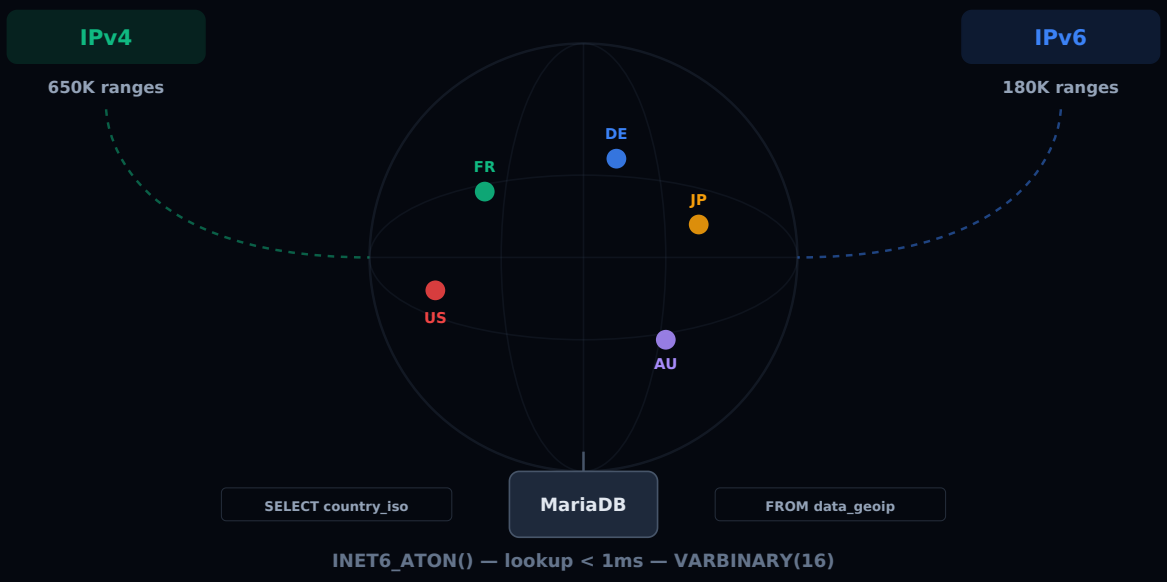
GeoIP w PmaControl: rozwiązywanie IPv4 i IPv6 bez odczytu pliku mmdb przy każdym żądaniu

Aurélien LEQUOY · April 15, 2026

PMACONTROL GEOIP IPV6 MARIADB PERFORMANCE ARCHITECTURE



Problem

Gdy monitorujesz ponad 100 serwerów MariaDB/MySQL rozproszonych po wielu centrach danych i krajach, przy każdym załadowaniu strony głównej pojawia się to samo pytanie: **gdzie jest ten serwer?**

Klasyczna odpowiedź: otworzyć plik GeoLite2 `.mmdb` od MaxMind, wykonać lookup dla każdego IP i wyświetlić flagę kraju. Proste... z wyjątkiem tego, że:

- Otwieranie pliku `.mmdb` o rozmiarze 70 MB **przy każdym żądaniu HTTP** jest kosztowne
- Przy 100 serwerach do rozwiązania oznacza to 100 odczytów pliku na stronę
- Plik jest drzewem binarnym zoptymalizowanym pod sekwencyjny odczyt, nie pod równoległe serie zapytań
- W PHP-FPM każdy worker ładuje plik niezależnie

W PmaControl strona `server/main` odświeża się **co sekundę** przez AJAX. Otwieranie `.mmdb` 100 razy na sekundę nie ma sensu.

Rozwiązanie: wszystko w MariaDB

Pomysł jest prosty: **zaimportować wszystkie zakresy GeoLite2 do tabeli MariaDB**, a następnie wykonywać standardowe zapytania SQL. `SELECT` z indeksem jest znacznie szybszy niż przeszukiwanie drzewa binarnego na dysku.

Schemat

```
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;
```

Wybór `VARBINARY(16)` jest kluczowy:

- **4 bajty** wystarczą dla IPv4 (32 bity)
- **16 bajtów** jest potrzebnych dla IPv6 (128 bitów)
- `VARBINARY(16)` przechowuje oba formaty jednolicie
- `INET6_ATON()` konwertuje dowolny adres IP (v4 lub v6) na porównywalną postać binarną

Zapytanie 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;
```

Wynik: `FR` (Francja). Czas wykonania: **< 1 ms**.

To samo zapytanie działa dla 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;
```

Wynik: `US` (Stany Zjednoczone — to publiczny DNS Google).

