Pamięć podręczna i indeksy są, a strona nadal działa wolno: analiza MariaDB w PmaControl

Aurélien LEQUOY · October 9, 2026

PMACONTROL MARIADB PERFORMANCE SQL BENCHMARK ARCHITECTURE



„Rysunki są już w pamięci podręcznej. Dlaczego wyświetlenie strony Architektura nadal trwa kilka sekund?”

To pytanie doprowadziło nas do bardzo krótkiej poprawki w PmaControl. **Nie brakowało indeksu. Nie było również potrzeby ponownego obliczania rysunków.** MariaDB wybrała kosztowny sposób wyszukania generacji wykresów przeznaczonych do wyświetlenia.

Oto metryki, SQL i ograniczenia tego porównania. Rozróżniamy **klienta**, który przedstawił problem, od instancji testowej, która już szybko wykonywała początkowe zapytanie.

Aktualizacja z 10 października 2026 r. Poniżej dodano pomiary SQL z godziny 13:29. Czasy pobierania zmierzono ponownie około 13:45 z serwera w sieci lokalnej, ponieważ połączenie użyte do pomiarów z 9 października było niestabilne. Każda kampania zachowuje własne środowisko, liczbę wykresów i zakres: SQL, pobieranie HTTP albo ładowanie w Chromium.

Wyjaśnienie na przykładzie pudełka z rysunkami

Wyobraź sobie bibliotekę pełną pudeł. Każde pudełko zawiera rysunki jednej generacji topologii. Rysunki są już gotowe: to jest nasza pamięć podręczna SVG.

Aby wyświetlić stronę, aplikacja musi jeszcze znaleźć właściwe pudełko, wyjąć z niego rysunki i rozmieścić je na ekranie. Katalog biblioteki już istnieje: to indeks SQL.

Jednak posiadanie katalogu nie wystarczy. W powolnym przypadku MariaDB zdecydowała się na iterację przez miliony wpisów, aby przygotować pośrednią listę dopasowań. Do interesującej nas weryfikacji wystarczy znalezienie **pojedynczego dopasowania**.

Poprawka wyraża tę kontrolę wprost. Zwracane są te same rysunki, a wyszukiwanie omija kosztowną drogę okrężną.

To, co zostało już zapisane w pamięci podręcznej

Pliki SVG są zapisywane w formacie `dot3_graph`. Tabela `dot3_cluster` wiąże je z generacjami topologii. Strona pobiera już wszystkie wykresy za pomocą **pojedynczego zapytania SQL**, które zwraca w szczególności:

```
SELECT dg.id, dg.svg, dg.height, dg.width,
       (dg.height * dg.width) AS area,
       dc.date_inserted AS date_refresh
FROM dot3_cluster dc
JOIN dot3_graph dg ON dg.id = dc.id_dot3_graph
-- Warunek wybiera generację gotową do wyświetlenia.
```

Przeglądarka otrzymuje więc rysunek, jego wysokość, szerokość i pole powierzchni. Kod w `App/Webroot/js/Architecture/topology.js` już rozmieszcza rysunki w przeglądarce. Ta poprawka nie zmienia tego algorytmu.

Skrytka pozwala uniknąć rysowania przy każdej wizycie. Nie unika wyszukiwania, przesyłania i wyświetlania rysunków. Ta pamięć podręczna SVG jest oddzielna od pamięci podręcznej zapytań MariaDB i pamięci podręcznej HTTP przeglądarki.

Nie, nie brakowało żadnego indeksu

Powolna instancja miała już ten unikalny indeks, co potwierdził `SHOW INDEX FROM dot3_cluster`:

```
-- Istniejący indeks przedstawiony jako definicja:
UNIQUE KEY id_dot3_information_2
    (id_dot3_information, id_dot3_graph)
```

Wyszukiwanie filtruje po `id_dot3_information`, czyli pierwszej kolumnie indeksu. Pozwala to odnaleźć klastry danej generacji.

Problem tkwił w **planie wykonania**, czyli strategii wybranej przez MariaDB do wykonania SQL. Początkowa wersja używała `EXISTS`, aby sprawdzić, czy generacja ma co najmniej jeden klaster. W tym środowisku wybór ten prowadził do materializacji: MariaDB przygotowywała wynik pośredni, skanując około **4,33 miliona historycznych wpisów**.

