Load Testing Tools

for Troubleshooting MySQL Concurrency Issues

May, 23, 2018

Sveta Smirnova

Introduction

- This is very personal webinar
  - No intended use
  - No best practices
  - No QA-specific tools
    - Real life don't generate queries
    - Each customer has its own corner case
    - Typical issues usually already found by QA team before first GA release date
  - Creative by nature

PERCONA

Introduction

- This is very personal webinar
- I usually don't do benchmarks
  - Unless have a reason to

Performance Schema Benchmarks: OLTP RW

PERCONA

Introduction

- This is very personal webinar
- I usually don't do benchmarks
- I use load tools for reproducible scenarios
  - Locking issues
  - Crashes
  - Custom slowdowns
  - Replication issues
  - Almost everything!

PERCONA

Introduction

- This is very personal webinar
- I usually don't do benchmarks
- I use load tools for reproducible scenarios
- It helps to fix them!

PERCONA

Few Examples

Create binary log events

```
3. Connect to master, create database foo.

4. In parallel client start some load:

mysqlslap --user=root --host=127.0.0.1 --port=13000 --create-schema=foo \
--query="create table if not exists t1(f1 int); insert into t1 values(1); \
drop table if exists t1;" -c 10 -i 10000

5. kill -9 slave process

6. Restart MTR in dirty mode:
...
```

PERCONA

Few Examples

Generate parallel load

```
6. Run concurrent environment:

sysbench --test=/home/sveta/src/sysbench/sysbench/tests/db/oltp.lua --mysql-engine-trx=yes \
--mysql-table-engine=innodb --oltp_table_size=1000 --oltp_tables_count=1 --mysql-user=root \
--mysql-host=127.0.0.1 --mysql-port=13000 --num-threads=8 --max-requests=1000 run

7. Turn OFF and ON slow log:

mysql> set global slow_query_log=0;
Query OK, 0 rows affected (0,09 sec)
mysql> set global slow_query_log=1;
Query OK, 0 rows affected (0,00 sec)

8. Now you have 4 descriptors:
...
```

PERCONA

Few Examples

Prepare data

```
...

--disable_query_log
--let $i=10
while($i)
{
insert into cartest values  (null, 1);
--dec $i
}
--enable_query_log
...
```

PERCONA

Table of Contents

- What Do I Do?
- Case Study: Slave Crash
- Why Do I Do Load Tests?
- Machines Which I Use
- Binaries Which I Use
- How Do I Scale
- Summary

PERCONA

What Do I Do?

Single Thread Issues: MySQL CLI

- Benchmarking tool by nature

```
mysql> SELECT id, k FROM sbtest2 WHERE id=5014;
+------+------+
| id   | k    |
+------+------+
| 5014 | 3972 |
+------+------+
1 row in set (0.16 sec)
```

PERCONA

Single Thread Issues: MySQL CLI

- Benchmarking tool by nature

```
mysql> SELECT id, k FROM sbtest2 WHERE id=5014;
+------+------+
| id   | k    |
+------+------+
| 5014 | 3972 |
+------+------+
1 row in set (0.16 sec)
```

- Access to
  - Status variables
  - Engine status
  - Performance Schema
  - All internal troubleshooting tools

PERCONA

More than One Query: SHELL

- Shell is our friend

```
$ while (true); do mysql -e "flush slow logs"; done
...
$ for i in ‘seq 1 1 100000‘; \
   do mysql -e "insert into t1(id, myfield) values \
   ($i, ’‘cat /dev/urandom | tr -dc A-Za-z0-9 | head -c $(( RANDOM % (250 - 5 + 1 ) + 5 ))‘’)"; \
   done
```


PERCONA

More than One Query: SHELL

- Shell is our friend

```
$ while (true); do mysql -e "flush slow logs"; done
...
$ for i in ‘seq 1 1 100000‘; \
   do mysql -e "insert into t1(id, myfield) values \
   ($i, ’‘cat /dev/urandom | tr -dc A-Za-z0-9 | head -c $(( RANDOM % (250 - 5 + 1 ) + 5 ))‘’)"; \
   done
```

