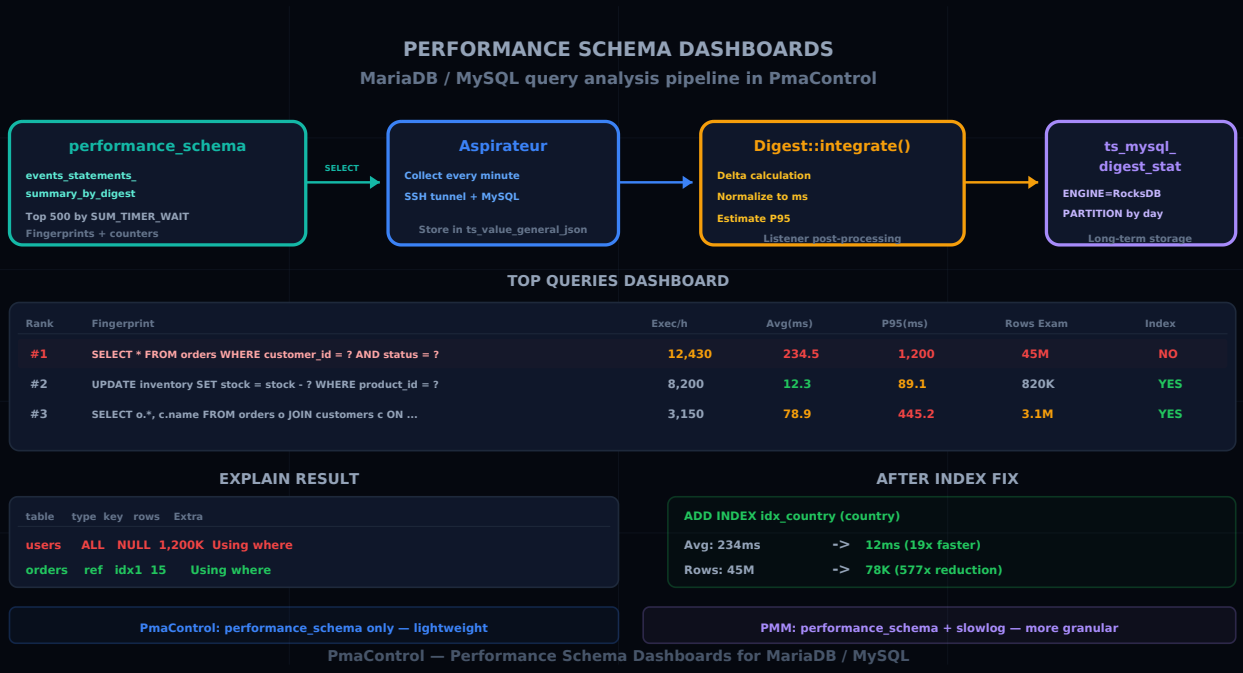
Performance Schema и дашборды PmaControl: отлов медленных запросов

Aurélien LEQUOY · April 13, 2026

MARIADB MYSQL PERFORMANCE-SCHEMA DIAGNOSTICS PMACONTROL



Performance Schema: неиспользованная золотая жила

`performance_schema` включён по умолчанию в MariaDB / MySQL уже много лет. И тем не менее большинство DBA не используют его в повседневной работе. Причина проста: сырые данные сложно читать. Десятки таблиц, миллионы строк, кумулятивные счётчики — без инструмента агрегации это просто шум.

PmaControl превращает этот шум в сигнал. Он собирает данные `performance_schema` через Аспиратор, агрегирует их через Listener (Digest::integrate) и представляет в эксплуатируемых дашбордах. Эта статья описывает полный конвейер, от источника до дашборда.

Проверка включения performance_schema

MariaDB

```
SHOW GLOBAL VARIABLES LIKE 'performance_schema';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| performance_schema | ON    |
+--------------------+-------+
```

Если `OFF`, добавьте в файл конфигурации:

```
[mysqld]
performance_schema = ON
```

Требуется перезапуск — эта переменная не динамическая.

MySQL

Аналогично для MySQL. Переменная доступна только для чтения и требует перезапуска:

```
[mysqld]
performance_schema = ON
```

Влияние на производительность

Классический вопрос: «замедляет ли performance_schema мой сервер?» Ответ в 2026 году — **нет, измеримо**. Накладные расходы составляют порядка 1-3% на типичных нагрузках. Выгода в виде видимости многократно компенсирует эту стоимость.

