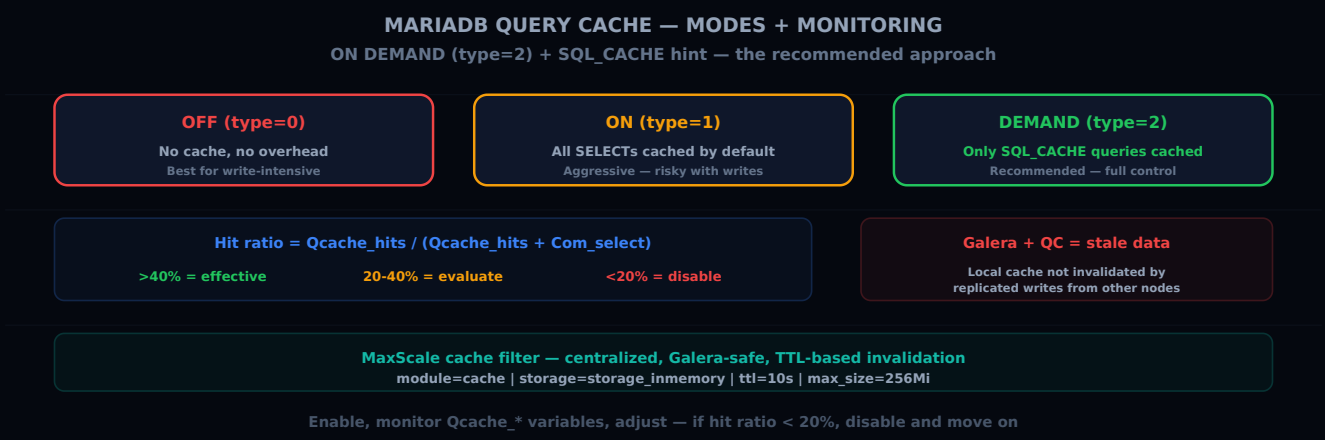
Eksperymenty z MariaDB: edycja query cache

Sylvain ARBAUDIE · June 30, 2025

MARIADB QUERY-CACHE PERFORMANCE TUNING



Koncepcja Query Cache

Query cache MariaDB / MySQL to wbudowany w serwer mechanizm przechowujący wyniki zapytań SELECT w pamięci. Gdy identyczne zapytanie jest ponownie przesyłane, serwer zwraca bezpośrednio wynik z cache bez wykonywania zapytania. To proste, eleganckie i potencjalnie bardzo skuteczne.

Słowo kluczowe to „potencjalnie". Query cache to jedna z najgorzej rozumianych i najgorzej konfigurowanych funkcji MariaDB / MySQL. Prawidłowo użyty, może podzielić czasy odpowiedzi przez 10. Źle skonfigurowany, może zniszczyć wydajność.

Trzy tryby

Query cache może działać w trzech trybach, kontrolowanych zmienną `query_cache_type`:

OFF (0)

Query cache jest całkowicie wyłączony. Żadne zapytanie nie jest cachowane, żadna weryfikacja cache nie jest wykonywana. To najbezpieczniejsza opcja dla obciążeń intensywnych w zapisie.

ON (1)

Wszystkie zapytania SELECT są domyślnie cachowane, z wyjątkiem oznaczonych hintem `SQL_NO_CACHE`. To najbardziej agresywny tryb.

```
-- To zapytanie zostanie cachowane
SELECT * FROM products WHERE category_id = 5;

-- To zapytanie NIE zostanie cachowane
SELECT SQL_NO_CACHE * FROM products WHERE category_id = 5;
```

ON DEMAND (2)

Żadne zapytanie nie jest domyślnie cachowane. Tylko zapytania jawnie oznaczone `SQL_CACHE` są cachowane. To najbardziej kontrolowany i często najbardziej skuteczny tryb.

```
-- To zapytanie NIE zostanie cachowane (zachowanie domyślne)
SELECT * FROM products WHERE category_id = 5;

-- To zapytanie ZOSTANIE cachowane
SELECT SQL_CACHE * FROM products WHERE category_id = 5;
```

Tryb `ON DEMAND` jest tym, który polecam w większości przypadków. Zmusza do zastanowienia się, które zapytania zasługują na cachowanie, zamiast cachować wszystko na oślep.

Invalidacja: pięta achillesowa

Query cache invaliduje CAŁY cache tabeli po każdym zapisie na tej tabeli. Nie tylko dotknięte wiersze — całą tabelę.

```
-- Załóżmy, że 1000 SELECT na tabeli 'products' jest w cache
UPDATE products SET price = 19.99 WHERE product_id = 42;
-- → Wszystkie 1000 wpisów cache dla 'products' jest zinvalidowanych
```

To brutalny, ale konieczny mechanizm gwarantujący spójność. Problem polega na tym, że dla często modyfikowanych tabel cache jest nieustannie invalidowany i odbudowywany, co zużywa więcej zasobów niż brak cache w ogóle.

Wymiarowanie cache

Rozmiar query cache jest kontrolowany przez `query_cache_size`:

```
[mysqld]
query_cache_type = 2
query_cache_size = 64M
query_cache_limit = 2M
query_cache_min_res_unit = 2048
```

- **query_cache_size**: całkowity rozmiar cache. Nie przekraczaj 256 MB — powyżej globalny mutex cache staje się wąskim gardłem.
- **query_cache_limit**: maksymalny rozmiar pojedynczego wyniku. Większe wyniki nie są cachowane.
- **query_cache_min_res_unit**: rozmiar bloku alokacji. Zmniejszenie tej wartości dla małych wyników redukuje fragmentację.

