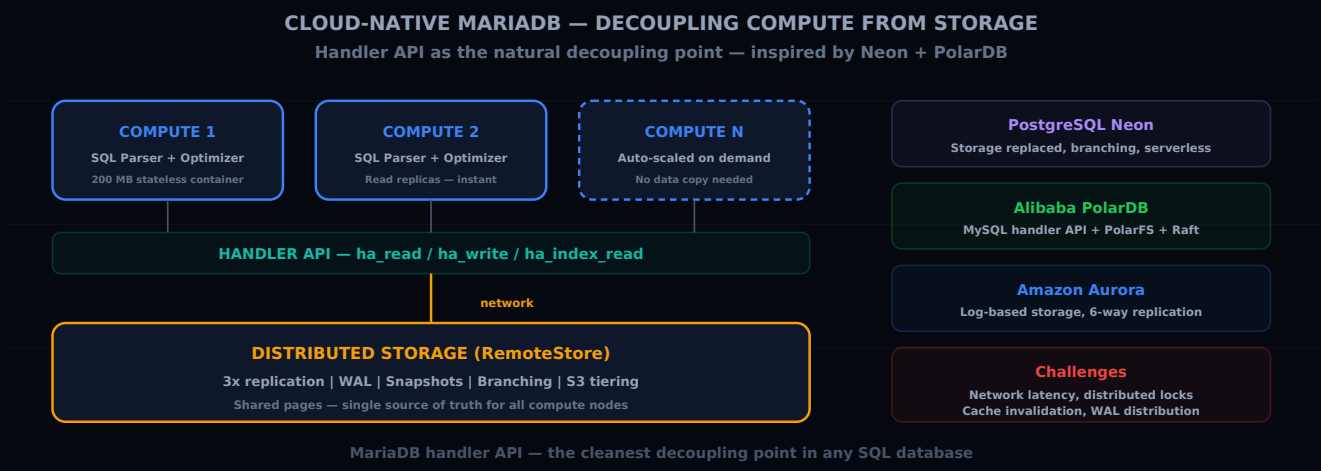
Rendre MariaDB cloud-native : découpler le calcul du stockage

Sylvain ARBAUDIE · 9 juillet 2025

MARIADB CLOUD-NATIVE ARCHITECTURE STORAGE



Le constat : MariaDB n'est pas cloud-native

Soyons honnêtes : MariaDB / MySQL dans leur forme actuelle ne sont pas des bases de données cloud-native. Elles ont été conçues pour fonctionner sur un serveur unique avec un stockage local. Même les architectures de réplication master-slave sont fondamentalement des copies complètes des données sur chaque nœud.

Dans le cloud, cela pose plusieurs problèmes fondamentaux :

- **Le stockage est couplé au calcul.** Si vous avez besoin de plus de CPU, vous devez provisionner un nouveau serveur avec tout le stockage. Si vous avez besoin de plus de stockage, vous devez redimensionner le serveur.
- **Le scaling horizontal est limité.** Chaque réplica contient une copie complète des données. Ajouter un slave de 2 To signifie copier 2 To de données.
- **Le failover implique du downtime.** Promouvoir un slave implique un temps de convergence pendant lequel les écritures les plus récentes peuvent être perdues (en réplication asynchrone) ou pendant lequel le cluster est indisponible (en réplication synchrone).

Les bases cloud-native modernes comme Amazon Aurora, Google AlloyDB, ou PostgreSQL Neon ont résolu ces problèmes en découplant le calcul du stockage.

L'inspiration : PostgreSQL Neon

PostgreSQL Neon est un projet fascinant. Il prend PostgreSQL et remplace la couche de stockage locale par un stockage distribué distant. Le moteur PostgreSQL fonctionne comme un nœud de calcul pur : il reçoit les requêtes, planifie l'exécution, et échange des pages de données avec un service de stockage distant via le réseau.

Les avantages sont spectaculaires :

- **Scaling instantané** : ajouter un nœud de calcul ne nécessite aucune copie de données
- **Branching** : créer une "branche" de la base (comme un branch Git) est instantané — c'est une opération de métadonnées, pas une copie physique
- **Pay-per-use** : les nœuds de calcul inactifs sont arrêtés, vous ne payez que le stockage

MariaDB pourrait-elle suivre le même chemin ?

Le handler API : le point de découplage naturel

MariaDB a un avantage architectural unique que PostgreSQL n'a pas : le **handler API**. C'est l'interface abstraite entre le moteur SQL (parseur, optimiseur, exécuteur) et les moteurs de stockage (InnoDB, Aria, RocksDB, ColumnStore, etc.).