Nie oznacza to, że przeglądarka otrzymała cztery miliony wykresów. Otrzymała **61 wykresów**. Nadmiarowa praca odbywała się w bazie danych przed pobraniem użytecznego wyniku. Liczba 4,33 miliona opisuje rząd wielkości skanowania ustalonego podczas pierwszej diagnozy, a nie liczbę wierszy przesłanych do PHP. Pomiar z 10 października pokazuje około **5,2 miliona wierszy** w tabeli tej instancji; wielkości danych z pierwotnego artykułu i nowej kampanii nie są więc identyczne.

Indeks można wykorzystać podczas bardzo szerokiego skanowania. Dlatego obecność indeksu sama w sobie nie gwarantuje krótkiego dostępu.

Konfiguracja i tabele MariaDB, InnoDB / RocksDB

Oba środowiska używają **MariaDB 11.8.9**, lecz z różnymi pakietami: **11.8.9-deb13 u klienta**, zgodnie z jego pomiarem z 10 października, oraz **11.8.9-MariaDB-deb12-log na instancji testowej**, co zweryfikowano bezpośrednio na serwerze. Klient używa PHP 8.4 do odtwarzania SQL, a instancja testowa FrankenPHP / PHP 8.5.9.

Poniższy diagram obejmuje **16 tabel aplikacji używanych przez 16 zapytań na stronie**, poza trzema tabelami w zapytaniu SVG. Przydatne kolumny, klucze i silniki pochodzą z rzeczywistego diagramu instancji testowej, zapisanego **10 października o 15:06**. Sprawdzanie obecności tabeli wysyła również zapytanie do `information_schema.tables`, widoku metadanych, który nie jest tabelą aplikacji, którą można sklasyfikować jako InnoDB lub RocksDB.

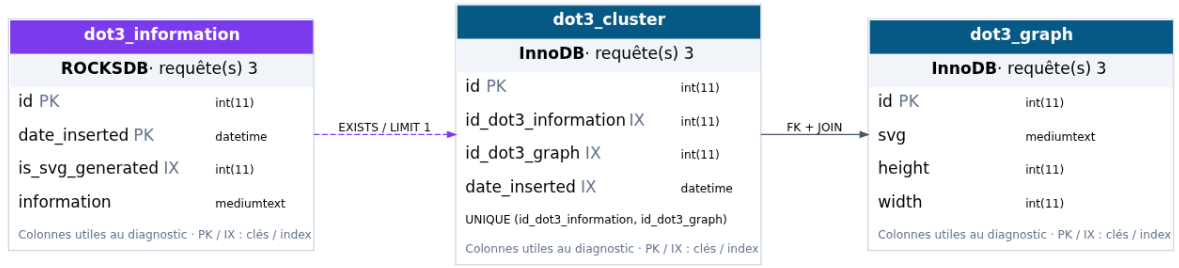
Tables de la page Architecture · partie 1/2 · instance de test

7 tables InnoDB (bleu) · 1 table RocksDB (violet) · relevé du 10 octobre à 15 h 06

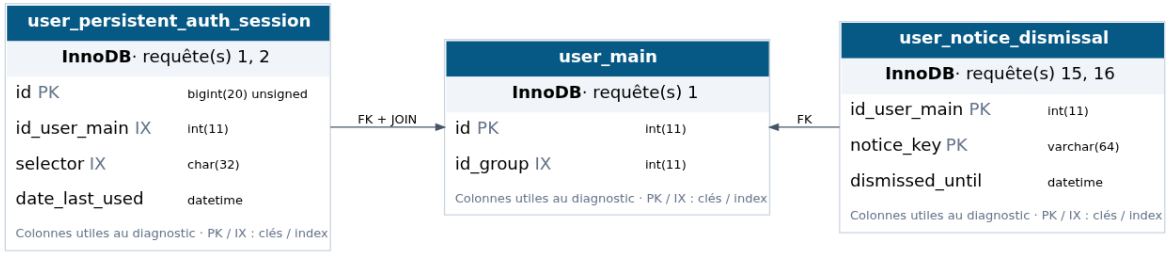
Trait plein : clé étrangère déclarée · Pointillé : jointure logique du SQL

Colonnes utiles au diagnostic ; schéma complet des 16 tables disponible en SVG.