Monitoring cache

Zmienne statusu `Qcache_*` są niezbędne do oceny efektywności:

```
SHOW GLOBAL STATUS LIKE 'Qcache%';
```

Kluczowe metryki:

Zmienna	Opis
`Qcache_hits`	Liczba zapytań obsłużonych z cache
`Qcache_inserts`	Liczba zapytań dodanych do cache
`Qcache_not_cached`	Zapytania niecachowane (zbyt duże, hinty itp.)
`Qcache_lowmem_prunes`	Eksmisje z powodu braku pamięci
`Qcache_free_memory`	Wolna pamięć w cache
`Qcache_total_blocks`	Całkowita liczba zaalokowanych bloków
`Qcache_free_blocks`	Wolne bloki (fragmentacja, jeśli wartość wysoka)

Współczynnik efektywności

Najważniejszy współczynnik to **hit ratio**:

```
Hit ratio = Qcache_hits / (Qcache_hits + Com_select) × 100
```

Interpretacja:

- **> 40%**: cache jest skuteczny, warto go mieć włączony
- **20-40%**: strefa szara, oceniać indywidualnie
- **< 20%**: cache nie jest skuteczny, rozważ wyłączenie

Drugi ważny współczynnik to **współczynnik eksmisji**:

```
Eviction ratio = Qcache_lowmem_prunes / Qcache_inserts × 100
```

Jeśli ten współczynnik przekracza 10%, cache jest zbyt mały — zwiększ `query_cache_size` lub ogranicz to, co cachujesz.

Pułapka współbieżności

Query cache używa **globalnego muteksu**. Oznacza to, że tylko jeden wątek może jednocześnie czytać lub pisać do cache. Na serwerze ze 100 równoczesnymi połączeniami ten muteks staje się poważnym wąskim gardłem.

To jest powód, dla którego MySQL 8.0 usunął query cache. Zespół Oracle uznał, że koszt muteksu przewyższa korzyści z cache w nowoczesnych obciążeniach (wysoka współbieżność, częste zapisy).

MariaDB zdecydowała się go zachować, uznając, że pozostaje użyteczny dla pewnych specyficznych przypadków użycia (obciążenia odczytowe, niska współbieżność, rzadko modyfikowane tabele).

Query cache i Galera: trudna kombinacja

Używanie query cache w klastrze Galera jest problematyczne. Galera replikuje zapisy na wszystkie węzły, ale query cache jest lokalny dla każdego węzła. Rezultat:

1. Węzeł A otrzymuje SELECT i cachuje wynik
2. Węzeł B otrzymuje UPDATE przez replikację Galera
3. Cache węzła A NIE jest invalidowany — nie wie, że dane się zmieniły

4. Następny SELECT na węźle A zwraca przestarzałe dane

Jedynym bezpiecznym sposobem użycia query cache z Galera jest ustawienie `wsrep_causal_reads = ON`, które wymusza weryfikację spójności przed każdym odczytem. Ale to w dużej mierze niweluje korzyści z cache.

Alternatywa: MaxScale cache filter

Dla architektur wymagających rozproszonego cache zapytań, filtr cache MaxScale jest lepszym podejściem:

```
[query-cache]
type = filter
module = cache
storage = storage_inmemory
ttl = 10s
max_size = 256Mi
```

Cache MaxScale jest scentralizowany (na poziomie proxy), co eliminuje problem niespójności między węzłami Galera. Ponadto MaxScale może invalidować cache w sposób inteligentny na podstawie typu zapytania, a nie tylko na podstawie zmodyfikowanej tabeli.

Kiedy włączyć query cache

Query cache jest odpowiedni, gdy:

- Obciążenie jest **głównie odczytowe** (> 80% SELECT)
- Tabele są **rzadko modyfikowane** (tabele konfiguracyjne, słowniki)
- **Współbieżność jest umiarkowana** (< 50 równoczesnych połączeń)
- **Te same zapytania są powtarzane** często (aplikacje webowe z cache miss)
- Korzystasz z **serwera standalone**, nie klastra Galera

Query cache NIE jest odpowiedni, gdy:

- Obciążenie jest **mieszane odczyt/zapis**
- Tabele są **często modyfikowane** (tabele transakcyjne)
- **Współbieżność jest wysoka** (globalny muteks)

- Korzystasz z **klastra Galera** (niespójność cache)

Podsumowanie

Query cache MariaDB to potężne, ale delikatne narzędzie. Tryb `ON DEMAND` z hintem `SQL_CACHE` oferuje najlepszą kontrolę. Monitoring przez zmienne `Qcache_*` jest niezbędny do oceny efektywności. A dla architektur rozproszonych filtr cache MaxScale jest często lepszym wyborem.

Jak każde narzędzie do optymalizacji wydajności, kluczem jest pomiar. Włącz, monitoruj, dostosowuj. Jeśli hit ratio pozostaje poniżej 20%, wyłącz i zajmij się czymś innym.

Ten artykuł został pierwotnie opublikowany na Medium.