MyRocks in the Wild Wild West!
Alternate Storage Engine for MySQL

Alkin Tezuysal
Sr. Technical Manager

FOSDEM – Feb 1st 2020
Who are we?

@ask_dba - Alkin Tezuysal

Born to Sail, Forced to Work

❖ Open Source Database Evangelist
❖ Global Database Operations Expert
❖ Story Teller
❖ Inspiring Technical and Strategic Leader
❖ Creative Team Builder
❖ Speaker, Mentor, and Coach
Agenda

• Intro and basics
• Advanced internals and limitations
• Benchmarks
• Tuning suggestions
• Conclusion
Overview of MyRocks

❖ What’s MyRocks?
- Storage engine for MySQL
- Based on RocksDB, a fork of LevelDB
- Persistent key-value store
- Implemented at Facebook and introduced in 2016
- Used by FB in production
- Was only available as source code at first
Overview of MyRocks

❖ What’s MyRocks?
  ● Percona Server:
    ○ Announced for Q1 2017
    ○ Fully supported: 5.7.20, 8.0
  ● MariaDB:
    ○ Plugin alpha since 10.2.5
    ○ Stable since 10.3.7/10.2.16
  ● Getting more mature
  ● Not widely used
Overview of MyRocks

- Based on LSM tree
- Optimized for writes
- Space-efficient
- Fast data load (with correct setup)
- Fast read-free replication
- No foreign keys, no serializable
- No Full Text or Spatial keys
- MyRocks has TTL for data
LSM vs B-tree

Image credit: b+tree lsm
http://www.benstopford.com/2015/02/14/log-structured-merge-trees/
# LSM vs B-tree

<table>
<thead>
<tr>
<th>LSM: write-optimized</th>
<th>B-tree: read-optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential writes first</td>
<td>In-place</td>
</tr>
<tr>
<td>Compaction in background</td>
<td>Live tree re-balancing</td>
</tr>
<tr>
<td>Fast access only to leaves in the fast levels: memory, L0</td>
<td>Fast access to all leaves</td>
</tr>
</tbody>
</table>
Innodb vs MyRocks

❖ MyRocks: better writes
❖ MyRocks: 2-5x less size than InnoDB
❖ InnoDB supports FKs and Serializable
❖ InnoDB supports XA
❖ Handle locking differently
Innodb vs MyRocks

❖ InnoDB can be used with advanced replication: Galera, Percona Xtradb Cluster, Group Replication
❖ InnoDB supports STATEMENT and MIXED binlog format
❖ MyRocks doesn’t support transactions larger than available memory
Why use MyRocks engine?

- Large datasets
  - Larger than memory available
    - 100G is not that large
  - Multiple indexes
- Write-intensive load
- Mostly point selects *(it’s complicated)*
- No FKs/Serializable/XA required
Why use MyRocks engine?

© Vadim Tkachenko “How to Rock with MyRocks”
Why use MyRocks engine?

❖ Costs

➢ Cloud costs specifically
➢ Good for Flash
➢ Resource utilization

https://www.percona.com/blog/2019/07/19/assessing-mysql-performance-a
Installation and Configuration

❖ Easily installed for Percona Server with percona-release.
# yum install Percona-Server-server-57.x86_64
# yum install Percona-Server-rocksdb-57.x86_64
# ps-admin --enable-rocksdb
mysql> SHOW ENGINES;
    ROCKSDB | YES     | RocksDB storage engine
mysql> create table test (id int primary key) engine=ROCKSDB;
Query OK, 0 rows affected (0.03 sec)
❖ No downtime required
Installation and Configuration

❖ Configuration options can be reviewed

```sql
mysql> SHOW VARIABLES LIKE 'rocksdb%';
rocksdb_block_cache_size: 536870912
rocksdb_default_cf_options:
  compression=kLZ4Compression;bottommost_compression=kLZ4Compression
```

❖ Percona Server 8.0 brings a lot of improvements to defaults
Installation and Configuration

❖ Some things are configurable per column family

CREATE TABLE t1 (a INT, b INT,
PRIMARY KEY(a) COMMENT 'cfname=cf1',
KEY kb(b) COMMENT 'cfname=cf2')

rocksdb_override_cf_options='cf1={compression=kNoCompression};
cf2={compression=kZSTD}''
Differences between distributions

❖ Compression
  ➢ Facebook: none, depends on what you compile with
  ➢ Percona Server: Zlib, ZSTD, LZ4, LZ4HC
  ➢ MariaDB: Snappy, Zlib (+ LZ4, LZ4HC on Ubuntu)