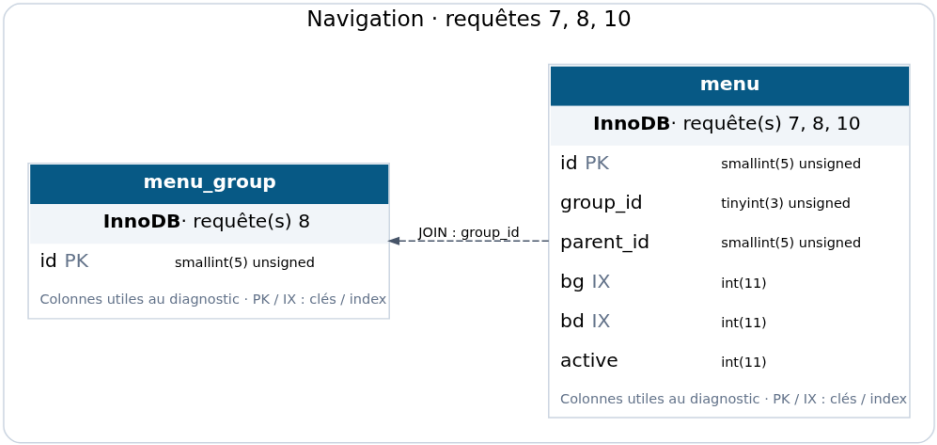
Sélection et lecture des SVG · requête 3



Session et masquage de bandeau · requêtes 1, 2, 15, 16



Navigation · requêtes 7, 8, 10





Otwórz duży diagram wektorowy. Linia ciągła wskazuje zadeklarowany klucz obcy; linia przerywana wskazuje połączenie logiczne używane przez SQL. Tabele bez wyświetlanego linku są odpytywane osobno. Rysunek przedstawia relacje, a nie kolejność wykonywania zapytań.

Tabela zapytania SVG na instancji testowej	Zaobserwowany silnik
`dot3_information`	**RocksDB**
`dot3_cluster`	**InnoDB**
`dot3_graph`	**InnoDB**

Pozostałych trzynaście tabel na stronie znajduje się w InnoDB. Pełny rysunek powyżej oraz inwentarz silników i rozmiarów do pobrania szczegółowo je wyszczególniają.

Zapytanie SVG przechodzi więc przez **dwa silniki** na instancji testowej. Indeks dostępności `dot3_information` należy do RocksDB, a unikalny indeks `dot3_cluster` i dane SVG w `dot3_graph` do InnoDB. Poprawka zachowuje te silniki i indeksy. Kampania porównuje dwie postacie SQL; nie mierzy konwersji tabel z InnoDB do RocksDB. Pomiar klienta nie podaje użytych silników, dlatego nie przypisujemy jego środowisku schematu zaobserwowanego w teście.

Zanonimizowana konfiguracja `.cnf` instancji testowej łączy sekcje serwera z dołączonych plików w kolejności ich wczytywania. Usunięto sekrety i identyfikatory sieciowe. Ponieważ kolejne pliki mogą nadpisywać ustawienia, faktycznie odczytane zmienne globalne pokazują aktywne wartości, w tym **2 GiB puli buforów InnoDB**, **2 GiB pamięci podręcznej bloków RocksDB**, **maksymalnie 128 połączeń**, **64 MiB dla `tmp_table_size` i `max_heap_table_size`** oraz **wyłączoną pamięć podręczną zapytań**.

Parametry zebrano o **15:14**, po pomiarach z 13:45; niektóre zmienne mogą zmieniać się dynamicznie. Plik `.cnf` opisuje zaobserwowaną konfigurację i jej nadpisania, z usuniętymi wartościami wrażliwymi; nie należy stosować go bezpośrednio. Pomiar klienta nie zawiera jego konfiguracji `.cnf`.

Modyfikacja SQL przed i po

Zmiana zlokalizowana jest w `App/Library/Dot3LatestGeneration.php`, w ramach wyboru najnowszej generacji.