- Helps to
  - Generate data
  - Craft queries with slightly various parameters

PERCONA

More than One Query: SHELL

- Shell is our friend

```
$ while (true); do mysql -e "flush slow logs"; done
...
$ for i in ‘seq 1 1 100000‘; \
   do mysql -e "insert into t1(id, myfield) values \
   ($i, ’‘cat /dev/urandom | tr -dc A-Za-z0-9 | head -c $(( RANDOM % (250 - 5 + 1 ) + 5 ))‘’)"; \
   done
```

- Helps to
  - Generate data
  - Craft queries with slightly various parameters
- Execute in multiple terminals
  - To imitate concurrent load

PERCONA

Simple Tests: mysqlslap

- Queries without variations

```
$ mysqlslap --concurrency=8 --number-of-queries=1000 \
   --query="REPLACE INTO mytest (ts, id, f1, f2, f3, f4) \
   VALUES (now(), 1000*rand(), 'field text', 1000000*rand(), 'more text', 1)" \
   --create-schema=test \
   --pre-query="CREATE TABLE test.mytest(ts TIMESTAMP, id INT NOT NULL PRIMARY KEY, \
   f1 VARCHAR(255), f2 INT, f3 CHAR(20), f4 INT) ENGINE=INNODB" \
   --post-query="DROP TABLE test.mytest"
   Benchmark
     Average number of seconds to run all queries: 113.591 seconds
     Minimum number of seconds to run all queries: 113.591 seconds
     Maximum number of seconds to run all queries: 113.591 seconds
     Number of clients running queries: 8
     Average number of queries per client: 125
```

PERCONA

Simple Tests: mysqlslap

- Queries without variations
- Supports auto-generated SQL

```
$ time mysqlslap --concurrency=8 --auto-generate-sql --number-of-queries=100
    Benchmark
     Average number of seconds to run all queries: 8.126 seconds
     Minimum number of seconds to run all queries: 8.126 seconds
     Maximum number of seconds to run all queries: 8.126 seconds
     Number of clients running queries: 8
     Average number of queries per client: 12


    real 1m37.932s
    user 0m0.032s
    sys 0m0.008s
```

PERCONA

Simple Tests: mysqlslap

- Queries without variations
- Supports auto-generated SQL
- Helpful for
  - Simple query patterns
  - When you need to imitate some activity
  - Generate binary log events

PERCONA

Simple Tests: mysqlslap

- Queries without variations
- Supports auto-generated SQL
- Helpful for
- Should be used with care
  - In auto-generate mode drops schema
    - No matter if it has other tables of not!
  - Always use option `--no-drop`
  - Do not use on production databases

PERCONA

Standard and Custom Tests: SysBench

- Comes with standard OLTP tests

```
$ sysbench --test=/home/sveta/src/sysbench/sysbench/tests/db/oltp.lua \
 --mysql-engine-trx=yes --mysql-table-engine=innodb --oltp_table_size=10000 \
 --oltp_tables_count=2 --num-threads=8 \
 --mysql-user=root --mysql-host=127.0.0.1 --mysql-port=13002 prepare
sysbench 0.5:  multi-threaded system evaluation benchmark

Creating table 'sbtest1'...
Inserting 10000 records into 'sbtest1'
Creating secondary indexes on 'sbtest1'...
Creating table 'sbtest2'...
Inserting 10000 records into 'sbtest2'
Creating secondary indexes on 'sbtest2'...
```

PERCONA

Standard and Custom Tests: SysBench

- Comes with standard OLTP tests

```
$ sysbench --test=/home/sveta/src/sysbench/sysbench/tests/db/oltp.lua \
 --mysql-engine-trx=yes --mysql-table-engine=innodb --oltp_table_size=10000 \
 --oltp_tables_count=2 --num-threads=8 \
 --mysql-user=root --mysql-host=127.0.0.1 --mysql-port=13002 run
sysbench 0.5:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 8
Random number generator seed is 0 and will be ignored


Threads started!
...

Threads fairness:
    events (avg/stddev):           1281.0000/12.29
    execution time (avg/stddev):   1564.7510/1.16
```

