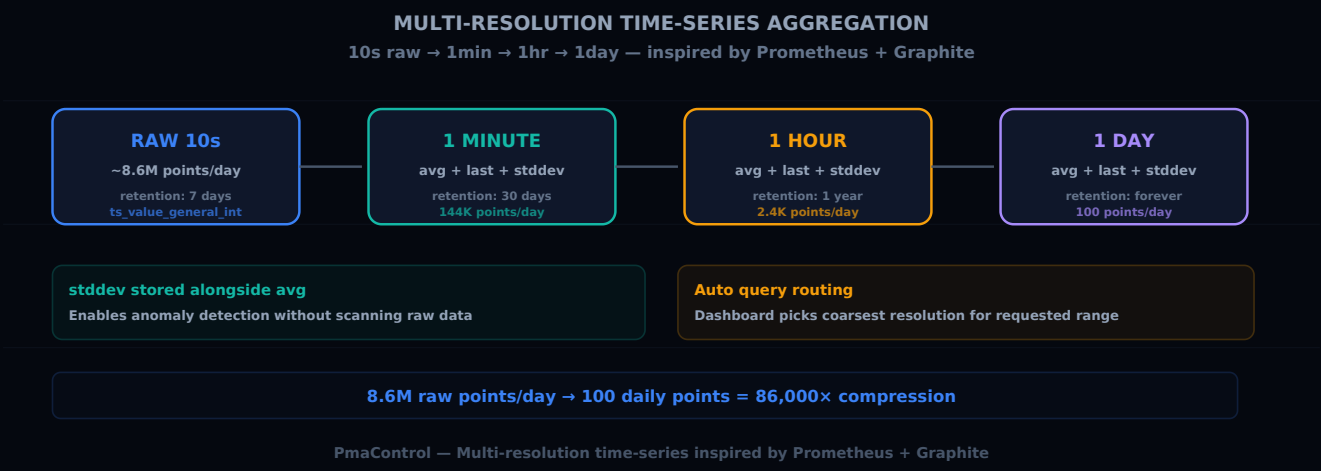
Agregacja szeregów czasowych: od milionów surowych punktów do szybkich zapytań

Aurélien LEQUOY · March 21, 2026

PMACONTROL TIME-SERIES AGGREGATION MONITORING ARCHITECTURE



Surowy wolumen

PmaControl zbiera metryki z każdej nadzorowanej instancji MariaDB / MySQL co **10 sekund**. Dla każdego serwera oznacza to:

- 6 punktów na minutę
- 360 punktów na godzinę
- 8 640 punktów dziennie
- **60 480 punktów tygodniowo**

Przy 100 serwerach i 50 metrykach na serwer:

```
100 serwerów × 50 metryk × 8 640 punktów/dzień = 43 200 000 punktów/dzień
```

43 miliony punktów dziennie. 302 miliony tygodniowo. Ponad miliard miesięcznie.

Przechowywanie tego wszystkiego w surowej rozdzielczości (10 sekund) w nieskończoność jest technicznie możliwe, ale praktycznie bezużyteczne. Nikt nie ogląda wykresu z rozdzielczością 10 sekund dla danych sprzed 6 miesięcy. A zapytania skanujące miliony wierszy w celu wyświetlenia

rocznego wykresu są wolne i kosztowne.

Inspiracja: Prometheus i Graphite

Problem nie jest nowy. Dwa systemy rozwiązały go w elegancki sposób:

- **Prometheus** ze swoimi **recording rules**: wstępnie obliczanymi zapytaniami PromQL, które agregują surowe dane w metryki pochodne w regularnych odstępach
- **Graphite** z formatem **Whisper**: systemem retencji wielorozdzielczej, w którym dane są automatycznie agregowane w miarę starzenia się

PmaControl czerpie inspirację z obu podejść, projektując własny system agregacji.

