



InnoDB Status&Roadmap in MariaDB

Marko Mäkelä
Brussels & London
February 2020

InnoDB Improvements in MariaDB 10.5

- **10.5.0** [MDEV-19514](#) Defer change buffer merges until pages requested
 - Prevents ‘random’ crashes due to change buffer corruption
- **10.5.0** [MDEV-16264](#) Implement a work queue for InnoDB background tasks
 - Removes a large number of InnoDB background threads
- **In progress:** [MDEV-18959](#) Engine transaction recovery through binlog
 - Only `fsync()` the binlog on transaction commit, not InnoDB redo log
- **Planned:** Remove `innodb_log_optimize_ddl` (write full ALTER TABLE log)
 - Enables [MDEV-19738](#) Doublewrite buffer is unnecessarily used for newly (re)initialized pages

I/O Scalability Improvements

- Raw idea: [MDEV-16260](#) Scale the purge effort according to the workload
- In progress: [MDEV-12353](#)/[MDEV-14425](#) Efficient redo log record format
- Early stages: [MDEV-16526](#) Overhaul the InnoDB page flushing
 - **Blocks:** [MDEV-14481](#) Execute InnoDB crash recovery in the background
- In progress: [MDEV-15058](#) Remove multiple InnoDB buffer pool instances
- **10.5.1** [MDEV-18115](#) Remove dummy tablespace for the redo log
- In progress: [MDEV-15528](#) Punch holes when pages are freed
 - [MDEV-12226](#) Avoid writes of freed (garbage) pages to InnoDB temporary tablespace
 - [MDEV-12227](#) Defer writes to the InnoDB temporary tablespace

