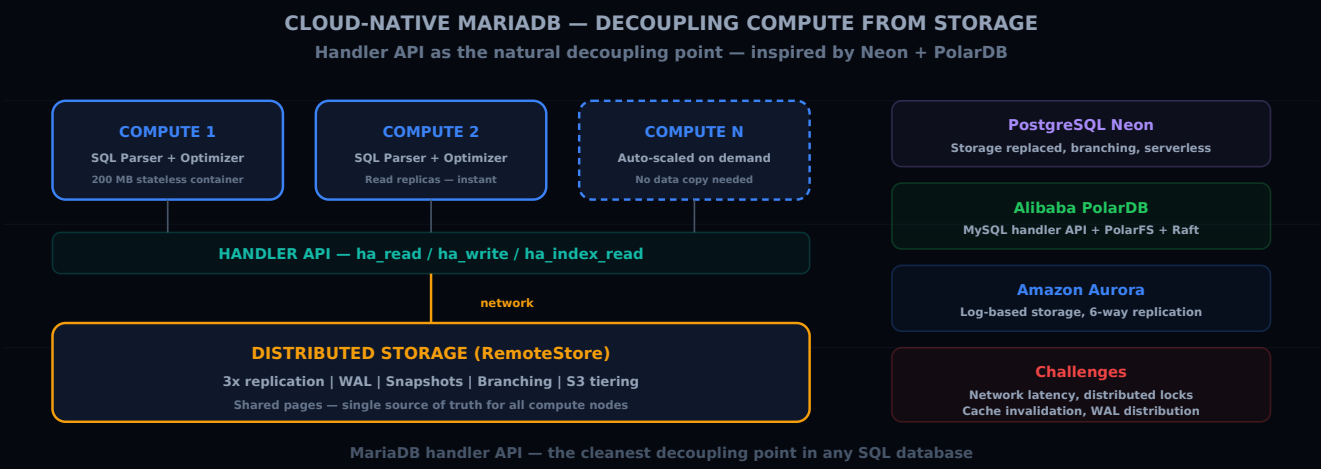
# MariaDB в облаке: разделение вычислений и хранения

**Sylvain ARBAUDIE** · July 9, 2025

MARIADB CLOUD-NATIVE ARCHITECTURE STORAGE




## Констатация: MariaDB не является cloud-native

Будем честны: MariaDB / MySQL в их нынешнем виде не являются cloud-native базами данных. Они были спроектированы для работы на одном сервере с локальным хранилищем. Даже архитектуры репликации master-slave по сути представляют собой полные копии данных на каждом узле.

В облаке это создаёт несколько фундаментальных проблем:

- **Хранилище связано с вычислениями.** Если вам нужно больше CPU, необходимо провизировать новый сервер со всем хранилищем. Если нужно больше хранилища, нужно изменить размер сервера.
- **Горизонтальное масштабирование ограничено.** Каждая реплика содержит полную копию данных. Добавление slave на 2 ТБ означает копирование 2 ТБ данных.
- **Failover подразумевает простой.** Промоутирование slave подразумевает время конвергенции, в течение которого последние записи могут быть потеряны (при асинхронной репликации) или кластер недоступен (при синхронной репликации).

Современные cloud-native базы данных, такие как Amazon Aurora, Google AlloyDB или PostgreSQL Neon, решили эти проблемы, разделив вычисления и хранилище.

## Вдохновение: PostgreSQL Neon

PostgreSQL Neon — это захватывающий проект. Он берёт PostgreSQL и заменяет уровень локального хранилища на распределённое удалённое хранилище. Движок PostgreSQL работает как чистый вычислительный узел: он получает запросы, планирует выполнение и обменивается страницами данных с удалённым сервисом хранения по сети.

Преимущества впечатляющие:

- **Мгновенное масштабирование**: добавление вычислительного узла не требует копирования данных
- **Ветвление**: создание «ветки» базы (подобно Git branch) мгновенно — это операция с метаданными, а не физическое копирование
- **Pay-per-use**: неактивные вычислительные узлы останавливаются, вы платите только за хранилище

Может ли MariaDB пойти тем же путём?

## Handler API: естественная точка разделения

У MariaDB есть уникальное архитектурное преимущество, которого нет у PostgreSQL: **handler API**. Это абстрактный интерфейс между SQL-движком (парсер, оптимизатор, исполнитель) и движками хранения (InnoDB, Aria, RocksDB, ColumnStore и т.д.).

