Une seule lecture MariaDB pour reconnaître un relevé d’alerte déjà enregistré

Aurélien LEQUOY · 21 novembre 2026

PMACONTROL MARIADB MYROCKS PERFORMANCE SQL BENCHMARK

PmaControl reçoit parfois deux fois le même relevé d’alerte. Pour reconnaître ce doublon, le logiciel cherchait la date du dernier relevé, puis le point reçu. **Chercher directement ce point suffit.**

Sur cinq contrôles réels, la médiane des lectures SQL passe de **6,645 à 0,363 ms** (−95 %). En laboratoire, la médiane de la transaction complète passe de **2,554 à 0,904 ms**, verrou et écritures compris. Le correctif est testé et attend son déploiement.

Le cas des fuseaux horaires, requête par requête

Le 21 novembre à 22 h 29 min 19 s, heure de Paris, le contrôle d’alignement des fuseaux horaires signale une alerte. Que fait le logiciel si ce même relevé lui parvient à nouveau ? L’ancienne version commence par la requête ci-dessous. Le `?` représente l’empreinte du contrôle, une valeur identique dans les deux requêtes.

```
EXPLAIN PARTITIONS
SELECT observed_at
FROM pmc_alert_level_point
WHERE fingerprint = UNHEX(?)
ORDER BY observed_epoch DESC, observed_at DESC
LIMIT 1;
```

Le code utilisait uniquement la présence de la ligne pour savoir si le contrôle avait un historique. Il cherchait ensuite le relevé exact et lisait son état. La version retenue commence directement ici. Si le point existe avec le même état, elle a reconnu le rejeu. Un état contradictoire au même instant déclenche une erreur.

```
EXPLAIN PARTITIONS
SELECT is_open
FROM pmc_alert_level_point
WHERE fingerprint = UNHEX(?)
```

```
  AND observed_epoch = 1795296559
  AND observed_at = '2026-11-21 22:29:19.000000';
```

Plans relevés sur le serveur pour ce même contrôle. La première colonne de résultats correspond à la lecture supprimée. La seconde montre la lecture qui suffit au rejeu.

EXPLAIN PARTITIONS	Dernière date	Point exact
Accès (`type`)	`range`	`const`
Index (`key`)	`PRIMARY`	`PRIMARY`
Octets utilisés (`key_len`)	32	48
Référence (`ref`)	—	`const,const,const`
Lignes estimées (`rows`)	8 617	1
Partitions au plan	45	1 (`p740307`)
Extra	`Using where; Using index`	—

Les deux requêtes utilisent la clé primaire. Avec `range`, MariaDB cherche dans la plage de clés de cette empreinte. Seuls les 32 octets de `fingerprint` sont fixés et 45 partitions figurent au plan. Avec `const`, l'empreinte, l'horodatage Unix (`observed_epoch`) et la date à la microseconde (`observed_at`) fixent les 48 octets de la clé. MariaDB identifie alors le jour et l'unique point possible dans cette partition.

Les 8 617 lignes de `rows` sont une estimation. Lors d'un `ANALYZE FORMAT=JSON`, l'itérateur de la première requête a renvoyé une ligne (`r_rows = 1`, `r_loops = 1`), conformément au `LIMIT 1`. Le nombre d'opérations internes dans les partitions reste inconnu. La fiche du digest comptait **1 643 071 appels** entre le 19 et le 21 novembre, pour **8 h 08 min** de temps SQL cumulé.

