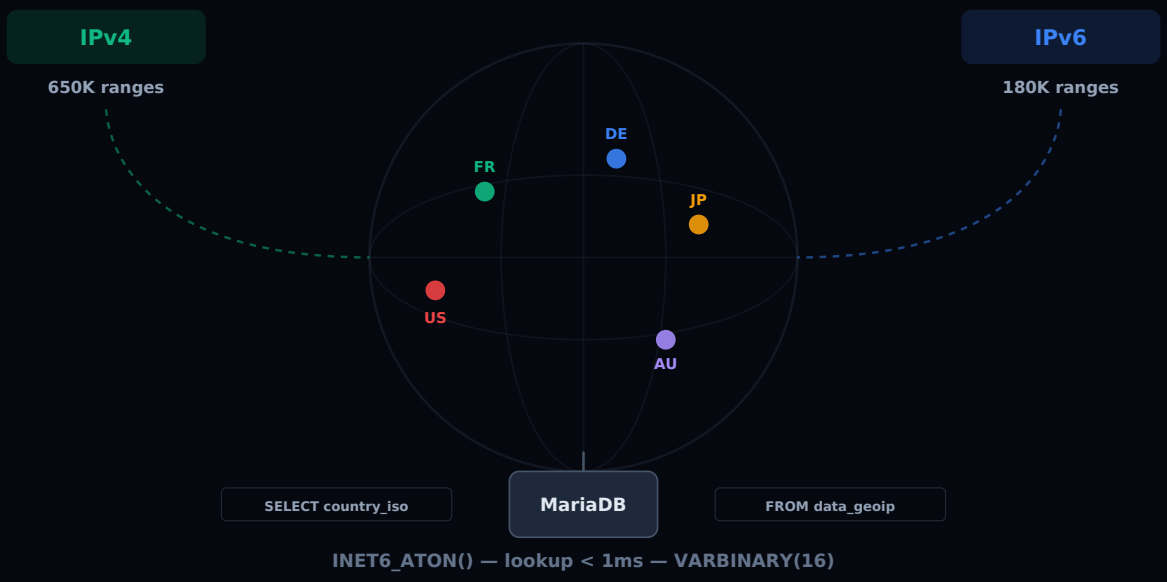
GeoIP в PmaControl: разрешение IPv4 и IPv6 без чтения файла mmdb при каждом запросе

Aurélien LEQUOY · April 15, 2026

PMACONTROL GEOIP IPV6 MARIADB PERFORMANCE ARCHITECTURE



Проблема

Когда вы мониторите 100+ серверов MariaDB / MySQL, распределённых по нескольким дата-центрам и странам, при каждом отображении главной страницы возникает один и тот же вопрос: **где находится этот сервер?**

Классический ответ: открыть файл GeoLite2 `.mmdb` от MaxMind, выполнить lookup для каждого IP и показать флаг страны. Просто… но есть нюансы:

- Открытие файла `.mmdb` размером 70 МБ **при каждом HTTP-запросе** обходится дорого
- При 100 серверах — это 100 файловых lookup'ов на страницу
- Файл представляет собой бинарное дерево, оптимизированное для последовательного чтения, а не для пакетного параллельного доступа
- В PHP-FPM каждый worker перезагружает файл независимо

В PmaControl страница `server/main` обновляется **каждую секунду** через AJAX. Открывать `.mmdb` 100 раз в секунду — абсурд.

Решение: всё поместить в MariaDB

Идея проста: **импортировать все диапазоны GeoLite2 в таблицу MariaDB**, а затем делать обычные SQL-запросы. `SELECT` с индексом работает значительно быстрее, чем обход бинарного дерева на диске.

Схема

```
CREATE TABLE data_geoip (
  id INT AUTO_INCREMENT PRIMARY KEY,
  network_start VARBINARY(16) NOT NULL,
  network_end   VARBINARY(16) NOT NULL,
  country_iso   CHAR(2) NOT NULL DEFAULT '',
  country_name  VARCHAR(100) NOT NULL DEFAULT '',
  INDEX idx_network_start (network_start)
) ENGINE=InnoDB;
```

Выбор `VARBINARY(16)` имеет принципиальное значение:

- **4 байта** достаточно для IPv4 (32 бита)
- **16 байт** необходимо для IPv6 (128 бит)
- `VARBINARY(16)` хранит оба формата единообразно
- `INET6_ATON()` преобразует любой IP (v4 или v6) в сравнимое бинарное представление

Запрос lookup

```
SELECT country_iso FROM data_geoip
WHERE network_start <= INET6_ATON('89.30.104.134')
  AND network_end   >= INET6_ATON('89.30.104.134')
LIMIT 1;
```

Результат: `FR` (Франция). Время выполнения: **< 1 мс**.

Тот же запрос работает для IPv6:

```
SELECT country_iso FROM data_geoip
WHERE network_start <= INET6_ATON('2001:4860:4860::8888')
  AND network_end   >= INET6_ATON('2001:4860:4860::8888')
LIMIT 1;
```

Результат: `US` (Соединённые Штаты — это публичный DNS Google).

