PERCONA LIVE ONLINE MAY 12 - 13th 2021

MyRocks - A 30,000 Foot View

Hello!

I am Mike Benshoof

Technical Account Manager @ Percona

1.
Agenda

What This Talk Is...

- High Level review of the MyRocks Storage Engine
- Discussion of some pros/cons of MyRocks
- Review of some potential use cases for MyRocks
- Providing an alternative approach to investigate when dealing with growing data

What This Talk Is Not...

- Deep dive into the MyRocks internals
- Discussion of implementation choices and tuning
- Full walkthrough of a migration from InnoDB to MyRocks
- A slide deck to encourage you to migrate your entire infrastructure from InnoDB to MyRocks the moment the talk is over...

2.
MyRocks Engine Overview

What the heck is MyRocks?!

- MySQL storage engine, built on RocksDB
  - Key-Value store designed for high performance storage
- Originally developed and open sourced by Facebook
  - Builds now available for Percona Server for MySQL and MariaDB
- Designed to optimize:
  - Space Utilization (Smaller on-disk footprint)
  - Write Efficiency (Less write amplification -> better write performance and longer SSD lifetime)
  - Database Storage Costs (Less Space + Longer Lifetime = Lower TCO)
- LSM-Based database engine
  - Great! But what does this mean and why does it matter?

I promised no “geeky” details, but I had to have a couple slides… please bear with me!

Log-Structured Merge-Tree (LSM-Tree)

Here are key points from the 1996 paper* introducing LSM-Trees:

“The LSM-tree uses an algorithm that defers and batches index changes, cascading the changes from a memory-based component through one or more disk components in an efficient manner reminiscent of merge sort”

“The algorithm has greatly reduced disk arm movements compared to a traditional access methods such as B-trees, and will improve cost-performance in domains where disk arm costs for inserts with traditional access methods overwhelm storage media costs”

“However, indexed finds requiring immediate response will lose I/O efficiency in some cases, so the LSM-tree is most useful in applications where index inserts are more common than finds that retrieve the entries”

** http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.2782&rep=rep1&type=pdf*

Log-Structured Merge-Tree (LSM-Tree)

Here are key points from the 1996 paper* introducing LSM-Trees:

"The LSM-tree uses an algorithm that defers and batches index changes, cascading the changes from a memory-[**Write Optimized**]ough one or more disk components in an efficient manner reminiscent of merge sort"

"The algorithm has greatly reduced disk arm movements compared to a traditional access methods such as B-[**Lower Storage Cost**]ost-performance in domains where disk arm costs for inserts with traditional access methods overwhelm storage media costs"

"However, indexed finds requiring immediate response will lose I/O efficiency in some cases, so the LSM-tree is most useful in applications where index inserts are more common than finds that retrieve the entries"

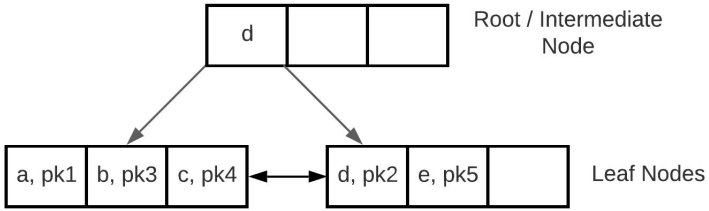
[**Read Overhead**]

** http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.44.2782&rep=rep1&type=pdf*

B+Tree Engine vs LSM-Tree Based Engine

Traditional B+Tree Engine (InnoDB)

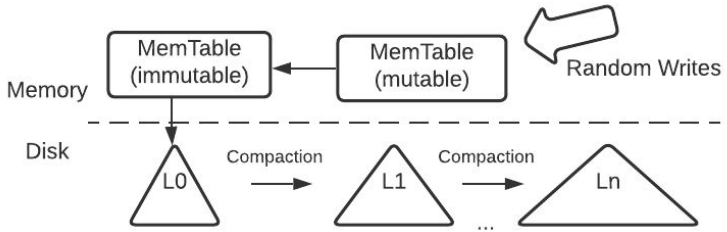
- Optimized for small, random reads
- Tables are wide and shallow -> fewer traversals to find most values
- Writes can be expensive with tree rebalancing (random I/O)
- Clustered Indexing



B+Tree Engine vs LSM-Tree Based Engine

LSM-Tree Based Engine (RocksDB)

- Optimized for write throughput
- Intermediate SST files are merged and compacted at different levels (similar to MergeSort)
- Converts random writes to sequential, append-only writes through MemTables



My MySQL instance is really big and I do lots of writes. I’m ready to migrate!