PERCONA

Standard and Custom Tests: SysBench

- Comes with standard OLTP tests
- Standard tests help to
  - Imitate some activity
  - Generate binary log events
  - Test effect of options/upgrades

PERCONA

Standard and Custom Tests: SysBench

- Comes with standard OLTP tests
- Standard tests help to
- Standard OLTP tests include
  - Read only
  - `INSERT`s only
  - `DELETE`s only
  - Mixed
  - More

PERCONA

Standard and Custom Tests: SysBench

- Comes with standard OLTP tests
- Standard tests help to
- Standard OLTP tests include
- SysBench has standard hardware tests

PERCONA

Standard and Custom Tests: SysBench

- Comes with standard OLTP tests
- Standard tests help to
- Standard OLTP tests include
- SysBench has standard hardware tests
- I use mostly OLTP for troubleshooting

PERCONA

Custom SysBench Tests

- True power for troubleshooting
- You can imitate any load
- Supports Lua scripting

PERCONA

SysBench: Test Structure

- `thread_init`
  - Initialize thread

PERCONA

SysBench: Test Structure

- `thread_init`
- `set_vars`
  - Set user variables

PERCONA

SysBench: Test Structure

- `thread_init`
- `set_vars`
- `prepare`
  - Prepare test

PERCONA

SysBench: Test Structure

- `thread_init`
- `set_vars`
- `prepare`
- `event`
  - Run event

PERCONA

SysBench: Test Structure

- `thread_init`
- `set_vars`
- `prepare`
- `event`
- `cleanup`
  - Cleanup test

PERCONA

SysBench: Test Structure

- `thread_init`
- `set_vars`
- `prepare`
- `event`
- `cleanup`
- Version 1.0+
  - `sysbench.cmdline.options`
  - `sysbench.cmdline.commands`

PERCONA

SysBench 0.5+: Variables

```
function set_vars()
    branches_size = branches_size or 1
    tellers_size = tellers_size or 10
    accounts_size = accounts_size or 100000
    scale = scale or 1
    mysql_table_engine = mysql_table_engine or "innodb"
end

function thread_init(thread_id)
    set_vars()
end
```

PERCONA

SysBench 1.0+: Variables

```
sysbench.cmdline.options = {
    branches_size = {"Size of branches table", 1},
    tellers_size = {"Size of tellers table", 10},
    accounts_size = {"Size of accounts table", 100000},
    scale = {"Test scale", 1},
    mysql_table_engine = {"MySQL storage engine", "innodb"}
}
```

PERCONA

SysBench 0.5+: Test Preparation

```
function prepare()
    set_vars()
    db_connect()
    db_query([[
create table accounts(
aid int not null primary key,
bid int,
abalance int,
filler char(84)
)
/*! ENGINE = ]] .. mysql_table_engine ..
[[ default character set=latin1 */
]])
    db_bulk_insert_init("insert into branches (bid, bbalance, filler) values")
    for i=1,(branches_size * scale) do
        db_bulk_insert_next("(" .. i .. ", 0, '')")
    end
    db_bulk_insert_done()
end
```


PERCONA

SysBench 0.5+: Test Run

```
function event(thread_id)
    local aid = sb_rand(1, accounts_size * scale)

    db_query("SELECT abalance FROM accounts WHERE aid = " .. aid)
end
```

PERCONA

SysBench 0.5+: Hot Options

`num-threads`	Number of threads to use
`max-requests`	Maximum number of events
`max-time`	Total execution time
`mysql-db`	Schema to run tests in
`mysql-host|user|port|...`	Connection options