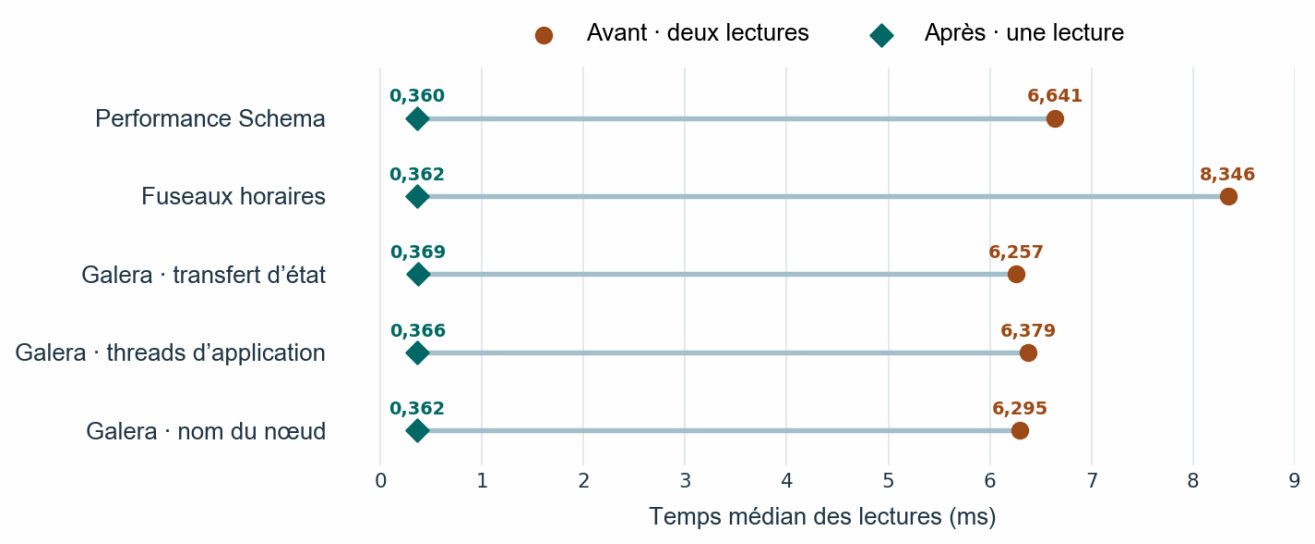
Ouvrir les 20 plans MariaDB complets des cinq contrôles (texte)

Les cinq contrôles du test

La fiche du digest montrait cinq exécutions de la même requête avec cinq valeurs de `fingerprint`. Chaque valeur identifie une condition surveillée. Nous avons retrouvé son nom dans le relevé correspondant. Les lettres entre parenthèses permettent de retrouver les mêmes lignes dans les CSV.

Contrôle (repère CSV)	Code de la règle	Points en alerte / total
Activation de Performance Schema (A)	`AdvisorPerformanceSchema`	0 / 5 692
Alignement des fuseaux horaires (B)	`AdvisorTimeZoneAlignment`	5 694 / 5 695
Galera · méthode de transfert d’état (SST) (C)	`AdvisorWsrepSstMethod`	0 / 5 692
Galera · threads d’application des transactions (D)	`AdvisorWsrepSlaveThreads`	0 / 5 692
Galera · nom du nœud (E)	`AdvisorWsrepNodeName`	0 / 5 692

Les cinq relevés rejoués portent la même heure, le 21 novembre à 22 h 29 min 19 s. Seul le contrôle des fuseaux était en alerte à cet instant. Sur le graphique, chaque trait relie la médiane de l’ancien parcours SQL à celle du parcours qui commence par le point exact.



Chaque contrôle a été mesuré 50 fois par version sur le serveur observé. La dernière ligne regroupe les 250 mesures de la version. Les deux tableaux donnent les temps SQL en millisecondes.

Rejouer un point connu · de deux lectures à une

Contrôle	Moyenne avant	Moyenne après	Médiane avant	Médiane après
Activation de Performance Schema	7,171	0,372	6,641	0,360

Contrôle	Moyenne avant	Moyenne après	Médiane avant	Médiane après
Alignement des fuseaux horaires	8,808	0,400	8,346	0,362
Galera · méthode de transfert d’état (SST)	7,372	0,575	6,257	0,369
Galera · threads d’application des transactions	6,970	0,389	6,379	0,366
Galera · nom du nœud	7,132	0,374	6,295	0,362
Les cinq contrôles	7,491	0,422	6,645	0,363

Ajouter un point à un historique existant · de quatre lectures à trois

Contrôle	Moyenne avant	Moyenne après	Médiane avant	Médiane après
Activation de Performance Schema	39,832	29,522	29,189	26,465
Alignement des fuseaux horaires	41,518	32,461	35,533	29,123
Galera · méthode de transfert d’état (SST)	39,431	31,138	31,509	28,157
Galera · threads d’application des transactions	36,855	31,667	33,790	25,943
Galera · nom du nœud	39,696	31,043	35,195	28,511
Les cinq contrôles	39,466	31,166	33,243	27,975

Pourquoi l’ajout d’un nouveau point gagne moins

Un relevé reçu en retard peut se placer entre deux points déjà enregistrés et modifier les bornes d’un incident. Si la clé exacte est absente, le logiciel cherche donc le point précédent et le point suivant dans l’ordre `(observed_epoch, observed_at)`. Dès qu’il trouve un voisin, il sait aussi que l’historique existe.

