Introduction to MySQL Query Tuning

for Dev[Op]s

October 4, 2019

Sveta Smirnova

PERCONA

Table of Contents

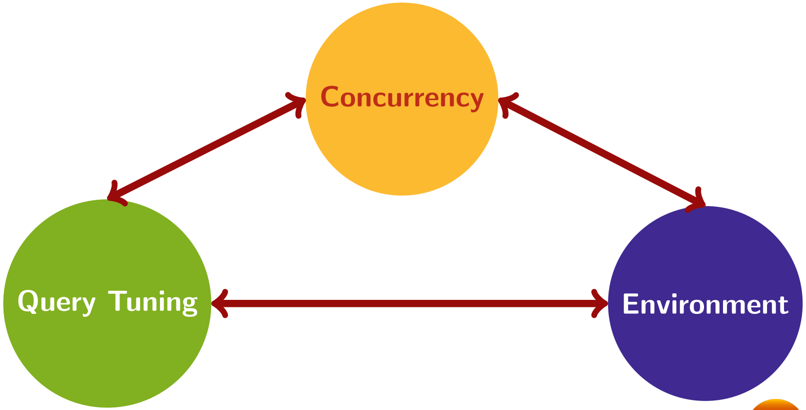
PERCONA

Sveta Smirnova

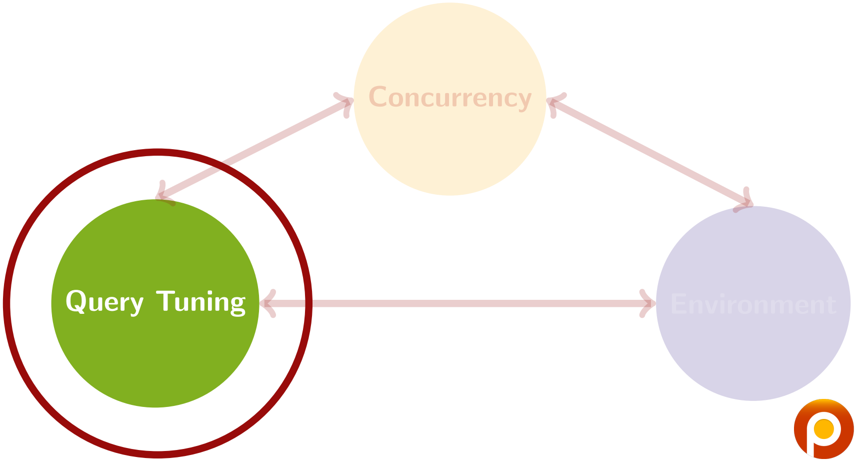
- MySQL Support engineer
- Author of
  - MySQL Troubleshooting
  - JSON UDF functions
  - FILTER clause for MySQL
- Speaker
  - Percona Live, OOW, Fosdem, DevConf, HighLoad...

Basics

Troubleshooting Workflow



Troubleshooting Workflow: This Session



The Query

```
$system = System::factory()
    ->setName($this->form->get(Field::NAME))
    ->setDescription(
        $this->form->get(Field::DESCRIPTION)
    );
DAO::system()->take($system);
```

The Query

```
$system = System::factory()
    ->setName($this->form->get(Field::NAME))
    ->setDescription(
        $this->form->get(Field::DESCRIPTION)
    );
DAO::system()->take($system);
```

The Query

```
cursor = conn.cursor()
q = '''UPDATE `foo` SET my_date=NOW(),
    subject = %s,
    msg = %s,
    address = %s,
    updated_at = NOW()
    WHERE id=%s
'''
cursor.execute(q, [
    remote_resp.get('subject'),
    remote_resp.get('msg'),
    remote_resp.get('address'),
    my_id
])
```

The Query

```
cursor = conn.cursor()
q = ’’’UPDATE ‘foo‘ SET my_date=NOW(),
    subject = %s,
    msg = %s,
    address = %s,
    updated_at = NOW()
    WHERE id=%s
’’’
cursor.execute(q, [
    remote_resp.get(’subject’),
    remote_resp.get(’msg’),
    remote_resp.get(’address’),
    my_id
])
```

PERCONA

The Query

```
SELECT dept_name, title, gender,
       min(salary) AS mins, max(salary) AS maxs
FROM employees
JOIN salaries USING(emp_no)
JOIN titles USING(emp_no)
JOIN dept_emp USING(emp_no)
JOIN departments USING(dept_no)
JOIN dept_manager USING(dept_no)
WHERE dept_manager.to_date = '9999-01-01'
GROUP BY dept_name, title, gender
ORDER BY gender, maxs DESC;
```

