PERCONA
Performance Consulting Experts

Switching to Innodb from MyISAM

Matt Yonkovit
Percona

DIAMOND SPONSORSHIPS

THANK YOU
TO OUR
DIAMOND SPONSORS

Who We Are

- Who I am
  - Matt Yonkovit – Principal Architect
  - Veteran of MySQL/SUN/Percona
  - 13 Years
- Who we are
  - Percona is a top MySQL consulting, support, training, and development firm
  - Creator of XtraDB and XtraBackup

What We Do

- Consulting
  - Architecture and Design
  - Performance Tuning
  - High Availability
  - Custom Builds
  - Troubleshooting
- Support
- Training
- Development
- Benchmarking

PERCONA
Performance Consulting Experts

Who we do it for

Disclaimer

- Goal is to walk you through the general differences that can cause issues with a migration
- This is not a presentation on Innodb or XtraDB.
  - Consider training or read supplemental materials for details on Innodb
- I assume the audience has some knowledge of MySQL and storage engines in general

MyISAM

- Long history
  - Been around since MySQL 3.x
  - Many applications still have optimizations for MyISAM
- Simple Storage Engine without a lot of complex moving parts
  - Is less better or worse?
  - There is a lot less configuration options to tweak performance
- Built in Support for Full-Text Indexes
- Compression of data

Issues With MyISAM

- Does Table Level Locking
- Not Crash Safe (Not ACID Compliant)
  - Who likes losing data
- Only index pages are in the Key Buffer
  - Non-indexed columns read from disk or filesystem cache
- You have to be aware of certain maintenance that should take place regularly
  - Optimize, analyze, check, etc.
- Limited options for instrumentation and tweaking
- Limited development
  - Maria/Aria was going to be the replacement

Enter Innodb

- Innodb is an ACID Compliant Storage Engine for MySQL.
  - As a storage engine it works as a drop in replacement for other storage engines like MyISAM (No SQL CHANGES! Yeah!)
- Added to 3.23 max & MySQL 4.x around 2001,
-

Innodb

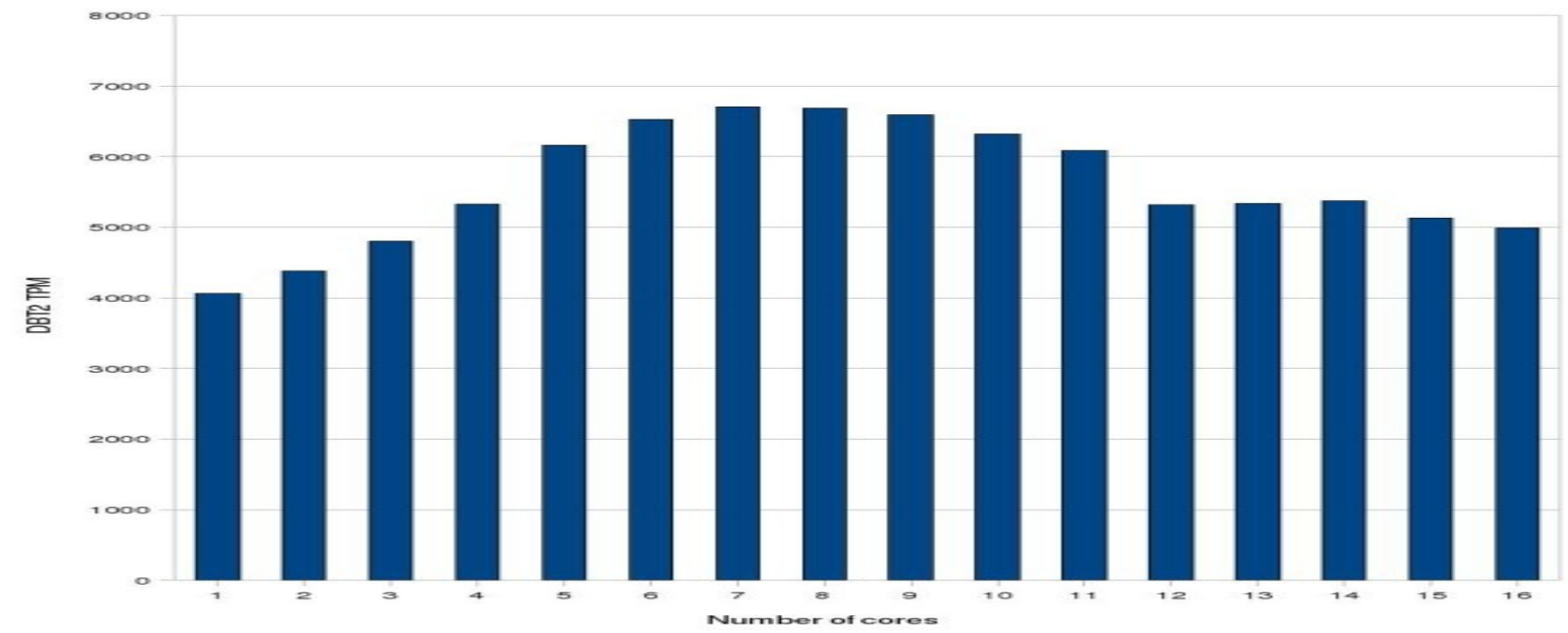
- Row Level Locking
- Crash Recovery
- Both Data Pages and Indexes can be stored in memory
- Foreign Key Support
- Automated maintenance tasks (i.e. building stats)