Handler API определяет операции такие как:

```
handler::ha_open()       // открыть таблицу
handler::ha_read_first() // прочитать первую строку
handler::ha_read_next()  // прочитать следующую строку
handler::ha_write_row()  // записать строку
handler::ha_update_row() // обновить строку
handler::ha_delete_row() // удалить строку
handler::ha_index_read() // чтение по индексу
```

Каждый движок хранения реализует эти методы. InnoDB реализует их, обращаясь к локальным B-tree страницам. Aria реализует их иначе. RocksDB использует LSM-деревья.

А что если бы движок хранения реализовал эти методы, обращаясь к **удалённым** страницам через сеть?

## Подход PolarDB (Alibaba)

Alibaba уже доказала, что это возможно, с PolarDB, основанной на коде MySQL. PolarDB использует:

- Распределённую файловую систему (PolarFS), заменяющую локальное хранилище
- Протокол консенсуса (Raft) для обеспечения долговечности
- Общий кэш страниц между вычислительными узлами

Результат — MySQL, в котором хранилище отделено от вычислений. Несколько вычислительных узлов могут одновременно читать одни и те же данные, а записи обрабатывает один узел.

PolarDB показывает, что handler API MariaDB/MySQL — жизнеспособная точка разделения. Alibaba не переписывала SQL-движок — они реализовали handler, обращающийся к удалённому хранилищу.

## Видение: удалённый handler API для MariaDB

Представим движок хранения MariaDB под названием `RemoteStore`:

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

Этот движок хранения не имел бы локальных данных. Каждый вызов handler API транслировался бы в сетевой запрос к распределённому сервису хранения. Сервис хранения управлял бы:

- Репликацией данных (минимум 3 копии)

- Долговечностью (распределённый write-ahead log)
- Снимками и ветвлением
- Сжатием и тирингом (горячие данные на SSD, холодные в S3)

## Компиляция MariaDB без встроенных движков

Первым шагом к этому видению была бы возможность компилировать MariaDB без встроенных движков хранения. Сегодня InnoDB скомпилирован в бинарный файл `mariadbd`. Опции компиляции позволяют отключить некоторые движки, но InnoDB остаётся глубоко интегрированным.

«Headless» MariaDB — чистый SQL-движок без хранилища — выглядел бы так:

```
MariaDB SQL Engine (parser + optimizer + executor)
       ↓ handler API
Plugin: RemoteStore → сеть → Распределённое хранилище
```

Этот headless MariaDB был бы лёгким (несколько сотен МБ RAM), запускался бы за миллисекунды и мог бы развёртываться как stateless-контейнер в Kubernetes.

## Технические вызовы

Это видение не без сложностей:

### Сетевая задержка

Каждое обращение к странице проходит через сеть. Сетевая задержка в дата-центре составляет порядка 100-500 микросекунд. Это в 100-1000 раз медленнее, чем доступ к локальному SSD. Агрессивный кэш страниц на уровне вычислительного узла необходим.

### Управление блокировками

InnoDB управляет блокировками локально. С удалённым разделяемым хранилищем управление блокировками должно быть распределённым. Это сложная проблема, которая может привести к дополнительным deadlock-ам и таймаутам.

### Распределённый buffer pool

Buffer pool InnoDB локален. В cloud-native архитектуре необходим механизм инвалидации кэша между вычислительными узлами. Когда один узел записывает страницу, кэши других узлов должны быть инвалидированы.

### Логирование транзакций

Redo log и undo log InnoDB локальны. В разделённой архитектуре WAL должен быть распределённым и доступным всем узлам.

## Что уже существует

Некоторые компоненты этого видения уже существуют в экосистеме MariaDB:

- **MariaDB ColumnStore**: колоночный движок хранения, способный использовать хранилище S3
- **Spider**: движок хранения, распределяющий данные по нескольким серверам MariaDB
- **CONNECT**: движок хранения, обращающийся к внешним источникам данных

Ни один из этих движков не реализует полное разделение вычислений и хранения, но они демонстрируют гибкость handler API.

## Заключение

Сделать MariaDB cloud-native — амбициозная, но не нереалистичная задача. Handler API предоставляет естественную точку разделения, которой нет в столь чистом виде ни у PostgreSQL, ни у Oracle MySQL.

PolarDB от Alibaba доказала, что это технически осуществимо. PostgreSQL Neon доказал, что рынок заинтересован. Вопрос не в том, «возможно ли это?», а в том, «кто сделает это первым в сообществе MariaDB?»

В тот день, когда MariaDB сможет стартовать как stateless-контейнер в 200 МБ, подключиться к распределённому хранилищу и обслуживать запросы за несколько миллисекунд — в этот день всё изменится.

---

Эта статья была первоначально опубликована на Medium.