



PERCONA
Performance Consulting Experts

Upgrading MySQL Best Practices

Apr 11-14, 2011

MySQL Conference and Expo

Santa Clara, CA

by Peter Zaitsev, Percona Inc

MySQL Upgrade

- How many of you have performed MySQL upgrade ?
- How many of you have done 1 major version transition ?
 - Two ? More ?
- How many of you have “surprises” during minor upgrades ?
 - Major upgrades ?

Why Upgrade ?

- Old MySQL Versions have bugs
- Old MySQL Versions have Security Issues
- New MySQL Versions have more features, scalability, performance
 - And new bugs added by your friends at development team
- Hard to find talent familiar with legacy versions
 - Is it harder to find COBOL or PHP developer ?
- Support becomes limited
 - Issues your run into may not be fixed in the same major release

Why NOT to Upgrade ?

- Every upgrade has risks
 - Do not fix what is not broken ?
- Major upgrades (such as 5.0 to 5.1) has more risks
- Jumping over many minor versions have more risks
 - 5.1.20->5.1.55 more risky than 5.1.52->5.1.55
- Upgrades from Pre-GA version are more risky
 - As they tend to have more aggressive changes
- Skipping major MySQL Version is risky
 - 4.1->5.1 is not tested as well as 5.0->5.1
 - May be good idea still with good testing

Reasons for Upgrade

- Running into current bug or performance issue
- Concern with bug which can potentially affect system
 - Bug mentioned in changelog which looks relevant
- Security Concern
- Moving away from very old version

How frequently to upgrade ?

- Depends on the complexity of upgrade process
 - Some companies need man years worth of effort just to go from one minor release to another
- When serious issues are discovered in your release
- Hard to name exact “frequency” in time
- Stay with current or previous GA version. Now MySQL 5.5 is out it is time to consider upgrade
 - Even if you do not have many problems with it.
- Note: there are people still running MySQL 3.23 out there which are just doing fine.

Minor upgrade vs Major upgrade

- Minor Upgrade
 - Upgrading to same major MySQL version with small minor number change 5.1.53->5.1.55
 - Relatively low risk
- Major upgrade
 - Upgrading to different major MySQL Version, or large minor number change (over 1 year between releases)
 - 5.1->5.5 or 5.1.20->5.1.55
 - Higher risk due to higher amount of changes
 - Upgrades to MySQL Forks such as Percona Server or MariaDB should be considered Major upgrades

Issues to Consider

- On Disk format changes
- Query Syntax and Result format
- Query Result
- Query Performance
- Ability to Store given data
- Scalability
- Resource Usage
- Replication

On Disk Format Changes

- MySQL has pretty good track record for on disk format support for MyISAM and Innodb
- Number of “edge case” changes over the years
 - Many related to fixing “sort order” for edge cases
- New disk format may be available in new version
- Solutions
 - Mysqldump and reload best for small databases
 - Can be combined with Replication (as explained later)
 - mysql_upgrade
 - Tool to identify most of potentially incompatible tables and fix them
 - Manual ALTER TABLE required for Innodb Tables

Query Syntax and Result Format

- Reserved keywords are added in new versions
- Parser changes may make it more strict
- MySQL 5.0 – changing how JOIN conditions are interpreted broke a lot of queries
- MySQL 4.1 - new TIMESTAMP format required changing a lot of applications
- No such massively problematic issues after MySQL 5.0
 - Still some applications require changes on upgrade

Query Result

- Queries may start returning different result
- Query “Meaning” may become different
- Non Deterministic queries may be executed differently
- Can be caused by changes in sort order, comparison etc

Query Performance

- Most Common Upgrade Regression Issue
- Can be caused by
 - Execution plan changes
 - Changes in the software
 - Many improvements come at cost
 - There may be bugs/unintended consequences

Solving these problems

- Mk-upgrade to check query result set, execution plan and performance
 - <http://www.maatkit.org/doc/mk-upgrade.html>
- Set up 2 servers with old and new version
- Get the relevant query sample
 - Query log or tcpdump
- Use the tool to run them on 2 servers and watch for differences

Ability to Store given Data

- Yes... your data may no longer be stored in the database
- Sometimes it is bug fixes
 - Storing 0 in AUTO_INCREMENT column
- It can also be storage engine changes
 - InnoDB Plugin has different restrictions than InnoDB built in MySQL 5.1
- Other changes done during upgrade
 - Such as character set may affect it

Scalability

- Problems with concurrent workload execution
 - Can't spot this with mk-upgrade
 - But benchmark with mk-log-player can help
- Relatively rate case, commonly scalability gets better
MySQL 5.0->5.1->5.1plugin->5.5
 - But there are reported edge cases

Replication

- Replication during upgrade
 - You may need cross version replication during upgrade
- Replication working right in version you're upgrading to
 - Performance
 - Consistency
- Mk-table-checksum can be used to check replication for consistency
- Remember: Slaves should be upgraded first !