PERCONA

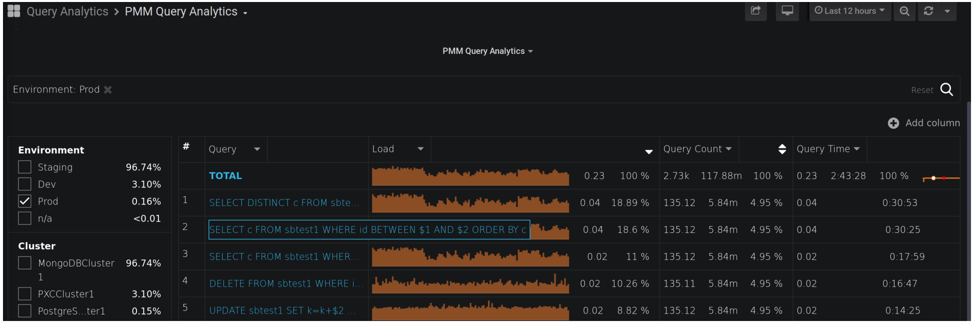
The Query

```
SELECT dept_name, title, gender,
       min(salary) AS mins, max(salary) AS maxs
FROM employees
JOIN salaries USING(emp_no)
JOIN titles USING(emp_no)
JOIN dept_emp USING(emp_no)
JOIN departments USING(dept_no)
JOIN dept_manager USING(dept_no)
WHERE dept_manager.to_date = '9999-01-01'
GROUP BY dept_name, title, gender
ORDER BY gender, maxs DESC;
```

PERCONA

Allways Tune Raw Query

- PMM QAN



Query Analytics > PMM Query Analytics

Last 12 hours

PMM Query Analytics

Environment: Prod ✖

Reset

Add column

Environment	
☐ Staging	96.74%
☐ Dev	3.10%
☑ Prod	0.16%
☐ n/a	<0.01

Cluster	
☐ MongoDBCluster 1	96.74%
☐ PXCCluster1	3.10%
☐ PostgreS...ter1	0.15%

#	Query	Load		Query Count			Query Time		
	TOTAL	0.23	100 %	2.73k	117.88m	100 %	0.23	2:43:28	100 %
1	SELECT DISTINCT c FROM sbte...	0.04	18.89 %	135.12	5.84m	4.95 %	0.04	0:30:53	
2	SELECT c FROM sbtest1 WHERE id BETWEEN $1 AND $2 ORDER BY c	0.04	18.6 %	135.12	5.84m	4.95 %	0.04	0:30:25	
3	SELECT c FROM sbtest1 WHER...	0.02	11 %	135.12	5.84m	4.95 %	0.02	0:17:59	
4	DELETE FROM sbtest1 WHERE i...	0.02	10.26 %	135.11	5.84m	4.95 %	0.02	0:16:47	
5	UPDATE sbtest1 SET k=k+$2 ...	0.02	8.82 %	135.12	5.84m	4.95 %	0.02	0:14:25	

Allways Tune Raw Query

- PMM QAN
- Slow Query Log

Allways Tune Raw Query

- PMM QAN
- Slow Query Log
- Application log

Allways Tune Raw Query

- PMM QAN
- Slow Query Log
- Application log
- ...

PERCONA

Slow is relative

- Mind you data!
- 75,000,000 rows
  - `(INT, INT)`
    - 75,000,000 * (4 + 4) = 600,000,000 bytes = 572 MB
  - `(INT, INT, DATETIME, VARCHAR(255), VARCHAR(255))`
    - 75,000,000 * (4 + 4 + 8 + 256 + 256) = 39,600,000,000 bytes = 37 G
  - 39,600,000,000 / 600,000,000 = 66

PERCONA

Slow is relative

- Mind you data!
- Mind use case
    - Popular website
    - Admin interface
    - Weekly cron job

Slow is relative

- Mind you data!
- Mind use case
- Mind location
  - Server, used by multiple connections
  - Dedicated for OLAP queries