Innodb History & Problems

- Innodb was originally written in the mid to late 90's, so much of the code was optimized for machines with limited horsepower
  - Comments in the code referred to 100 disk io limit of current hardware
  - Multi-core machines did not exist, so lots of optimization for 1 or 2 CPU Cores
- As commodity hardware costs declined, these assumptions were pushed and invalidated

Not so distant past:More is Less

- Changes were made to increase scalability, but problems still persisted, and regressions happened



PERCONA
Performance Consulting Experts

Community Responds

- Google Patches
- Percona Patches
- Innodb Plugin
- XtraDB

Development work on innodb

- Majority of development being done around innodb, the performance benefits have been substantial.
- Updated several times a year, (10 releases since 2008)
- Makes sense for this to be the default storage engine going forward

Default = Innodb

- Coming in MySQL 5.5, Innodb will become the default storage engine in MySQL.
  - This means that people not specifying the engine on create will start getting Innodb based tables, where previously they had gotten MyISAM
  - If these people are not paying attention performance is going to suffer as the database is more then likely not designed and optimized for both storage engines.
  - This does not mean you have to move from MyISAM

What tables are you using?

- There are several ways to check to see what storage engine you are using including:
  - Show create table 'xxxx';
  - Show table status;
  - select TABLE_SCHEMA,TABLE_NAME,ENGINE from information_schema.tables;
- If your already all Innodb based, your in good shape for the change.
- What if you find some tables that are accidently MyISAM?
  - --default-storage-engine

Migrate storage engines

Lets Assume Your Ready to Migrate to Innodb

Preparing for the change

- Treat this like any other upgrade
- Plan lots of testing
  - Regression testing
  - Benchmarking
  - Unit Tests
- Check the latest readme and storage engine specific information

Feature Gotcha's

- Full Text Indexes
- Compact/Compress
  - Innodb has the ability to compress data automatically
- Multi-column Auto-Inc PK's
- GIS/Spartial Indexes
- Merge Tables
- Segmented Key Caches
- Deal With Deadlocks
- Locking

Full Text Indexes

- Innodb does not support full text indexes
  - Check for full-text indexes before you begin
- This is one of the big reasons people continue to use MyISAM in some limited capacity
  - If you do this make sure you keep enough memory allocated to MyISAM
- Alternative is you could consider switching to sphinx, lucene, or some other alternative
  - Sphinx has a MySQL storage engine
  - Doing full text searches outside the DB can scale better

Table Locking Be Gone!

- One of the key benefits people look for when migrating is Innodb's locking mechanism
  - Instead of doing table level locking, row level locking is used
  - Locking is much more in align with other RDBM's on the market
- If you issue “lock table” statements, innodb will gladly lock the table

Locking Exceptions

- Older versions of Innodb, and newer versions with innodb_autoinc_lock_mode = 0 will perform a low level lock on the table in order to grab the next auto-inc key when needed
- “Insert into select * from”, load data, bulk inserts, etc will still lock the table!
  - Select into outfile is does not lock
  - Note that some of these locking issues are being fixed with row-based replication
- Be Careful: In MyISAM autocommit = 0 means nothing, In Inndob you may never commit... very evil

What if you need

- If you need a MyISAM feature for a table or two you can have both
  - Later we will talk about the disadvantages to this
- Consider running Innodb on the Master, MyISAM on the slave
- Look to alternatives for features
  - i.e. sphinx for full text search

Benchmarking

- Capture a query stream for some amount of time and replay it against a database that has MyISAM and one that has Innodb and compare the results.
  - Look for regressions
  - Look for performance boosts
  - Mk-upgrade or Mk-query-digest are your friends

Do you need more hardware?

