ProxySQL 2.7 → 3.0 : des audits du cache environ 4× plus rapides dans PmaControl

Aurélien LEQUOY · 27 septembre 2026

PMACONTROL PROXYSQL BENCHMARK MARIADB MYSQL OBSERVABILITY

PmaControl BENCHMARK NOTES

ProxySQL 2.7 → 3.0

Same complete cache audit. Same two Admin queries.

ProxySQL 2.7.3
8.112 ms/audit

ProxySQL 3.0.5
2.015 ms/audit

−75.2% mean elapsed time **4.03× faster**

150 timed audits per version · Ubuntu 24.04.4 · 2 vCPU · 4 GiB · PHP 8.4.26

Client-observed Admin workload: SQL + loopback + fetch + PHP. Query-routing throughput is outside scope.

Une belle amélioration observée pendant les benchmarks de PmaControl : le même contrôle complet du cache passe de **8,112 à 2,015 millisecondes en moyenne** entre les builds ProxySQL 2.7.3 et 3.0.5 testés. Cela représente **75,16 % de temps en moins**, soit un temps moyen divisé par **4,03**.

Nous publions ici les chiffres, leur définition, les requêtes et le protocole pour permettre une lecture précise de ce résultat. Il s’agit du contrôle d’efficacité du cache exécuté par PmaControl via l’interface **Admin** de ProxySQL.

Les résultats, à travail identique

Mesure	ProxySQL 2.7.3	ProxySQL 3.0.5	Baisse du temps
Moyenne, ms/audit	8.112204	2.014760	75.16 %
Médiane des moyennes de lots, ms/audit	7.895812	2.053810	73.99 %

Mesure	ProxySQL 2.7.3	ProxySQL 3.0.5	Baisse du temps
Maximum des moyennes de lots, ms/audit	11.914402	2.919234	75.50 %
Pic alloué PHP, Mio	4	4	inchangé

Les temps du tableau sont exprimés en **ms/audit**, sauf la mémoire. La médiane et le maximum portent sur **15 moyennes de lots de 10 audits** : le maximum n’est donc ni le pire appel individuel, ni un percentile de latence. Les fichiers de données conservent la précision originale ; l’infographie arrondit les durées à trois décimales.

Le pic PHP de 4 Mio correspond à la mémoire allouée au processus PHP. Il ne mesure pas la mémoire du serveur ProxySQL.

PmaControl

BENCHMARK NOTES / 27 SEP 2026

ProxySQL 2.7 → 3.0

A nice improvement during PmaControl cache-audit benchmarks.

−75.2%
MEAN ELAPSED TIME PER AUDIT

4.03×
faster complete cache audits

Mean time · complete cache audit

ms/audit · lower is better

Supporting measurements

	2.7.3	3.0.5	UNIT
Median batch mean	7.896	2.054	ms/audit
Maximum batch mean	11.914	2.919	ms/audit

Median / maximum across 15 batch means; each batch averages 10 audits.

What “ms/audit” measures

One complete cache audit = two read-only ProxySQL Admin queries.

- 01 SQL execution
- 02 Loopback + fetch
- 03 PHP audit logic

Client elapsed time. Connection setup, process startup, HTTP and UI rendering are outside the timer.

The benchmark setup

Ubuntu 24.04.4 · 2 vCPU · 4 GiB RAM · PHP 8.4.26

150 timed complete audits per version

15 batches × 10 audits · 3 warmups per batch

Same audit code and SQL · 1 + 7 rows returned per audit

PHP allocated peak: 4 MiB for both versions

Scope: this cache-audit workload on a shared test VM.
These are client-observed Admin timings. Query-routing throughput and standalone server execution time were outside this benchmark.

Source: PmaControl PR #6389 · fixed complete audit on both versions
Builds: 2.7.3-12-g50b7f85 / 3.0.5-60-g7e9e009

Nice improvement, ProxySQL.

Ce que signifie « ms/audit »

Un audit désigne ici **un appel complet à `QueryCacheEffectiveness::run()`**, et non l'ensemble des contrôles d'un serveur. Il effectue deux SELECT en lecture seule : le premier lit les règles de cache actives et leurs compteurs de correspondance ; le second lit les compteurs du cache et la taille configurée. La fixture renvoie **une ligne**, puis **sept lignes**.

Le chronomètre entoure le contrôle PHP. Il inclut l'exécution des requêtes SQL natives, les échanges réseau en loopback, la récupération des résultats et leur traitement par PmaControl. La connexion initiale, le lancement du processus PHP, HTTP et le rendu de l'interface restent hors chronométrage. Ces durées sont donc observées côté client, sans isoler le temps d'exécution interne de ProxySQL.

Voici les deux requêtes, présentées avec une mise en forme lisible ; leur logique est identique sur les deux versions.

```
SELECT r.rule_id, r.cache_ttl, r.digest,
       r.match_digest, r.match_pattern,
       COALESCE(h.hits, 0) AS hits
FROM runtime_mysql_query_rules r
LEFT JOIN stats_mysql_query_rules h
       ON h.rule_id = r.rule_id
WHERE r.active = 1 AND r.cache_ttl > 0
ORDER BY r.rule_id;
```

```
SELECT Variable_Name AS name, Variable_Value AS value
FROM stats_mysql_global
WHERE Variable_Name IN (
    'Query_Cache_count_GET', 'Query_Cache_count_GET_OK',
    'Query_Cache_count_SET', 'Query_Cache_Memory_bytes',
    'Query_Cache_Entries', 'Query_Cache_Purged'
)
UNION ALL
SELECT variable_name, variable_value
FROM global_variables
WHERE variable_name = 'mysql-query_cache_size_MB';
```

Le champ `stats_mysql_query_rules.hits` compte les correspondances d'une règle, pas les hits du cache attribués à cette règle. Les compteurs de cache sont cumulatifs : ils ne donnent pas, seuls, un taux récent. La valeur `mysql-query_cache_size_MB` lue dans MAIN est une cible de purge configurée, pas une limite dure ; sa valeur runtime n'est pas vérifiée par ce contrôle.