Why Query can be Slow

- MySQL performs a job to execute a query

Why Query can be Slow

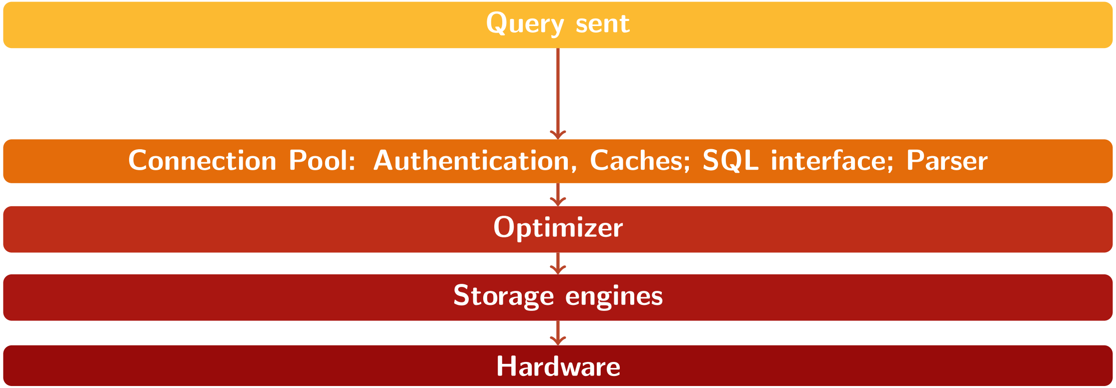
- MySQL performs a job to execute a query
- In worst case scenario it will do a full table scan
  - `CREATE INDEX`
  - `ANALYZE TABLE ... UPDATE HISTOGRAM ON`

PERCONA

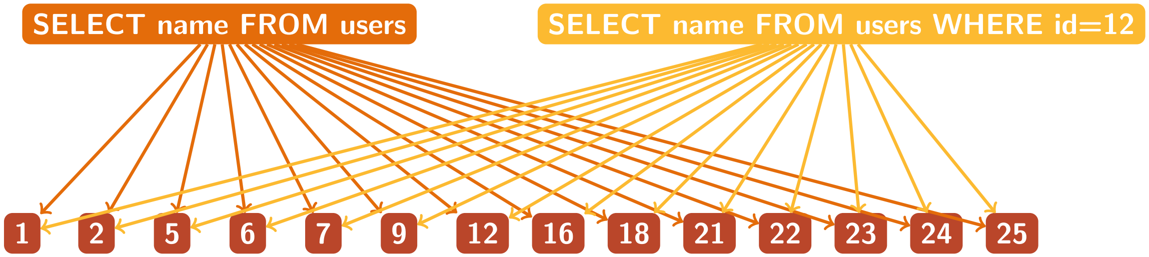
Why Query can be Slow

- MySQL performs a job to execute a query
- In worst case scenario it will do a full table scan
  - `CREATE INDEX`
  - `ANALYZE TABLE ... UPDATE HISTOGRAM ON`
- Incorrect index can be used

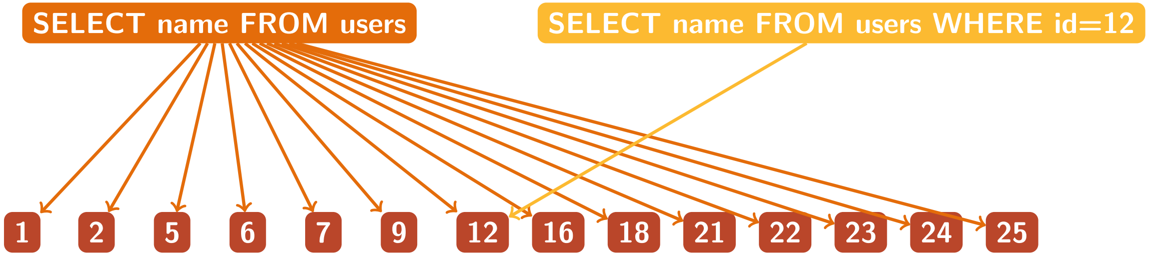
Query Execution Workflow


Query sent
Connection Pool: Authentication, Caches; SQL interface; Parser
Optimizer
Storage engines
Hardware

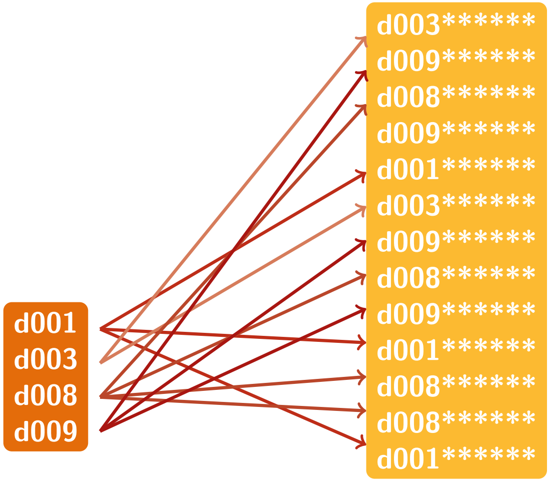
Full Table Scan



After Index Added



MySQL Indexes



- B-Tree (Mostly)
- Fractal Tree
- LSM Tree
- R-Tree (Spatial)
- Hash (Memory SE)
- Engine-dependent