PERCONA

SysBench 1.0+: Hot Options

`threads`	Number of threads to use
`events`	Maximum number of events
`time`	Total execution time
`db-ps-mode`	Prepared statements mode
`mysql-db`	Schema to run tests in
`mysql-host|user|port|...`	Connection options

PERCONA

SysBench 1.0+: Major Scripting Improvements

- Scripts can declare options
  - Default values
  - SysBench can validate options
  - Options are visible in `help` output

PERCONA

SysBench 1.0+: Major Scripting Improvements

- Scripts can declare options
- Parallel execution

PERCONA

SysBench 1.0+: Major Scripting Improvements

- Scripts can declare options
- Parallel execution
- Processing result sets

PERCONA

SysBench 1.0+: Major Scripting Improvements

- Scripts can declare options
- Parallel execution
- Processing result sets
- Custom commands

PERCONA

SysBench 1.0+: Major Scripting Improvements

- Scripts can declare options
- Parallel execution
- Processing result sets
- Custom commands
- Better reporting

PERCONA

SysBench 1.0+: Major Scripting Improvements

- Scripts can declare options
- Parallel execution
- Processing result sets
- Custom commands
- Better reporting

sysbench 1.0: teaching an old dog new tricks

PERCONA

Case Study: Slave Crash

The Issue

- Multi-channel slave
- One master runs load on a table
- Another master runs `FLUSH SLOW LOGS`
- Is this `FLUSH` reason for the crash?

PERCONA

Setup

- Slave and two masters

Setup

- Slave and two masters
- Copy options from production

PERCONA

Setup

- Slave and two masters
- Copy options from production
- To imitate load on updating master

```
$ mysqlslap --defaults-file=./.my_updating_master.cnf  --concurrency=50 \
   --number-of-queries=500000000 \
   --pre-query="CREATE TABLE IF NOT EXISTS test.t(ts varchar(26), \
   id int unsigned not null primary key, f1 varchar(255), \
   f2 bigint unsigned, f3 varchar(255), f4 bigint unsigned);" \
   --query="REPLACE INTO t (ts, id, f1, f2, f3, f4) VALUES \
   (now(), 1000*rand(), md5(rand()), 1000000*rand(), 'fixed', 10000*rand())" \
   --create-schema=test
```

PERCONA

Setup

- Slave and two masters
- Copy options from production
- To imitate load on updating master
- Load on flushing master

```
$ while (true); do mysql -e "flush slow logs"; done
```

PERCONA

Setup

- Slave and two masters
- Copy options from production
- To imitate load on updating master
- Load on flushing master
- Standard Sysbench OLTP load on slave

```
$ sysbench --threads=16 --events=0 --time=0 \
   --mysql-host=127.0.0.1 --mysql-port=13002 --mysql-user=root \
   ./oltp_read_only.lua --tables=4 --table_size=10000 prepare
```

PERCONA

Setup

- Slave and two masters
- Copy options from production
- To imitate load on updating master
- Load on flushing master
- Standard Sysbench OLTP load on slave

```
$ sysbench --threads=16 --events=0 --time=0 \
   --mysql-host=127.0.0.1 --mysql-port=13002 --mysql-user=root \
   ./oltp_read_only.lua --tables=4 --table_size=10000 run
```

PERCONA

Setup

- Slave and two masters
- Copy options from production
- To imitate load on updating master
- Load on flushing master
- Standard Sysbench OLTP load on slave
- Slave did not crash
  - I will create better test

PERCONA

Custom SysBench Script on Updating Master

- Options

```
sysbench.cmdline.options = {
    table_size = {"Number of rows per table", 1000},
    tables = {"Number of tables", 1}
}
```

Custom SysBench Script on Updating Master

- Commands

```
sysbench.cmdline.commands = {
    prepare = {cmd_prepare, sysbench.cmdline.PARALLEL_COMMAND},
    run = {cmd_run, sysbench.cmdline.PARALLEL_COMMAND}
}
```

- Must be after function definitions

Custom SysBench Script on Updating Master

- Thread initialization for run command

```
function thread_init()
    drv = sysbench.sql.driver()
    con = drv:connect()
end
```


