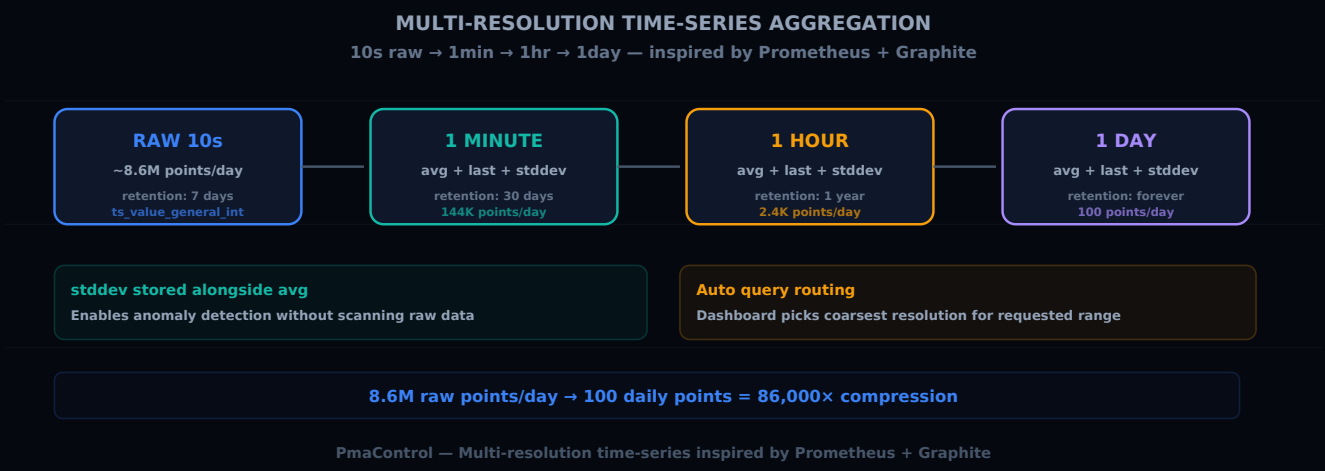
Агрегация временных рядов: от миллионов сырых точек к быстрым запросам

Aurélien LEQUOY · March 21, 2026

PMACONTROL TIME-SERIES AGGREGATION MONITORING ARCHITECTURE



Объём сырых данных

PmaControl собирает метрики с каждого наблюдаемого экземпляра MariaDB / MySQL каждые **10 секунд**. Для каждого сервера это:

- 6 точек в минуту
- 360 точек в час
- 8 640 точек в сутки
- **60 480 точек в неделю**

При 100 серверах и 50 метриках на каждый:

```
100 серверов × 50 метрик × 8 640 точек/сутки = 43 200 000 точек/сутки
```

43 миллиона точек в сутки. 302 миллиона в неделю. Более миллиарда в месяц.

Хранить всё это в сыром разрешении (10 секунд) бесконечно технически возможно, но практически бесполезно. Никто не смотрит на график с 10-секундным разрешением по данным полугодовой давности. А запросы, сканирующие миллионы строк для отображения

годового графика, медленны и дороги.

Вдохновение: Prometheus и Graphite

Проблема не нова. Два системы решили её элегантно:

- **Prometheus** с **recording rules**: предварительно вычисляемые PromQL-запросы, агрегирующие сырые данные в производные метрики через регулярные интервалы
- **Graphite** с форматом **Whisper**: система мультирезолюционного хранения, где данные автоматически агрегируются по мере старения

PmaControl заимствует оба подхода для проектирования собственной системы агрегации.

Схема мультирезолюции

Четыре уровня разрешения:

Уровень	Интервал	Хранение	Расчётный объём (100 серв.)
Raw	10 секунд	7 дней	302M точек/нед.
1 минута	1 минута	30 дней	216M точек/мес.
1 час	1 час	1 год	43.8M точек/год
1 день	1 день	Бессрочно	1.8M точек/год

Суммарный объём, хранимый в любой момент (при 100 серверах):

```
Raw (7 дней):      302M точек
1min (30 дней):    216M точек
1hr (1 год):        43.8M точек
1day (всё):          ~2M точек
Итого:             ~564M точек
```

Без агрегации хранение года сырых данных составило бы **15,8 миллиарда точек**. Агрегация сокращает объём хранения в 28 раз.