Przed:

```
AND EXISTS (
    SELECT 1
    FROM dot3_cluster dc_ready
    WHERE dc_ready.id_dot3_information = di.is_svg_generated
)
```

Po:

```
AND (
    SELECT dc_ready.id_dot3_information
```

```
    FROM dot3_cluster dc_ready
    WHERE dc_ready.id_dot3_information = di.is_svg_generated
    LIMIT 1
) IS NOT NULL
```

Dlaczego tutaj jest to równoważne? Żądanie zewnętrzne już narzuca `di.is_svg_generated > 0`. Jeśli istnieje pasujący klaster, podzapytanie zwraca ten dodatni identyfikator. W przeciwnym razie podzapytanie skalarne bez wyników zwraca `NULL`. Dlatego warunek zawsze sprawdza istnienie klastra dla tej generacji.

Podzapytanie skalarne pozwala uniknąć planu materializacji obserwowanego w przypadku `EXISTS`. Silnik może przeszukać generację w indeksie i zatrzymać się na pierwszym dopasowaniu. Rozwiązania awaryjne stosowane w przypadku braku gotowej generacji pozostają takie same.

To nie jest reguła mówiąca, że `EXISTS` jest powolny. W naszym drugim przypadku stare zapytanie już korzystało z wydajnej strategii `FirstMatch`. Przepisanie całego `EXISTS` byłoby złym wnioskiem: trzeba sprawdzić i zmierzyć plan każdego zapytania.

Klient: nowe pomiary SQL z 10 października

Pomiar wykonany u **klienta** około **13:29** porównuje po dziesięć przebiegów każdego wariantu na danych produkcyjnych, po odrzuceniu jednego przebiegu rozgrzewającego i z naprzemienną kolejnością przed/po. Kod produkcyjny zawiera już poprawkę: „przed” oznacza tu porównawcze wykonanie starego zapytania `EXISTS`, a „po” podzapytanie skalarne z `LIMIT 1`.

Pomiar SQL	**Przed (`EXISTS`)**	**Po (`LIMIT 1`)**
Zapytanie SVG, minimum–maksimum	4 434,17–5 297,38 ms	2,69–6,20 ms
Zapytanie SVG, mediana	**5 044,59 ms**	**3,32 ms**
Łącznie 16 zapytań, mediana	5 068,52 ms	12,92 ms
Zwrócone wykresy	61	61
Objętość SVG	995 307 bajtów	995 307 bajtów
Licznik odczytów indeksu zestawienia	5 199 176	3

Dominujące zapytanie skraca się z **5,04 sekundy do 3,32 ms**, czyli w tym pomiarze działa około **1520 razy szybciej**. Wszystkie 61 wykresów jest identycznych przed poprawką i po niej. Obecna

objętość różni się o 130 bajtów od pierwszej diagnozy, która wskazywała 995 437 bajtów; są to dwa różne terminy pomiaru, natomiast oba warianty nowej kampanii zwracają tę samą treść.

Plan `EXPLAIN` przed poprawką **szacuje 4 406 092 wpisy** dla materializacji, a po poprawce **81 wpisów** dla zależnego podzapytania używającego indeksu. Te wartości nie są liczbą faktycznie wykonanych odczytów. Licznik odczytów indeksu wskazuje **5 199 176 przed i 3 po**; `dot3_cluster` zawierał wtedy **5 198 990 wierszy**. Szacunek planu, odczyty i przesłane wykresy to trzy różne miary.

Medians of the remaining 15 queries stay below 10 ms. Niektóre wartości się wahają: zliczanie alarmów spada na przykład z 9,21 do 0,85 ms, a przed poprawką chwilami osiąga 15,55 ms. Nie dowodzi to, że poprawka przyspiesza zliczanie: obciążenie serwera wynosiło około 6, a te zapytania wykonywano bezpośrednio po zapytaniu trwającym pięć sekund. Zmiana kodu dotyczy wyłącznie wyboru SVG.

Jedyne żądanie zmiany odtworzone przez tę kampanię, odświeżenie sesji, zostało wykonane, a następnie anulowane. **Ten raport mierzy SQL; nie mierzy pobierania ani pełnego ładowania przeglądarki.** Czasy obejmują pełne pobranie wyników przez mysqli, w tym około 1 MB pliku SVG.