PERCONA

Custom SysBench Script on Updating Master

- Values templates

```
local ts_value_template = "####-##-#####:##:##.######"
local f1_value_template = "######-###.######"
local f3_value_template = "#####-###.######"
```


PERCONA

Custom SysBench Script on Updating Master

- Prepare command

```
function cmd_prepare()
    local drv = sysbench.sql.driver()
    local con = drv:connect()

    for i = sysbench.tid % sysbench.opt.threads + 1, sysbench.opt.tables, sysbench.opt.threads
    do
        create_table(con, i)
    end
end
```

PERCONA

Custom SysBench Script on Updating Master

- `create_table` definition

```
function create_table(con, table_num)
    local query

     print(string.format("Creating table 'sbtest%d'...", table_num))

    query = string.format([[
CREATE TABLE sbtest%d(
ts varchar(26),
id int unsigned not null primary key,
f1 varchar(255),
f2 bigint unsigned,
f3 varchar(255),
f4 bigint unsigned
) engine=innodb]],
    table_num)

    con:query(query)
```


PERCONA

Custom SysBench Script on Updating Master

- `create_table` definition

```
    if (sysbench.opt.table_size > 0) then
        print(string.format("Inserting %d records into 'sbtest%d'",
                            sysbench.opt.table_size, table_num))
    end
    query = "INSERT INTO sbtest" .. table_num .. "(ts, id, f1, f2, f3, f4) VALUES"
    con:bulk_insert_init(query)
    for i = 1, sysbench.opt.table_size do
        query = string.format("('%s', %d, '%s', %d, '%s', %d)",
                              sysbench.rand.string(ts_value_template), i,
                              sysbench.rand.string(f1_value_template),
                              sb_rand(1, sysbench.opt.table_size),
                              sysbench.rand.string(f3_value_template),
                              sb_rand(1, sysbench.opt.table_size))
    con:bulk_insert_next(query)
    end
    con:bulk_insert_done()
end
```


PERCONA

Custom SysBench Script on Updating Master

- Load test itself

```
function event()
    local table_name = "sbtest" .. sysbench.rand.uniform(1, sysbench.opt.tables)

    local query = string.format("REPLACE INTO %s (ts, id, f1, f2, f3, f4) VALUES " ..
                            "('%s', %d, '%s', %d, '%s', %d)",
                            table_name, sysbench.rand.string(ts_value_template),
                            sb_rand(1, sysbench.opt.table_size),
                            sysbench.rand.string(f1_value_template),
                            sb_rand(1, sysbench.opt.table_size),
                            sysbench.rand.string(f3_value_template),
                            sb_rand(1, sysbench.opt.table_size))

    con:query(query)
end
```

PERCONA

Custom SysBench Script on Updating Master

- Preparing...

```
$ sysbench --threads=16 --events=0 --time=0 \
--mysql-host=127.0.0.1 --mysql-port=13001 --mysql-user=root \
slave_crash.lua --tables=4 --table_size=10000 prepare
WARNING: Both event and time limits are disabled, running an endless test
sysbench 1.0.14-aa52c53 (using bundled LuaJIT 2.1.0-beta2)

Initializing worker threads...

Creating table 'sbtest2'...
Creating table 'sbtest1'...
Creating table 'sbtest4'...
Creating table 'sbtest3'...
Inserting 10000 records into 'sbtest1'
Inserting 10000 records into 'sbtest2'
Inserting 10000 records into 'sbtest4'
Inserting 10000 records into 'sbtest3'
```

PERCONA

Custom SysBench Script on Updating Master

- And executing

```
$ sysbench --threads=16 --events=0 --time=0 \
--mysql-host=127.0.0.1 --mysql-port=13001 --mysql-user=root \
slave_crash.lua --tables=4 --table_size=10000 run
WARNING: Both event and time limits are disabled, running an endless test
sysbench 1.0.14-aa52c53 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 16
Initializing random number generator from current time


Initializing worker threads...

Threads started!
```

PERCONA

Custom SysBench Script on Updating Master

- Now wait ☺

PERCONA

Why Do I Do Load Tests?

