MySQL Upgrade

Best Practices

Peter Zaitsev, CEO
Percona Inc
April 18, 2012

MySQL Upgrade: Agenda

- Look at Different types of Upgrade
- Discuss when to upgrade and not
- Issues you can discover during Upgrade
- Look through the process you can use for upgrade

What is your Experience ?

- How many have done MySQL Upgrades ?
- How many of you have done 1 major version transition ?
  - Two ? More ?
- Did you get any “surprises” during upgrade ?
- Did you ever have to roll back ?

Upgrade as Version Changes

- Issues are similar for upgrade and over version changes
- Moving between MySQL, Percona Server, MariaDB, proprietary solution ?
- Downgrading ?

Reasons to Upgrade

- Bugs in old versions
- Security problems in old version
- New Version benefits
  - Features, scalability, performance, new bugs
- Hard to find people familiar with old versions
  - Anyone remembers MySQL 3.23 restrictions ?
- Support Becomes limited

Case to stay on old MySQL version

- Application is on isolated network
- Application is not being actively developed
- Application is not growing actively
- No changes in Platform
  - Hardware or Operating system

Which Upgrades are Risky ?

- Major upgrades, such as 5.1 to 5.5 are riskier
- MySQL to Percona Server or MariaDB
- Jumping over many minor versions at once
  - 5.1.20 to 5.1.61
- Upgrades from PreGA versions
- Skipping one or more major releases
  - 4.1 to 5.5 is tested less than 5.1 to 5.5

Typical reasons for MySQL Upgrade project

- Running into bug or performance issue
- Concern with bug which can potentially affect system
  - Checking change logs for relevant bugs
- Security Concerns
- Generally keeping up with fresh versions

Issues to consider with Upgrade

- Data
- Queries
  - Reads and Writes
- Performance and Scalability
- Replication
- Resource Usage
- Advanced Features

MySQL Upgrade Data Issues

- On Disk format changes
- Changes to MySQL Types
- Sorting order Changes
- Data Presentation Issues
  - TIMESTAMP text format changed in
- Changes to limits
- Reserved Words
- Statistics

Issues with Queries

- Syntax might have changed
- Query has different result
  - Changes in query “meaning”
  - Non deterministic queries
- Query Produces warnings or Error

Performance

- Query Execution Time
- Query Plan Changes
- Performance at Concurrency
  - Scalability
  - Locking

Replication

- Can workload be still replicated
  - And are there any warnings in error logs ?
  - Any “data drift” causing inconsistent slave
- Is replication performance adequate ?

Resource Usage

- Memory is main resource to care
- Check memory usage for new version
  - Can be memory leaks or simply more memory needed

Advanced Features

- Complex Features = More things to break
- Pay special attention to
  - Stored Procedures/Functions
  - Plugins
  - Triggers
  - Events
  - Views

Upgrading Multiple Envinronments

- Development, Staging, Production
- Do we upgrade Development first or last ?
  - Both have their problems
- Creating special Upgraded Development environment is best

Drop In or Replication Based Upgrades

- Drop-In for small systems
  - Which can handle downtime or risks
  - LVM snapshot great for quick rollback
    - http://bit.ly/yQwm2v
- Drop in for “shards”
  - When in place upgrade is well tested
- Replication based upgrades are best default

Upgrading in the Cloud

- Great Possibilities !
- Inexpensive way to temporary increase your infrastructure size
  - Can build a “clone” on updated version and test throughly

Many Changes at Once

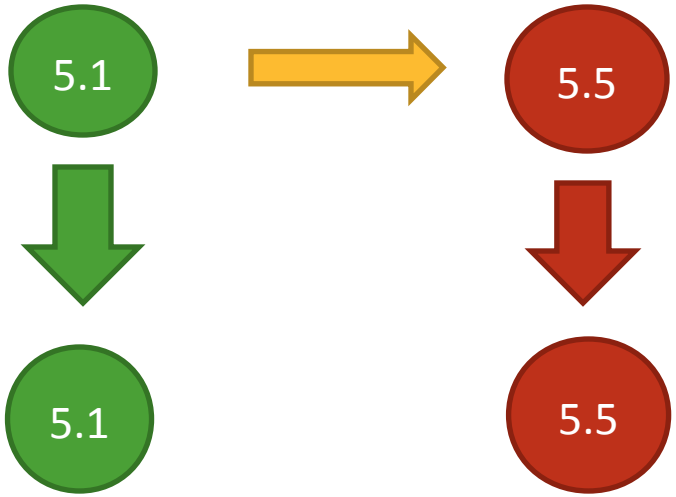
- Are you upgrading Hardware, OS, Application, Changing Configuration at the same time ?
- Good
  - Usually can invest more time in testing
- Bad
  - If something breaks you would not be sure what to blame.
  - Many more moving parts. Riskier.

