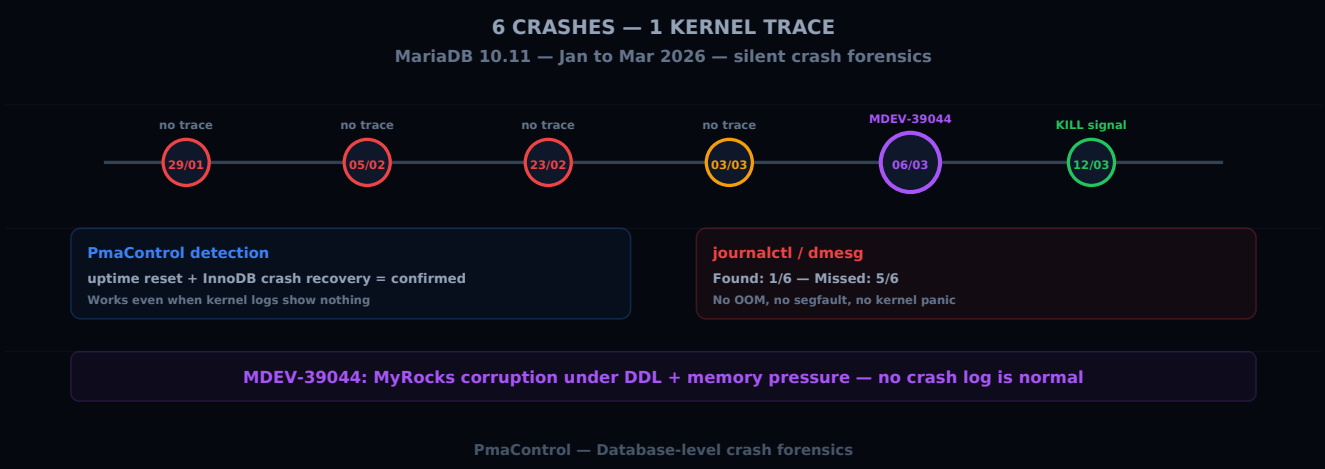
Silent MariaDB Crashes: When Log Silence Masks Critical Failures

Aurélien LEQUOY · March 12, 2026

MARIADB CRASH-ANALYSIS FORENSICS INCIDENT-RESPONSE ROCKSDB



The problem

Between January and March 2026, we observed **6 abnormal uptime resets** on a production MariaDB 10.11.15 server monitored by PmaControl. The server hosts very large partitioned RocksDB tables (time-series metrics).

A DBA's classic reflex when facing a crash: check the system logs. `journalctl`, `dmesg`, `/var/log/syslog`. Look for `OOM`, `Killed process`, `segfault`, `kernel panic`.

Out of 6 crashes, only one had an exploitable kernel trace. The other 5: complete silence from the system side.

Detection method

Rather than trusting system logs, we used PmaControl to detect crashes via the `uptime` **time series**:

1. Retrieve `uptime` values from `ts_value_general_int`
2. Filter abnormal resets (uptime dropping back to 0)

3. Correlate with MariaDB logs (`error.log`)
4. Correlate with `journalctl` looking for kernel signatures
5. Analyse metrics for the preceding hour (threads, CPU, memory)

This is the most reliable approach: **an uptime reset + an `InnoDB crash recovery` signature = confirmed crash**, even without a system trace.

The 6 identified crashes

Date	Classification	Main signature
Jan 29	probable crash	`InnoDB crash recovery` + binlog recovery
Feb 5	probable crash	crash recovery + `Event invalid` in binlog
Feb 23	probable crash	`InnoDB crash recovery` + `Crash table recovery`
Mar 3	probable crash	`Too many connections` then crash recovery
Mar 6	major incident	**MyRocks corruption**: `truncated record body` , `.frm mismatch`
Mar 12	confirmed crash	`systemd: status=9/KILL` + crash recovery

The March 6 crash: MDEV-39044 correlation

The most severe incident was on March 6. The errors were different:

```
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
```

This pattern matches exactly MariaDB ticket **MDEV-39044**: MyRocks corruption triggered by:

- `ALTER TABLE` on large partitioned RocksDB tables
- continuous heavy write load
- simultaneous InnoDB memory pressure

The ticket explicitly confirms that **the absence of crash logs or OOM killer is normal in this scenario**.

Why system logs are not enough

Out of 6 incidents, `journalctl` found only **one exploitable trace** (the `status=9/KILL` on March 12).

For the other 5:

- no `Out of memory`
- no `Killed process`
- no `segfault`
- no `kernel panic`

The inference is simple: **the absence of a kernel signature does not clear an incident**. This is even consistent with the MDEV-39044 pattern, which documents crashes without system traces.

What PmaControl detects that logs don't show

PmaControl monitors `uptime` continuously (every 10 seconds). A reset = immediate alert. The agent then automatically correlates:

- metrics from the preceding hour (threads, memory, CPU)
- presence of `crash recovery` in the error log
- metadata errors (`.frm mismatch`)

This allows classifying the incident **even without kernel cooperation**.

Recommendations

1. **Never rely solely on system logs** to detect MariaDB crashes
2. **Monitor `uptime` as the primary stability indicator**
3. **Correlate with the MariaDB error log**, not with `journalctl`
4. **If you use RocksDB**: limit DDL on large partitioned tables, especially under write load

5. **Follow MDEV-39044** for a potential MyRocks fix

Conclusion

A MariaDB server can crash **6 times in 6 weeks** without any system log documenting it. Only database-dedicated monitoring — that understands MariaDB's internal recovery signatures — can detect and classify these incidents.

That is exactly what PmaControl does.