Pour les fuseaux horaires, voici les plans de ces deux lectures avec le paramètre d'ajout du banc d'essai, daté du 22 novembre à 22 h 29 min 19 s. Leurs plans incluent toujours les 45 partitions. Cela explique pourquoi l'ajout bénéficie moins de la lecture supprimée.

Point cherché	Accès	Index	Partitions	Lignes estimées
Précédent	`range`	`PRIMARY`	45	8 653
Suivant	`range`	`PRIMARY`	45	81

V1 est la version initiale. **V2** supprime la lecture de la dernière date et utilise la recherche du point exact puis, si nécessaire, celle de ses voisins. Si les trois recherches sont vides, le logiciel vérifie si une ancienne installation a laissé un état à réimporter. Dans la première campagne, cette restauration est passée de 9,665 à 12,963 ms en médiane parce que V2 cherchait à nouveau des voisins dans un historique qu'elle venait de trouver vide.

V3, la version retenue, traite ce cas précis. Si l'historique est vide et qu'il n'existe aucun ancien incident, elle restaure directement l'unique relevé sain mémorisé, avec sa date, son epoch et son contenu. Elle vérifie ensuite le point entrant sous le même verrou.

Transaction complète en laboratoire, moyenne / médiane en ms. Le chronomètre PHP couvre la prise du verrou, les lectures, les écritures, le `COMMIT` et la libération du verrou.

Opération	V1 · initiale	V2 · lecture supprimée	V3 · restauration optimisée
Rejouer un point connu	3,954 / 2,554	1,586 / 0,924	1,518 / 0,904
Ajouter à un historique existant	23,907 / 17,297	23,516 / 13,954	22,895 / 14,017
Créer le premier point	28,373 / 10,477	20,031 / 8,689	22,708 / 8,660
Restaurer un ancien point sain	23,609 / 10,422	29,137 / 13,423	17,151 / 7,916

Pour l'ajout courant, V2 garde un léger avantage en médiane (13,954 contre 14,017 ms). La restauration départage les deux versions, avec 13,423 ms pour V2 et 7,916 ms pour V3. Les trois versions ont produit les mêmes décisions, points, incidents, intervalles et derniers états.

Peut-on faire de fingerprint la clé primaire ?

Le 20 novembre, nous avons compté **543 719 points pour 9 478 empreintes**. Une même empreinte apparaissait 83 fois dans la journée. L'empreinte identifie le contrôle. Le triplet `(fingerprint, observed_epoch, observed_at)` identifie un relevé précis de ce contrôle.

`SHOW INDEX` estimait 3 672 630 valeurs pour le premier préfixe de la clé et 2 448 420 pour les deux préfixes plus longs. Ajouter une colonne conserve ou augmente le nombre réel de combinaisons distinctes. La baisse affichée vient des estimations du moteur. Sur la copie partitionnée par jour, `PRIMARY KEY(fingerprint)` a donné l'erreur 1503 car toute clé unique doit inclure `observed_at`. Sur la copie sans partitions, la deuxième observation de la même empreinte a donné l'erreur de doublon 1062. MariaDB · SHOW INDEX · MariaDB · Partitioning

Les autres façons de tester l'existence d'un historique

Les temps ci-dessous portent sur les cinq contrôles ensemble. Le plan illustré est celui des **fuseaux horaires**. Un accès `ref` recherche les entrées ayant la même valeur pour le préfixe indexé. La flèche indique le recours à l'historique complet si la première recherche échoue. Le tiret désigne une table sans partitions. Chaque série a sa propre mesure de référence.