Doing many changes at once

- Changing hardware
- Moving to different storage engine
- Changing character set
- Major configuration changes
- Application architecture changes
- Benefit
 - Just need to test everything once
- Drawbacks
 - Do not know exactly where gains/losses come from
 - Many moving parts, risk if something goes wrong

Reckless or Paranoid ?

- There is wide range of Reckless to Paranoid upgrade processes
- You should ask yourself
 - How many times have you done upgrade ?
 - Did you ever encourage any problems in this type of upgrade ?
 - How complicated your application ?
 - How much are you concerned about downtime or data corruption ?

Extra Resources during upgrade

- Do you have extra servers during upgrade ?
 - Having them makes it easier, safer, less downtime
 - This is where Cloud is really helpful
- Using Replication as part of upgrade process is helpful even for single server

General Upgrade Process

- Take a backup (as with any significant change)
- Setup the slave (S1) which will be target for upgrade
 - With target MySQL version
- Setup slave S2 which with old MySQL data and same data as S1
- Do **mysql_upgrade** on S1
 - Or **mysqlcheck -A --check-upgrade**
 - Alternatively do mysqldump and reload on S1
 - Do mysqldump before upgrading MySQL server version
- Check S1=S2
 - Loaded/Altered data may not be the same

General Upgrade Process 2

- Check S1=S2
 - **mk-table-checksum s1 s2 | mk-checksum-filter**
 - May need to use `–algorithm=BIT_XOR`
 - Checksum table can give false positives sometimes
- Run `mk-upgrade` comparing S1 to S2
 - Check for read queries regressions, wrong results, different plans
- Setup replication M->S1, M->S2 let it run for 24h+
 - Run **mk-table-checkum –replicate**
 - Check cross-version replication is working fine
 - Also checks write queries work the same on both versions

General Upgrade Process 3

- Validate replication performance
- Enable **log-slow-slave-statements=1** for statement based replication
 - Set `long_query_time=0`
 - Run `mk-query-digest` on S1 and S2 slow query log
 - You should not see highly different time taken by same statement
- **Alternatively**
 - Stop replication on S1 and S2 at the same position,
 - Wait couple of hours
 - Start replication and time how long it takes to catch up

General Upgrade 4

- Setup S2 as slave off S1
- Repeat replication consistency check
 - Ensure same version replication works fine
 - Especially makes sense if you're switching to ROW level replication as part of upgrade.
- Do Stress Test on S2
 - We will not need that database any more so you can change its data
 - Mk-log-player can be helpful
 - Or use your own stress tool

General Upgrade Process 5

- Check Replication works in reverse
 - Load old MySQL version & data on S2
 - Check replication S1->S2 is working or consistent
- You may have to run STATEMENT based replication until you're ready to give up quick roll back to 5.0
- It is not an option at all for some MySQL version upgrades, and it is not officially supported.

General Upgrade 6

- Promote S1 to the master and move traffic to it
 - Depends on your replication topology a lot
 - MMM (<http://mysql-mmm.org/>) can be great tool to assist
 - Keep S2 with old version as it slave
- Validate your application is working with new master

Upgrade Tips

- If upgrading in sharded environment you can do full testing only on one/few shards
- If you have master+many slaves
 - Upgrade one slave. Validate its working fine
 - Upgrade all slaves
 - Upgrade master last
 - Note master often have different kind of read queries
- Always have rollback plan
- Test Schema migration on smaller schema
- Test all process on development environment

Upgrading Environments

- What If you have Development, Staging, Production ?
- Consider separate Dev environment for upgrade
 - Developing on different version can be dangerous
- Keep Staging close to production
- Minimize time when environments have different versions
 - Most problems appear if development is forgotten to be upgraded for years or runs much newer version.

Who is helping you ?

- Upgrades are different
 - MySQL 4.0 → 4.1 upgrade had different set of issues than 5.0 → 5.1
 - Team often has limited experience doing same upgrade
 - External help and advice is often an option
 - We at Percona can help

The End

- Thank you !
- Send your feedback to pz@percona.com
- Percona Does MySQL Support, Consulting, Training
- We're creators of

