ORACLE OPEN WORLD

#MySQL #oow17

InnoDB: What’s new in 8.0

Sunny Bains
Director Software Development

ORACLE

MySQL

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

What Is New In 8.0
Labs release
Q&A

Legacy Multiple Data Dictionaries

Problems

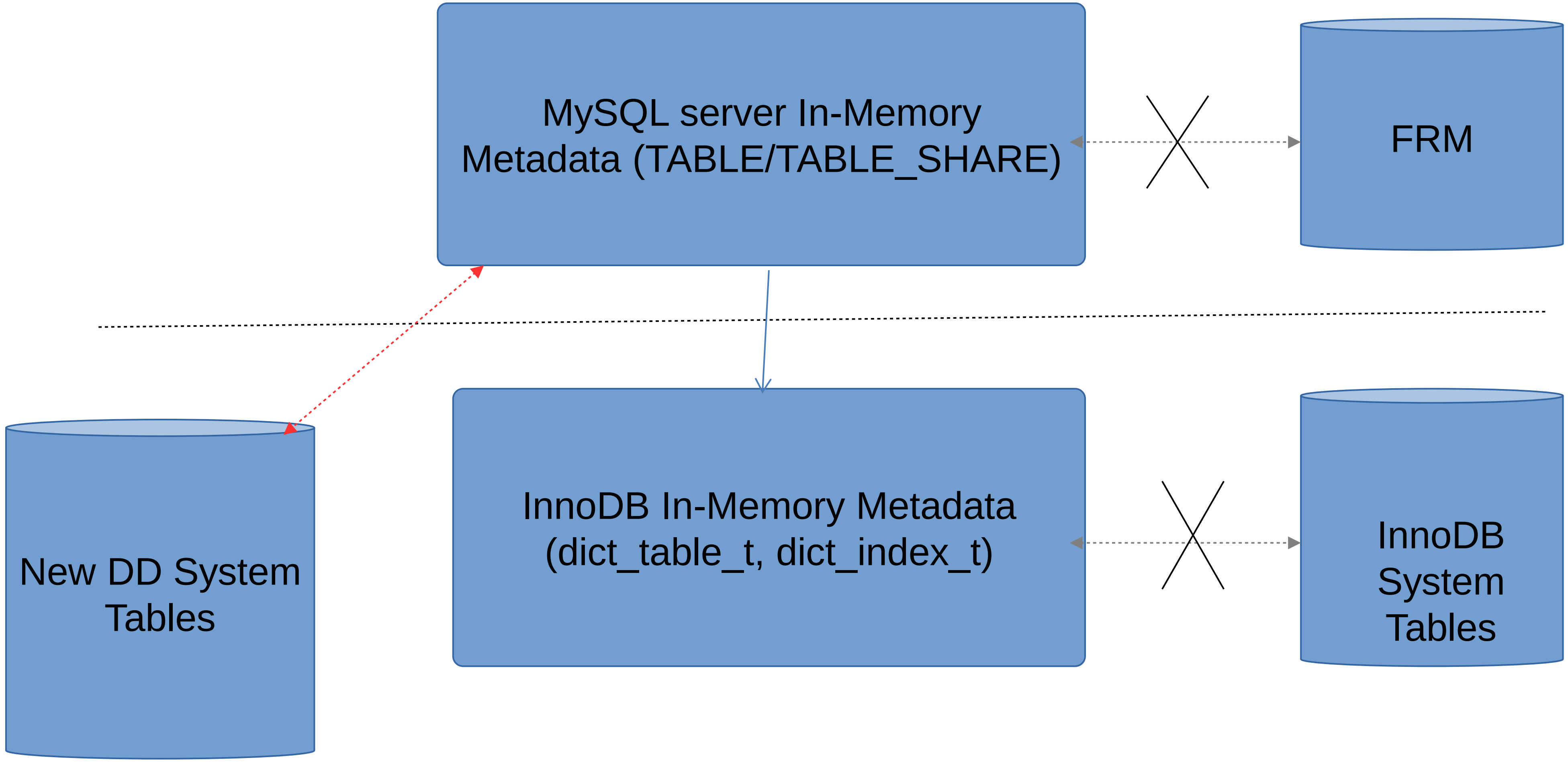
- Up to 5.7 two separate data dictionaries (.frm & InnoDB DD)
- Changes were not atomic
- Mismatch between .frm files and InnoDB's meta-data
    - .frm file updates were not transactional
- Concurrent access had to be very carefully managed
    - Separate locking/latching mechanisms
        - MDL, dict_sys_t::mutex, dict_sys_t::rw_lock etc.
- Not crash proof