Reckless or Paranoid

- For some upgrade is as simple as “yum update”
- Others take months to upgrade
- A lot depends on application complexity and your risk tolerance
  - We’ll look at reasonable conservative approach in this talk.

Upgrading Replication

- Upgrade Slaves first
  - Upgrade Inactive Master In Master-Master Pair
- Setup extra Slaves or “Replication Tree if possible”



Upgrading Sharded Environment

- You can take shortcuts
- No full testing for each shard might be needed
- Upgrade one shard, create “script” and monitor
- Update couple of more shards with full testing
- Update over shards with reduced testing

Doing an Upgrade

MySQL Upgrade Process in Details

Read the Release Notes

- Understanding changes helps you to know what to expect
- Major Releases Come with great Summary documentation
  - http://dev.mysql.com/doc/refman/5.5/en/mysql-nutshell.htm

Perform Preliminary testing

- Setup QA environment with new release
- See if it works
  - Helps to understand amount of effort needed for upgrade

Adjust MySQL Configuration

- Focus on limited changes, you can fine tune later.
  - remove depreciated options
  - Adjust defaults you might relay on
    - **storage_engine=MYISAM**
  - Set options which is hard to change later
    - **innodb_file_format=Barracuda**
  - New compatibility requirements
    - READ-COMITTED does not work with Statement replication in MySQL 5.5

Moving The Data

- Mysqldump and import back
  - The most clean way to move data, but slowest
  - Helpful for upgrading very old databases or when you want to move to new features
- Mysql_upgrade
  - Will check tables for compatibility and “fix” incompatible tables
  - Run this in any case as it upgrade system tables

Check the data is the same

- In both cases data may start to “look” differently
- Check data is the same in both versions
  - Compare restored backup to upgraded restored backup
  - CHECKSUM TABLE may report false positives
  - Pt-table-checksum
  - Mk-table-checksum can compare standalone servers.

Replication Old->New

- You will run this some way if you do not schedule downtime
  - Let replication run for 24 hours or so
  - Check it is in sync with pt-table-checksum
  - Check error logs on the Slave for errors or warnings

Replication New->Old

- Needs to be tested too if this is roll back strategy
  - Old->New->Old chain good for testing
    - Does not 100% represent production use case
  - Run New->Old in QA to be sure
- Unsupported but works in most cases
  - MySQL 5.0-> MySQL 5.5 works in most cases.

Replication New->New

- Is not expected to give problems if previous cases work
- Might wish to check it for extra safety
- Often validated post factum after upgrade
  - As normal replication consistency check practice

Validate Replication Performance

- You want to ensure replication is not slower in new version
- Measure catch up speed in All Versions
- See replication thread utilization in Percona Server or MariaDB

Getting Queries for Test

- Get representative live traffic from production
  - Full query log/ slow query log with long_query_time=0
  - Tcpdump (pt-query-digest)
- Get several samples for each query
  - **pt-query-digest --sample 5 --print --no-report queries.log > samples.log**

Checking Queries

- Single connection test
  - Run pt-upgrade on sample of queries
- High concurrency test
  - Run pt-log-player with full traffic
  - Use 2 sets; one for warmup over for real run
- Looking at pt-query-digest report on full query log is good to check in both cases
  - Outliers; number of examined rows etc.

Do you have load testing setup ?

- Full application load testing is great final step
- Few companies have it setup well

Consider Getting Help

- Upgrades is something you do not get a lot of training working on single applications
- Common upgrade issues are different all the time 4.0->4.1 and 5.1->5.5 are very different
- We at Percona can help !

Thank You !

- pz@percona.com
- http://www.percona.com
- @percona
  - Use #perconalive
- http://www.facebook.com/Percona
- Slides will be published on the conference site