MySQL DBaaS on Intel® Next Gen Xeon Platform

David Cohen – Storage Solutions CTO & Senior Principal Engineer, Intel®
Steve Scargall – Persistent Memory Software Architect, Intel®
Agenda

• Overview of Intel’s Next Gen Platform
• Overview of Persistent Memory as Block Storage
• Benchmark Environment
  • Percona Server + MyRocks/RocksDB on Persistent Memory in Block Storage Mode
• Performance Results
• Future Work
• Q&A
Intel’s Next Gen Platform

Intel Cascade-Lake and Optane™ DC Persistent Memory
Redefining the Memory/Storage Hierarchy

Expanding data insights with:

1. **Persistent and Large Memory**
   - Intel® Optane™ DC persistent memory module

2. **Massively Extended Memory**
   - Intel® Optane™ DC SSD with software

Multiple affordable solutions
Intel® Optane™ DC Persistent Memory
(Optane™ based Memory Module for the Data Center)

- DIMM Capacity
  - 128GB, 256GB, 512GB
- Speed
  - 2666 MT/sec
- Capacity per CPU
  - 3TB (not including DRAM)

- DDR4 electrical & physical
- Close to DRAM latency
- Cache line size access
- Byte Addressable

* DIMM population shown as an example only.

Launched April, 2019
Intel® Optane™ DC Persistent Memory
Operational Modes

App Direct Mode
- Persistent
- High availability / less downtime
- Significantly faster storage

Memory Mode
- High capacity
- Affordable
- Ease of adoption†
Using Persistent Memory in Block Storage Mode
Storage over App Direct
Block Storage Mode – Atomic & Non-Atomic

- No Code Changes Required
- Operates in Blocks like SSD/HDD
  - Traditional read/write
  - Works with Existing File Systems
  - Atomicity at block level
  - Block size configurable
    - 4K, 512B*
- NVDIMM Driver required
  - Support starting Kernel 4.2
  - Configurable as Boot Device
  - Higher Endurance than Enterprise SSDs
  - High Performance Block Storage
  - Low Latency, Higher B/W, High IOPS

*Requires Linux

MySQL Handles Atomic Writes (Default Behavior)

Future Work
- Code changes may be required*
- Bypasses file system page cache
- Requires DAX enabled file system
  - XFS, EXT4, NTFS
- No Kernel Code or interrupts
- No interrupts
- Fastest IO path possible

* Code changes required for load/store direct access if the application does not already support this.

No Page Cache

Persistent Memory

Disable MySQL Atomic Writes
Benchmark Environment

Percona Server with MyRocks/RocksDB Storage Engine using Persistent Memory in Block Storage Mode
Benchmark Environment

Server | Intel® Server System S2600WF
-- | --
BIOS | Version: SE5C620.086B.0D.01.0180.110720181502
 | Release Date: 11/07/2018
CPU | 2 x Intel(R) Xeon(R) Platinum 8260L CPU @ 2.30GHz
 | 24 cores per socket, 2 threads per core
Memory | DDR4 Dual Rank ECC 192GB (12*16GB@2667MHz)
 | 1.5TB Intel® Optane™ DC Persistent Memory
 | (12*128GB@2666MHz)
Storage | 2 x 4TB Intel P4510 NVMe SSDs
 | 2 x 480GB Intel P545 SSDs (boot disks)
 | 1 x Intel 800GB SSD (scratch space)
OS | CentOS 7.5 (Kernel 4.17.5)
Filesystem | XFS
DB Version | Percona Server 5.7.22 with RocksDB
Test Details | SysBench-TPCC
 | ● 5000 Warehouses (500 Warehouses with 10 tables each)
 | ● Up to 48 Database Threads
 | ● Test Duration: ≈900seconds
Dataset Sizes | The following shows the approximate on-disk size of the datasets used for testing
 | MyRocks
 | ● lz4/16KB - 90GB
 | ● lz4/4KB - 98GB
 | ● nocomp/16KB - 351GB
 | ● nocomp/4KB - 353GB