So is MyRocks just a “better” InnoDB?

- In a word… NO!
- Like everything, there are benefits and disadvantages
- Very much workload / use case dependent

… Let’s look at some pros and cons of MyRocks

MyRocks, some pros and cons

Pros:

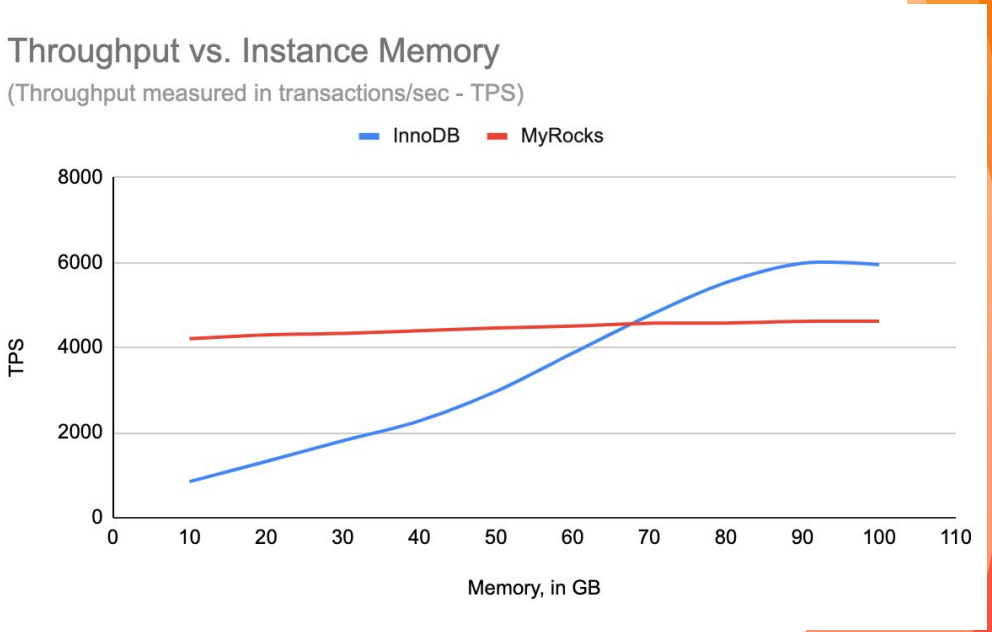
- LZ4 default compression + 100% filled SST files -> smaller disk footprint
- Write optimized database
  - Write performance relatively steady when data larger than memory
- Can allow cheaper storage to be used for write intensive workloads
  - Cost of cloud storage increases when more IOPS are required
- Available in Percona Server 5.7 and 8.0

Cons:

- Statement based replication not supported
- No Gap Locking
- No Foreign Key support
  - Some may put this in the “pro” column…
- Generally slower read performance
- ORDER BY ASC or DESC have different performance characteristics
  - One is fast, the other is slow
- Range scans / table scans are notably slower
- Transportable Tablespace, Spatial Index, and Fulltext Index are not supported

Consistent Write Throughput

- InnoDB benefits greatly from additional memory until dataset fits in memory
- MyRocks doesn't benefit from additional memory, but maintains consistent performance
- Once the majority of the dataset is in memory, InnoDB outperforms MyRocks