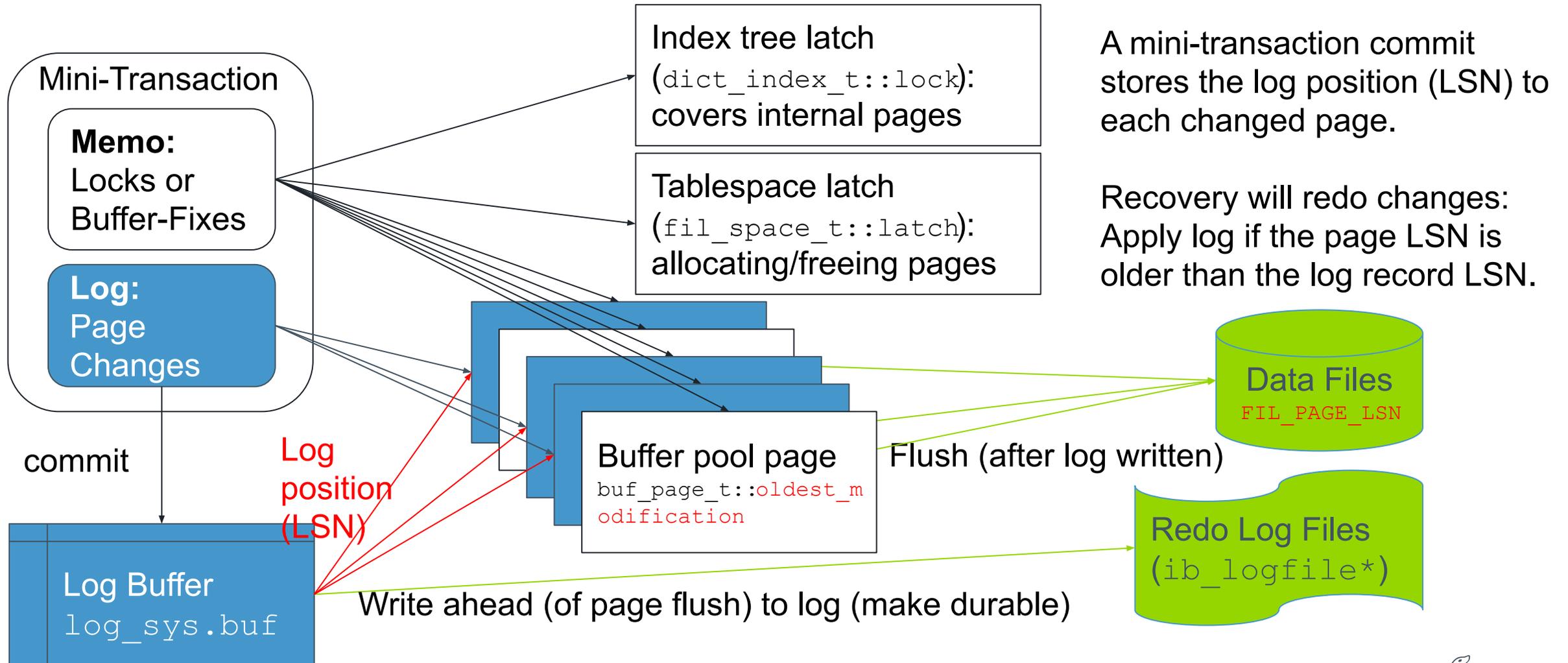
I/O Subsystem Overhaul

Page Flushing and Log Checkpoints

Write Dependencies and ACID

- Log is written by *mini-transactions*, to atomically update pages.
 - Transactional ACID (record locks, rollback, MVCC) builds upon this.
 - Mini-transactions are *totally ordered* by log sequence number (*LSN*).
 - A mini-transaction is *durable* if everything up to its *LSN* has been written to log
 - A user transaction `COMMIT` is durable if the mini-transaction of is durable
- Write-ahead logging: Must **write log before dirty pages**, at least up to the `FIL_PAGE_LSN` of the dirty page that is about to be written
- Log checkpoint: **write dirty pages older than the checkpoint *LSN***
 - Recovery will have to process log from the checkpoint *LSN* to last durable *LSN*
- [MDEV-16264](#) Implement a common work queue... simplifies page flushing
 - `io_submit()` from only one thread, `io_getevents()` from another

Mini-Transactions: RW-Latches and Redo Logs



Optimizing Log Writes

- Current situation: Mutex contention: Any thread that issues writes can:
 - write or fsync the log \Rightarrow contention on `log_sys.mutex` or `log_sys.write_mutex`
 - invoke `log_checkpoint()` by `log_free_check()`
 - Checkpoint is also initiated by master thread, and log writes by page writes!
- Idea: Have a **dedicated log writer task** that is signalled by other threads
 - Page flush would skip “too new” pages instead of waiting for durable log write
 - Avoid mutex: `log_sys.last_flushed_lsn.load()`
 - Remove `buf_page_t::newest_modification` and just use `FIL_PAGE_LSN`
 - **Dedicated log checkpoint task**
 - `log_free_check()` would submit a task (if needed) and wait for completion
- Most `mtr_t::commit()` could return immediately (just transfer the `mtr_t::m_log` ownership); a special durable variant would wait

Redo Log Format Redesign

Compact, extensible format, faster recovery

Planned Redo Log Changes in 10.5

- [MDEV-12353](#) Efficient redo log record format
 - Done: Replace physio-logical log records with purely physical ones
 - Removed: `innodb_log_optimize_ddl` (always write redo log)
 - Done: Reimplemented redo log record codec (© MariaDB Corporation)
 - Opens possibility for “smart storage” à la Amazon Aurora or Alibaba PolarDB
 - InnoDB writes only log (no page flushing, no log checkpoints!)
 - InnoDB reads back pages as of a specified *LSN*. (Easy “flashback” to any time.)
- [MDEV-14425](#) InnoDB redo log format for better performance
 - `ib_logfile0` will be a dummy file, or at most contain checkpoint information
 - Write information about persistent files and checkpoints into one file
 - Separate, circular page-level log file (similar to the existing format):
 - Efficient with persistent memory (NVDIMM): `mmap (MAP_SYNC)` with `mount -o dax`

Redo Log File Format (1/2)

- Partitioning the log was considered and rejected in [MDEV-14425](#)
 - Requires `fsync()` of all log files at `COMMIT`, destroying any performance benefit
- Compact, easy-to-parse log record format to cover changes to pages
 - [MDEV-12353](#): Each record starts with type and length code
 - One type code will be completely ignored by InnoDB, can be used for anything
 - Could allow variable-size blocks (terminate each durable snippet with checksum)
- For more flexibility, make *LSN* count mini-transactions, not payload bytes
 - `mariabackup --incremental` can inject additional records to the redo log!
 - `mariabackup --prepare` might be performed by normal server startup

Redo Log File Format (2/2)