Единственное исключение: серверы с экстремальными нагрузками (>100 000 запросов/секунду), где каждый процент на счету. В этом случае отключайте ненужные инструменты, а не performance_schema целиком.

Источник: events_statements_summary_by_digest

Ключевая таблица, которую использует PmaControl:

```
SELECT * FROM performance_schema.events_statements_summary_by_digest
ORDER BY SUM_TIMER_WAIT DESC
LIMIT 10\G
```

Эта таблица содержит сводку по **фингерпринту** (нормализованному отпечатку) каждого выполненного запроса. Вот наиболее полезные столбцы:

Столбец	Описание
`DIGEST`	Уникальный хеш фингерпринта
`DIGEST_TEXT`	Нормализованный текст запроса (параметры заменены на `?`)
`COUNT_STAR`	Общее количество выполнений
`SUM_TIMER_WAIT`	Общее время выполнения (в пикосекундах)
`AVG_TIMER_WAIT`	Среднее время на выполнение
`SUM_ROWS_EXAMINED`	Всего просмотренных строк
`SUM_ROWS_SENT`	Всего возвращённых строк
`FIRST_SEEN`	Первое выполнение
`LAST_SEEN`	Последнее выполнение

Фингерпринт — краеугольный камень: он нормализует `SELECT * FROM users WHERE id = 42` и `SELECT * FROM users WHERE id = 1337` в единый фингерпринт `SELECT * FROM users WHERE id = ?`. Это позволяет агрегировать статистику независимо от значений параметров.

Конвейер PmaControl

Этап 1: Сбор Аспиратором

Аспиратор периодически выполняет следующий запрос на каждом контролируемом сервере:

```
SELECT
    DIGEST,
    DIGEST_TEXT,
    COUNT_STAR,
    SUM_TIMER_WAIT,
    AVG_TIMER_WAIT,
    SUM_ROWS_EXAMINED,
    SUM_ROWS_SENT,
```

```
    SUM_NO_INDEX_USED,
    SUM_NO_GOOD_INDEX_USED,
    FIRST_SEEN,
    LAST_SEEN
FROM performance_schema.events_statements_summary_by_digest
WHERE DIGEST IS NOT NULL
ORDER BY SUM_TIMER_WAIT DESC
LIMIT 500;
```

`LIMIT 500` намеренный: PmaControl фокусируется на 500 самых затратных запросах по суммарному времени. Быстрые и редкие запросы не представляют интереса для оптимизации.

Результаты сохраняются в `ts_value_general_json` с временной меткой.

Этап 2: Обработка Listener

Когда Listener обнаруживает новые данные дайджестов, он запускает `Digest::integrate()`. Эта функция:

1. **Вычисляет дельты**: поскольку `performance_schema` предоставляет кумулятивные счётчики (с момента последнего `TRUNCATE` или перезапуска), Digest::integrate вычисляет разницу между двумя сборами для получения метрик за период.
2. **Нормализует время**: пикосекунды конвертируются в миллисекунды для отображения.
3. **Вычисляет перцентили**: P95 (95-й перцентиль) времени выполнения оценивается из распределений.
4. **Сохраняет в ts_mysql_digest_stat**: специализированная таблица для статистики дайджестов, партиционированная по дням и использующая движок RocksDB для сжатия.

```
Аспиратор
  → SELECT FROM performance_schema (every minute)
  → INSERT INTO ts_value_general_json

Listener
  → Detect new data (ts_max_date changed)
  → Digest::integrate()
     → Delta calculation (current - previous)
```

```
     → Normalize to milliseconds
     → Estimate P95
     → INSERT INTO ts_mysql_digest_stat
```

Таблица ts_mysql_digest_stat

Это долгосрочное хранилище статистики дайджестов:

```
CREATE TABLE ts_mysql_digest_stat (
    id BIGINT UNSIGNED AUTO_INCREMENT,
    server_id INT UNSIGNED,
    digest VARCHAR(64),
    digest_text TEXT,
    period_start DATETIME,
    period_end DATETIME,
    exec_count BIGINT UNSIGNED,
    total_time_ms DECIMAL(20,3),
    avg_time_ms DECIMAL(15,3),
    p95_time_ms DECIMAL(15,3),
    rows_examined BIGINT UNSIGNED,
    rows_sent BIGINT UNSIGNED,
    no_index_used BIGINT UNSIGNED,
    PRIMARY KEY (id),
    KEY idx_server_digest (server_id, digest, period_start)
) ENGINE=ROCKSDB
PARTITION BY RANGE (TO_DAYS(period_start)) (
    PARTITION p20260413 VALUES LESS THAN (TO_DAYS('2026-04-14')),
    PARTITION p20260414 VALUES LESS THAN (TO_DAYS('2026-04-15')),
    ...
);
```