Schemat wielorozdzielczy

Cztery poziomy rozdzielczości:

Poziom	Interwał	Retencja	Szacowany wolumen (100 srv)
Raw	10 sekund	7 dni	302M punktów/tydzień
1 minuta	1 minuta	30 dni	216M punktów/miesiąc
1 godzina	1 godzina	1 rok	43.8M punktów/rok
1 dzień	1 dzień	Nieokreślona	1.8M punktów/rok

Całkowity wolumen przechowywany w dowolnym momencie (przy 100 serwerach):

```
Raw (7 dni):       302M punktów
1min (30 dni):     216M punktów
1hr (1 rok):        43.8M punktów
1day (wszystko):     ~2M punktów
Razem:             ~564M punktów
```

Bez agregacji przechowywanie roku surowych danych oznaczałoby **15,8 miliarda punktów**. Agregacja zmniejsza ilość przechowywanych danych 28-krotnie.

Co przechowujemy na każdym poziomie agregacji

Dla każdego zagregowanego punktu przechowywane są trzy wartości:

```
CREATE TABLE ts_aggregated_1min (
    server_id     INT,
    metric_id     INT,
    timestamp     DATETIME,
    last_value    DOUBLE,    -- ostatnia wartość interwału
    avg_value     DOUBLE,    -- średnia interwału
    stddev_value  DOUBLE,    -- odchylenie standardowe interwału
    PRIMARY KEY (server_id, metric_id, timestamp)
);
```

Dlaczego `last_value`?

Dla metryk typu licznikowego (liczba zapytań, wysłane bajty) ostatnia wartość interwału jest często bardziej istotna niż średnia. Reprezentuje najnowszy stan.

Dlaczego `avg_value`?

Dla metryk typu gauge (wykorzystanie CPU, pamięć, aktywne wątki) średnia jest najwierniejszą reprezentacją zachowania w danym interwale.

Dlaczego `stddev_value`? Kluczowy wgląd

To główna innowacja tego projektu. **Przechowywanie odchylenia standardowego obok średniej umożliwia wykrywanie anomalii bez surowych danych.**

Rozważmy dwie godziny z taką samą średnią CPU wynoszącą 45%:

- **Godzina A**: CPU stabilne między 42% a 48%. `avg=45%, stddev=2%`
- **Godzina B**: CPU oscylujące między 5% a 85%. `avg=45%, stddev=28%`

Bez stddev te dwie godziny są nieodróżnialne w zagregowanych danych. Ze stddev godzina B jest natychmiast identyfikowalna jako anomalna.

Pozwala to budować alerty oparte na historycznym stddev:

```
JEŚLI stddev_bieżący > 3 × stddev_średni_30_ostatnich_dni
WTEDY alert: wykryto nienormalne zachowanie
```

Proces agregacji

Agregacja działa kaskadowo, sterowana przez zadanie cron:

Etap 1: Raw → 1 minuta

Co minutę worker odczytuje 6 ostatnich surowych punktów dla każdej pary (serwer, metryka) i oblicza:

```
INSERT INTO ts_aggregated_1min (server_id, metric_id, timestamp, last_value, avg_value,
stddev_value)
SELECT
    server_id,
    metric_id,
    DATE_FORMAT(timestamp, '%Y-%m-%d %H:%i:00') AS minute,
    -- last_value: podzapytanie dla ostatniego punktu
    (SELECT value FROM ts_raw r2
     WHERE r2.server_id = ts_raw.server_id
       AND r2.metric_id = ts_raw.metric_id
       AND r2.timestamp >= DATE_FORMAT(ts_raw.timestamp, '%Y-%m-%d %H:%i:00')
       AND r2.timestamp < DATE_FORMAT(ts_raw.timestamp, '%Y-%m-%d %H:%i:00') + INTERVAL 1
MINUTE
     ORDER BY r2.timestamp DESC LIMIT 1),
    AVG(value),
    STDDEV(value)
FROM ts_raw
WHERE timestamp >= NOW() - INTERVAL 1 MINUTE
GROUP BY server_id, metric_id, minute;
```

Etap 2: 1 minuta → 1 godzina

Co godzinę worker agreguje 60 punktów 1-minutowych w jeden punkt 1-godzinny. Obliczanie połączonego stddev wykorzystuje wzór na wariancję łączoną:

```
σ_połączone = sqrt( mean(σ²_i) + var(μ_i) )
```

Gdzie `σ_i` to odchylenia standardowe podinterwałów, a `μ_i` ich średnie. Ten wzór jest matematycznie dokładny i nie wymaga surowych danych.