❖ Data file location
  ➢ Facebook and Percona Server: $datadir/.rocksdb
  ➢ MariaDB: $datadir/#rocksdb

❖ Gap lock detection
  ➢ Percona Server and Facebook: yes (FB off by default)
  ➢ MariaDB: no
Advanced Internals and Limitations

- Mem Table
- WAL (Write Ahead Log)
- Leveled LSM Structure
- Compaction
- Column Family
- ... and more
MyRocks Engine Architecture

Memory

Write Request

Active MemTable

Switch

MemTable

Flush

Persistent Storage

WAL

Switch

WAL

SST Files

Compaction
How does LSM handle writes?

INSERT INTO ..

WAL/ MemTable Sort

New SST

Existing SSTs Merge & Compact

New SST
MemTable(s)

❖ **Store writes in MyRocks**
- Associated with each column family
- Changes go to WAL
- Limited to 64Mb

Ref:
WAL (Write Ahead Log)

- Immediate writes
- Act as redo-log
LSM Leveled Compaction

Compaction

❖ **LSM compaction on Row level is better**
  ➢ Aligned to OS sector (4Kb unit)
  ➢ Negligible OS page alignment overhead

❖ **Percona Server LZ4 as default algorithm**
  ➢ All levels compressed
  ➢ Zstd available
  ➢ Column families allow per table/index
Compression Results

Traditional zlib is slow

Great speed/ratio balance

Best speed
Column Family

- Provides query atomicity between different key spaces.
  - MemTables and SST files
  - Shared transaction logs
- Index mapping is 1 to N
- MyRocks configuration parameters are per CF
- Index Comment per CF
LSM on Disk

- Innodb (Write Amplification on B+Tree)
  - Lower write penalty vs Reduced fragmentation
  - B+Tree Fragmentation over space
  - Compression issues
- Higher read penalty
- Good fit for write heavy workloads
LSM on Flash

❖ Pros
  ➢ Smaller space with compression
  ➢ Lower write amplification
❖ Cons
  ➢ Higher read penalty
❖ Good fit for write heavy workloads
MyRocks Engine Architecture

Memory

Active MemTable
Bloom Filter

MemTable
Bloom Filter

Index and Bloom Filters cached

Block Cache

Persistent Storage

WAL

WAL

SST Files

Read Request
Data Structure & Query Optimizer

❖ Supports Primary and Secondary Keys
  ➢ PK is clustered, single step lookup
  ➢ FK not supported
❖ Tables don’t exist
❖ Online DDL not possible
❖ Fast on scanning forward, slow on ORDER BY DESC
❖ Reverse column families can make DESC scan fast
Data Structure & Query Optimizer

❖ Optimizer Statistics
   ➢ Table statistics (rocksdb_table_stats_sampling_pct; the default value is 10%)
   ➢ Index cardinality
   ➢ Records-in-range estimates
   ➢ SHOW ENGINE ROCKSDB STATUS \G
   ➢ Case Sensitive and Binary Collations
      ■ CREATE TABLE myrocks ENGINE=ROCKSDB COLLATE latin1_bin
Data Structure & Query Optimizer

❖ Optimizer Statistics

➢ SST files stores index statistics
  ■ *Idx name, size, # of rows, disk space, deletes*
  ■ *Distinct # of keys*

➢ Calculated during flush/compaction
  ■ *Ability to force using ANALYZE TABLE syntax (small tables)*

➢ Multi Range Read (MRR) is not supported
Data Dictionary

- Column Family ID
- Index ID
- Global Index ID : Column Family ID + Index ID
- Information Schema
Locking & Isolation Levels

❖ **Row locking**
  ➢ Read-Committed
  ➢ Repeatable-Read

❖ **Gap Lock** - Not Supported
  ➢ Error on statement for Repeatable-Read
  ➢ Percona Server will detect and error out
Replication

❖ **RBR binlog_format=ROW**
  ➢ Large binlogs
  ➢ No triggers on slaves
  ➢ Schema incompatibilities

❖ **SBR causes issues with Gap Locks**
  ➢ Can use on slaves
  ➢ If safe set rocksdb_unsafe_for_binlog=1
Backup and Recovery

- **XtraBackup**
  - Only in 8.0 with xtrabackup 8.0.6+
  - Optimized for Innodb and MyRocks
  - No partial backups for MyRocks