Série et requête	Accès / index	Partitions	Moy. / méd. (ms)
I · Dernière date	`range / PRIMARY`	45	7,577 / 6,660
I · Dernière date, `FORCE INDEX (PRIMARY)`	`ref / PRIMARY`	45	4,927 / 4,283
I · `SELECT 1 … LIMIT 1`	`ref / level_neighbour`	45	6,361 / 5,434
I · `SELECT EXISTS(…)`	`ref / level_neighbour`	45	6,870 / 5,625
I · Vérifier le dernier état dans l'historique, puis repli	`const / PRIMARY` + `const / PRIMARY` → `ref / level_neighbour`	— + 45 → 45	4,543 / 3,995

Série et requête	Accès / index	Partitions	Moy. / méd. (ms)
II · Dernière date	`range / PRIMARY`	45	6,401 / 5,623
II · Chercher dans la journée, puis repli	`ref / level_neighbour_epoch` → `ref / level_neighbour`	1 → 45	3,739 / 3,411
II · Lire uniquement le dernier état	`const / PRIMARY`	—	0,277 / 0,257

Le dernier état seul a renvoyé la même date dans 250 comparaisons sur 250. Cette table peut toutefois contenir un état hérité d'une ancienne version ou dont les points ont été purgés. Elle ne suffit donc pas à décider si l'historique doit être reconstruit. Parmi les sondes qui vérifient réellement l'historique, commencer par le jour du relevé donne la meilleure médiane de la série II (3,411 ms). Pour un doublon, la lecture exacte du point fournit déjà cette réponse.

Lire le SQL et les plans de toutes les variantes testées (texte)

Ce que changent l'index temporel et le partitionnement

Pour isoler le choix de structure, nous avons rejoué la requête initiale ordonnée sur **621 865 points identiques** dans MariaDB 11.8.10 avec MyRocks. Les plans sont ceux du contrôle des fuseaux. L'index temporel testé s'appelle `when_observed` et couvre `(observed_epoch, observed_at)` .

Structure de la table	Accès / index	Partitions au plan	Moy. / méd. (ms)
Clé composée, découpage journalier	`range / PRIMARY`	45	612,119 / 549,937
Index temporel ajouté	`range / PRIMARY`	45	637,779 / 525,603
Index temporel forcé	`index / when_observed`	45	64,429 / 48,306
Table sans partitions	`range / PRIMARY`	—	14,052 / 7,676

Structure de la table	Accès / index	Partitions auplan	Moy. / méd.(ms)
16 partitions par empreinte	`ref / PRIMARY`	1	0,786 / 0,410

L'optimiseur a conservé `PRIMARY` pour les cinq contrôles après l'ajout de l'index. Quand on force `when_observed`, l'accès `index` parcourt les dates jusqu'à rencontrer l'empreinte. En répartissant les points par empreinte, MariaDB peut cibler une seule partition sur seize, mais la purge par journée doit alors être repensée.

Le laboratoire disposait d'un CPU et de 96 Mio de cache de blocs MyRocks. Sur une première copie de 571 957 points, la référence avait une médiane de 1,350 ms. Sur la copie étendue, elle atteint 549,937 ms. L'écart entre ces campagnes reste inexpliqué. La comparaison porte sur les structures d'une même série. Les temps de migration et de construction des index restent à mesurer. Le temps des écritures pour entretenir l'index secondaire reste également à mesurer.

Protocole et données à télécharger

Nous avons figé les cinq exemples affichés ensemble par la fiche digest. À chacun des **50 tours**, nous avons mélangé l'ordre des cinq paramètres et celui des variantes avec une graine fixe. Cela donne **50 mesures par contrôle et par variante, 250 par variante**. Les premières mesures et les pointes de latence sont incluses. Sur le serveur observé, les comparaisons ont utilisé des instantanés en lecture seule, le cache de résultats désactivé et les temps de `SHOW PROFILES`. Le tirage aléatoire porte sur l'ordre de passage des cinq exemples sélectionnés par la fiche.

Les transactions avec écriture ont été testées sur des copies distinctes à 45 partitions, avec les 28 463 clés historiques des cinq contrôles et du contenu d'alerte synthétique. Les tests couvrent aussi l'arrivée tardive, le changement d'heure, les conflits et la reprise après annulation. Le dossier contient chaque mesure, les formes SQL et les plans complets par contrôle.

Télécharger les mesures, la correspondance des cinq contrôles et les plans SQL (ZIP)

Le choix pour PmaControl

Reprenons le relevé des fuseaux horaires. Sur ses 50 rejeux, les lectures SQL passent de **8,808 à 0,400 ms en moyenne**, et de **8,346 à 0,362 ms en médiane**. **Nous gardons V3 et la clé primaire composée.** Pour ce relevé, une recherche suffit à retrouver l'alerte déjà enregistrée. Si le prochain relevé arrive en retard, ses voisins permettent toujours de le replacer dans

l’historique.