Etap 3: 1 godzina → 1 dzień

Ta sama zasada, raz dziennie, 24 punkty 1-godzinne stają się jednym punktem 1-dniowym.

Etap 4: Czyszczenie starych danych

Po każdej agregacji dane poza retencją są usuwane:

```
DELETE FROM ts_raw WHERE timestamp < NOW() - INTERVAL 7 DAY;
DELETE FROM ts_aggregated_1min WHERE timestamp < NOW() - INTERVAL 30 DAY;
DELETE FROM ts_aggregated_1hr WHERE timestamp < NOW() - INTERVAL 1 YEAR;
-- ts_aggregated_1day: nigdy nie czyszczone
```

Routing zapytań

Gdy dashboard PmaControl wyświetla wykres, musi wybrać odpowiednią rozdzielczość. Zasada jest prosta: **użyj najgrubszej rozdzielczości, która pokrywa żądany zakres**.

```
function selectResolution(int $timeRangeSeconds): string {
    if ($timeRangeSeconds <= 3600) {       // <= 1 godzina
        return 'ts_raw';                   // rozdzielczość 10s
    } elseif ($timeRangeSeconds <= 86400 * 2) {  // <= 2 dni
        return 'ts_aggregated_1min';       // rozdzielczość 1min
    } elseif ($timeRangeSeconds <= 86400 * 90) { // <= 90 dni
        return 'ts_aggregated_1hr';        // rozdzielczość 1hr
    } else {
        return 'ts_aggregated_1day';       // rozdzielczość 1day
    }
}
```

Rezultat: wykres na 1 rok ładuje tylko **365 punktów** (rozdzielczość 1 dzień) zamiast 3,1 miliona (rozdzielczość 10 sekund). Zapytanie przechodzi z kilku sekund do kilku milisekund.

Wpływ na zapytania

Żądany zakres	Rozdzielczość	Załadowane punkty	Czas zapytania
1 godzina	10s (raw)	360	< 10 ms
24 godziny	1 min	1 440	< 20 ms
30 dni	1 godzina	720	< 15 ms
1 rok	1 dzień	365	< 10 ms

Czasy zapytań stają się **niezależne od zakresu czasowego**. Wykres roczny jest tak samo szybki jak wykres godzinny.

Wykrywanie anomalii z przechowywanym stddev

Dzięki wstępnie obliczonemu stddev PmaControl może wykrywać anomalie na zagregowanych danych bez odwoływania się do surowych danych:

1. **Obliczanie linii bazowej**: średnia i stddev z stddev z ostatnich 30 dni dla każdej metryki
2. **Porównanie**: stddev bieżącej godziny jest porównywany z linią bazową
3. **Alert**: jeśli stddev przekracza 3-krotność linii bazowej, zachowanie jest nienormalne

Konkretny przykład:

- Linia bazowa threads_running: `avg_stddev = 2.1, stddev_stddev = 0.8`
- Bieżąca godzina: `stddev = 14.3`
- Wynik: `(14.3 - 2.1) / 0.8 = 15.25` sigm — **pewna anomalia**

Ten mechanizm wykrywa anomalie, które prosta średnia by przeoczyła: serwer, którego CPU oscyluje gwałtownie, ale zawsze wraca do prawidłowej średniej.

Podsumowanie

Agregacja wielorozdzielcza jest kluczem do zarządzania danymi szeregów czasowych na dużą skalę. Przechowywanie stddev obok średniej jest nietypowym, ale potężnym wyborem projektowym: zachowuje informację o zmienności, umożliwiając wykrywanie anomalii nawet na zagregowanych danych.

Dzięki temu systemowi PmaControl może nadzorować ponad 100 serwerów MariaDB / MySQL przez pełny rok, gwarantując jednocześnie zapytania dashboardowe w mniej niż 20 milisekund.