Single New Data Dictionary

Benefits

- One source of truth - Server meta-data
- Atomic DDL
  - No more .frm and InnoDB data dictionary mismatch issues
- Required for transactional DDL (future)
- Data dictionary tables stored in a transactional engine
- Control meta-data access using a single locking mechanism (MDL)
- Server supports the concept of Tablespaces
  - .frm files were per table, made general tablespace support messy
- .frm files not created for temporary tables - meta-data in memory only

New Data Dictionary in 8.0


MySQL server In-Memory Metadata (TABLE/TABLE_SHARE)

FRM

InnoDB In-Memory Metadata (dict_table_t, dict_index_t)

InnoDB System Tables

New DD System Tables

New Data Dictionary in 8.0

- Single set of persisted metadata for all Storage Engines
  - Multi-copies of the metadata in server and SE caused ambiguity, which copy is correct?
  - Simple and cleaner interfaces
- The Data Dictionary system tables are stored in InnoDB
  - Stored in a transactional engine with full transactional support (vs. the old frm without any trx support)
  - Create and update of these system tables can be done within a single transaction for a DDL
  - Makes Atomic DDL possible!

New Data Dictionary in 8.0

- InnoDB serves 2 roles
  - Data Dictionary store for all storage engines
  - Supports atomic (crash safe) DDLs

- InnoDB now gets its metadata from server
  - InnoDB no longer fetches its metadata directly from the system tables
  - It is the server's job to open those system tables and read the content
  - InnoDB acts like all other storage engines, that obtain the metadata info from server

Example - DROP SCHEMA at a high level

MySQL 5.7

Delete tables
InnoDB will starts its own transactions to delete table/index metadata from InnoDB system tables and commit.
Server will delete TRN/TRG/FRM files without transaction support.

Delete stored programs
Metadata rows in MyISAM (non-transactional)

Delete schema
Metadata in DB.OPT file

Problems
Mix of filesystem
Non-transactional/transactional storage
Multiple commits.
Non-atomic, could result in in-consistent state at various stages

MySQL 8.0

Delete tables
Server starts transaction.
Metadata in DD system tables marked as deleted.
InnoDB will not drop physical artefacts at this stage, it only logs a record in the DDL_LOG, to ensure that the physical deletion happens when trx commits (recovery implications too).

Delete stored programs
Metadata rows in InnoDB (within the same transaction)

Delete schema
Metadata rows in InnoDB (within the same transaction)

Benefits
Updates to transactional storage, one commit
InnoDB physically deletes all indexes/tablespaces etc.

ORACLE

Atomic DDL

Single set of persistent system tables in InnoDB

- A Single Atomic Transaction for all updates/inserts/deletes to the Data Dictionary
  - Since it is a single transaction, all updates to the Dictionary can be rolled back and are crash safe.
  - Need to make sure there is no intermediate commits in SE during DDL
  - SQL layer will invoke a post_ddl() hook, so SE can do post commit/rollback work.
  - As part of the transaction, write to binary log.

Atomic DDL with InnoDB

DDL_LOG to record physical operations for InnoDB DDLs

- DDL_LOG is a table in the **mysql** tablespace, no DDL and no USER DML allowed
- It is created to track tablespace (file) creation/drop, index trees creation/drop, file rename etc.
- This covers physical operations in a DDL that cannot be rolled back by a transaction.
- DDL transaction (mentioned in previous slide) and this table makes Atomic (Crash safe DDL) possible

Example – Create table with Atomic DDL

MySQL 5.7

Call to SE handler::create(name,...)

