ProxySQL 2.7 → 3.0: około 4× szybszy audyt pamięci podręcznej w PmaControl

Aurélien LEQUOY · September 27, 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.

Wyraźna poprawa zaobserwowana podczas benchmarków PmaControl: ten sam kompletny test pamięci podręcznej trwał średnio **8,112 milisekundy w wersji 2.7.3 i 2,015 milisekundy w wersji 3.0.5** w badanych kompilacjach ProxySQL. Nowsza kompilacja potrzebowała **o 75,16% mniej czasu**, czyli średni czas był **4,03 raza krótszy**.

Poniżej publikujemy wyniki, ich definicje, zapytania i metodę pomiaru. Badane obciążenie to kontrola skuteczności pamięci podręcznej wykonywana przez PmaControl przez interfejs **Admin** ProxySQL.

Wyniki przy takim samym zakresie pracy

Miara	ProxySQL 2.7.3	ProxySQL 3.0.5	Spadek czasu
Średnia, ms/audit	8.112204	2.014760	75.16 %
Mediana średnich partii, ms/audit	7.895812	2.053810	73.99 %

Miara	ProxySQL 2.7.3	ProxySQL 3.0.5	Spadek czasu
Maksymalna średnia partii, ms/audit	11.914402	2.919234	75.50 %
Szczyt alokacji PHP, MiB	4	4	bez zmian

Czasy w tabeli podano w **ms/audit**, z wyjątkiem pamięci. Mediana i maksimum dotyczą **15 średnich z partii po 10 audytów**. Maksimum nie jest zatem czasem najwolniejszego pojedynczego wywołania ani percentylem opóźnienia. Pliki danych zachowują pierwotną precyzję, a infografika zaokrągla czasy do trzech miejsc po przecinku.

Szczytowe 4 MiB oznacza pamięć przydzieloną procesowi PHP. Nie jest to pomiar pamięci serwera 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.

Co oznacza „ms/audit”

Audyt oznacza tutaj **jedno kompletne wywołanie** `QueryCacheEffectiveness::run()`, a nie wszystkie kontrole serwera. Wykonuje ono dwa zapytania SELECT tylko do odczytu: pierwsze pobiera aktywne reguły pamięci podręcznej i liczniki dopasowań, drugie — liczniki pamięci podręcznej i skonfigurowany rozmiar. Dane testowe zwracają **jeden wiersz**, a następnie **siedem wierszy**.

Pomiar obejmuje całą kontrolę PHP: wykonanie natywnych zapytań SQL, komunikację przez interfejs loopback, pobranie wyników i przetwarzanie w PmaControl. Nawiązanie połączenia, uruchomienie procesu PHP, HTTP i renderowanie interfejsu nie są mierzone. Są to czasy obserwowane przez klienta, bez wyodrębnienia czasu wykonania wewnątrz ProxySQL.

Poniżej znajdują się oba zapytania, sformatowane dla czytelności. Ich logika jest taka sama w obu wersjach.

```
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';
```

Pole `stats_mysql_query_rules.hits` zlicza dopasowania reguły, a nie trafienia pamięci podręcznej przypisane do tej reguły. Liczniki pamięci podręcznej są kumulacyjne i same nie określają niedawnego współczynnika trafień. Wartość `mysql-query_cache_size_MB` odczytana z MAIN jest skonfigurowanym celem mechanizmu usuwania wpisów, a nie twardym limitem; kontrola nie

weryfikuje jej wartości runtime.

Metodyka benchmarku

Benchmark źródłowy wykonano na **współdzielonej maszynie testowej z Ubuntu 24.04.4 LTS**, **2 vCPU**, **4 GiB RAM** i **PHP 8.4.26**. Połączenia korzystały z loopback i następujących natywnych kompilacji ProxySQL:

- `2.7.3-12-g50b7f85` dla serii 2.7;
- `3.0.5-60-g7e9e009` dla serii 3.0.

Dla każdej wersji opublikowany zestaw obejmuje **15 partii po 10 mierzonych audytów**, poprzedzonych **3 wywołaniami rozgrzewającymi na partię**: 150 kompletnych audytów na wersję, łącznie 300 audytów i 600 SELECT. Kod pomocniczy PHP i teksty zapytań SQL są identyczne; jego skrót oraz liczba zwracanych wierszy są sprawdzane w każdej partii i wywołaniu.