How to Create an Index

- Single column

```
CREATE INDEX index_name ON
the_table(the_column)
```

- Multiple columns

```
CREATE INDEX index_name ON
the_table(column1, column2)
```

How to Create an Index

- Single column
  ```
  ALTER TABLE table_name ADD INDEX
  [index_name] (the_column)
  ```
- Multiple columns
  ```
  ALTER TABLE table_name ADD INDEX
  [index_name] (column1, column2)
  ```

PERCONA

When MySQL Uses Indexes

PERCONA

Conditions

- `WHERE the_column = a_value`
- `WHERE the_column IN(value1, value2, value3)`
- `WHERE the_column LIKE 'value%'`
- ~~WHERE the_column LIKE '%value'~~

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~~

Joins

- `table1 JOIN table2 ON table1.column1 = table2.column2`

Joins

- `table1 JOIN table2 ON table1.column1 = table2.column2`
- Same as `FROM table1, table2 WHERE table1.column1 = table2.column2`

PERCONA

GROUP BY

- `GROUP BY the_column`
- `GROUP BY left_part, right_part`
- ~~GROUP BY right_part, left_part~~
- ~~GROUP BY the_index, another_index~~

ORDER BY

- ORDER BY the_column
- ORDER BY left_part, right_part
- ~~ORDER BY right_part, left_part~~
- ~~ORDER BY the_index, another_index~~

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)

Expressions

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

Expressions

- Deterministic, built-in
  - Return same value for the same argument
  - `WHERE the_column = FLOOR(123.45)`
- Non-deterministic
  - Return different values for different invocations
  - ~~WHERE the_column = RAND() * 100~~

Expressions

- Deterministic, built-in
  - Return same value for the same argument
  - `WHERE the_column = FLOOR(123.45)`
- Non-deterministic
  - Return different values for different invocations
  - ~~`WHERE the_column` = RAND() * 100~~
- Stored functions and UDFs
  - Indexes are not used
  - Use generated column indexes

Diagnostics

Diagnostics

EXPLAIN: estimation on how Optimizer works

PERCONA

How to Find how MySQL Uses Indexes

- `EXPLAIN`
  - Estimates what happens during query execution

5.6- `EXTENDED`
5.6- `PARTITIONS`
5.6+ `FORMAT=JSON`
8.0+ `FORMAT=TREE`

PERCONA

How to Find how MySQL Uses Indexes

- `EXPLAIN`
  - Estimates what happens during query execution

5.6- `EXTENDED`
5.6- `PARTITIONS`
5.6+ `FORMAT=JSON`
8.0+ `FORMAT=TREE`

- `INFORMATION_SCHEMA.OPTIMIZER_TRACE`
  - Real data, collected after query was executed
  - Advanced topic

PERCONA

Effect of Indexes: Before

```
mysql> explain select * from t1\G
*************************** 1. row ***************************
...
rows: 12
Extra: NULL

mysql> explain select * from t1 where f2=12\G
*************************** 1. row ***************************
...
key: NULL
...
rows: 12
Extra: Using where
```

Same number of examined rows for both queries

Effect of Indexes: After

```
mysql> alter table t1 add index(f2);
Query OK, 12 rows affected (0.07 sec)
Records: 12  Duplicates: 0  Warnings: 0

mysql> explain select * from t1 where f2=12\G
*************************** 1. row ***************************
    ...
    key: f2
key_len: 5
    ref: const
   rows: 1
  Extra: NULL
1 row in set (0.00 sec)
```

Much more effective!
Only 1 row examined

PERCONA

EXPLAIN: overview

```
mysql> explain extended select * from t1 join t2 where t1.int_key=1;
+----+-------------+-------+-------+---------------+---------+---------+-------+------+------+---------
| id | select_type | table | type  | possible_keys | key     | key_len | ref   | rows | f... | Extra
+----+-------------+-------+-------+---------------+---------+---------+-------+------+------+---------
|  1 | SIMPLE      | t1    | ref   | int_key,ik    | int_key | 5       | const |    4 | 100. | NULL
|  1 | SIMPLE      | t2    | index | NULL          | pk      | 9       | NULL  |    6 | 100. | Using in
Using join buffer           |
(
+----+-------------+-------+-------+---------------+---------+---------+-------+------+------+---------
2 rows in set, 1 warning (0.00 sec)
```

Number of select

Tables, for which information is printed

Possible keys

Length of the key

Number of examined rows

Additional informatio

Table, for which information is printed

Product of rows here: how many rows in all tables will be accessed
For this example estimated value is 4*6 = 24