InnoDB creates the physical tablespaces/files(file-per-table) or Cluster index tree and other index trees

InnoDB starts its own transactions to insert new table/indexes metadata to InnoDB own System tables

Contd.

MySQL 8.0

Call to SE handler::create(name,...)

SE creates the physical tablespaces/files(file-per-table) or Cluster index tree and other index trees

InnoDB logs the operations in the DDL_LOG

Note: No separate SE transaction

Contd.

Example – Create table with Atomic DDL

MySQL 5.7

If server crashes after physical tablespace/files created, before metadata updated, these files will be orphan files.

MySQL 8.0

SQL layer commits or rollback

Call to SE post_ddl(). If rollback, the post_ddl() physically deletes the tablespace/ibd (file-per-table) and drops the index trees for the table

Performance

- Cost Based Optimiser statistics
  - Number of pages in RAM per index
- Remove the buffer pool mutex (Percona contribution)
  - Took a long time to fix problems in the contributed patch
    - QA team found lots of problems in edge cases
  - Foundation for more improvements in the future

Performance (cont.)

- CATS (Contention Aware Transaction Scheduling) (was called VATS earlier)
    - Contributed by University of Michigan DB researchers
    - No configuration required
    - Switches between FIFO and CATS automatically
        - Threshold is >= 32 waiting threads

Performance (cont.)

- Group records by table id when purging
  - Reduces contention of the dict_index_t::lock when multiple purge threads enabled
- --innodb_stats_include_delete_marked := bool
  - Include/Exclude rows that are delete marked (in 8.0.1)
- --innodb_deadlock_detect := bool (dynamic)
  - On high concurrent loads, rely on --innodb_lock_wait_timeout and rollback
- Internal read ahead row buffer set by the Optimiser

Feature Improvements

- Memcache improvements
  - Support multiple get and range search
- Persistent auto increment
  - Doesn’t reset to SELECT MAX(AUTOINC_COL) FROM T; on restart
  - Probably the most requested feature since v3.x
  - Bug 199 - Created on 27 March 2003

Information Schema

- A new INFORMATION_SCHEMA table, INNODB_CACHED_INDEXES
  - Report pages cached in the InnoDB buffer pool for each index.

Undo tablespace improvements

- Change the undo roll ptr format – upgrade/downgrade impact
  - CREATE UNDO TABLESPACE ‘rbs01’ ADD DATAFILE ‘rbs01.ibu’;
  - DROP UNDO TABLESPACE ‘rbs01’;
  - ALTER UNDO TABLESPCE ‘rbs01’ SET OFFLINE/ONLINE;
- More flexible tablespace management
- Implications for upgrade
- Default will be two undo tablespaces
- SQL syntax to manage undo logs dynamically
- Exact syntax work in progress
- Undo truncate is on by default

Miscellaneous

- Avoid intermediate commits that would occur every 10000 rows
  - e.g. ALTER TABLE … ALGORITHM=COPY
- Remove .isl files (InnoDB Symbolic Link files)
  - Was used when creating tablespace data files outside of the MySQL data directory.
- --innodb-read-only semantics change
  - If ON then affects entire MySQL instance
  - Because DD tables are stored in InnoDB

Deprecations / Removals

- Deprecated parameters that have been removed
  - innodb_file_format
  - innodb_file_format_check
  - innodb_file_format_max
  - innodb_large_prefix
  - innodb_stats_sample_pages
  - innodb_locks_unsafe_for_binlog
  - innodb_checksums
  - innodb_support_xa (always ON)

Better Tablespace Management

- Versioning for tablespaces
  - Support multiple tablespace/page/row formats
  - Easier to introduce new features/capabilities
    - E.g., a page/row format that gives better compression
  - Support the Server native row format, avoid conversions
  - Improve upgrade process
  - Helps with maintaining backward compatibility

Better Tablespace Management

- SQL for managing UNDO logs/tablespaces
- Tablespaces will be self describing:
  - Serialized Dictionary Information (SDI)
  - Embedded inside the table space (Separate file (.sdi) for MyISAM)
  - Improve import/export - long term objective is to make it instant
- Getting rid of the legacy “system tablespace” a.k.a ibdata