Customer's Performance Issues

- An option makes our server X % slower
- Feature is slow
- Query is slow
- Benchmarks techniques

PERCONA

Effect of Options

- Before recommending for customers
- To identify improvements in new versions
- To find out if common guess is working

PERCONA

To Repeat

- Locking issues
- Crashes
- Other concurrency issues

PERCONA

Generate Data

- Built-in data generation functionality
- Scriptable data distribution

```
function set_vars()
    branches_size = branches_size or 1
    tellers_size = tellers_size or 10
    accounts_size = accounts_size or 100000
    scale = scale or 1
    mysql_table_engine = mysql_table_engine or "innodb"
end
```

PERCONA

Generate Data

- Built-in data generation functionality
- Scriptable data distribution

```
function prepare()
    set_vars()
    db_connect()
    ...
    db_bulk_insert_init("insert into branches (bid, bbalance, filler) values")
    for i=1,(branches_size * scale) do
        db_bulk_insert_next("(" .. i .. ", 0, '')")
    end
    db_bulk_insert_done()
    ...
```

PERCONA

Generate Data

- Built-in data generation functionality
- Scriptable data distribution

```
function prepare()
...
db_bulk_insert_init("insert into tellers (tid, bid, tbalance, filler) values")
for i=1,(tellers_size * scale) do
    db_bulk_insert_next("(" .. i .. ", " .. ((i % (branches_size * scale)) + 1) .. ", 0, '')")
end
db_bulk_insert_done()

db_bulk_insert_init("insert into accounts (aid, bid, abalance, filler) values")
for i=1,(accounts_size * scale) do
    db_bulk_insert_next("(" .. i .. ", " .. ((i % (branches_size * scale)) + 1) .. ", 0, '')")
end
db_bulk_insert_done()
...
```

PERCONA

Machines Which I Use

Exclusive Lab Machine

- Ideal solution
- Requirement for true benchmarks
- Performance tests
- Test which require use of all machine resources

PERCONA

Percona Shared Lab Machine

- Tests which needs to be shared
- High resource usage tests
- Long running tests
- Can affect and be affected by tests of others

PERCONA

My Own Laptop

- Short tests
- Not requiring powerful hardware
- Test drafts before porting them to lab machines

PERCONA

Binaries Which I Use

Standard Binaries

- Same as customers use
- With thoroughly tested options
- Optimized for performance

PERCONA

Self-Compiled

- Quick tests
- Special options
- Debug builds for crashes

PERCONA

How Do I Scale

Lack of disk space

- Copy only relevant data
  - Only tables which participate in the problem
  - Part of data if table is huge
    - `LIMIT`
    - `WHERE`
    - `mysqldump --where`
  - Only columns which participate in the query
- Not always leads to issue repetition!

PERCONA

Lack of CPU cores

- Some options must be scaled
  - `innodb_thread_concurrency`
  - `innodb_concurrency_tickets`
    - If you changed `innodb_thread_concurrency`, check the user manual 
  - `innodb_commit_concurrency`
  - `innodb_purge_threads`
  - `innodb_[read|write]_io_threads`
  - `thread_concurrency`
  - `thread_cache_size`
    - Only if you scale number or connections in your test

Lack of CPU cores

- Some options must be scaled
- Use scale coefficient
  - `SCALE = NUMBER_OF_CPU_CORES_ON_PRODUCTION / NUMBER_OF_CPU_CORES_ON_TEST`

PERCONA

Lack of CPU cores

- Some options must be scaled
- Use scale coefficient
  - `SCALE = NUMBER_OF_CPU_CORES_ON_PRODUCTION / NUMBER_OF_CPU_CORES_ON_TEST`
- Example
  - `innodb_thread_concurrency_test = innodb_thread_concurrency / SCALE`
  - Only if greater than number of CPU cores on the test server or unlimited