Le protocole du benchmark

Le benchmark source utilise une **VM de test partagée sous Ubuntu 24.04.4 LTS**, avec **2 vCPU**, **4 Gio de RAM** et **PHP 8.4.26**. Les connexions utilisent la boucle locale et les binaires ProxySQL natifs suivants :

- `2.7.3-12-g50b7f85` pour la série 2.7 ;
- `3.0.5-60-g7e9e009` pour la série 3.0.

Pour chaque version, la sélection publiée contient **15 lots de 10 audits chronométrés**, après **3 appels d'échauffement par lot** : 150 audits complets par version, 300 au total et 600 SELECT. Le helper PHP et les chaînes SQL sont identiques ; son empreinte et le nombre de lignes retournées sont contrôlés à chaque lot et appel.

L'expérience d'origine alternait l'ancienne et la nouvelle variante PmaControl en 15 paires AB/BA, avec un processus PHP distinct par variante. Cette publication sélectionne uniquement la variante corrigée. Dans chaque processus, ProxySQL 2.7.3 était testé avant 3.0.5 : **l'ordre des versions ProxySQL n'était pas alterné**. Les 600 appels et 900 SELECT de l'expérience complète, ancienne variante incluse, ne sont pas les effectifs de la comparaison présentée ici.

Pourquoi retenir le contrôle complet

Le benchmark accompagnait un correctif fonctionnel : une règle valide portant l'identifiant signé `rule_id=-1`, avec un TTL de 5 000 ms, faisait auparavant arrêter le contrôle après la première lecture. Le résultat indiquait à tort que les statistiques étaient indisponibles ; la seconde requête n'était jamais atteinte.

Comparer cette sortie incomplète au contrôle corrigé ferait varier le travail effectué. Nous retenons donc **le même helper corrigé sur les deux versions de ProxySQL**, avec les deux lectures et les observations complètes. Le gain publié compare les builds ProxySQL dans ce scénario ; il n'est pas attribué au correctif de validation de PmaControl. Les mesures existantes ont été revérifiées, sans lancer un nouveau benchmark pour cet article.

Ce que cette amélioration permet de conclure

Dans cet environnement, la moyenne, la médiane des moyennes de lots et leur maximum sont tous plus faibles avec le build 3.0 testé. Le pic de mémoire allouée PHP reste inchangé. C'est une

amélioration mesurable pour ce travail de supervision.

Le résultat concerne une VM partagée, des jeux de résultats réduits et un contrôle précis. Il ne certifie ni le débit des requêtes applicatives, ni la latence du routage frontend, ni un gain identique sous forte concurrence. Le test n'isole pas les phases serveur prepare/step/copy/finalize, ne prouve pas un budget SQL inférieur à 1 ms et n'évalue pas la capacité de production ou la durabilité. Il ne fournit pas d'intervalle de confiance et l'ordre des versions est fixe.

Ces limites n'enlèvent rien au constat : **le contrôle complet du cache prend environ quatre fois moins de temps en moyenne dans cette mesure**. Nice improvement, ProxySQL !

Recalculer les résultats

Les données publiques contiennent les 30 lots retenus, les 300 durées individuelles, les cardinalités et les métadonnées du protocole. Les chemins internes et les messages de diagnostic ont été retirés ; les mesures sont inchangées. Le script Python utilise uniquement la bibliothèque standard et recalcule le tableau : il ne relance pas le benchmark.

La baisse se calcule avec `100 × (1 − temps_3.0 / temps_2.7)` ; le rapport avec `temps_2.7 / temps_3.0`. Les lots ont tous la même taille. L'empreinte ci-dessous identifie le helper réellement chargé ; le commit est celui de la PR contenant cette implémentation.

```
curl -fsSLO https://pmacontrol.com/downloads/proxysql-2.7-vs-3.0/benchmark-data.json
curl -fsSLO https://pmacontrol.com/downloads/proxysql-2.7-vs-3.0/analyze.py
python3 analyze.py benchmark-data.json
```

```
PmaControl revision:
b4e7b4c6ca72cf27077ddbdb1f2c89af05800b5a
QueryCacheEffectiveness helper SHA-256:
c47a169935f665259554bd0fbaa85e90af9cbcd3983bf7ad3548e2caaf63d4dc
```

Visuels, partage et sources

- Infographie complète en PNG, 1600 × 2000
- Source SVG modifiable, tous les éléments inclus
- Données du benchmark en JSON

- Script de recalcul Python
- Post LinkedIn en anglais
- Archive du visuel et de ses scripts de génération

Le visuel est en anglais pour le partage ; les explications et mesures sont intégralement reprises dans cet article. Le bouton PDF de la page permet également d’exporter l’article avec ses images.

Traçabilité du travail d’origine : PR PmaControl #6389 et issue de communication #6390. Ces références Forgejo demandent un accès au dépôt ; les fichiers téléchargeables ci-dessus sont publics.