Pierwotny eksperyment przeplatał stary i poprawiony wariant PmaControl w 15 parach AB/BA, z osobnym procesem PHP dla każdego wariantu. Ta publikacja uwzględnia wyłącznie wariant poprawiony. W każdym procesie najpierw uruchamiano ProxySQL 2.7.3, potem 3.0.5: **kolejność wersji ProxySQL nie była zmieniana**. Łączne 600 wywołań i 900 SELECT z całego eksperymentu, obejmującego stary wariant, nie stanowią liczebności prezentowanego porównania.

Dlaczego porównujemy kompletną kontrolę

Benchmark towarzyszył poprawce funkcjonalnej: poprawna reguła z identyfikatorem ze znakiem `rule_id=-1` i TTL 5 000 ms powodowała wcześniej zakończenie kontroli po pierwszym odczycie. Kontrola błędnie zgłaszała niedostępność statystyk; do drugiego zapytania nigdy nie dochodziło.

Porównanie takiego niepełnego wyniku z poprawioną kontrolą oznaczałoby porównanie różnego zakresu pracy. Dlatego używamy **tego samego poprawionego kodu na obu wersjach ProxySQL**, z dwoma odczytami i kompletnymi obserwacjami. Opublikowany zysk dotyczy kompilacji ProxySQL w tym scenariuszu, a nie poprawki walidacji PmaControl. Istniejące pomiary zostały ponownie zweryfikowane; na potrzeby artykułu nie uruchomiono nowego benchmarku.

Co wynika z tej poprawy

W tym środowisku średnia, mediana średnich partii i maksymalna średnia partii są niższe dla badanej kompilacji 3.0. Szczyt alokacji pamięci PHP pozostał bez zmian. Jest to mierzalna poprawa pracy związanej z monitoringiem.

Wynik dotyczy współdzielonej VM, niewielkich zestawów wyników i jednej konkretnej kontroli. Nie potwierdza przepustowości zapytań aplikacyjnych, opóźnienia routingu frontend ani identycznego zysku przy dużej współbieżności. Test nie rozdziela faz serwera prepare/step/copy/finalize, nie dowodzi budżetu SQL poniżej 1 ms i nie bada wydajności produkcyjnej ani trwałości danych. Nie wyznaczono przedziału ufności, a kolejność wersji jest stała.

W tych granicach **kompletna kontrola pamięci podręcznej trwała średnio około cztery razy krócej**. Dobra robota, ProxySQL!

Ponowne obliczenie wyników

Publiczne dane zawierają 30 wybranych partii, 300 pojedynczych czasów, liczebności wyników i metadane metodyki. Usunięto ścieżki wewnętrzne i komunikaty diagnostyczne; czasy są niezmienione. Skrypt Python używa tylko biblioteki standardowej i przelicza tabelę — nie uruchamia ponownie benchmarku.

Spadek czasu obliczamy jako `100 × (1 − czas_3.0 / czas_2.7)`, a stosunek jako `czas_2.7 / czas_3.0`. Wszystkie partie mają tę samą wielkość. Poniższy skrót identyfikuje rzeczywiście załadowany kod pomocniczy; commit wskazuje rewizję PR zawierającą tę implementację.

```
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
```

Grafiki, udostępnianie i źródła

- Pełna infografika PNG, 1600 × 2000
- Edytowalny plik SVG ze wszystkimi elementami

- Dane benchmarku JSON
- Skrypt Python przeliczający wyniki
- Wpis LinkedIn po angielsku
- Archiwum grafiki ze skryptami generującymi

Grafika jest po angielsku, aby ułatwić udostępnianie; artykuł zawiera wszystkie objaśnienia i pomiary. Przycisk PDF na stronie eksportuje również artykuł wraz z obrazami.

Prace źródłowe: PmaControl PR #6389 i issue dotyczące publikacji #6390. Te odnośniki Forgejo wymagają dostępu do repozytorium; powyższe pliki są publiczne.