Serialized Dictionary Information (SDI)

```
{
    "sdi_version": 1,
    "dd_version": 1,
    "dd_object_type": "Table",
    "dd_object": {
        "name": "tbl1",
        "mysql_version_id": 80000,
        "created": 20160922042352,
        "last_altered": 20160922042352,
…
        "columns": [
            {
                "name": "id",
                "type": 4,
                "is_nullable": false,
…
        ],
…
"indexes": [
…
        ],
        "foreign_keys": [],
        "partitions": [],
        "collation_id": 8
…
```

SDI tools

- A tool for extracting Serialized Dictionary Information (SDI)
  - ibd2sdi
  - Works offline and online
  - Extracts the SDI record id, type, data in JSON format
  - Useful during disaster recovery
    - e.g., Table corrupted in a tablespace with multiple tables
    - Extract the meta-data from the .ibd file into a separate .SDI file
    - Remove corrupt table meta-data by editing .SDI file
    - Use edited .SDI file to import the tablespace and ignore the corrupted table

New In-Memory Storage Engine (temptable)

- Currently for internal use only (Optimizer joins etc.)
- Not shared across connection
- Lifetime limited to query life time
- Limited size, bounded by memory allocated

Encryption and Generalised Tablespace Improvements

- Encryption of redo and undo log
- Generalised/Shared tablespaces
  - Support Encryption (WIP)
  - Support Compression (WIP)
  - Support Import/Export (WIP)

Better LOB design

- More flexible BLOB handling
  - Allow partial fetch and update
  - Plan is to make streaming easier
- Performance improvement for large LOBs
  - Up to 14x in our internal tests (WIP)

Encryption and Generalised Tablespace Improvements

- Encryption of redo and undo log
  - --innodb-redo-log-encrypt := bool
  - --innodb-undo-log-encrypt := bool
- Generalised/Shared tablespaces
  - Support Encryption
  - Support Compression
  - Support Import/Export

NO_WAIT/SKIP LOCKED

- If NO_WAIT set for a query
- Return immediately without waiting for the row lock to be released
- SELECT * FROM T WHERE C 1= n and C2 = m FOR UPDATE NO_WAIT;

- If SKIP LOCKED set for a query
- Skip locked row, without waiting for the row lock to be released
- SELECT * FROM T WHERE C1 = N AND C2 = m LIMIT 1 FOR UPDATE SKIP LOCKED;

Descending Indexes

- Change buffering is not supported
- If secondary index contains a descending index key column
- If the primary key includes a descending index column
- Supported for all data types for which ascending indexes are available.
- Supported for ordinary and generated columns (both VIRTUAL and STORED)
- Not supported for full text indexes and RTree
- A little slower than ascending indexes, due to page layout issues

Dedicated Server

- --innodb-dedicated-server := boolean (default OFF)
- Sets default values based on physical memory available
- If below variables not explicitly set to non-defaults
- --innodb-log-file-size based on physical memory size
- --innodb-buffer-pool-size based on physical memory size
- --innodb-flush-method=O_DIRECT_NO_FSYNC

ORACLE

Dedicated Server (contd.)

- --innodb-buffer-pool-size

```
If phy_mem_size < 1G
    Use InnoDB default value
Else If phy_mem_size <= 4GB
     Use 50% of phy_mem_size
Else
     Use 75% of phy_mem_size
End
```

Dedicated Server (contd.)

