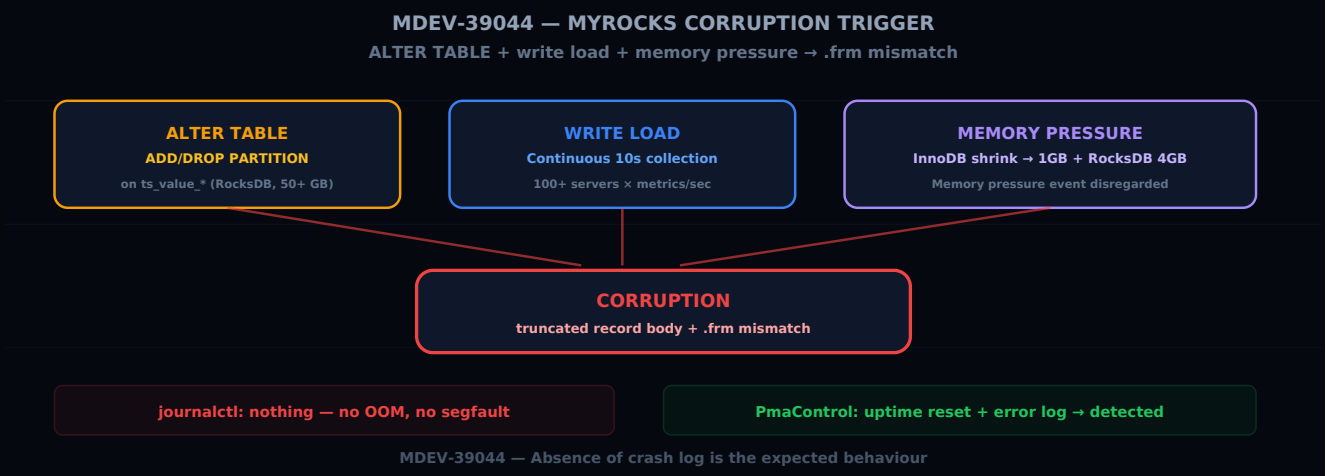
MyRocks Under Load: When ALTER TABLE Triggers Corruption

Aurélien LEQUOY · March 6, 2026

MARIADB ROCKSDB CORRUPTION DDL INCIDENT-RESPONSE MDEV-39044



Context

On March 6, 2026, a production MariaDB 10.11.15 server monitored by PmaControl suffered a **major incident**. Unlike typical crashes (OOM, segfault), this one presented unique symptoms:

```
RocksDB: Error opening instance, Status Code: 2,
  Status: Corruption: truncated record body
Incorrect information in file: './pmacontrol/ts_value_general_int.frm'
Can't init tc log
Aborting
```

The server restarted in a loop several times before stabilising, with `.frm mismatch` errors on multiple time-series tables.

The MDEV-39044 ticket

After investigation, we correlated this incident with MariaDB ticket **MDEV-39044**:

> *MyRocks corruption after restart during/after ALTER workload: Corruption: truncated record body, .frm mismatch, no crash log, no OOM killer*

What the ticket describes

The ticket documents a reproducible corruption scenario:

1. **Large partitioned RocksDB tables** — exactly what PmaControl uses for metrics (`ts_value_*` tables partitioned by day)
2. `ALTER TABLE` **under write load** — adding partitions while the application writes continuously
3. **Simultaneous InnoDB memory pressure** — InnoDB and RocksDB tables coexist on the same server
4. **No kernel trace** — no OOM killer, no segfault, no crash log

Why it's insidious

The most dangerous aspect of the ticket: **the absence of a crash log is the expected behaviour in this scenario**. The server restarts, performs `InnoDB crash recovery`, but the RocksDB metadata is corrupted (`.frm mismatch`).

A DBA who only checks `journalctl` or `dmesg` will find nothing. They'll classify the incident as an "unexplained restart" and move on.

Our concrete case

Affected tables

All partitioned RocksDB tables with heavy daily writes:

- `ts_value_general_int` — integer metrics (status variables, counters)
- `ts_value_general_json` — complex JSON metrics
- `ts_mysql_digest_stat` — query statistics (digests)
- `ts_value_general_text` — text metrics
- `ts_value_slave_int` — replication metrics
- `ts_value_slave_text` — detailed replication states

The likely trigger

PmaControl automatically maintains partitions on these tables: adding next day's partition, dropping expired partitions. These are `ALTER TABLE ... ADD PARTITION` / `DROP PARTITION` operations on tables weighing tens of GB, **while collection workers write continuously** (every 10 seconds per monitored server).

Memory pressure signals

Before the crash, the MariaDB log shows:

```
InnoDB: Memory pressure event disregarded
```

The MDEV-39044 ticket explicitly cites this pattern as an aggravating factor. InnoDB memory pressure doesn't directly cause the corruption, but it creates the context in which the RocksDB DDL becomes non-atomic.

How PmaControl detected the incident

1. **Uptime reset** detected within 10 seconds via the `ts_variable.uptime` time series
2. **Telegram alert** sent immediately
3. **Automatic correlation** with the error log: detection of `crash recovery` + `truncated record body` signatures
4. **Retrospective analysis**: metrics from the preceding hour (threads, memory, CPU) were normal — confirming this is not a typical load issue

Recommendations

Immediate actions

1. **Do not run DDL on RocksDB tables under write load**. Schedule `ALTER TABLE ... ADD/DROP PARTITION` during low-activity windows.
2. **Monitor `.frm` errors** in the error log. This is the first indicator of post-DDL corruption.
3. **Follow ticket MDEV-39044** for an official fix.

Structural actions

4. **Separate engines**: if possible, do not mix InnoDB and RocksDB on the same server for critical tables.
5. **Consider migrating hot tables to InnoDB**. RocksDB excels at sequential writes, but its DDL operations are not atomic under load.
6. **Size memory properly** to avoid the InnoDB pressure that aggravates the problem. See our article on the OOM killer for worst-case calculations.

What it is not

- It is **not** a hardware problem (disk, RAM)
- It is **not** a MySQL configuration problem (parameters are correct)
- It is **not** reproducible on demand (it's a race condition in the RocksDB/DDL engine)

It is an **engine bug** documented by MariaDB themselves.

Conclusion

MDEV-39044 is a reminder that using alternative storage engines (RocksDB, TokuDB) on production workloads requires particular vigilance around DDL. The absence of a crash log does not mean the absence of corruption.

PmaControl detects these incidents through `uptime` monitoring + error log correlation, where standard tools see nothing.