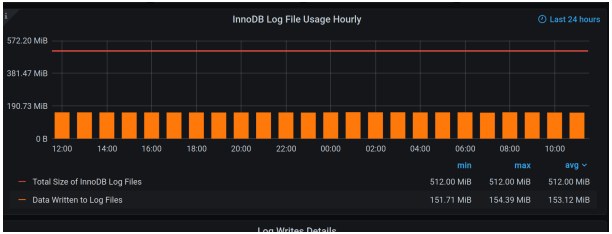
- innodb_buffer_pool_size
- innodb_log_file_size
  - Should hold changes for an hour
  - Too low



PERCONA

InnoDB

- innodb_buffer_pool_size
- innodb_log_file_size
  - Should hold changes for an hour
  - Good



PERCONA

InnoDB

- `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
  - You may need to change it if adjust hardware options in a Cloud
  - **Do not set too high!**

InnoDB

- innodb_buffer_pool_size
- innodb_log_file_size
- innodb_io_capacity
- innodb_flush_method
  - In most cases: O_DIRECT
  - **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

Synchronization

- **Changing these compromize durability!**

Synchronization

- **Changing these compromize durability!**
- `innodb_flush_log_at_trx_commit`
  1: full ACID, **default**

Synchronization

- **Changing these compromize 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** `INSERT`s per second
     - Safe with PXC, Galera and InnoDB Clusters

Synchronization

- **Changing these compromize 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

Synchronization

- **Changing these compromize 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

- **Changing these compromize 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
    - Value of `Opened_table_definitions` is larger than `Open_table_definitions`

Table Handlers

- `table_open_cache`
- `table_definition_cache`
- Increase OS `open files` limit if needed

Query Tuning

Heart of the Application

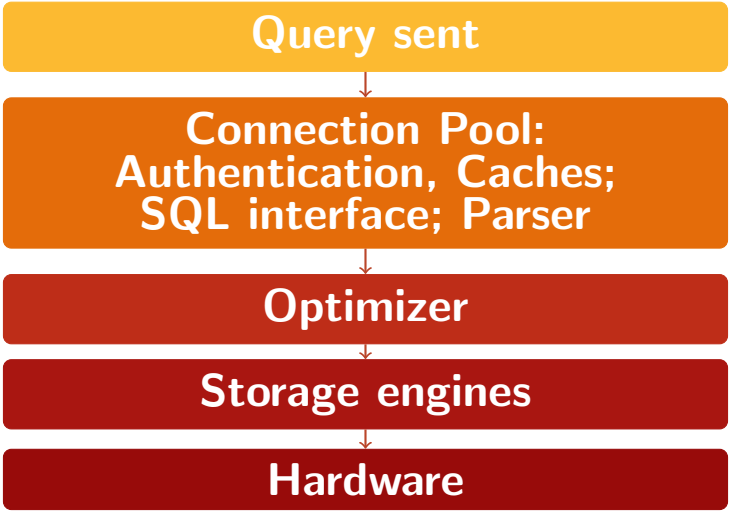
- You communicate with database using queries
  - Even via NoSQL interface
  - They are not SQL queries, but still queries

Heart of the Application

- 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

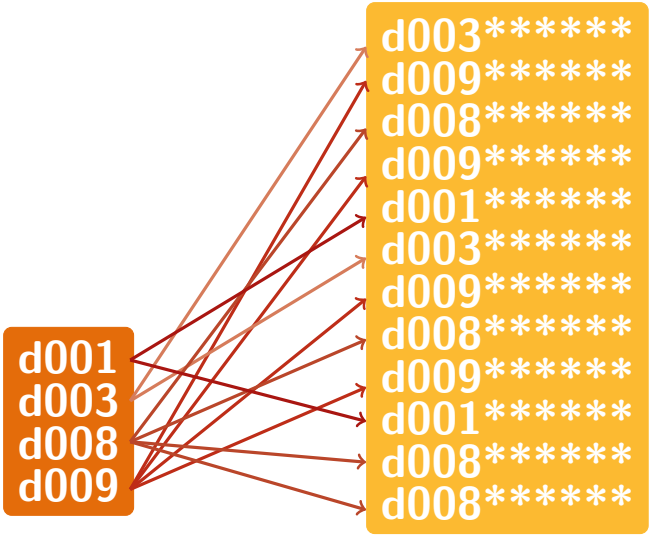


Query Tuning

Indexes

MySQL Indexes

- B-Tree Mostly
- LSM Tree
- Fractal Tree
- R-Tree Spatial
- Hash Memory SE
- Engine's



PERCONA

When MySQL Uses Indexes:

Conditions

- `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:

Conditions

- `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`

When MySQL Uses Indexes

GROUP BY

- 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

- 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 **de**scending
- right_part **must** be **a**scending
- the_index(left_part DESC, right_part ASC)

PERCONA

When MySQL Uses Indexes:

Expressions

- Deterministic, **built-in**
  - Return same value for the same argument
  - `WHERE the_column = FLOOR(123.45)`

When MySQL Uses Indexes:

Expressions

- 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`~~

When MySQL Uses Indexes:

Expressions

- 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`~~
- Stored functions and UDFs
  - Indexes are not used

MySQL: Use indexes on generated columns

MariaDB: Use indexes on generated columns

Query Tuning

Optimizer Configuration

Increase Size of Optimizer Temporary Objects

- 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

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`
    - `SELECT INTO OUTFILE`
    - `SELECT INTO DUMPFILE`

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!**

Conclusion

- Hardware

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

Conclusion

- 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

More Information

Troubleshooting hardware resources

Troubleshooting configuration issues

MySQL Query Tuning for Dev[Op]s

Percona Monitoring and Management

Percona Kubernetes Operators

Thank you!

www.slideshare.net/SvetaSmirnova

twitter.com/svetsmirnova

github.com/svetasmirnova

PERCONA