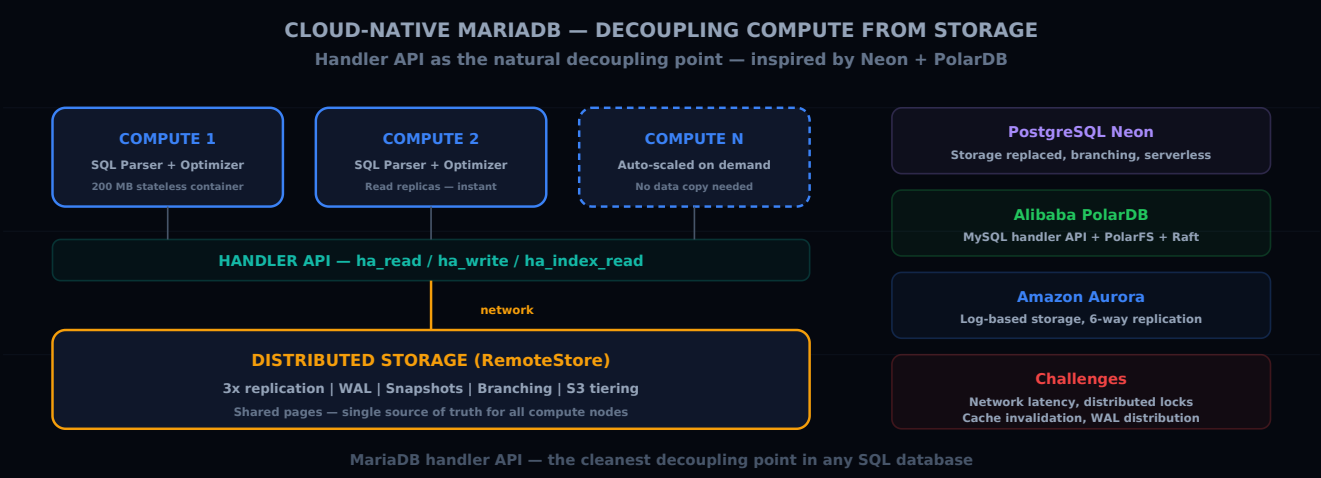
Uczynienie MariaDB cloud-native: oddzielenie obliczeń od przechowywania

Sylvain ARBAUDIE · July 9, 2025

MARIADB CLOUD-NATIVE ARCHITECTURE STORAGE



Stwierdzenie: MariaDB nie jest cloud-native

Bądźmy szczerzy: MariaDB / MySQL w obecnej formie nie są bazami danych cloud-native. Zostały zaprojektowane do pracy na pojedynczym serwerze z lokalnym przechowywaniem. Nawet architektury replikacji master-slave to fundamentalnie kompletne kopie danych na każdym węźle.

W chmurze stwarza to kilka fundamentalnych problemów:

- **Przechowywanie jest sprzężone z obliczeniami.** Jeśli potrzebujesz więcej CPU, musisz zaaprowizować nowy serwer z całym przechowywaniem. Jeśli potrzebujesz więcej przestrzeni dyskowej, musisz zmienić rozmiar serwera.
- **Skalowanie horyzontalne jest ograniczone.** Każda replika zawiera pełną kopię danych. Dodanie slave'a o pojemności 2 TB oznacza skopiowanie 2 TB danych.
- **Failover oznacza przestój.** Promowanie slave'a wiąże się z czasem konwergencji, w trakcie którego najnowsze zapisy mogą zostać utracone (w replikacji asynchronicznej) lub w trakcie którego klaster jest niedostępny (w replikacji synchronicznej).

Nowoczesne bazy cloud-native, takie jak Amazon Aurora, Google AlloyDB czy PostgreSQL Neon, rozwiązały te problemy, oddzielając obliczenia od przechowywania.

Inspiracja: PostgreSQL Neon

PostgreSQL Neon to fascynujący projekt. Bierze PostgreSQL i zastępuje lokalną warstwę przechowywania rozproszonym przechowywaniem zdalnym. Silnik PostgreSQL działa jako czysty węzeł obliczeniowy: odbiera zapytania, planuje wykonanie i wymienia strony danych z usługą zdalnego przechowywania przez sieć.

Zalety są spektakularne:

- **Natychmiastowe skalowanie**: dodanie węzła obliczeniowego nie wymaga żadnej kopii danych
- **Branching**: tworzenie "brancha" bazy (jak branch Git) jest natychmiastowe — to operacja na metadanych, a nie fizyczna kopia
- **Pay-per-use**: nieaktywne węzły obliczeniowe są zatrzymywane, płacisz tylko za przechowywanie

Czy MariaDB mogłaby podążyć tą samą drogą?

Handler API: naturalny punkt oddzielenia

MariaDB ma unikalną przewagę architektoniczną, której PostgreSQL nie posiada: **handler API**. To abstrakcyjny interfejs między silnikiem SQL (parser, optymalizator, executor) a silnikami przechowywania (InnoDB, Aria, RocksDB, ColumnStore itp.).

Handler API definiuje operacje takie jak:

```
handler::ha_open()       // otwarcie tabeli
handler::ha_read_first() // odczyt pierwszego wiersza
handler::ha_read_next()  // odczyt następnego wiersza
handler::ha_write_row()  // zapis wiersza
handler::ha_update_row() // aktualizacja wiersza
handler::ha_delete_row() // usunięcie wiersza
handler::ha_index_read() // odczyt przez indeks
```

Każdy silnik przechowywania implementuje te metody. InnoDB implementuje je, uzyskując dostęp do lokalnych stron B-tree. Aria implementuje je inaczej. RocksDB używa LSM-trees.