- **Mariabackup**
  - 10.2.16+, 10.3.8+
  - No partial backups for MyRocks
Backup and Recovery

❖ **myrocks_hotbackup**
  ➢ Original backup tool
  ➢ Doesn’t work with 8.0
  ➢ Copies RocksDB checkpoint + WAL
  ➢ MyRocks only, won’t do anything for innodb
  ➢ Supports rolling checkpoint
    ■ *Less WAL to apply on restore till replication*
Backup and Recovery

- **mysqldump**
  - Optimization can be enabled for import
  - rocksdb_bulk_load=1
  - mysqldump in Percona Server detects MyRocks automatically

- **Snapshots**
  - Quite difficult to do right when mixing engines
  - MyRocks: checkpoint + wal
Crash recovery

- Corrupted immutable files: not recoverable
- WAL file: recoverable
  - Variable rocksdb_wal_recovery_mode
    - 1: Fail to start, do not recover
    - 0: If corrupted last entry: truncate and start
    - 2: Truncate everything after corrupted entry
    - 3: Truncate only corrupted entry (unsafe)
# Tool compatibility

## Percona tools generally work with MyRocks

<table>
<thead>
<tr>
<th>Tool</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMM</td>
<td>Supported</td>
<td>Built-in dashboards for MyRocks</td>
</tr>
<tr>
<td>xtrabackup</td>
<td>Supported</td>
<td>Since xtrabackup 8.0.6 (MySQL 8.0 only)</td>
</tr>
<tr>
<td>pt-online-schema-change</td>
<td>Partial</td>
<td>Only in read committed</td>
</tr>
<tr>
<td>pt-table-checksum</td>
<td>Not supported</td>
<td>Only ROW is supported by MyRocks</td>
</tr>
<tr>
<td>pt-table-sync</td>
<td>Not supported</td>
<td>Only ROW is supported by MyRocks</td>
</tr>
</tbody>
</table>
Benchmarks

sysbench/TPCC (100 warehouses, 10 tables)
Percona Server 8.0.16

Engine: INNODB  ROCKSDB
Benchmarks

sysbench/OLTP_RW(8tables, 50M rows)
Percona Server 8.0.16

Engine: INNODB ROCKSDB
Tuning suggestions

❖ **Directory Structure**
  ➢ All files are under .rocksdb directory
  ➢ No file per table option (not even per db)
  ➢ Log file verbosity is high

❖ **Beware of bulk load is problematic**
  ➢ Set rocksdb_bulk_load=1
  ➢ Set rocksdb_commit_in_the_middle=1
Tuning suggestions

❖ Memory Cache Blocks
  ➢ rocksdb_block_cache_size

❖ DirectIO (bypass OS cache)
  ➢ rocksdb_use_direct_reads=ON
  ➢ rocksdb_use_direct_io_for_flush_and_compaction=ON
Tuning suggestions

❖ Simulation cache

➤ rocksdb_sim_cache_size

■ *Simulates block cache (for reads)*
■ *Set to larger/smaller value (restart)*
■ *Costs ~2% of that value*
■ *Show engine rocksdb status*

• rocksdb.sim.block.cache.hit COUNT : 346684
• rocksdb.sim.block.cache.miss COUNT : 86667
Tuning suggestions

❖ **Background jobs**
  ➢ `rocksdb_max_background_jobs=<num_cpu_cores/4>`
  ➢ `rocksdb_max_total_wal_size=4G`

❖ **Better compression**
  ➢ `rocksdb_block_size=16384`
Tuning suggestions

❖ Memory limits
  ➢ rocksdb_db_write_buffer_size

❖ Unless using Percona Server 8.0 with optimized defaults
  ➢ rocksdb_default_cf_options
    ■ Use 8.0 defaults, at least enable bloom filters
    ■ block_based_table_factory= {filter_policy=bloomfilter:10:false;};
Conclusion

❖ Big data sets over 100Gb
❖ Multiple indexes
❖ Write-intensive workloads
❖ Concurrent reads without range scans
❖ Cloud efficient and cheaper to run
  ➢ Less IOPS, Memory, Storage
❖ Write and Read immediately
Special Thanks to...

- Yoshinori Matsunobu @matsunobu
- Vadim Tkachenko @VadimTk
- Sveta Smirnova @svetsmirnova
- Mark Callaghan for doing the extensive research and development.
- Engineering, Experts and Services Teams at Percona
Credits & References

https://www.slideshare.net/matsunobu/myrocks-deep-dive
https://www.percona.com/resources/webinars/how-rock-myrocks