Zapytanie od klienta	Przed (ms)	Po (ms)
1. Przeczytaj sesję logowania	0,34	0,34
2. Odśwież sesję, a następnie anuluj	0,16	0,19
3. Załaduj 61 wykresów i ich wymiary	5 044,59	3,32
4. Lista klientów	0,53	0,29
5. Lista środowisk	0,78	0,15
6. Lista tagów	0,60	0,22
7. Znajdź kanał nawigacji	1,59	0,64
8. Zbuduj menu	5,26	4,24
9. Zliczanie aktywnych alarmów	9,21	0,85
10. Wybierz punkt menu	0,78	0,66
11. Sprawdź klucz tłumaczenia	0,53	0,15

Zapytanie od klienta	Przed (ms)	Po (ms)
12. Przeczytaj tłumaczenie	0,48	0,18
13. Sprawdź tabelę konfiguracyjną	0,20	0,17
14. Przeczytaj ustawienia licencji	0,17	0,15
15. Sprawdź ukrytą tabelę banerów	0,18	0,15
16. Przeczytaj maskowanie banera licencji	0,30	0,13
Łącznie, mediana sum dla przebiegu	**5 068,52**	**12,92**

Zanonimizowany raport SQL z zakresami czasów i dwoma planami uzupełnia tę tabelę.

Czas pobierania u klienta: założenie porównawcze

Aby zakończyć porównanie HTTP, **bierzemy czas pobierania z serwera zmierzony na instancji testowej**, przedstawiony w następnej sekcji. Zakładamy, że te czasy trwania są identyczne dla klienta. **Te wartości są założeniem wspólnym dla obu środowisk; nie stanowią nowego środka HTTP u klienta.**

Pobieranie dokumentu HTML, przyjęte jako założenie	Przed	Po
Mediana Chromium na instancji testowej	42,90 ms	37,45 ms

Nie zmierzono ponownie czasu do pierwszego bajtu ani pełnego ładowania u klienta. Nie można użyć sum z testu, ponieważ przed poprawką samo zapytanie SVG klienta trwało pięć sekund. Zmierzony SQL i założony transfer pozostają oddzielne; nie tworzymy sztucznego czasu przeglądarki.

Pobieranie i przeglądarka: nowy pomiar z serwera

Aby usunąć niepewność związaną z połączeniem użytym do wcześniejszych pomiarów, **10 października około 13:45** przeprowadzono kolejną kampanię z serwera Debian 12 do instancji testowej w sieci lokalnej. Użyto tej samej strony Architektura bez filtra, tego samego tymczasowego konta tylko do odczytu, **30 wykresów**, okna 1440 × 900 pikseli i Chromium 152. Wyłączono pamięć podręczną HTTP przeglądarki; odrzucono po jednym ładowaniu rozgrzewającym, a następnie zachowano **dziesięć ładowań przed i dziesięć po**.

Serwer docelowy używa MariaDB 11.8.9 i FrankenPHP / PHP 8.5.9. Kod aplikacji odpowiada commitowi `4207dfbe5e7d1709dbbd2efdd2cff8c1e6619dc1`; w wariancie „przed” tylko `Dot3LatestGeneration.php` pochodzi z commitu nadrzędnego `366b200703ff7550ec01d33e2c7d49c0b570ab15`. Pomiary korzystają z tymczasowego serwera HTTPS udostępniającego worktree w sieci lokalnej. Produkcja pozostaje na poprawionym kodzie.

Medians in milliseconds:

Etap mierzony w Chromium	Przed (`EXISTS`)	Po (`LIMIT 1`)
Pierwszy bajt, PHP i transport w cenie	62,35	50,90
Pobierz dokument HTML	42,90	37,45
Otrzymano dokument → zdarzenie `load`	416,05	392,20
Pełne ładowanie → `load`	**523,15**	**488,40**



Pobieranie dokumentu trwa od **33,2 do 61,3 ms przed** i od **32,3 do 55,1 ms po**. Pełne ładowanie trwa od **472,3 do 681,5 ms przed** i od **444,5 do 669,7 ms po**. Zakresy się nakładają: na tej trasie ładowanie trwa około pół sekundy, bez wykazania powtarzalnego zysku 34,75 ms w przeglądarce.

