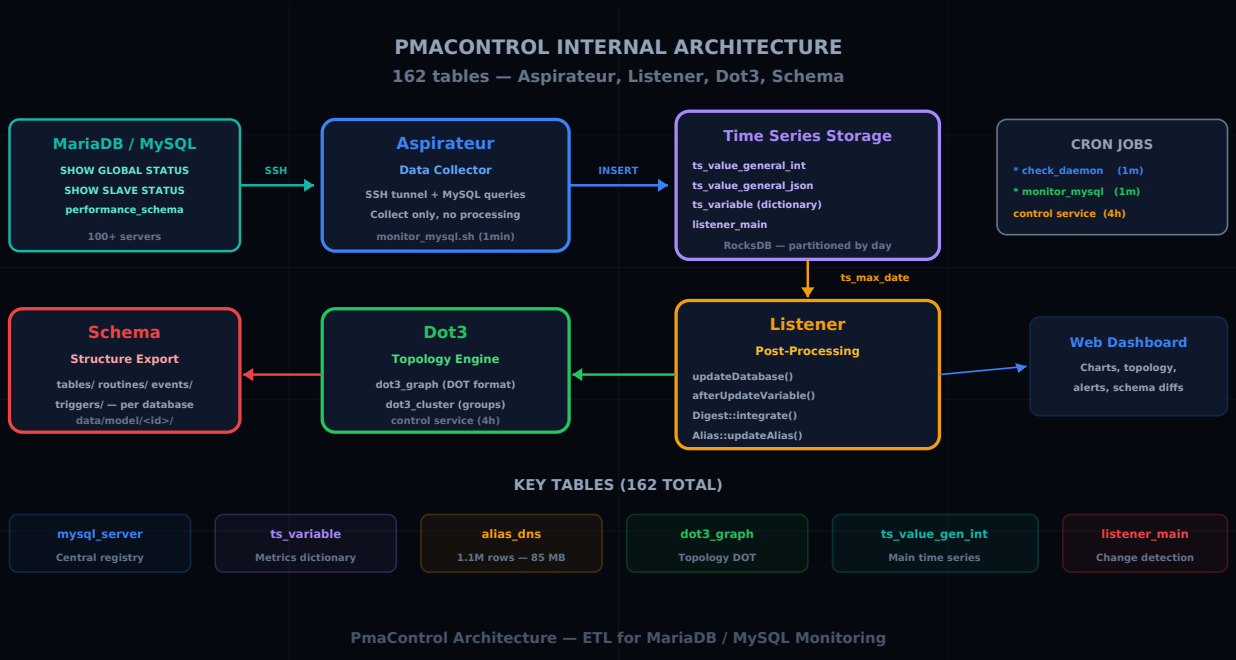
Architektura wewnętrzna PmaControl: Aspirateur, Listener, Dot3 i 162 tabele

Aurélien LEQUOY · April 13, 2026

PMACONTROL ARCHITECTURE AGENTS CRON MONITORING



162 tabele, żadna zbędna

PmaControl to nie prosty dashboard. To system rozproszony, który zbiera, przechowuje, transformuje i udostępnia metryki setek serwerów MariaDB / MySQL w czasie rzeczywistym. Wewnętrzna baza danych zawiera **162 tabele** — każda z precyzyjną rolą w potoku przetwarzania danych.

Ten artykuł szczegółowo opisuje architekturę wewnętrzną: cztery główne komponenty (Aspirateur, Listener, Dot3, Schema), przepływ danych od początku do końca, zadania cron orkiestrujące całość oraz kluczowe tabele do poznania w celu debugowania lub rozszerzania systemu.

Cztery filary

Aspirateur: kolektor danych

Aspirateur to komponent pobierający metryki z każdego nadzorowanego serwera. Jego działanie jest proste, ale skuteczne:

1. Łączy się z serwerem przez **SSH** (tunel), a następnie otwiera lokalne połączenie **MySQL**
2. Wykonuje serię zapytań: `SHOW GLOBAL STATUS`, `SHOW GLOBAL VARIABLES`, `SHOW SLAVE STATUS`, `SHOW PROCESSLIST`, zapytania do `performance_schema` itp.
3. Zapisuje wyniki do tabel `ts_value_*` (szeregi czasowe) bazy PmaControl

Prefiks `ts_` jest wszechobecny: oznacza **time series** (szeregi czasowe). Każda metryka jest opatrzona znacznikiem czasu i przechowywana z identyfikatorem serwera źródłowego.

```
Aspirateur → tunel SSH → lokalne MySQL
           → SHOW GLOBAL STATUS
           → SHOW SLAVE STATUS
           → zapytania performance_schema
           → INSERT INTO ts_value_general_int (...)
           → INSERT INTO ts_value_general_json (...)
```

Aspirateur **nie wykonuje żadnego przetwarzania**. Zbiera i zapisuje. To fundamentalna zasada projektowa: oddzielenie zbierania od przetwarzania, aby można je było skalować niezależnie.