192 GB

DRAM & PMEM Solution

1536 GB

1728 GB

Intel P4510 NVMe SSDs
Benchmark Environment

Sysbench/OLTP

```bash
# numactl --cpubind=1 --interleave=1
/usr/bin/envLD_PRELOAD=/usr/lib64/libjemalloc.so.1 sysbench --num-threads=48
oltp_read_write.lua --tables=32 --table-size=12000000 --report-interval=1 --rand-type=uniform --db-driver=mysql --forced-shutdown=1 --time=1800 --events=0 --percentile=99 --mysql-user=root --mysql-db=sbtest32t12M --mysql-storage-engine=ROCKSDB
--mysql-socket=/tmp/mysql.sock --index-updates=0 --non-index-updates=1 --simple-ranges=0 --skip-trx=on --sum-ranges=0 --order-ranges=0 --distinct-ranges=0 --point-selects=0 --delete Inserts=0 run
```

Sysbench-TPCC

```bash
# numactl --cpubind=1 --interleave=1
/usr/bin/envLD_PRELOAD=/usr/lib64/libjemalloc.so.1 sysbench --num-threads=48 tpcc.lua
--tables=10 --report-interval=1 --rand-type=pareto --db-driver=mysql --forced-shutdown=1 --time=900 --events=0 --percentile=99 --mysql-user=root --mysql-db=tpcc500w10t --mysql-storage-engine=ROCKSDB --mysql-socket=/tmp/mysql.sock --scale=500 --trx_level=RC run
```
Performance Results
Effect of different Block Sizes (4KB/16KB)

• Reducing block size to 4KB can be helpful for setups with persistent memory and without block cache, providing up to 10% improvement in throughput and latency for both lz4 and uncompressed datasets.
• Reducing block size may increase dataset size up to 10%.
• When the block cache is enabled, RocksDB will amortize reads, providing a negligible difference between 16KB and 4KB.
• Note: Filesystem page cache is disabled. The top-level cache is decompressed. Block cache is only using DRAM, not the 2nd level cache (which can be on different media).
• MyRocks performs well with a page size of 4KB or 16KB.
Compression Vs No Compression

Comparison of results for MyRocks engine with lz4 compression and uncompressed
Percona Server 5.7.22/RocksDB # direct, no_binlog # Sysbench-TPCC/5000 warehouses, 1 socket, 48 DB connections

- Results obtained for persistent memory for both lz4 and uncompressed datasets show almost no differences. Compression facts:
  - lz4 dataset 4x smaller than uncompressed
  - lz4 reduces reads rate up to 2.5x, writes rate up to 2.5x
  - CPU overhead of lz4 compression only 2-3%

- Compression decreases the number of writes which improves media endurance while maintaining a similar level of throughput and latency compared with uncompressed data. It also saves storage space and costs.
Moving Read Cache to Persistent Memory

Comparison of results for MyRocks engine with disabled block cache (0G) and enabled (30G)
Percona Server 5.7.22/RocksDB # direct, no_binlog # sysbench-TPCC/5000 warehouses, 1 socket, 48 DB connections

- rockdb_block_cache
- block cache **enabled**: the advantage of persistent memory over NVMe is up to 25%
- block cache **disabled**: persistent memory shows a slight drop in performance versus runs with block cache and in the same time outperforms NVMe up to 4x
Conclusion

- Up to 5.7x Query Latency using Persistent Memory Vs NVMe
- Up to 3x Transactions/Queries per second Vs NVMe
- Persistent Memory as Storage saw up to 59% better CPU Utilization
  - Removed NVMe %iowait overhead
- Minimal Compression Overhead
  - Storage cost savings with compression enabled
- Reduce DRAM footprint by disabling MyRocks caching
Future Work
Call to Action

• Community driven persistent memory enablement
  • Atomicity & Immediate Persistence
  • Faster Reads/Writes & Logging
  • Faster Replication & Consistency (RDMA)
  • Data Tiering
  • Higher IOPS/TPS/QPS, Lower Latency
• Many more innovations
Resources

- Intel Cascade Lake & Optane DC Persistent Memory are available
  - In the Cloud
  - From your OEM/ODM
- Persistent Memory Home – https://pmem.io
- Docs - https://docs.pmem.io
  - Getting Started, User Guides, Tutorials, etc
- Persistent Memory Development Kit (PMDK) – https://pmem.io/pmdk
- Intel Developer Zone (Persistent Memory) - https://software.intel.com/pmem
- PMEM Google Group - https://groups.google.com/forum/#!forum/pmem
Please Rate Our Session

Introducing gh-ost: triggerless, painless, trusted online schema migrations

Rate & Review

Tap a star to rate

Feedback (optional)

SUBMIT