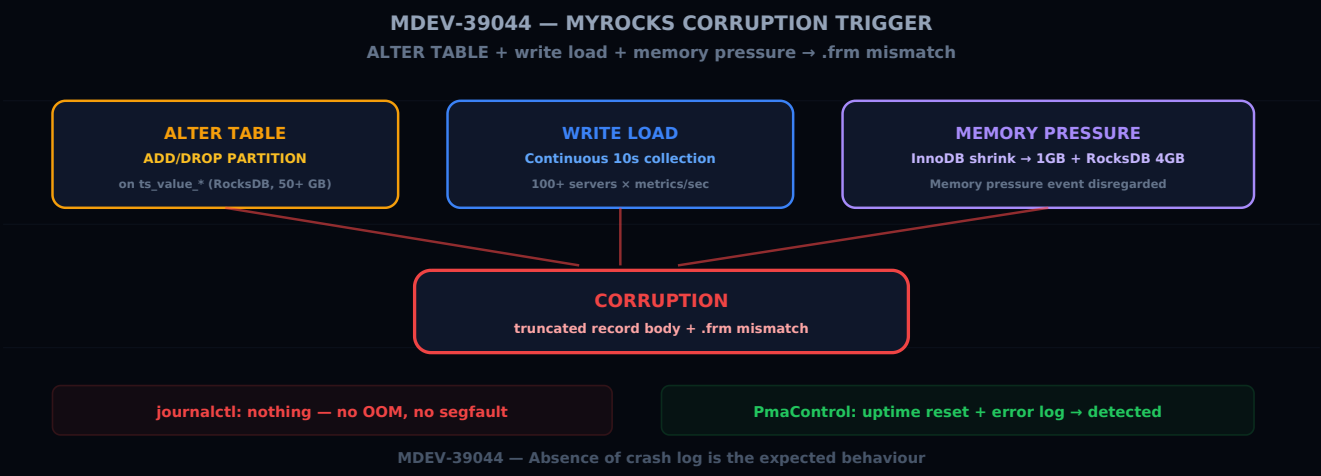
MyRocks sous Charge : Quand ALTER TABLE Provoque une Corruption

Aurélien LEQUOY · 6 mars 2026

MARIADB ROCKSDB CORRUPTION DDL INCIDENT-RESPONSE MDEV-39044



Le contexte

Le 6 mars 2026, un serveur MariaDB 10.11.15 de production supervisé par PmaControl a subi un **incident majeur**. Contrairement aux crashs habituels (OOM, segfault), celui-ci présentait des symptômes inédits :

```
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
```

Le serveur a redémarré en boucle plusieurs fois avant de se stabiliser, avec des erreurs `.frm mismatch` sur plusieurs tables time-series.

Le ticket MDEV-39044

Après investigation, nous avons corrélé cet incident avec le ticket MariaDB **MDEV-39044** :

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

Ce que le ticket décrit

Le ticket documente un scénario de corruption reproductible :

1. **Tables RocksDB partitionnées volumineuses** — exactement ce que PmaControl utilise pour les métriques (tables `ts_value_*` partitionnées par jour)
2. `ALTER TABLE` **sous charge d'écriture** — ajout de partitions pendant que l'application écrit en continu
3. **Pression mémoire InnoDB simultanée** — les tables InnoDB et RocksDB cohabitent sur le même serveur
4. **Aucune trace kernel** — pas d'OOM killer, pas de segfault, pas de crash log

Pourquoi c'est insidieux

Le point le plus dangereux du ticket : **l'absence de crash log est le comportement attendu dans ce scénario**. Le serveur redémarre, fait un `InnoDB crash recovery`, mais les métadonnées RocksDB sont corrompues (`.frm mismatch`).

Un DBA qui ne regarde que `journalctl` ou `dmesg` ne trouvera rien. Il classera l'incident comme "restart inexpliqué" et passera à autre chose.

Notre cas concret

Tables impactées

Toutes des tables RocksDB partitionnées par jour, massivement sollicitées en écriture :

- `ts_value_general_int` — métriques entières (status variables, compteurs)
- `ts_value_general_json` — métriques JSON complexes
- `ts_mysql_digest_stat` — statistiques de requêtes (digests)
- `ts_value_general_text` — métriques textuelles
- `ts_value_slave_int` — métriques de réplication
- `ts_value_slave_text` — états de réplication détaillés

Le déclencheur probable

PmaControl maintient les partitions de ces tables automatiquement : ajout de la partition du jour suivant, suppression des partitions expirées. Ce sont des `ALTER TABLE ... ADD PARTITION` / `DROP PARTITION` exécutés sur des tables de plusieurs dizaines de Go, **pendant que les workers d'aspiration écrivent en continu** (toutes les 10 secondes par serveur supervisé).

Les signaux de pression mémoire

Avant le crash, le log MariaDB montre :

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

Le ticket MDEV-39044 cite explicitement ce motif comme facteur aggravant. La pression mémoire InnoDB ne cause pas directement la corruption, mais elle crée le contexte dans lequel le DDL RocksDB devient non atomique.

Comment PmaControl a détecté l'incident

1. **Reset d'uptime** détecté en 10 secondes via la série temporelle `ts_variable.uptime`
2. **Alerte Telegram** envoyée immédiatement
3. **Corrélation automatique** avec l'error log : détection des signatures `crash recovery` + `truncated record body`
4. **Analyse rétrospective** : les métriques de l'heure précédente (threads, mémoire, CPU) étaient normales — confirmant que ce n'est pas un problème de charge classique

Recommandations

Actions immédiates

1. **Ne pas exécuter de DDL sur des tables RocksDB sous charge d'écriture**. Planifier les `ALTER TABLE ... ADD/DROP PARTITION` pendant les fenêtres de faible activité.
2. **Monitorer les erreurs** `.frm` dans l'error log. C'est le premier indicateur de corruption post-DDL.
3. **Suivre le ticket MDEV-39044** pour un correctif officiel.

Actions structurelles

4. **Séparer les moteurs** : si possible, ne pas mélanger InnoDB et RocksDB sur le même serveur pour les tables critiques.
5. **Envisager la migration des tables chaudes vers InnoDB**. RocksDB est excellent pour l'écriture séquentielle, mais ses DDL ne sont pas atomiques sous charge.
6. **Dimensionner la mémoire** pour éviter la pression InnoDB qui aggrave le problème. Voir notre article sur l'OOM killer pour le calcul du pire cas.

Ce que ce n'est pas

- Ce n'est **pas** un problème de hardware (disque, RAM)
- Ce n'est **pas** un problème de configuration MySQL (les paramètres sont corrects)
- Ce n'est **pas** reproductible à la demande (c'est une race condition dans le moteur RocksDB/DDL)

C'est un **bug moteur** documenté par MariaDB eux-mêmes.

Conclusion

MDEV-39044 est un rappel que l'utilisation de moteurs de stockage alternatifs (RocksDB, TokuDB) sur des workloads de production nécessite une vigilance particulière sur les DDL. L'absence de crash log ne signifie pas l'absence de corruption.

PmaControl détecte ces incidents via la surveillance `uptime` + corrélation error log, là où les outils classiques ne voient rien.