- `ib_logfile0` could just contain a special header that indicates new format
- Page-level data file(s) start with a header
 - Identifies the format, creator version, maybe also timestamp of creation
 - We might introduce an append-only format with archiving later
- Checkpoint information file:
 - All files created, deleted, renamed, modified since the previous checkpoint
 - Checkpoint *LSNs* and corresponding log file names and byte offsets
 - Can contain multiple checkpoints, written sequentially
 - Can be rotated (rewritten) upon reaching a configured maximum size
 - Might use the existing infrastructure for log file rotation (Aria log, binlog)

Optimizing Write Performance

Smarter Page Writes, Fewer `fsync()`

Optimizing Dirty Page Flushing

- [MDEV-16526](#) Overhaul the InnoDB page flushing
 - Remove `BUF_FLUSH_SINGLE_PAGE`
 - Do we need separate `BUF_FLUSH_LRU` (w/ `evict`) and `BUF_FLUSH_LIST`?
 - Always sort the `buf_pool_t::flush_list` like during recovery (`flush_rbt`)?
 - Remove separate page cleaner mode for crash recovery
- [MDEV-14481](#) Execute InnoDB crash recovery in the background
 - Allow `innodb_read_only` even when crash recovery is needed
- [MDEV-15058](#) Remove multiple buffer pool instances
 - Find and remove the bottlenecks that motivated this feature in MySQL 5.5
 - [MDEV-15053](#) Split `buf_pool_t::mutex` (and use more `std::atomic`)

Reducing `fsync()` Operations

- Important state change at *LSN* is persisted by durable write of log $\geq LSN$
 - COMMIT, (SQL-level) XA PREPARE, XA ROLLBACK, XA COMMIT
 - Future idea: Set up a “log write completion” event that sends OK packet to the client
 - Binlog-driven transaction: XA PREPARE in InnoDB (durable log write), then `write(); fsync()` binlog, and finally XA COMMIT in the InnoDB log buffer
 - [MDEV-18959](#): Do binlog `write(); fsync()` and non-durable COMMIT in InnoDB
- `fsync()` is overkill for ‘write barriers’. **Leverage liburing at some point?**
 - Before data page flush at *LSN*, complete durable write of redo log $\geq LSN$
 - Before completing log checkpoint, we `fsync()` all data files
 - Before binlog rotation (discarding the start of binlog), [MDEV-18959](#) must `fsync()` the InnoDB redo log up to the *LSN* of the first remaining commit in the binlog

Longer-Term Ideas

What to improve in InnoDB after 10.5

More Performance and Flexibility (1/2)

- Leverage `liburing` to avoid `fsync()` for ‘write barriers’
- Move things out of the system tablespace, to prepare for its removal
 - [MDEV-11634](#) Logical change buffer, exploited also for `ROLLBACK`
 - [MDEV-11659](#) Move the InnoDB doublewrite buffer to flat files
 - [MDEV-19506](#) Remove the global sequence `DICT_HDR_ROW_ID` for `DB_ROW_ID`
 - [MDEV-15020](#) Store persistent statistics in `.ibd` file (or remove the code?)
 - Note: InnoDB system tables will remain until [MDEV-11655](#)
- [MDEV-18518](#) Atomic `CREATE` of partitioned table; crash-safe `DROP INDEX`
- [MDEV-11658](#) Simpler, faster `IMPORT` of InnoDB tables
- Improve record locks: [MDEV-10962](#), [MDEV-16406](#), [MDEV-16232](#), [MDEV-11215](#), [MDEV-20612](#); replace table locks with MDL?

More Performance and Flexibility (2/2)

- Move FOREIGN KEY out of InnoDB: [MDEV-12483](#), [MDEV-10393](#), ...
- Non-blocking COMMIT: Send OK packet after transaction is durable
 - Allow interleaved execution of the next transaction while log flush is pending
- [MDEV-16232](#) Use fewer mini-transactions
 - Implicit record locks in UPDATE, DELETE, INSERT...ODKU, REPLACE
 - Remove the row prefetch buffer from InnoDB
- [MDEV-515](#) Bulk insert into empty table or partition (TRUNCATE on ROLLBACK)
- [MDEV-18746](#) Reduce the amount of `mem_heap_create()` or `malloc()`
- ALTER TABLE: [MDEV-16356](#) ADD CONSTRAINT, ALGORITHM=NOCOPY, [MDEV-16281](#) parallel ADD INDEX, [MDEV-9260](#) Improve progress reporting



Thank you