30 wyświetlanych SVG ma ten sam SHA-256 podczas każdego z dwudziestu ładowań. Rozpakowany dokument ma **2 240 909 bajtów** w obu wariantach. Skompresowana treść ma około **671 254 bajtów**, czyli około 671 554 bajtów z nagłówkami liczonymi przez Chromium. Ta objętość HTML i 30 wykresów należą do instancji testowej; nie odpowiadają 995 307 bajtom SVG z 61 wykresów klienta.

Oddzielna kontrola z `curl --compressed`, dziesięć żądań na wariant po nagrzaniu, potwierdza, że transfer pozostaje zmienny: mediana pobierania **70,85 ms przed i 93,80 ms po**, pełna odpowiedź HTTP **119,97 i 145,62 ms**. `curl` nie ładuje zasobów strony ani nie uruchamia JavaScript. Dekompresja, czytnik HTTP, negocjowany protokół i obciążenie serwera przyczyniają się do rozbieżności z Chromium; różnice te nie dowodzą regresji SQL.

Kamienie milowe są mierzone podczas każdej wizyty, a następnie podsumowywane osobno: **suma median niekoniecznie jest medianą całości**. Czas po otrzymaniu dokumentu obejmuje inne zasoby, analizę HTML, JavaScript i renderowanie. Nie przypisujemy tego samemu rozmieszczeniu wykresów. Pierwszy bajt zawiera już przygotowanie PHP; do sumy nie jest dodawany szacowany czas PHP.

Dawne czasy ładowania 12 948, 2 952 i 843 ms pozostają historycznymi obserwacjami z 9 października, dostępnymi w archiwalnym CSV. Pobieranie od 470 do 535 ms zależało od niestabilnego wówczas połączenia, dlatego nie stanowi już punktu odniesienia dla sieci. Nowa kampania na 30 wykresach nie odtwarza czasu przeglądarki dla instancji z 61 wykresami.

Nowe porównanie SQL w instancji testowej, dziesięć naprzemiennych par w pojedynczej migawce tylko do odczytu z wyłączoną pamięcią podręczną zapytań MariaDB dla sesji, daje wynik **6,94 ms przed i 5,81 ms po**. Pełne wyniki i **2 172 396 bajtów SVG** są identyczne. Początkowy plan nadal wykorzystuje `FirstMatch`: nawet przy 6 591 509 wierszach w `dot3_cluster` ta instancja nie odtwarza kosztownej materializacji drugiego serwera. Dlatego sam rozmiar tabeli nie wystarczy, aby przewidzieć powolność.

Instancja testowa: szczegóły 16 żądań z 9 października

Poniższa tabela zachowuje szczegóły kampanii pomiarowej z **9 października**: strona bez filtra, konto tylko do odczytu i **30 wykresów**. Odrzucono pierwsze ładowanie rozgrzewające, a następnie zachowano po pięć ładowań każdej wersji. Tabela nie jest przedstawiana jako nowy pomiar 16 zapytań z 10 października.

Profiler aplikacji wykrywa **16 żądań na każdym z dziesięciu ładowań**. Każda linia poniżej odpowiada zapytaniu wykonanemu raz przed i raz po; ostatnia linia pokazuje sumę. Poniższe czasy to mediany wywołań SQL mierzone po stronie sterownika PHP, w milisekundach. Reprezentują one nie tylko czas wewnętrzny silnika MariaDB.

Zapytanie	Przed (ms)	Po (ms)
1. Przeczytaj sesję logowania	0,31	0,34
2. Odśwież sesję	0,29	0,27
3. Załaduj 30 wykresów i ich wymiary	5,86	5,57
4. Lista klientów	0,37	0,34
5. Lista środowisk	0,23	0,27
6. Lista tagów	0,23	0,29
7. Znajdź kanał nawigacji	0,71	1,07
8. Zbuduj menu	3,02	3,04
9. Zliczanie aktywnych alarmów	1,11	1,09
10. Wybierz punkt menu	0,67	0,67
11. Sprawdź klucz tłumaczenia	0,33	0,31
12. Przeczytaj tłumaczenie	0,30	0,29
13. Sprawdź tabelę konfiguracyjną	0,33	0,31
14. Przeczytaj ustawienia licencji	0,22	0,23
15. Sprawdź ukrytą tabelę banerów	0,32	0,31
16. Przeczytaj maskowanie banera licencji	0,18	0,22
Ogółem, mediana	**14,53**	**14,59**

Suma jest medianą sum obliczonych dla każdego ładunku. Dodanie median każdego wiersza niekoniecznie daje tę wartość.