- Innodb stores its data completely differently then MyISAM, as a result its footprint on disk can be larger. Do you have enough space to handle this?
- Additionally the extra space translates into extra memory usage

Quick Tip: Indexing

- One of the largest contributors to excessive space usage is Innodb is a large primary key.
  - Innodb uses the Primary Key as a clustered index, keeping all data sorted by the Primary Key on disk. Additionally the primary key is copied into every secondary index to be used as a pointer back to the row on disk. Here size matters!
- Sometimes converting large multi-key primary keys into a single auto-inc field can save a lot of space and boost performance
- Also note, because the data is stored in PK order on disk, this can allow you to boost performance
  - Example: If you always search based on date, including date as the first column in the PK can boost performance

Space Consideration

- Why Does Innodb appear to take more space?
  - Indexes contain the PK
  - 768 Byte Prefix for externally stored large columns
  - Transaction details
  - Undo Space
  - Row Versions
  - Pages are not filled to 100%

New Set of my.cnf params

- MyISAM and Innodb Don't share the exact same set of parameters
  - Key Buffer is the most important MyISAM parameter, this should be reduced (but can not be eliminated completely) when you move fully to innodb
  - The Innodb_buffer_pool_size is the the closest thing to an equvilant
    - Key difference here is the innodb buffer pool contains data and index pages, while the key buffer contains only index pages
    - Innodb buffer pool is also used for other internal operations
      - i.e. insert buffer

Quick Tips!

- While it may seem subtle, the difference between the key buffer and Innodb buffer pool can make a huge performance difference.
  - Double edge sword, memory in MyISAM is Index only... With Innodb its Indexes + data...
  - Less index pages in memory as its shared
- Don't over-allocate memory!

Other Key Parameters

- innodb_flush_log_at_trx_commit
- innodb_flush_method
- innodb_log_file_size
- innodb_thread_concurrency

Pro Tip

innodb_flush_log_at_trx_commit = 2

Innodb Plugin & Xtradb

- innodb_adaptive_flushing
- innodb_io_capacity
- innodb_read_ahead
- innodb_read_io_threads
- innodb_write_io_threads

How are you going to migrate?

- Could be as simple as “alter table XXX engine = innodb”... or a simple dump and reload.. but it could be much more complex every situation is unique
- Size makes a huge difference
  - Alter table engine= will cause locking, so plan for application downtime, as the data size is larger the time require gets longer

TIP: Use replication

- Replication is a great way to stage your changes without massive amounts of downtime on a large database.
  - Alter tables on the slaves to change the engine to be innodb
  - Run your tests against the slave to verify all is working
  - Plan to promote the master to be a slave

Stats Gathering

- With MyIsam the optimizer benefited from updating your statistics with the analyze command, this does not work the same with innodb
  - Stats are gathered behind the scenes when certain thresholds are reached
  - Larger datasets can cause issues
    - Helpful sometimes to use: innodb_stats_auto_update and innodb_stats_sample_pages

Backups

- No Longer forced to do mysqldump or cold backups
- Mysqldump
  - use --single-transaction
- Hot Backups Can be done with Innodb using SAN Snapshots, LVM Snapshots, or XtraBackup
  - Note this can still cause resource contention (IO) if run on your production server
  - MyISAM Can use these but requires table locking

Recovery

- One of the big benefits of Innodb is better recoverability and durability
  - This does not mean data will not become corrupt
  - This does not also mean you will never lose data
    - innodb_flush_log_at_trx_commit
    - Replication issues
    - Bugs and Errors
- Corruption in MyISAM is bad, as the data lost can have no pattern to it.

Performance

- Select count(*) on the full table in innodb is much slower then in MyISAM
- In Innodb,PK Order can make operations faster or slower
- Realize Innodb will require more tuning to achieve optimal performance then MyISAM

Caution

- When converting from MyISAM beware of the changing your bottleneck
- EXAMPLE
  - Locking bottleneck freed, increased cpu because more threads active
  - System stats are good, response time is better
  - IO can increase!
  - Some apps optimized for MyISAM

Pro Tip

- Absolutes are difficult in the performance world, in many causes the answer is: It Depends!
- Don't believe everything you hear, for instance:
- “MyISAM is faster on Reads”

More then Just the Database

- Your durability and recoverability is only as good as your infrastructure
  - Using Raid?
  - Battery Backed Cache on your Raid Card?
  - Journaling File-system?

Mix Innodb & MyISAM

- Joining the two table types in one statement will cause innodb and MyISAM will revert to the lowest set of features
  - Table Locking
  - No Transactions
  - Etc.
- Each uses separate memory pools that can not share
  - Split resources

Percona Training

- Looking for help with understanding how Innodb works? Percona offers 1 day and multi day training classes

Percona Consulting

- Percona offers a full range of support and consulting services
- We can help you asses and execute a migration from MyISAM to Innodb

PERCONA
Performance Consulting Experts