Listener: silnik przetwarzania końcowego

Listener to mózg PmaControl. Monitoruje tabele szeregów czasowych i wyzwala akcje, gdy pojawiają się nowe dane. Jego mechanizm opiera się na tabeli pivot: `listener_main`.

Tabela `listener_main` zawiera:

Kolumna	Rola
`ts_file`	Plik źródłowy danych
`ts_max_date`	Ostatni przetworzony znacznik czasu
`ts_date_by_server`	Ostatni znacznik czasu według serwera

Listener działa w pętli. W każdej iteracji porównuje zapisany `ts_max_date` z najnowszym znacznikiem czasu w tabelach `ts_value_*`. Jeśli wykryto różnicę, oznacza to, że Aspirateur zapisał nowe dane — Listener wyzwala wtedy łańcuch przetwarzania końcowego:

```
Pętla Listenera:
  1. Sprawdź ts_max_date vs rzeczywisty max(timestamp)
  2. Jeśli zmieniony → wyzwól potok:
     a. updateDatabase()      — aktualizuje metadane serwera
     b. afterUpdateVariable() — wyzwala reguły warunkowe
     c. Digest::integrate()   — agreguje metryki performance_schema
     d. Alias::updateAlias()  — odświeża aliasy DNS
```

updateDatabase() synchronizuje podstawowe informacje: wersja serwera, stan replikacji, rozmiar baz danych, liczba aktywnych połączeń.

afterUpdateVariable() to silnik reguł. Porównuje nowe wartości z skonfigurowanymi progami i w razie potrzeby generuje alerty. Na przykład, jeśli `Seconds_Behind_Master` przekroczy 60, tworzony jest alert Warning.

Digest::integrate() przetwarza dane z `performance_schema`. Agreguje statystyki zapytań (czas wykonania, przeanalizowane wiersze, częstotliwość) i przechowuje je w tabelach digest PmaControl. To dane zasilające dashboardy wydajności.

Alias::updateAlias() utrzymuje tabelę `alias_dns`, która mapuje przyjazne nazwy na rzeczywiste adresy IP. Ta tabela jest jedną z największych: **1,1 miliona wierszy, 85 MB danych**. Aliasy są używane w całym interfejsie do wyświetlania czytelnych nazw zamiast adresów IP.

Dot3: topologia w czasie rzeczywistym

Dot3 to komponent mapowania topologii. Analizuje relacje replikacji między serwerami i generuje graf skierowany w formacie DOT (Graphviz).

Proces:

1. Dot3 odczytuje metadane replikacji każdego serwera (master/slave, GTID, kanał)
2. Buduje graf zależności: kto jest masterem kogo, kto jest slave'em kogo
3. Generuje wizualną reprezentację z klastrami (grupy powiązanych serwerów)

Zaangażowane tabele:

- **`dot3_graph`**: kompletny graf w formacie DOT, gotowy do renderowania
- **`dot3_cluster`**: klastry serwerów (klaster = grupa replikacji)

Dot3 jest szczególnie przydatny do wykrywania uszkodzonych topologii: slave wskazujący na nieistniejący serwer, nieoczekiwana pętla replikacji kołowej lub izolowany serwer, który powinien być w klastrze.

Schema: eksport struktury

Komponent Schema eksportuje kompletną strukturę każdej nadzorowanej bazy danych. Dla każdego serwera tworzy drzewo plików:

```
data/model/<server_id>/databases/<db_name>/
├── schema/
│   └── tables/
│       ├── users.sql
│       ├── orders.sql
│       └── ...
├── routines/
│   ├── calculate_total.sql
│   └── ...
├── events/
│   ├── daily_cleanup.sql
│   └── ...
└── triggers/
    ├── before_insert_users.sql
    └── ...
```

Każdy plik zawiera odpowiedni `CREATE TABLE`, `CREATE PROCEDURE`, `CREATE EVENT` lub `CREATE TRIGGER`. Umożliwia to:

- Wersjonowanie struktury w Git (diff między dwoma eksportami)
- Porównywanie struktury między produkcją a stagingiem
- Wykrywanie dryfu schematu (indeks dodany ręcznie na produkcji, kolumna zmodyfikowana bez migracji)

CLI Glial

PmaControl jest zbudowany na frameworku Glial, który zapewnia standaryzowany interfejs wiersza poleceń:

```
./glial <controller> <action> [params]
```

Konkretne przykłady:

```
# Sprawdzenie stanu demonów
./glial agent check_daemon

# Wymuszenie cyklu zbierania
./glial control service

# Eksport schematu serwera
./glial schema export 42

# Regeneracja topologii
./glial dot3 generate
```

CLI jest używany zarówno ręcznie (debugowanie, konserwacja), jak i przez zadania cron do automatycznej orkiestracji.