Что хранится на каждом агрегированном уровне

Для каждой агрегированной точки сохраняются три значения:

```
CREATE TABLE ts_aggregated_1min (
    server_id     INT,
    metric_id     INT,
    timestamp     DATETIME,
    last_value    DOUBLE,    -- последнее значение интервала
    avg_value     DOUBLE,    -- среднее за интервал
    stddev_value  DOUBLE,    -- стандартное отклонение за интервал
    PRIMARY KEY (server_id, metric_id, timestamp)
);
```

Зачем `last_value`?

Для метрик типа счётчик (количество запросов, отправленные байты) последнее значение интервала часто информативнее среднего. Оно отражает самое актуальное состояние.

Зачем `avg_value`?

Для метрик типа gauge (загрузка CPU, память, активные потоки) среднее — наиболее точное отображение поведения на интервале.

Зачем `stddev_value`? Ключевой инсайт

Это главная инновация данного дизайна. **Хранение стандартного отклонения вместе со средним позволяет обнаруживать аномалии без сырых данных.**

Рассмотрим два часа с одинаковой средней загрузкой CPU 45%:

- **Час А**: CPU стабильно между 42% и 48%. `avg=45%, stddev=2%`
- **Час В**: CPU колеблется между 5% и 85%. `avg=45%, stddev=28%`

Без stddev эти два часа неразличимы в агрегированных данных. Со stddev час В немедленно идентифицируется как аномальный.

Это позволяет строить алерты на основе исторического stddev:

```
ЕСЛИ текущий_stddev > 3 × средний_stddev_за_30_дней
ТОГДА алерт: обнаружено аномальное поведение
```

Процесс агрегации

Агрегация работает каскадно, управляемая cron-задачей:

Этап 1: Raw → 1 минута

Каждую минуту воркер считывает 6 последних сырых точек для каждой пары (сервер, метрика) и вычисляет:

```
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: подзапрос для последней точки
    (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;
```

Этап 2: 1 минута → 1 час

Каждый час воркер агрегирует 60 минутных точек в одну часовую. Расчёт комбинированного stddev использует формулу пулированной дисперсии:

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

Где `σ_i` — стандартные отклонения подинтервалов, а `μ_i` — их средние. Эта формула математически точна и не требует сырых данных.

Этап 3: 1 час → 1 день

Тот же принцип, раз в сутки 24 часовые точки превращаются в одну суточную.

Этап 4: Очистка старых данных

После каждой агрегации данные за пределами окна хранения удаляются:

```
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: никогда не очищается
```

Маршрутизация запросов

Когда дашборд PmaControl отображает график, он должен выбрать правильное разрешение. Принцип прост: **использовать наиболее грубое разрешение, покрывающее запрашиваемый диапазон**.

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

Результат: годовой график загружает только **365 точек** (суточное разрешение) вместо 3,1 миллиона (10-секундное разрешение). Запрос выполняется за миллисекунды вместо нескольких секунд.

Влияние на производительность запросов

Запрашиваемый диапазон	Разрешение	Загружено точек	Время запроса
1 час	10s (raw)	360	< 10 мс
24 часа	1 мин	1 440	< 20 мс
30 дней	1 час	720	< 15 мс

Запрашиваемый диапазон	Разрешение	Загружено точек	Время запроса
1 год	1 день	365	< 10 мс

Время запросов становится **не зависящим от временного диапазона**. Годовой график так же быстр, как часовой.

Обнаружение аномалий с хранимым stddev

Благодаря предварительно вычисленному stddev PmaControl может обнаруживать аномалии на агрегированных данных без обращения к сырым данным:

1. **Расчёт базовой линии**: среднее и stddev от stddev за последние 30 дней для каждой метрики
2. **Сравнение**: stddev текущего часа сравнивается с базовой линией
3. **Алерт**: если stddev превышает базовую линию в 3 раза — аномальное поведение

Конкретный пример:

- Базовая линия threads_running: `avg_stddev = 2.1, stddev_stddev = 0.8`
- Текущий час: `stddev = 14.3`
- Оценка: `(14.3 - 2.1) / 0.8 = 15.25` сигм — **бесспорная аномалия**

Этот механизм обнаруживает аномалии, которые пропустил бы простой мониторинг среднего: сервер, чей CPU бешено колеблется, но всегда возвращается к нормальному среднему.

Заключение

Мультирезолюционная агрегация — ключ к управлению данными временных рядов в масштабе. Хранение stddev вместе со средним — необычное, но мощное проектное решение: оно сохраняет информацию о вариативности, позволяя обнаруживать аномалии даже на агрегированных данных.

С этой системой PmaControl может мониторить 100+ серверов MariaDB / MySQL на протяжении полного года, гарантируя выполнение запросов дашборда менее чем за 20 миллисекунд.