Select type

Key, which was actually used

% of filtered rows
rows x filtered / 100 — number of row
which will be joined with another tabl

How data is accessed

Which columns were compared with the index

```
Note (Code 1003): /* select#1 */ select `test`.`t1`.`pk` AS `pk`,`test`.`t1`.`int_key` AS `int_key`,`te
AS `pk`,`test`.`t2`.`int_key` AS `int_key` from `test`.`t1` join `test`.`t2` where ((`test`.`t1`.`int_ke
```

Actual (optimized) query as executed by MySQL Server

EXPLAIN in Details

```
mysql> explain extended select * from t1 join t2 where...
+----+-------------+-------+-------+***
| id | select_type | table | type  |***
+----+-------------+-------+-------+***
| 1  | SIMPLE      | t1    | ref   |***
| 1  | SIMPLE      | t2    | index |***
+----+-------------+-------+-------+***
2 rows in set, 1 warning (0.00 sec)
```

SIMPLE;PRIMARY;UNION;DEPENDENT UNION;UNION RESULT;
SUBQUERY;DEPENDENT SUBQUERY;DERIVED;MATERIALIZED

system
const
eq_ref
ref
fulltext
ref_or_null
index_merge
unique_subquery
index_subquery
range
index
ALL

EXPLAIN in Details: keys

Keys, which can be used for resolving the query

Actual length of the key (Important for multiple-column keys)

```
mysql> explain extended select * from t1 join t2 where t1.int_key=1;
***+---------------+---------+---------+-------+***
***| possible_keys | key     | key_len | ref   |***
***+---------------+---------+---------+-------+***
***| int_key,ik    | int_key | 5       | const |***
***| NULL          | pk      | 9       | NULL  |***
***+---------------+---------+---------+-------+***
2 rows in set, 1 warning (0.00 sec)
```

Constant
Numeric in our case

Index used
to resolve rows

Only one key was actually used

Which columns were compared with the index

EXPLAIN in Details: rows

Number of rows accessed

% of rows filtered

Additional information: how query is resolved

Using filesort
Using temporary
etc.

```
mysql> explain extended select * from t1 join t2 where t1.int_key=1;
***+-------+----------+----------------------------------------------------+
***| rows  | filtered | Extra                                              |
***+-------+----------+----------------------------------------------------+
***|     4 |   100.00 | NULL                                               |
***|     6 |   100.00 | Using index; Using join buffer (Block Nested Loop) |
***+-------+----------+----------------------------------------------------+
2 rows in set, 1 warning (0.00 sec)
```

4X6 = 24

All rows used

EXPLAIN Type by Example: ALL

```
mysql> explain select count(*) from employees where hire_date > '1995-01-01'
********************** 1. row **********************
           id: 1
  select_type: SIMPLE
        table: employees
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 300157
        Extra: Using where
1 row in set (0.00 sec)
```

All rows in the table examined
Worst plan ever!

PERCONA

EXPLAIN Type by Example: range

- We need to add index to table `employees` first

```
mysql> alter table employees add index(hire_date);
Query OK, 0 rows affected (3.48 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

PERCONA

EXPLAIN Type by Example: range

```
mysql> explain select count(*) from employees where hire_date>'1995-01-01'\G
********************** 1. row **********************
           id: 1
  select_type: SIMPLE
        table: employees
         type: range
possible_keys: hire_date
          key: hire_date
      key_len: 3
          ref: NULL
         rows: 68654
        Extra: Using where; Using index
1 row in set (0.00 sec)
```

Only rows from given range used

Compare with ALL:
300157/68654 = 4.3720
4 times less rows examined!

PERCONA

Combined Indexes

- Consists of two or more columns

Combined Indexes

- Consists of two or more columns
- Only leftmost part used

```
mysql> alter table City add key
    -> comb(CountryCode, District, Population),
    -> drop key CountryCode;
```

PERCONA

Combined Indexes: example 1

```
mysql> explain select * from City where CountryCode = 'USA'\G
********************** 1. row ******************
        table: City
         type: ref
possible_keys: comb
          key: comb
      key_len: 3
          ref: const
         rows: 273
```

Uses first field from the comb key

Combined Indexes: example 2

```
mysql> explain select * from City where \
-> District = 'California' and population > 10000\G
********************** 1. row ******************
        table: City
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3868
```

Can't use combined index: not a leftmost part

Does not have the CountryCode in the where clause = can't use comb index

Combined Indexes: key_len

- `Key_len = total size` (in bytes)
- Index
  - `comb(CountryCode, District, Population)`

Explain:	Fields:
`key: comb`	`CountryCode char(3)`
`key_len: 3`	`District char(20)`
	`Population int(11)`

3 ->Char(3) ->First field is used

PERCONA

EXPLAIN Type by Example: index