Import: iteracja po przestrzeni IP

IPv4: od 0.0.0.0 do 255.255.255.255

Przestrzeń IPv4 obejmuje 2^32 = 4,3 miliarda adresów. Nie iterujemy po nich jeden po jednym — używamy `getWithPrefixLen()` z readera MaxMind, która zwraca pełny CIDR dla każdego adresu:

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

    // Oblicz koniec sieci
    $networkSize = 1 << (32 - $prefixLen);
    $networkEnd = $ip + $networkSize - 1;

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

    // Przeskocz cały blok CIDR
    $ip = $networkEnd + 1;
}
```

Wynik: **~650 000 zakresów** zaimportowanych w kilka sekund. Każdy zakres obejmuje cały blok CIDR (np. `89.30.104.0/22` → 1024 adresy w jednym wierszu).

IPv6: przestrzeń 2000::/3

Przestrzeń IPv6 obejmuje 2^128 adresów — niemożliwe do iterowania jak w przypadku IPv4. Ale publiczne adresy routowalne znajdują się w bloku `2000::/3` (Global Unicast), a alokacje GeoIP to zazwyczaj /32 do /48.

Zasada jest taka sama: przesuwamy się o rozmiar zwróconego prefiksu. Różnica polega na tym, że pracujemy z 16-bajtowymi adresami binarnymi.

```
$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 z UNHEX(bin2hex(...))
    }

    $current = binIncrement($endBin); // +1 w arytmetyce 128-bitowej
}
```

Arytmetyka binarna 128-bitowa jest zaimplementowana w czystym PHP (bez wymagania GMP):

```
// Inkrementacja adresu IPv6 o 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
}
```

Wyniki na produkcji

Wolumeny

Tabela	IPv4	IPv6	Razem
`data_geoip` (kraj)	~650K	~180K	~830K zakresów
`data_geoip_city` (miasto)	~3,7M	~1,2M	~4,9M zakresów

Wydajność

Operacja	Czas
Import country (IPv4 + IPv6)	~30 sekund
Import city (IPv4 + IPv6)	~5 minut
Lookup 1 IP	< 1 ms
Lookup 100 IP (strona server/main)	~15 ms łącznie
Odświeżenie AJAX (1x/sekundę)	pomijalne

Wyświetlanie

Na stronie głównej PmaControl każdy serwer wyświetla flagę emoji obok adresu 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
```

Prywatne adresy IP (10.x, 172.16-31.x, 192.168.x, 127.x) nie mają rekordu GeoLite2 — flaga jest po prostu nieobecna.

Aktualizacja

MaxMind aktualizuje GeoLite2 co tydzień. Aby odświeżyć:

```
# Pobierz nowy plik .mmdb do data/
# Następnie uruchom ponownie import:
php App/Webroot/index.php server loadGeoip       # country (~30s)
php App/Webroot/index.php server loadGeoipCity   # city (~5min)
```

Import wykonuje `TRUNCATE`, a następnie wstawia wszystko od nowa. Nie trzeba robić diff ani migracji — to jednorazowy cache.

Tabela city: idąc dalej

Tabela `data_geoip_city` dodaje region, miasto, współrzędne GPS i strefę czasową:

```
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;
```

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

To otwiera drzwi do kartografii serwerów, wykrywania opóźnień między centrami danych lub po prostu bogatszego wyświetlania w interfejsie.

Dlaczego nie po prostu LEFT JOIN?

Można by chcieć zrobić `LEFT JOIN data_geoip g ON g.network_start <= INET6_ATON(s.ip) AND g.network_end >= INET6_ATON(s.ip)` bezpośrednio w zapytaniu o serwery. Problem: przy 650K zakresów ten range-join jest kosztowny dla optymalizatora. Wolimy N indywidualnych lookupów (1 na unikalny IP), które są natychmiastowe dzięki indeksowi.

Podsumowanie

Importując dane GeoLite2 do MariaDB, eliminujemy zależność od pliku `.mmdb` przy każdym renderowaniu strony. Lookup staje się indeksowanym `SELECT` w czasie < 1 ms, kompatybilnym z IPv4 i IPv6, a aktualizacja to prosty cotygodniowy `TRUNCATE + INSERT`.

Kod źródłowy jest dostępny na GitHub — wkład mile widziany.