A gdyby silnik przechowywania implementował te metody, uzyskując dostęp do **zdalnych** stron przez sieć?

Podejście PolarDB (Alibaba)

Alibaba już udowodniło, że to możliwe, dzięki PolarDB, opartemu na kodzie MySQL. PolarDB używa:

- Rozproszonego systemu plików (PolarFS) zastępującego lokalne przechowywanie
- Protokołu konsensusu (Raft) dla trwałości
- Współdzielonego cache stron między węzłami obliczeniowymi

Rezultatem jest MySQL, którego przechowywanie jest oddzielone od obliczeń. Wiele węzłów obliczeniowych może jednocześnie odczytywać te same dane, a tylko jeden węzeł zarządza zapisami.

PolarDB pokazuje, że handler API MariaDB/MySQL jest realnym punktem oddzielenia. Alibaba nie przepisała silnika SQL — zaimplementowali handler, który uzyskuje dostęp do zdalnego przechowywania.

Wizja: zdalny handler API dla MariaDB

Wyobraźmy sobie silnik przechowywania MariaDB o nazwie `RemoteStore`:

```
CREATE TABLE users (
    id BIGINT PRIMARY KEY,
    name VARCHAR(255)
) ENGINE=RemoteStore
CONNECTION='storage-cluster.internal:9000/db1';
```

Ten silnik przechowywania nie miałby żadnych lokalnych danych. Każde wywołanie handler API byłoby tłumaczone na wywołanie sieciowe do rozproszonej usługi przechowywania. Usługa przechowywania zarządzałaby:

- Replikacją danych (minimum 3 kopie)

- Trwałością (rozproszony write-ahead log)
- Snapshotami i branchingiem
- Kompresją i tieringiem (gorące dane na SSD, zimne na S3)

Kompilowanie MariaDB bez wbudowanych silników

Pierwszym krokiem w kierunku tej wizji byłoby umożliwienie kompilacji MariaDB bez żadnego wbudowanego silnika przechowywania. Dziś InnoDB jest wkompilowany w plik binarny `mariadbd`. Opcje kompilacji pozwalają wyłączyć niektóre silniki, ale InnoDB pozostaje głęboko zintegrowany.

"Bezgłowy" MariaDB — czysty silnik SQL bez przechowywania — wyglądałby tak:

```
MariaDB SQL Engine (parser + optimizer + executor)
       ↓ handler API
Plugin: RemoteStore → sieć → Rozproszone przechowywanie
```

Taki bezgłowy MariaDB byłby lekki (kilkaset MB RAM), uruchamiałby się w milisekundach i mógłby być wdrażany jako bezstanowy kontener w Kubernetes.

Wyzwania techniczne

Ta wizja nie jest pozbawiona wyzwań:

Opóźnienie sieciowe

Każdy dostęp do strony przechodzi przez sieć. Opóźnienie sieciowe w centrum danych wynosi rzędu 100 do 500 mikrosekund. To 100 do 1000 razy wolniej niż lokalny dostęp do SSD. Agresywny cache stron na poziomie węzła obliczeniowego jest niezbędny.

Zarządzanie blokadami

InnoDB zarządza blokadami lokalnie. Przy zdalnym współdzielonym przechowywaniu zarządzanie blokadami musi być rozproszone. To trudny problem, który może wprowadzać dodatkowe deadlocki i timeouty.

Rozproszony buffer pool

Buffer pool InnoDB jest lokalny. W architekturze cloud-native potrzebny jest mechanizm inwalidacji cache między węzłami obliczeniowymi. Gdy jeden węzeł zapisuje stronę, cache pozostałych węzłów muszą zostać zinwalidowane.

Logowanie transakcji

Redo log i undo log InnoDB są lokalne. W architekturze oddzielonej WAL musi być rozproszony i dostępny dla wszystkich węzłów.

Co już istnieje

Niektóre elementy tej wizji istnieją już w ekosystemie MariaDB:

- **MariaDB ColumnStore**: kolumnowy silnik przechowywania, który może korzystać z przechowywania S3
- **Spider**: silnik przechowywania, który rozkłada dane na wiele serwerów MariaDB
- **CONNECT**: silnik przechowywania, który może uzyskiwać dostęp do zewnętrznych źródeł danych

Żaden z tych silników nie realizuje pełnego oddzielenia obliczeń od przechowywania, ale demonstrują elastyczność handler API.

Podsumowanie

Uczynienie MariaDB cloud-native to ambitne, ale nie nierealistyczne wyzwanie. Handler API oferuje naturalny punkt oddzielenia, którego ani PostgreSQL, ani Oracle MySQL nie mają w tak czystej formie.

PolarDB od Alibaba udowodnił, że jest to technicznie wykonalne. PostgreSQL Neon udowodnił, że rynek tego oczekuje. Pytanie nie brzmi "czy to możliwe?" lecz "kto zrobi to pierwszy w społeczności MariaDB?"

Dnia, w którym MariaDB będzie mógł wystartować jako bezstanowy kontener o pojemności 200 MB, połączyć się z rozproszonym przechowywaniem i obsługiwać zapytania w kilka milisekund — tego dnia nastąpi zmiana zasad gry.

Ten artykuł został pierwotnie opublikowany na Medium.