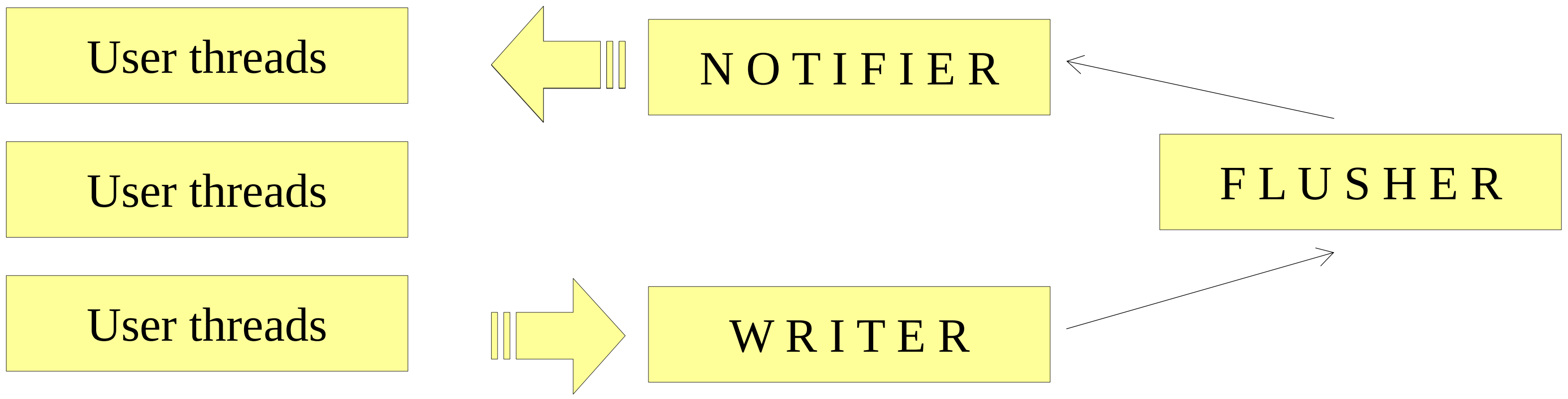
- --innodb-log-file-size

```
If phy_mem_size < 1G
    Use InnoDB default value
Else If phy_mem_size <= 4GB
     Set to 128 MB
Else if phy_mem_size <= 8 GB
     Set to 512 MB
Else if phy_mem_size <= 16 GB
      Set to 1GB
Else
     Set to 2GB
End
```

Scalable redo log

- Dedicated redo log threads
  - log_writer - writes from log buffer to file
  - log_flusher - executes fsync()
  - log_notifier - notifies user threads about finished fsync
  - log_checkpointer - writes checkpoints

Scalable redo log design (New)



Increased concurrency of mtr commits

- Removed log_flush_order mutex
- Removed log_sys mutex
- Decreased latency between: fsync()→trx committed

