Introduction to troubleshooting How to create test setup

Sveta Smirnova
Principal Support Engineer

March, 2, 2016

Table of Contents

Before troubleshooting begins

Importance of tests

- When something goes wrong it is often not clear what causes the issue
- Best way to clarify it is to repeat the problem
  - You can set breakpoints
  - You can experiment with input
  - You can rewrite query or code

Can we ignore tests?

- Experienced users sometimes can know what leads to one or another issue
  - Because they repeated it in their environments dozens of times!
- Not all issues can be solved without testing
  - Brand new bugs with new features
  - Issues, badly affected by data
  - Most of issues!
- I myself test something new everyday

Troubleshooting webinars

- In future webinars I would often ask you to repeat something
- In all such cases I would expect you have properly set test environment
- Tests on production are dangerous
  - Big data uses a lot of resources, required by your application
  - Repeating crashing issues can kill production server

Troubleshooting webinars

- In future webinars I would often ask you to repeat something
- In all such cases I would expect you have properly set test environment
- Tests on production are dangerous
- How to create test environment more effectively?
    - This is the topic of our today webinar.

Test setup

First considerations

Why don't use development server?

- When working on an application we usually first develop it on our own machine
- In case of database application we also usually have small database with test data

Table : User

Id	Name
1	test
2	Foo
...	
100000	Bar

Production database

- Usually has much more data

Table : User

Id	Name
100001	John
100002	Ignacio
100003	Sveta
...	
99999999999999999999999999	Brock

Production database

- Usually has much more data
- Data is not ordered and depends from random factors
- Can have extremely powerful hardware which costs too much to be duplicated on development server

Simplest setup

Ideal case

- If you can
  - Use same hardware as on production server for your test setup
  - Can copy data without modifications
  - Can make full binary backup of production server

Ideal case

- If you can
- Install same version as on production
  - Just install same package
  - Use MySQL Sandbox
    - On UNIX, Linux, Mac OS X

```
perl Makefile.PL
make
[make test] - optionally
sudo make install
make_sandbox your_MySQL/Percona_server.tar.gz
make_replication_sandbox your_MySQL/Percona_server.tar.gz
```

Ideal case

- If you can
- Install same version as on production
- Make backup and restore
    - Make full binary backup of your production server
        - innobackupex /data/backups
        - cp -R /path/to/datadir /data/backups/datadir
    - Restore it on the test server
    - Copy option file from production to test server
    - Start and use test server

We live in not ideal world

- You can have less power server for tests
  - Less space
  - Smaller number of CPU
  - Smaller amount of RAM
  - Have one test server and master-slave issue on production
  - Have other limitations
- There are ways to workaround these limitations

Still we need to have in mind the ideal

- MySQL/Percona/MariaDB Server must use same version as production
  - There is no excuse to test on different version!
- Same options
  - Not always possible
- Same data
  - Not always possible

Two kinds of issues with databases

Two groups of database issues

- Wrong behavior
  - You get results which you don’t expect
  - Wrong rows updated
  - Database crashes
  - Slave has dataset different from master’s
  - Etc.
- Performance issues

Two groups of database issues

- Wrong behavior
- Performance issues
  - Query runs slowly
  - Overall server performance is slower than you need
  - Queries hang up from time to time
  - Etc.

Two groups of database issues

- Wrong behavior
  - Most, but not all, issues can be repeated on development server
- Performance issues
  - Most issues can not be repeated on development server

Avoiding limitations

Lack of disk space

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

Sensitive data

- Copy only columns which do not have sensitive data
  - Example
    - Id, dates, public article content - all these are either generic or public already
    - Email, password, names - these can be sensitive
- Encrypt sensitive data
  - Non-symmetric encryption functions, such as md5, sha
  - Use substr and concat to have values of length, similar to data you have
- Replace sensitive data with random values

Lack of CPU cores

- Some options must be scaled
  - innodb_thread_concurrency
  - innodb_concurrency_tickets
    - If you changed innodb_thread_concurrency, check user manual
  - innodb_commit_concurrency
  - innodb_purge_threads
  - innodb_read_io_threads
  - innodb_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
  - innodb_thread_concurrency_test = innodb_thread_concurrency_production / SCALE
  - Makes sense only if such options are greater than number of CPU cores on the test server or unlimited

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

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
- Use scale coefficient: SCALE = RAM_ON_PRODUCTION / RAM_ON_TEST
- Scale expected return time for performance tests as well
- You can use less data to repeat similar slowdown as on production
  - 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, usually not needed
    - It is unlikely what you will have to tune InnoDB Buffer Pool resizing

RAM-related options: scale them

- innodb_buffer_pool_size
- innodb_log_file_size
- innodb_buffer_pool_chunk_size
- innodb_buffer_pool_instances
  - only if InnoDB Buffer Pool is very large and 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

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
- Do not adjust session and operation-specific options unless they set to very large values

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 and operation-specific options unless they set to very large values
- table_open_cache, table_definition_cache
    - Do not adjust

Session options: do not scale

- join_buffer_size
- [max_]binlog_[stmt_]cache_size
- max_heap_table_size
- tmp_table_size
- myisam_*
- net_buffer_length
- parser_max_mem_size
- read_buffer_size
- read_rnd_buffer_size
- sort_buffer_size

If you have to test on production

- Do not test crashes and issues which lead to full server hang!
- Create copy of database
  - CREATE DATABASE test_copy;
  - Copy all involved objects (tables, views and so on) into new database
  - Use any backup-restore technique
    - mysqldump test |mysql test_copy
    - mysqldbcopy –source=root:@127.0.0.1:13000 –destination=root:@127.0.0.1:13000 test:test_copy –multiprocess=4
    - Your favorite method

If you have to test on production

- Do not test crashes and issues which lead to full server hang!
- Create copy of database
- Create copy of single table

```
CREATE TABLE test_copy LIKE test;
INSERT INTO test_copy SELECT * FROM test;
INSERT INTO test_copy SELECT * FROM test WHERE condition;
-- Useful when your query returns wrong results
INSERT INTO test_copy SELECT * FROM test LIMIT 1000000;
-- Speeds up performance tests on big data
```

Any backup-restore technique which can copy-restore single table

Summary

Summary

- There is no excuse to test on different version!
- 100 % copy of data and options is ideal
- Ideal does not always happen in real life
- Care about
  - Data
  - Options
- Use test environment for experiments

References

- MySQL Sandbox
- Percona XtraBackup
- mysqldump
- mysqldbcopy
- MySQL option tables

Place for your questions

???

Thank you!

http://www.slideshare.net/SvetaSmirnova

https://twitter.com/svetsmirnova