Импорт: обход адресного пространства

IPv4: от 0.0.0.0 до 255.255.255.255

Адресное пространство IPv4 составляет 2^32 = 4,3 миллиарда адресов. Перебирать их по одному не нужно — используется `getWithPrefixLen()` из reader'a MaxMind, который возвращает полный CIDR для каждого адреса:

```
$ip = 0;
while ($ip <= 4294967295) {
    [$record, $prefixLen] = $reader->getWithPrefixLen(long2ip($ip));

    // Вычислить конец сети
    $networkSize = 1 << (32 - $prefixLen);
    $networkEnd = $ip + $networkSize - 1;

    if ($record && !empty($record['country']['iso_code'])) {
        // INSERT в data_geoip
    }

    // Перейти к следующему блоку CIDR
    $ip = $networkEnd + 1;
}
```

Результат: **~650 000 диапазонов**, импортированных за несколько секунд. Каждый диапазон покрывает целый блок CIDR (например, `89.30.104.0/22` — 1024 адреса в одной строке).

IPv6: пространство 2000::/3

Адресное пространство IPv6 составляет 2^128 адресов — перебирать его как IPv4 невозможно. Но публичные маршрутизируемые адреса находятся в блоке `2000::/3` (Global

Unicast), а аллокации GeoIP обычно имеют размер от `/32` до `/48`.

Принцип тот же: продвигаемся на размер возвращённого префикса. Разница в том, что работа идёт с бинарными адресами длиной 16 байт.

```
$current = inet_pton('2000::');
$end6    = inet_pton('3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff');

while ($current <= $end6) {
    [$record, $prefixLen] = $reader->getWithPrefixLen(inet_ntop($current));
    $endBin = binNetworkEnd($current, $prefixLen);

    if ($record && !empty($record['country']['iso_code'])) {
        // INSERT с UNHEX(bin2hex(...))
    }

    $current = binIncrement($endBin); // +1 в 128-битной арифметике
}
```

128-битная бинарная арифметика реализована на чистом PHP (без зависимости от GMP):

```
// Инкрементировать IPv6-адрес на 1
function binIncrement(string $bin): string|false
{
    $bytes = unpack('C16', $bin);
    for ($i = 15; $i >= 0; $i--) {
        $bytes[$i]++;
        if ($bytes[$i] <= 255) return pack('C16', ...$bytes);
        $bytes[$i] = 0; // carry
    }
    return false; // overflow
}
```

Результаты в продакшене

Объёмы

Таблица	IPv4	IPv6	Итого
`data_geoip` (country)	~650K	~180K	~830K диапазонов

Таблица	IPv4	IPv6	Итого
`data_geoip_city` (city)	~3.7M	~1.2M	~4.9M диапазонов

Производительность

Операция	Время
Импорт country (IPv4 + IPv6)	~30 секунд
Импорт city (IPv4 + IPv6)	~5 минут
Lookup 1 IP	< 1 мс
Lookup 100 IP (страница server/main)	~15 мс суммарно
Обновление AJAX (1 раз/секунду)	пренебрежимо мало

Отображение

На главной странице PmaControl каждый сервер показывает эмодзи-флаг рядом с IP:

```
□□□ 89.30.104.134:3306    PIXID-MDB-MASTER1
□□□ 136.243.1.1:3306       Hetzner-Slave
□□□ 210.171.224.1:3306     NTT-Tokyo
□□□ 8.8.8.8:3306           Google-Test
```

Приватные IP-адреса (10.x, 172.16-31.x, 192.168.x, 127.x) не имеют записей в GeoLite2 — флаг просто отсутствует.

Обновление

MaxMind обновляет GeoLite2 каждую неделю. Для обновления:

```
# Скачать новый .mmdb в data/
# Затем запустить импорт:
php App/Webroot/index.php server loadGeoip       # country (~30c)
php App/Webroot/index.php server loadGeoipCity   # city (~5мин)
```

Импорт выполняет `TRUNCATE`, а затем вставляет всё заново. Diff или миграция не нужны — это одноразовый кеш.

Таблица city: расширенные возможности

Таблица `data_geoip_city` добавляет регион, город, GPS-координаты и часовой пояс:

```
SELECT country_iso, region_name, city, latitude, longitude, time_zone
FROM data_geoip_city
WHERE network_start <= INET6_ATON('136.243.1.1')
  AND network_end   >= INET6_ATON('136.243.1.1')
LIMIT 1;
```

Результат: `DE | Saxony | Falkenstein | 50.4779 | 12.3713 | Europe/Berlin`

Это открывает возможности для картографии серверов, определения задержек между дата-центрами или просто более информативного отображения в интерфейсе.

Почему не просто LEFT JOIN?

Можно было бы сделать `LEFT JOIN data_geoip g ON g.network_start <= INET6_ATON(s.ip) AND g.network_end >= INET6_ATON(s.ip)` прямо в запросе к серверам. Проблема: при 650K диапазонов такой range-join дорого обходится оптимизатору. Мы предпочитаем N отдельных lookup'ов (по 1 на уникальный IP), которые выполняются мгновенно благодаря индексу.

Заключение

Импортируя данные GeoLite2 в MariaDB, мы устраняем зависимость от файла `.mmdb` при каждом рендеринге страницы. Lookup превращается в индексированный `SELECT` за < 1 мс, совместимый с IPv4 и IPv6, а обновление сводится к еженедельному `TRUNCATE + INSERT`.

Исходный код доступен на GitHub — мы рады вашим контрибуциям.