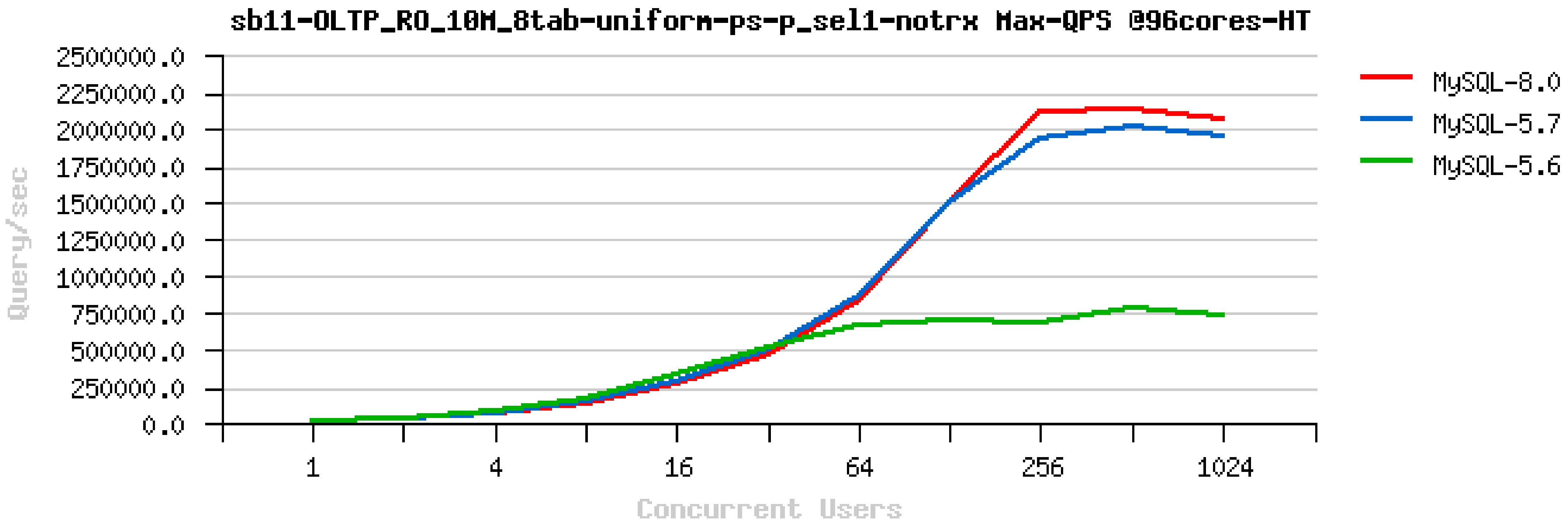
Benefits of the new design

- User threads write concurrently to log buffer
- User threads don't wait for each other
- Log writer tracks pending / finished writes
- Log writer keeps writing log buffer to disk
- Log buffer can be resized in runtime
- Overall much better throughput and latency

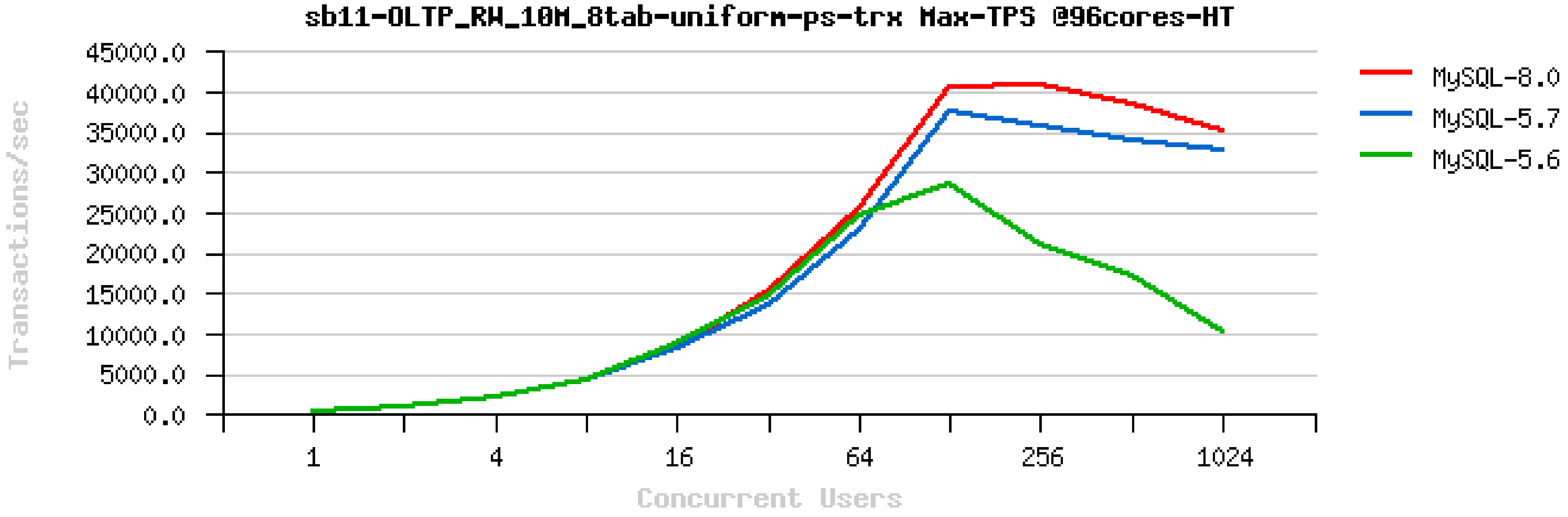
Scalable IO layer (fixed fil_sys mutex conention)

- Split the file IO layer into 64 shards
- Dedicated redo log shard
- 4 Dedicated undo log shards
- Remaining shards for user tablespaces