Zadania cron: orkiestracja

Trzy niezbędne zadania cron napędzają PmaControl:

1. `./glial agent check_daemon` — co minutę

To najczęstsze zadanie cron. Sprawdza, czy wszystkie procesy agentów żyją i restartuje je w razie potrzeby. Martwy agent oznacza lukę w danych — ten cron gwarantuje ciągłość zbierania.

```
* * * * * cd /srv/www/pmacontrol && ./glial agent check_daemon >> /tmp/pmacontrol_agent.log 2>&1
```

Jeśli agent nie odpowiada po 3 próbach, wysyłany jest alert Telegram.

2. `./glial control service` — co 4 godziny

To zadanie cron wykonuje ciężkie zadania konserwacyjne:

- Przeliczanie dziennych agregacji
- Czyszczenie wygasłych danych (konfigurowalna retencja)
- Regeneracja topologii Dot3
- Synchronizacja metadanych serwera
- Weryfikacja spójności między tabelami

Cztery godziny to dobry kompromis między świeżością a obciążeniem: te operacje są kosztowne i nie muszą być realizowane w czasie rzeczywistym.

```
0 */4 * * * cd /srv/www/pmacontrol && ./glial control service >> /tmp/pmacontrol_control.log 2>&1
```

3. `./monitor_mysql.sh` — co minutę

Ten skrypt jest punktem wejścia Aspirateura. Wyzwala kompletny cykl zbierania:

```
* * * * * cd /srv/www/pmacontrol && ./monitor_mysql.sh >> /tmp/pmacontrol_monitor.log 2>&1
```

Skrypt zarządza paralelizacją: jeśli nadzorujesz 200 serwerów, nie kontaktuje się z nimi sekwencyjnie. Rozkłada pracę na równoległe partie, z konfigurowalną liczbą workerów.

Kluczowe tabele

Oto najważniejsze tabele do poznania, aby zrozumieć lub debugować PmaControl:

`mysql_server`

Centralna tabela. Każdy wiersz reprezentuje **jedną nadzorowaną instancję** — nie tylko serwery MariaDB / MySQL, ale także:

- **Serwery MariaDB / MySQL** (główny przypadek)
- **Proxy**: MaxScale, ProxySQL, HAProxy
- **VIP** (Virtual IP)

Kolumny `is_proxy` i `is_vip` rozróżniają typy:

is_proxy	is_vip	Typ
0	0	Klasyczny serwer MariaDB / MySQL
1	0	Proxy (MaxScale, ProxySQL, HAProxy)
0	1	VIP (Virtual IP)

```
-- Tylko serwery MariaDB/MySQL
SELECT id, ip, port, name, display_name, id_environment
FROM mysql_server
```

```
WHERE is_deleted = 0 AND is_proxy = 0 AND is_vip = 0;

-- Proxy (MaxScale, ProxySQL, HAProxy)
SELECT id, ip, port, name, display_name
FROM mysql_server
WHERE is_deleted = 0 AND is_proxy = 1;

-- VIP
SELECT id, ip, port, name, display_name
FROM mysql_server
WHERE is_deleted = 0 AND is_vip = 1;
```

Proxy i VIP są przechowywane w tej samej tabeli co serwery MySQL, aby uprościć joiny i topologię. Dot3 używa ich do rysowania połączeń między warstwami sieci (VIP → Proxy → Master → Slave). Kolumna `timeout` jest obliczana dynamicznie: 11 sekund dla proxy (które odpowiadają wolniej na sprawdzenia), 1 sekunda dla klasycznych serwerów.

Dedykowane tabele uzupełniają szczegóły specyficzne dla każdego typu proxy:

- **`maxscale_server`** / **`maxscale_server__mysql_server`** — konfiguracja MaxScale i jej backendy
- **`proxysql_server`** — konfiguracja ProxySQL
- **`haproxy_main`** / **`haproxy_main_input`** / **`haproxy_main_output`** / **`link__haproxy_main_output__mysql_server`** — konfiguracja HAProxy (listenery, frontendy, backendy)
- **`vip_server`** — szczegóły VIP

`ts_variable`

Słownik metryk. Każda zbierana zmienna (na przykład `Threads_connected`, `Innodb_buffer_pool_pages_data`) ma wpis w tej tabeli ze swoim identyfikatorem numerycznym.

```
SELECT id, name, source
FROM ts_variable
WHERE name LIKE 'Innodb%';
```

`ts_value_general_int`

Główny magazyn metryk numerycznych. To najbardziej wolumenowa tabela — otrzymuje tysiące insertów na sekundę i może osiągnąć kilka miliardów wierszy dziennie na największych instalacjach PmaControl.

```
SELECT server_id, variable_id, value, timestamp
FROM ts_value_general_int
WHERE server_id = 42
  AND variable_id = 107  -- Threads_connected
  AND timestamp > NOW() - INTERVAL 1 HOUR;
```

Ta tabela jest partycjonowana według dni, aby umożliwić szybkie czyszczenie starych danych (`ALTER TABLE ... DROP PARTITION`).