```
mysql> explain select count(*) from titles where title='Senior Engineer'\G
********************** 1. row **********************
           id: 1
  select_type: SIMPLE
        table: titles
         type: index
possible_keys: NULL
          key: emp_no
      key_len: 4
          ref: NULL
         rows: 444033
Extra: Using where; Using index
1 row in set (0.11 sec)
```

No row in the table was accessed to resolve the query!
Only index used
Still all records in the index were scanned

Covered Indexes

- Covered index = cover all fields in the query

```
select name from City
where CountryCode = 'USA' and District = 'Alaska' and population > 10000

mysql> alter table City add key
    -> cov1(CountryCode, District, population, name);
```

1. Where part

2. Group By/Order (not used now)

3. Select part

Uses all fields in the query in particular order

EXPLAIN by Example: Covered Indexes

```
mysql> explain select name from City  where CountryCode = 'USA' \
-> and District = 'Alaska' and population > 10000\G
*************************** 1. row ***********
        table: City
         type: range
possible_keys: cov1
          key: cov1
      key_len: 27
          ref: NULL
         rows: 1
        Extra: Using where; Using index
```

Covered index is used
MySQL will only use index
Will not go to the data file

PERCONA

Diagnostics

Real Numbers: Inside Storage Engine

PERCONA

Handler_* Status Variables

- EXPLAIN is optimistic

```
mysql> explain select * from ol
    -> where thread_id=10432 and site_id != 9939 order by id limit 3\G
*************************** 1. row ***************************
           id: 1              |          ref: NULL
  select_type: SIMPLE         |         rows: 33
        table: ol             |     filtered: 8.07
   partitions: NULL           |        Extra: Using where
         type: index
possible_keys: thread_id
          key: PRIMARY
      key_len: 4
1 row in set, 1 warning (0,00 sec)
```

Handler_* Status Variables

- Status variables 'Handler_*' show truth

```
mysql> flush status; select * from ol
    -> where thread_id=10432 and site_id != 9939 order by id limit 3;
mysql> show status like 'Handler%';
+----------------------------+---------+
| Variable_name              | Value   |
+----------------------------+---------+
...
| Handler_read_first         | 1       |
| Handler_read_key           | 1       |
| Handler_read_last          | 0       |
| Handler_read_next          | 100000  |
...
```

Diagnostics

Real Numbers: Inside the Server

PROCESSLIST

- `SHOW [FULL] PROCESSLIST`
- `INFORMATION_SCHEMA.PROCESSLIST`
- `performance_schema.THREADS`

PROCESSLIST

- `SHOW [FULL] PROCESSLIST`
- `INFORMATION_SCHEMA.PROCESSLIST`
- `performance_schema.THREADS`
- Your first alert in case of performance issue

PROCESSLIST

- `SHOW [FULL] PROCESSLIST`
- `INFORMATION_SCHEMA.PROCESSLIST`
- `performance_schema.THREADS`
- Your first alert in case of performance issue
- Shows all queries, running at the moment

Execution Stages

- Can be seen in PROCESSLIST

```
mysql> show processlist\G
************************ 1. row ************************
     Id: 7
   User: root
   Host: localhost:48799
     db: employees
Command: Query
   Time: 2
  State: Sending data
   Info: select count(*) from employees join titles using(emp_no)
         where title='Senior Engineer'
         ...
```

Execution Stages

- Can be seen in `PROCESSLIST`
  - Very useful when you need to answer on question: ''What is my server doing now?''

Execution Stages

- PERFORMANCE_SCHEMA.EVENTS_STAGES_*

```
mysql> select eshl.event_name, substr(sql_text, 1, 15) as ‘sql‘,
    -> eshl.timer_wait/1000000000000 w_s from  events_stages_history_lon
    -> eshl join  events_statements_history_long esthl on
    -> (eshl.nesting_event_id = esthl.event_id) where
    -> esthl.current_schema=’employees’ and sql_text like
    -> ’select count(*) from employees%’ order by eshl.timer_start asc;
+--------------------------------+-----------------+--------+
| event_name                     | sql             | w_s    |
+--------------------------------+-----------------+--------+
| stage/sql/starting             | select count(*) | 0.0002 |
| stage/sql/checking permissions | select count(*) |    000 |
...
```

Execution Stages

- PERFORMANCE_SCHEMA.EVENTS_STAGES_*