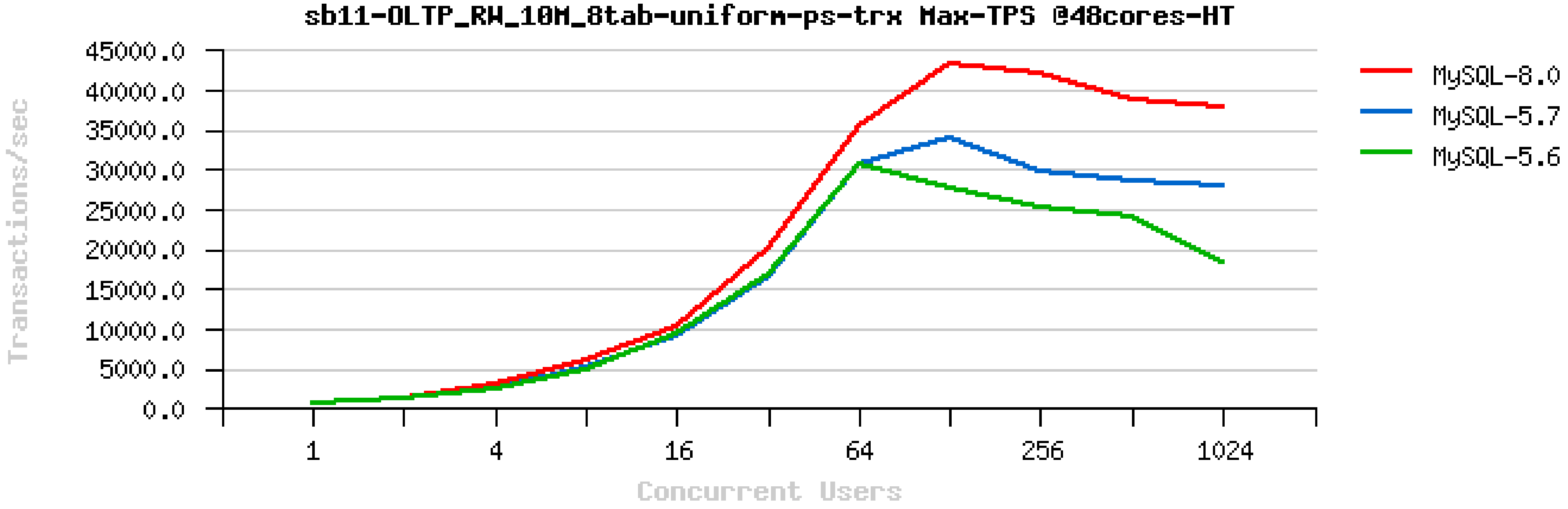
OLTP_RO-point-selects 10Mx8tab-uniform @96cores-HT



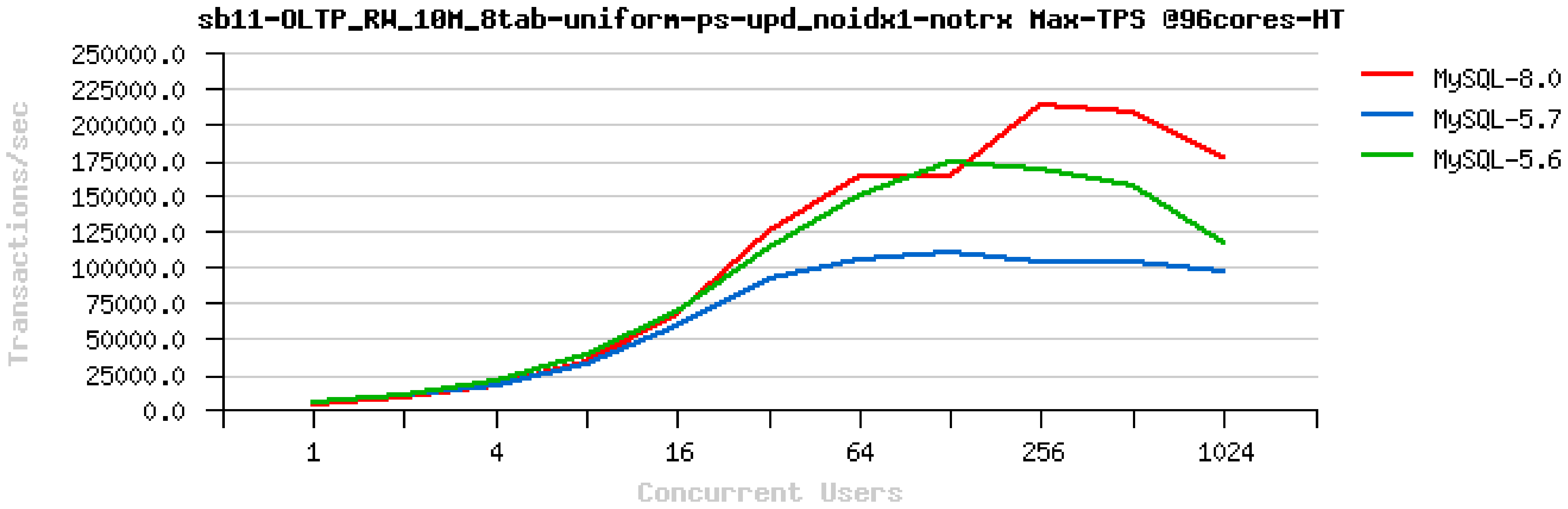
OLTP_RW-mixed 10Mx8tab-uniform @96cores-HT