`ts_value_general_json`

Dla złożonych metryk, które nie mieszczą się w liczbie całkowitej: wyniki `SHOW PROCESSLIST` , tabele `performance_schema` (digesty zapytań, blokady, I/O tabel), `SHOW ENGINE INNODB STATUS` itp. Format JSON pozwala na przechowywanie dowolnych struktur. Metryki replikacji (`SHOW SLAVE STATUS`) mają dedykowane tabele (`ts_value_slave_*`).

`alias_dns`

Tabela aliasów DNS — 1,1 miliona wierszy, 85 MB. Mapuje adresy IP na czytelne nazwy i jest używana w całym interfejsie.

```
SELECT ip, alias, source, updated_at
FROM alias_dns
WHERE ip = '10.0.1.42';
```

`dot3_graph` i `dot3_cluster`

Tabele topologii. `dot3_graph` zawiera kompletny graf DOT, `dot3_cluster` logiczne grupy serwerów.

Kompletny przepływ danych

Podsumujmy drogę metryki od źródła do ekranu:

Etap	Komponent	Akcja
1	**CRON** `monitor_mysql.sh` (co minutę)	Uruchamia Aspirateura
2	**ASPIRATEUR**	Tunel SSH → MySQL → `SHOW GLOBAL STATUS`

Etap	Komponent	Akcja
		Zapisuje do `ts_value_general_int` / `ts_value_general_json`
3	**LISTENER** (wykrywa zmianę `ts_max_date`)	`updateDatabase()` — aktualizuje metadane serwera
		`afterUpdateVariable()` — alerty przy przekroczeniu progów
		`Digest::integrate()` — agregacja `performance_schema`
		`Alias::updateAlias()` — odświeża `alias_dns`
4	**DOT3** (pętla co ~3s)	Regeneruje topologię replikacji w czasie rzeczywistym
5	**CRON** `control service` (co 4h)	Czyszczenie i dzienna agregacja
6	**INTERFEJS WEB**	Odczytuje zagregowane tabele → dashboardy, wykresy, topologia

Wymiarowanie

Dla typowego wdrożenia 100 serwerów MariaDB / MySQL:

- **Baza PmaControl**: około 15 GB danych (zdominowana przez tabele `ts_value_*`)
- **CPU**: 2-4 rdzenie wystarczą (Listener jest najbardziej wymagający)
- **RAM**: minimum 4 GB, zalecane 8 GB (dla buffer pool samej bazy PmaControl)
- **Dysk**: obowiązkowe SSD — tabele szeregów czasowych generują dużo losowych I/O

Zalecany silnik przechowywania dla tabel `ts_value_*` to **RocksDB** (przez MyRocks): lepsza kompresja, lepsza wydajność sekwencyjnych zapisów i natywne partycjonowanie według dni.

Debugowanie

Gdy coś nie działa, oto lista kontrolna:

1. **Czy agenci działają?** `./glial agent check_daemon` — jeśli agent jest martwy, dane nie są już zbierane dla zarządzanych przez niego serwerów
2. **Czy Listener działa?** Sprawdź `ts_max_date` w `listener_main` — jeśli nie postępuje, Listener jest zablokowany
3. **Czy zadania cron się wykonują?** Sprawdź `/tmp/pmacontrol_*.log` pod kątem błędów
4. **Czy łączność SSH jest OK?** Przetestuj ręcznie `ssh -p <port> <user>@<host>` z skonfigurowanym kluczem
5. **Czy baza PmaControl jest zdrowa?** Sprawdź przestrzeń dyskową, blokady, wolne zapytania na samej bazie PmaControl

Podsumowanie

Architektura PmaControl podąża za klasycznym modelem ETL (Extract-Transform-Load) dostosowanym do monitoringu:

- **Extract**: Aspirateur zbiera bez transformowania
- **Transform**: Listener stosuje reguły i agreguje
- **Load**: dashboardy odczytują przetransformowane dane

162 tabele nie są przypadkiem złożoności — odzwierciedlają bogactwo danych zbieranych z każdego serwera MariaDB / MySQL. Zrozumienie tej architektury jest niezbędne dla każdego, kto chce zdebugować problem ze zbieraniem danych, rozszerzyć PmaControl o nowy typ metryki lub zoptymalizować wydajność samego systemu nadzoru.