```
...
| stage/sql/checking permissions | select count(*) | 0.0000 |
| stage/sql/Opening tables       | select count(*) | 0.0000 |
| stage/sql/init                 | select count(*) | 0.0001 |
| stage/sql/System lock          | select count(*) | 0.0000 |
| stage/sql/optimizing           | select count(*) | 0.0000 |
| stage/sql/statistics           | select count(*) | 0.0001 |
| stage/sql/preparing            | select count(*) | 0.0000 |
| stage/sql/executing            | select count(*) | 0.0000 |
| stage/sql/Sending data         | select count(*) | 5.4915 |
| stage/sql/end                  | select count(*) |    000 |
...
```

Temporary tables and other job

- Status variables

```
mysql> flush status;
Query OK, 0 rows affected (0,01 sec)
mysql> select count(*) from employees join titles using(emp_no)
    -> where title='Senior Engineer';
+----------+
| count(*) |
+----------+
|    97750 |
+----------+
1 row in set (5,44 sec)
```

Temporary tables and other job

- Status variables

```
mysql> select * from performance_schema.session_status
    -> where variable_name in ('Created_tmp_tables',
    -> 'Created_tmp_disk_tables', 'Select_full_join',
    -> 'Select_full_range_join', 'Select_range',
    -> 'Select_range_check', 'Select_scan', 'Sort_merge_passes',
    -> 'Sort_range', 'Sort_rows', 'Sort_scan') and variable_value > 0;
+------------------------+----------------+
| VARIABLE_NAME          | VARIABLE_VALUE |
+------------------------+----------------+
| Select_scan            | 2              |
+------------------------+----------------+
1 row in set (0,00 sec)
```

Temporary tables and other job

- PERFORMANCE_SCHEMA.EVENTS_STATEMENTS_*

```
mysql> select * from performance_schema.events_statements_history_long
    -> where sql_text like 'select count(*) from employees join %'\G
*************************** 1. row ***************************
...
             ROWS_SENT: 1               SELECT_RANGE_CHECK: 0
         ROWS_EXAMINED: 541058                 SELECT_SCAN: 1
CREATED_TMP_DISK_TABLES: 0               SORT_MERGE_PASSES: 0
     CREATED_TMP_TABLES: 0                      SORT_RANGE: 0
       SELECT_FULL_JOIN: 0                       SORT_ROWS: 0
 SELECT_FULL_RANGE_JOIN: 0                       SORT_SCAN: 0
           SELECT_RANGE: 0                   NO_INDEX_USED
```

Temporary tables and other job

- sys.statement_analysis

```
mysql> select * from statement_analysis where query like 'SELECT COUNT
-> ( * ) FROM ‘emplo%' and db='employees'\G
*************************** 1. row ***************************
        query: SELECT COUNT ( * ) FROM ‘emplo ...  ‘emp_no‘ ) WHE...
           db: employees          max_latency: 5.59 s
    full_scan:                    avg_latency: 5.41 s
   exec_count: 7                 lock_latency: 2.24 ms
    err_count: 0                    rows_sent: 7
   warn_count: 0                rows_sent_avg: 1
total_latency: 37.89 s          rows_examined: 3787406
```

Temporary tables and other job

- sys.statement_analysis

```
rows_examined_avg: 541058
    rows_affected: 0
rows_affected_avg: 0
       tmp_tables: 0
  tmp_disk_tables: 0
      rows_sorted: 0
sort_merge_passes: 0
           digest: 4086bc3dc6510a1d9c8f2fe1f59f0943
       first_seen: 2016-04-14 15:19:19
        last_seen: 2016-04-14 16:13:14
```

PERCONA

How to Affect Query Plans

PERCONA

What has Effect on Query Optimizer Plans?

- Index statistics
- Histogram statistics
- Optimizer switches
- Bugs in optimizer

Index Statistics

- Collected by storage engine

Index Statistics

- Collected by storage engine
- Used by Optimizer

Index Statistics

- Can be examined by `SHOW INDEX` command

```
mysql> show index from sbtest1;
+---------+----------+-------------+-------------+
| Table   | Key_name | Column_name | Cardinality |
+---------+----------+-------------+-------------+
| sbtest1 | k_1      | k           |       49142 |
+---------+----------+-------------+-------------+
mysql> select count(distinct id), count(distinct k) from sbtest1;
+--------------------+-------------------+
| count(distinct id) | count(distinct k) |
+--------------------+-------------------+
|             100000 |             17598 |
+--------------------+-------------------+
```

PERCONA

Index Statistics

- Can be updated
  - `ANALYZE TABLE`
  - If does not help: rebuild table
    - `OPTIMIZE TABLE`
    - `ALTER TABLE ENGINE=INNODB; ANALYZE TABLE`

Histogram Statistics

- Since version 8.0

Histogram Statistics

- Since version 8.0
- Collected and used by the Optimizer

Histogram Statistics

- Since version 8.0
- Collected and used by the Optimizer
- Can be examined in Information Schema