Партиционирование по дням позволяет:

- Быструю очистку: `ALTER TABLE ts_mysql_digest_stat DROP PARTITION p20260401;`
- Быстрые запросы по диапазону дат
- Тонкий контроль ретенции

Дашборды

Представление Top Queries

Основной дашборд отображает самые затратные запросы, отсортированные по суммарному времени:

```
Ранг  Фингерпринт                                  Вып/ч    Сред(мс)  P95(мс)  Строк просм
────  ──────────────────────────────────────────── ──────── ──────── ──────── ─────────
  1   SELECT * FROM orders WHERE customer_id      12,430     45.2     234.5   1,245,000
      = ? AND status = ?
  2   UPDATE inventory SET stock = stock - ?       8,200     12.3      89.1     820,000
      WHERE product_id = ?
  3   SELECT o.*, c.name FROM orders o JOIN        3,150     78.9     445.2   3,150,000
      customers c ON o.customer_id = c.id
  4   INSERT INTO audit_log (...)                 45,600      1.2       5.3           0
  5   SELECT COUNT(*) FROM sessions WHERE           980    234.5     890.1  98,000,000
      last_active < ?
```

Каждая строка кликабельна для доступа к детализации.

Детальное представление фингерпринта

При клике на фингерпринт PmaControl отображает:

- **Полный текст** нормализованного запроса
- **Историю**: эволюцию среднего времени выполнения и P95 за последние 30 дней
- **Соотношение** rows_examined / rows_sent — высокое соотношение (>100:1) указывает на сканирование таблицы или отсутствующий индекс
- **Флаг no_index_used** — сколько выполнений не использовали ни одного индекса

Выявление отсутствующих индексов

Соотношение rows_examined / rows_sent — самый мощный индикатор. Рассмотрим пример:

```
Фингерпринт: SELECT * FROM orders WHERE customer_id = ?
Rows examined: 1,245,000 (всего)
Rows sent: 12,430 (всего)
Соотношение: 100:1
```

Соотношение 100:1 означает, что MariaDB / MySQL просматривает 100 строк, чтобы вернуть 1. Это классический признак full table scan или неэффективного индекса.

Действие: проверить наличие индекса на `customer_id`:

```
SHOW INDEX FROM orders WHERE Column_name = 'customer_id';
```

Если индекс не существует:

```
ALTER TABLE orders ADD INDEX idx_customer_id (customer_id);
```

Флаг SUM_NO_INDEX_USED

PmaControl выделяет красным запросы, где `SUM_NO_INDEX_USED` высок. Этот флаг активируется, когда MariaDB / MySQL выполняет full table scan — это часто проблема производительности номер один.

EXPLAIN из PmaControl

Для запросов, определённых как проблемные, PmaControl может выполнить `EXPLAIN` напрямую:

```
EXPLAIN SELECT * FROM orders WHERE customer_id = 42 AND status = 'pending';
```

Результат отображается с цветовой кодировкой:

- **Зелёный**: `type = ref` или `type = eq_ref` — использование индекса, хорошо
- **Жёлтый**: `type = range` — сканирование диапазона, приемлемо
- **Красный**: `type = ALL` — full table scan, нужно исправить

Интеграция с агентом Marina+

Marina+ — агент автоматической оптимизации PmaControl. Он анализирует данные дайджестов и предлагает рекомендации:

1. **Отсутствующие индексы**: обнаруживает запросы с высоким соотношением rows_examined/rows_sent и предлагает индексы для создания
2. **Запросы для переписывания**: выявляет неэффективные паттерны (SELECT *, коррелированные подзапросы, ORDER BY по неиндексированному столбцу)
3. **Конфигурация**: корректирует параметры сервера на основе паттернов запросов (sort_buffer_size, join_buffer_size и т.д.)

Marina+ ничего не изменяет автоматически — он генерирует рекомендации, которые DBA проверяет и применяет.