PERCONA

Scale coefficient example

Table : `innodb_thread_concurrency` scale

Production: 16 cores	Test: 4 cores	Adjust number of client threads?
0	0	It depends
4	4	Not
16	4	Yes
32	8	Yes
128	32	Yes

PERCONA

Disk-related options

- `innodb_io_capacity`
  - Depends on disk speed!
  - Can give very different results if disk speed is not taken in account
- `innodb_io_capacity_max`

Lack of RAM

- Some options must be scaled

Lack of RAM

- Some options must be scaled
- Use scale coefficient
  - `SCALE = RAM_ON_PRODUCTION / RAM_ON_TEST`

PERCONA

Lack of RAM

- Some options must be scaled
- Use scale coefficient
  - `SCALE = RAM_ON_PRODUCTION / RAM_ON_TEST`
- Scale return time for performance tests

PERCONA

Lack of RAM

- Some options must be scaled
- Use scale coefficient
  - `SCALE = RAM_ON_PRODUCTION / RAM_ON_TEST`
- Scale return time for performance tests
- Use less data to repeat production slowdowns
  - Care about data distribution!
  - Check `EXPLAIN` output
  - More waiting time can be better for testing: easier to notice difference

RAM-related options: scale them

- `innodb_buffer_pool_size`
- `innodb_log_file_size`
  - only if changed buffer pool size
- `innodb_buffer_pool_chunk_size`
  - only if very large on production
  - It is unlikely what you will have to tune InnoDB Buffer Pool resizing

PERCONA

RAM-related options: scale them

- `innodb_buffer_pool_size`
- `innodb_log_file_size`
- `innodb_buffer_pool_chunk_size`
- `innodb_buffer_pool_instances`
  - Only if current number of instances does not make sense on test server.
  - **Example:** 96G buffer pool on production with 12 instances and you test on laptop with 8G memory

PERCONA

RAM-related options: scale them

- innodb_buffer_pool_size
- innodb_log_file_size
- innodb_buffer_pool_chunk_size
- innodb_buffer_pool_instances
- query_cache_size, key_buffer_size
  - only if larger than available memory

PERCONA

RAM-related options: scale them

- innodb_buffer_pool_size
- innodb_log_file_size
- innodb_buffer_pool_chunk_size
- innodb_buffer_pool_instances
- query_cache_size, key_buffer_size
- Do not adjust session options

PERCONA

RAM-related options: scale them

- innodb_buffer_pool_size
- innodb_log_file_size
- innodb_buffer_pool_chunk_size
- innodb_buffer_pool_instances
- query_cache_size, key_buffer_size
- Do not adjust session options
- table_open_cache, table_definition_cache
  - Do not adjust

PERCONA

Session options: do not scale

- join_buffer_size
- [max_]binlog_[stmt_]cache_size
- max_heap_table_size
- tmp_table_size
- net_buffer_length
- parser_max_mem_size
- read_[rnd_]buffer_size
- sort_buffer_size

PERCONA

Summary

Tools Described

MySQL Command Line client

- Always

PERCONA

Tools Described

MySQL Command Line client

BASH

- Many queries in single thread

PERCONA

Tools Described

MySQL Command Line client

BASH

mysqlslap

- Simple queries concurrently

PERCONA

Tools Described

MySQL Command Line client

BASH

mysqlslap

SysBench

- Custom tests

Summary

- Benchmarking tools
  - More than just measuring performance
  - Help when more than single query involved
  - Imitate real life workload

PERCONA

Summary

- Benchmarking tools
  - More than just measuring performance
  - Help when more than single query involved
  - Imitate real life workload
- Do much more than described in theirs user manuals if used creatively!

PERCONA

Call for Questions!

Future webinar with MySQL benchmarking experts

Send your questions

https://twitter.com/svetsmirnova/status/999066023755223040

PERCONA

Thank you!

http://www.slideshare.net/SvetaSmirnova

https://twitter.com/svetsmirnova

PERCONA