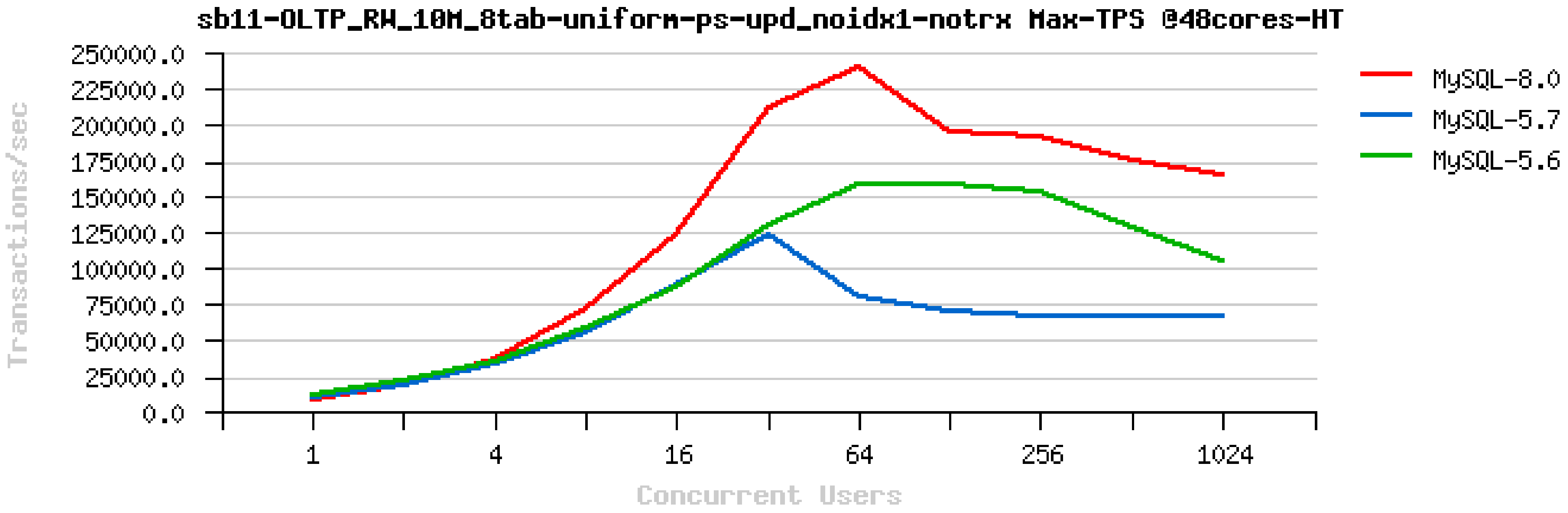
OLTP_RW-mixed 10Mx8tab-uniform @48cores-HT



OLTP_RW-update_only 10Mx8tab-uniform@96cores-HT



OLTP_RW-update_only 10Mx8tab-uniform@48cores-HT



ORACLE

Labs release Source and Binaries

http://labs.mysql.com

8.0.3 + InnoDB

ORACLE®