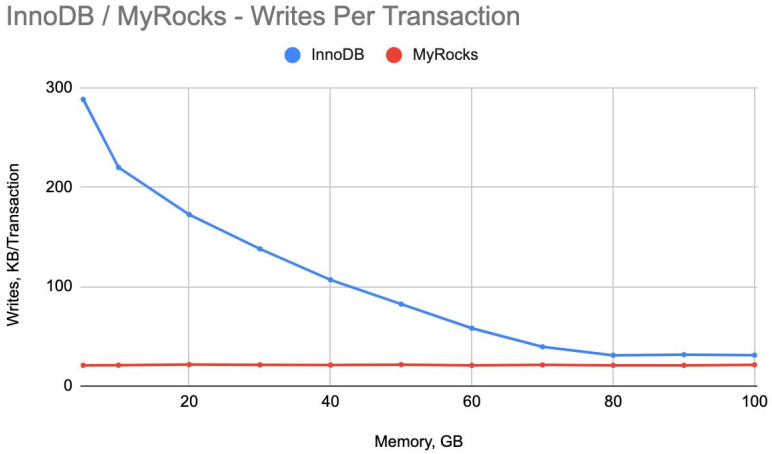


* InnoDB data size ~90G

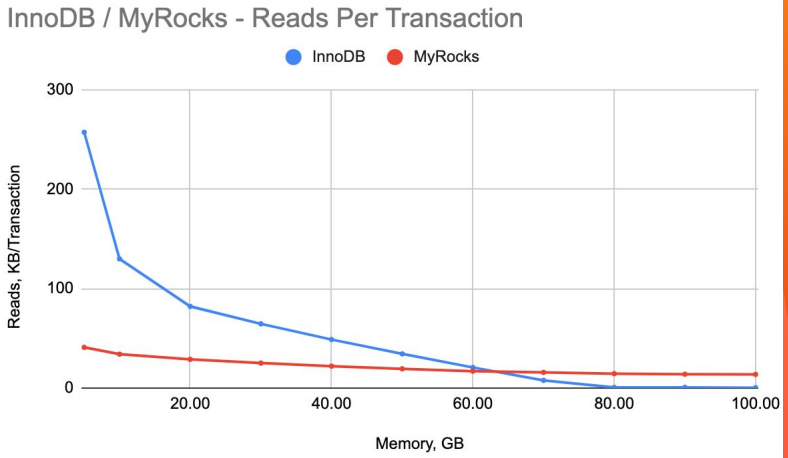
* MyRocks data size ~20G

Based on benchmarks by Vadim Tkachenko:
https://www.percona.com/blog/2018/04/30/a-look-at-myrocks-performance/

IO Utilization



- Demonstrates how MyRocks is a write optimized engine
- InnoDB write I/O improvement requires memory sized to the working data set



- Demonstrates read amplification for MyRocks - more memory only reduces read I/O to a point
- InnoDB is optimized for reads when data fits in memory

Based on benchmarks by Vadim Tkachenko:
https://www.percona.com/blog/2018/04/30/a-look-at-myrocks-performance/

So when does MyRocks make sense?

- Very large datasets
  - Working dataset is larger than memory
- Write intensive workloads
  - I/O bound, not CPU bound
- Limited storage hardware
  - Or looking to reduce cloud storage costs
- Limited range scan queries
  - Read queries are typically point lookups
  - Not ideally suited for reporting queries
- No requirements for:
  - FKs, Fulltext Indexes, Unique Keys (supported, but performance killer), or Spatial Indexes

3.

Some potential use-cases for MyRocks

Potential Use-Case #1

Raw Financial Transaction History

- Maintain detailed transaction history
- Requires years of archive data that MAY need to be selected on demand
- Transactions are primarily insert only
- Live, streaming transactions with a need for “predictable” response time, regardless of data set size

Caveats:

- Not good for ad-hoc reporting (i.e. how much did user X spend in Q1 2019)

Potential Use-Case #2

Medical records

- Notes for all procedures, visits
- Regulations require several years of historical records
- Data compresses nicely
- Unable to horizontally shard existing schema
- Combined with standard, OLTP tables in InnoDB, but rarely joined
- Bulk loading of records (patient transfers) is common

4.
Operational Caveats and Notes

Some things to consider with MyRocks

- No ONLINE DDL
  - Tools like pt-online-schema-change or gh-ost can help
- Statement Based Replication not supported
  - Tools like pt-table-checksum require SBR
- Xtrabackup support
  - 5.7 -> No, you must use a different hot backup option like myrocks_hotbackup
  - 8.0 -> Yes!
- No singular tablespace files
  - Any reporting based on filenames wouldn't be supported with SSTs
- PMM supports MyRocks with pre-defined panels

Monitoring with PMM

5.
Final Thoughts

Let’s review some concepts

Fit

MyRocks is not a silver bullet that will fix all of my RDBMS issues.

Proper Tech

In some circumstances, a different technology all together may be the proper solution

Analysis

Fully understand your existing workload and requirements before choosing MyRocks

Evaluate

TEST, TEST, and then TEST some more before deploying MyRocks in production

Implement

In cases where MyRocks is a fit, you can see great improvements in write performance and cost

Monitor / Iterate

Leverage PMM to ensure you are seeing the benefits you expected and tune MyRocks as needed

Thanks!

Any questions?

You can find me at:

- michael.benshoof@percona.com

THANK YOU !

PERCONA
LIVEONLINE
MAY 12 - 13th
2021