Na tej instancji testowej **nie wykazano istotnej ogólnej poprawy SQL**. Różnica 0,06 ms jest minimalna, ponieważ MariaDB wybierała już dobry plan dla starego zapytania. Poprawka jest

skierowana na kosztowny plan napotkany w instancji produkcyjnej.

Liczba ta obejmuje żądania aplikacji początkowego renderowania PHP w trzech połączeniach odnotowanych przez profiler. Nie obejmuje kolejnych wywołań AJAX i poleceń logowania wewnętrznego. Tych 16 żądań nie należy przypisywać do wszystkich profili użytkowników ani wszystkich instalacji.

Sprawdź także, czy wyniki pozostają prawidłowe

Szybsza strona jest użyteczna tylko wtedy, gdy wyświetla właściwe wykresy. Weryfikacja skupiała się na równości wyników przed i po, pokoleniach gotowych lub nieobecnych oraz rozwiązaniach awaryjnych.

Wykonano dziewięć scenariuszy z danymi testowymi zdefiniowanymi w CTE MariaDB, bez wstawiania do tabel aplikacji. Porównania SQL obejmowały także stronę niefiltrowaną, stronę filtrowaną i agregację wykresu, a wyniki porównano w tej samej migawce transakcyjnej. Test JavaScript dotyczący rozmieszczania zweryfikował zachowanie na różnych szerokościach ekranu.

Oddzielna seria piętnastu sparowanych pomiarów na instancji testowej wykazała niewielkie mediany narzutu na ścieżkach filtrowanej i zagregowanej: około **0,34 ms** i **0,16 ms**. Uwzględniamy je w podsumowaniu, ponieważ poprawka nie przyspiesza wszystkich scenariuszy. Te ukierunkowane kontrole nie zastępują pełnego zestawu testów produktu.

Powtórz diagnostykę na innej stronie

Zacznij od oddzielenia trzech pytań: jak długo serwer przygotowuje odpowiedź, ile czasu zajmuje dotarcie dokumentu i ile czasu zajmuje ładowanie przeglądarki?

Następnie policz żądania i zmierz każde z nich. Dla tego, który dominuje, porównaj plan przed/po i równość wyników. Sprawdź w szczególności liczbę przeszukiwanych wpisów, wyniki pośrednie oraz efektywność wykorzystania indeksów. Powtarzaj pomiary: pojedyncza obserwacja może wprowadzić w błąd.

W tym przypadku **zapytanie pozostało zapytaniem**, a indeks pozostał taki sam. Zmienił się sposób wyszukiwania rysunków znajdujących się już w pamięci podręcznej.

Szczegółowe dane i SQL

- 16 zapytań, ich lokalizacja w kodzie i pełny kod SQL przed/po
- Dziesięć oprzyrządowanych obciążeń: czas trwania każdego żądania w formacie CSV
- Dwadzieścia ładowań Chromium z serwera 10 października, w formacie CSV
- Kontrola HTTP przez curl z 10 października, przed i po
- Porównanie SQL z 10 października na instancji testowej: wyniki i plany w JSON
- Anonimowa instrukcja SQL od klienta: 16 zapytań, zakresów i planów
- Anonimowa konfiguracja instancji testowej MariaDB `.cnf` i aktywne zmienne, wersja i dołączone pliki
- Schemat 16 tabel w SVG, definicje SQL, spis silników i rozmiarów w CSV oraz kolumny, indeksy i relacje w JSON
- Skrypt SQL, podany w tekście, skrypt Chromium, skrypt HTTP i tymczasowa konfiguracja serwera HTTPS
- Metoda i kroki odtwarzania kampanii z 10 października
- Archiwum: trzy obciążenia produkcyjne z 9 października na niestabilnym połączeniu

W przykładach do pobrania identyfikatory sesji i konta zostały zastąpione parametrami. Pomiary produkcyjne i testowe są oddzielone, aby umożliwić ich interpretację bez mieszania tych dwóch środowisk.