Сравнение с PMM (Percona Monitoring and Management)

PMM и PmaControl используют один и тот же источник данных (`performance_schema`), но с разными подходами:

Аспект	PmaControl	PMM
Источник	performance_schema	performance_schema + slowlog
Агент	Аспиратор (SSH + MySQL)	mysqld_exporter + QAN
Хранилище	ts_mysql_digest_stat (RocksDB)	ClickHouse (QAN)
Фингерпринтинг	На стороне сервера (нативный MariaDB / MySQL)	На стороне клиента (Percona agent)
P95	Оценён из распределений	Вычислен из slowlog
История	Партиционирована по дням, настраиваемая ретенция	ClickHouse, настраиваемая ретенция
Действия	Встроенный EXPLAIN, предложения Marina+	Query Analytics + PMM UI

Основное различие: **PMM комбинирует performance_schema и slowlog** для более точных перцентилей. PmaControl основывается исключительно на performance_schema, что легче, но менее гранулярно.

Преимущество PmaControl: интеграция с остальной экосистемой (репликация, топология, оповещения, действия). PMM лучше для чистого анализа запросов, PmaControl лучше для глобального видения инфраструктуры.

Практический случай: найти и исправить медленный запрос

Сценарий: дашборд PmaControl показывает запрос, потребляющий 40% общего времени сервера.

Этап 1: Определить

В дашборде Top Queries:

```
#1  SELECT u.*, p.* FROM users u
    JOIN purchases p ON u.id = p.user_id
    WHERE p.created_at > ? AND u.country = ?

    Вып/ч: 5,200   Сред: 234мс   P95: 1,200мс   Строк просм: 45M
```

Этап 2: Проанализировать

Соотношение rows_examined / rows_sent катастрофическое: 45 миллионов просмотренных строк для ~5 200 результатов в час.

EXPLAIN из PmaControl:

```
+----+------+----------+------+------+------+----------+
| id | type | table    | key  | rows | filt | Extra    |
+----+------+----------+------+------+------+----------+
|  1 | ALL  | users    | NULL | 1.2M | 10%  | where    |
|  1 | ref  | purchases| idx1 | 15   | 33%  | where    |
+----+------+----------+------+------+------+----------+
```

Проблема: `users` сканируется полностью (`type = ALL`). Нет индекса по `country` .

Этап 3: Исправить

```
ALTER TABLE users ADD INDEX idx_country (country);
```

Этап 4: Проверить

После добавления индекса дашборд PmaControl показывает улучшение в течение часа:

```
#1  SELECT u.*, p.* FROM users u
    JOIN purchases p ON u.id = p.user_id
    WHERE p.created_at > ? AND u.country = ?

    Вып/ч: 5,200   Сред: 12мс   P95: 45мс   Строк просм: 78K
```

Среднее время снизилось с 234мс до 12мс (x19), а просмотренные строки с 45M до 78K (x577).

Лучшие практики

1. Не делать TRUNCATE performance_schema вручную

PmaControl вычисляет дельты между двумя сборами. Если вы выполните `TRUNCATE TABLE performance_schema.events_statements_summary_by_digest`, счётчики обнулятся и первая дельта будет некорректной. Предоставьте управление PmaControl.

2. Увеличить performance_schema_digests_size при необходимости

По умолчанию MariaDB / MySQL хранит N первых фингерпринтов. Если у вашего приложения больше уникальных запросов, чем лимит, наименее частые вытесняются:

```
[mysqld]
performance_schema_digests_size = 10000  ; по умолчанию ~5000
```

3. Коррелировать с slow query log

PmaControl через performance_schema даёт «что» (какие запросы медленные). Slow query log даёт «когда» (в какой именно момент). Оба дополняют друг друга.

4. Следить за соотношением rows_examined / rows_sent

Это самый практичный индикатор. Соотношение > 100:1 — почти всегда отсутствующий индекс. Соотношение > 1000:1 — срочная проблема.

5. Использовать P95, а не среднее

Среднее скрывает выбросы. Запрос со средним 10мс, но P95 в 500мс имеет прерывистую проблему (lock contention, cold cache, нестабильный план выполнения). P95 выявляет эти проблемы.

Заключение

Performance_schema — лучший источник данных для оптимизации запросов MariaDB / MySQL. PmaControl автоматизирует сбор, агрегацию и представление этих данных через

конвейер Аспиратор → Digest::integrate → ts_mysql_digest_stat → дашборд.

Результат: непрерывная видимость самых затратных запросов, отсутствующих индексов и эволюции производительности во времени. В сочетании с Marina+ для автоматизированных рекомендаций это полный рабочий процесс оптимизации производительности — от обнаружения до исправления.