```
mysql> select HISTOGRAM from information_schema.column_statistics
    -> where table_name='example'\G
*************************** 1. row ***************************
HISTOGRAM: {"buckets": [[1, 0.6], [2, 0.8], [3, 1.0]],
"data-type": "int", "null-values": 0.0, "collation-id": 8,
"last-updated": "2018-11-07 09:07:19.791470",
"sampling-rate": 1.0, "histogram-type": "singleton",
"number-of-buckets-specified": 3}
1 row in set (0.00 sec)
```

PERCONA

Histogram Statistics

- Since version 8.0
- Collected and used by the Optimizer
- Can be examined in Information Schema

More details

Optimizer Switches

```
mysql> select @@optimizer_switch\G
*************************** 1. row ***************************
@@optimizer_switch: index_merge=on,index_merge_union=on,
index_merge_sort_union=on,index_merge_intersection=on,
engine_condition_pushdown=on,index_condition_pushdown=on,
mrr=on,mrr_cost_based=on,
block_nested_loop=on,batched_key_access=off,
materialization=on,semijoin=on,loosescan=on,firstmatch=on,
duplicateweedout=on,subquery_materialization_cost_based=on,
use_index_extensions=on,condition_fanout_filter=on,derived_merge=on
1 row in set (0,00 sec)
```

Optimizer Switches

- Turn `ON` and `OFF` particular optimization

Optimizer Switches

- Turn `ON` and `OFF` particular optimization
- Can be not helpful
  - Especially for queries, tuned for previous versions

Optimizer Switches

- Turn `ON` and `OFF` particular optimization
- Can be not helpful
- Work with them as with any other option
  - Turn `OFF` and try

```
SET optimizer_switch = 'use_index_extensions=off';
SELECT ...
EXPLAIN SELECT ...
```

PERCONA

Optimizer Switches

- Turn `ON` and `OFF` particular optimization
- Can be not helpful
- Work with them as with any other option
  - If helps implement in queries

```
SELECT /*+ SEMIJOIN(FIRSTMATCH, LOOSESCAN) */ * FROM t1 ...;
SELECT /*+ BKA(t1) NO_BKA(t2) */ * FROM t1 INNER JOIN t2 WHERE ...;
```

Bugs in Optimizer

- Optimizer choses wrong index for no reason

Bugs in Optimizer

- Optimizer choses wrong index for no reason
- Statistics is up to date
- Histograms are not usable

Bugs in Optimizer

- Optimizer choses wrong index for no reason
- Statistics is up to date
- Histograms are not usable
- Solution
  - Use index hints
    - `FORCE INDEX`
    - `IGNORE INDEX`

Bugs in Optimizer

- Optimizer choses wrong index for no reason
- Statistics is up to date
- Histograms are not usable
- Solution
- On every upgrade
  - Remove index hints
  - Test if query improved
  - You must do it even for minor version upgrades!

PERCONA

Summary

- `EXPLAIN` is essential for query tuning
- Real job is done by storage engine
- Index statistics affect query execution plan
- All index hints, optimizer hints and other workarounds must be validated on each upgrade

More information

- EXPLAIN Syntax
- EXPLAIN FORMAT=JSON is Cool! series
- Troubleshooting Performance add-ons
- Optimizer Satistics aka Histograms
- Optimizer Hints
- Tracing the Optimizer

PERCONA

Special thanks

Alexander Rubin for combined and covered index examples

Thank you!

www.slideshare.net/SvetaSmirnova

twitter.com/svetsmirnova

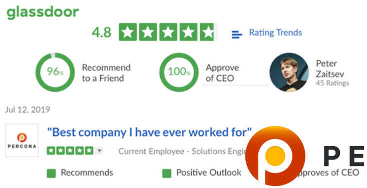
github.com/svetasmirnova

We're Hiring!

Percona's open source database experts are true superheroes, improving database performance for customers across the globe.

Percona's open source database experts are true superheroes, improving database performance for customers across the globe.

Discover what it means to have a Percona career with the smartest people in the database performance industries, solving the most challenging problems our customers come across.



glassdoor

4.8 ★★★★★ Rating Trends

96% Recommend to a Friend

100% Approve of CEO

Peter Zaitsev 45 Ratings

Jul 12, 2019

"Best company I have ever worked for"

★★★★★ Current Employee - Solutions Engi

🟩 Recommends 🟩 Positive Outlook pproves of CEO

PERCONA

DATABASE PERFORMANCE MATTERS