Le handler API définit des opérations comme :

```
handler::ha_open()       // ouvrir une table
handler::ha_read_first() // lire la première ligne
handler::ha_read_next()  // lire la ligne suivante
handler::ha_write_row()  // écrire une ligne
handler::ha_update_row() // mettre à jour une ligne
handler::ha_delete_row() // supprimer une ligne
handler::ha_index_read() // lecture par index
```

Chaque moteur de stockage implémente ces méthodes. InnoDB les implémente en accédant à des pages B-tree locales. Aria les implémente différemment. RocksDB utilise des LSM-trees.

Et si un moteur de stockage implémentait ces méthodes en accédant à des pages **distantes** via le réseau ?

L'approche PolarDB (Alibaba)

Alibaba a déjà prouvé que c'est possible avec PolarDB, qui est basé sur le code MySQL. PolarDB utilise :

- Un système de fichiers distribué (PolarFS) qui remplace le stockage local
- Un protocole de consensus (Raft) pour la durabilité
- Un cache de pages partagé entre les nœuds de calcul

Le résultat est un MySQL dont le stockage est découplé du calcul. Plusieurs nœuds de calcul peuvent lire les mêmes données simultanément, et un seul nœud gère les écritures.

PolarDB montre que le handler API de MySQL/MariaDB est un point de découplage viable. Alibaba n'a pas réécrit le moteur SQL — ils ont implémenté un handler qui accède au stockage distant.

La vision : un handler API distant pour MariaDB

Imaginons un moteur de stockage MariaDB appelé `RemoteStore` :

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

Ce moteur de stockage n'aurait aucune donnée locale. Chaque appel au handler API serait traduit en appel réseau vers un service de stockage distribué. Le service de stockage gérerait :

- La réplication des données (3 copies minimum)
- La durabilité (write-ahead log distribué)
- Le snapshot et le branching
- La compression et le tiering (données chaudes en SSD, froides en S3)

Compiler MariaDB sans moteurs embarqués

La première étape vers cette vision serait de pouvoir compiler MariaDB sans aucun moteur de stockage embarqué. Aujourd'hui, InnoDB est compilé dans le binaire `mariadbd`. Les options de compilation permettent de désactiver certains moteurs, mais InnoDB reste profondément intégré.

Un MariaDB "headless" — un pur moteur SQL sans stockage — ressemblerait à :

```
MariaDB SQL Engine (parser + optimizer + executor)
       ↓ handler API
Plugin: RemoteStore → réseau → Stockage distribué
```

Ce MariaDB headless serait léger (quelques centaines de Mo de RAM), démarrerait en millisecondes, et pourrait être déployé comme un container stateless dans Kubernetes.

Les défis techniques

Cette vision n'est pas sans défis :

Latence réseau

Chaque accès à une page passe par le réseau. La latence réseau en datacenter est de l'ordre de 100 à 500 microsecondes. C'est 100 à 1 000 fois plus lent qu'un accès SSD local. Un cache de pages agressif au niveau du nœud de calcul est indispensable.

Gestion des verrous

InnoDB gère les verrous localement. Avec un stockage distant partagé, la gestion des verrous doit être distribuée. C'est un problème difficile qui peut introduire des deadlocks et des timeouts supplémentaires.

Buffer pool distribué

Le buffer pool d'InnoDB est local. Dans une architecture cloud-native, il faut un mécanisme d'invalidation de cache entre les nœuds de calcul. Quand un nœud écrit une page, les caches des autres nœuds doivent être invalidés.

Transaction logging

Le redo log et l'undo log d'InnoDB sont locaux. Dans une architecture découplée, le WAL doit être distribué et accessible par tous les nœuds.

Ce qui existe déjà

Certains composants de cette vision existent déjà dans l'écosystème MariaDB :

- **MariaDB ColumnStore** : un moteur de stockage colonnaire qui peut utiliser du stockage S3
- **Spider** : un moteur de stockage qui répartit les données sur plusieurs serveurs MariaDB
- **CONNECT** : un moteur de stockage qui peut accéder à des sources de données externes

Aucun de ces moteurs ne réalise le découplage complet calcul/stockage, mais ils démontrent la flexibilité du handler API.

Conclusion

Rendre MariaDB cloud-native est un défi ambitieux mais pas irréaliste. Le handler API offre un point de découplage naturel que ni PostgreSQL ni Oracle MySQL n'ont de manière aussi propre.

PolarDB d'Alibaba a prouvé que c'est techniquement faisable. PostgreSQL Neon a prouvé que le marché est demandeur. La question n'est pas "est-ce possible ?" mais "qui le fera en premier dans la communauté MariaDB ?"

Le jour où MariaDB pourra démarrer comme un container stateless de 200 Mo, se connecter à un stockage distribué, et servir des requêtes en quelques millisecondes — ce jour-là, le game change.

Cet article a été initialement publié sur Medium.