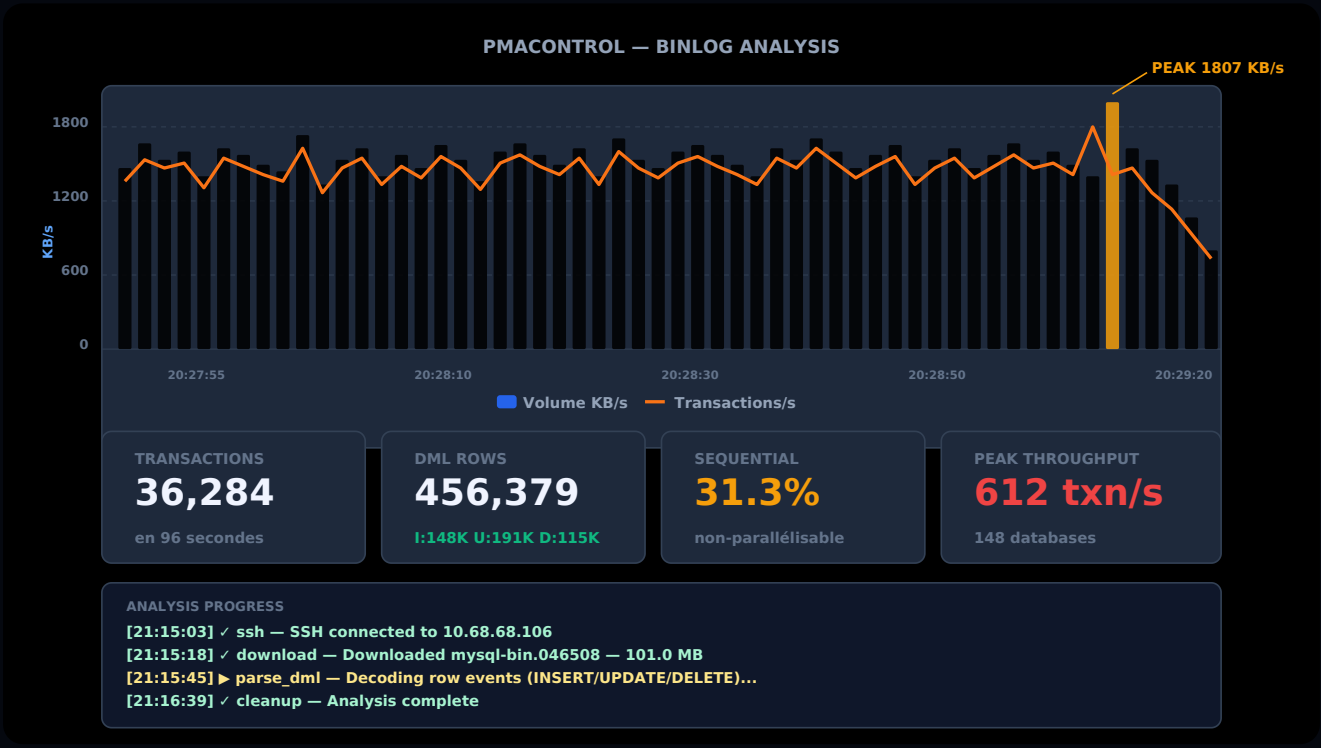
Анализ binlog для диагностики лага репликации

Aurélien LEQUOY · April 13, 2026

PMACONTROL BINLOG REPLICATION MYSQL MARIADB PERCONA-SERVER PERFORMANCE LAG MYSQLBINLOG



Лаг репликации — тихий бич

Каждый DBA переживал этот момент: мониторинг показывает 300 секунд задержки на реплике, сыплются оповещения, и вопрос всегда один и тот же — **почему?**

`Seconds_Behind_Master` (или `Seconds_Behind_Source` в MySQL 8) — это бинарный индикатор: есть лаг или нет. Он ничего не говорит о **причине**. Это массовый пакетный DELETE? Недостаток параллелизма? Транзакция размером 800 КБ, блокирующая SQL-поток? Чтобы узнать, нужно анализировать binlog мастера — а это требует SSH-доступа, нужного бинарника `mysqlbinlog`, времени и grep.

PmaControl теперь автоматизирует всё это.

Новая функция: Binlog Analysis

Принцип

Со страницы **slave/show** любой реплики теперь можно:

1. **Выбрать временной диапазон** прямо на графике лага (клик-перетаскивание мышью) или через поля datetime
2. **Запустить анализ** одним кликом — всё выполняется в фоновом режиме
3. **Следить за прогрессом в реальном времени** шаг за шагом
4. **Просмотреть отчёт** с интерактивным графиком и детальными метриками
5. **Получить уведомление в Telegram** с кратким резюме

Что происходит за кулисами

При запуске анализа PmaControl выполняет конвейер из 15 фаз:

```
[21:15:02] ✓ init — Analysis #42 — slave=85 master=106 range=[20:27:51 → 20:29:27]
[21:15:02] ✓ master_info — Master: prod-db-01 — 10.68.68.106:3306
[21:15:02] ✓ version — Version: 8.0.44
[21:15:02] ✓ binary — Using: mysqlbinlog-8.0 — Ver 8.0.42
[21:15:03] ✓ credentials — MySQL user: pmacontrol@10.68.68.106:3306
[21:15:04] ✓ find_binlogs — Found 1 binlog file(s): mysql-bin.046508
[21:15:18] ✓ fetch — Fetched mysql-bin.046508 — 101.0 MB (via --read-from-remote-server,
cached)
[21:15:19] ✓ file_ranges — 1 file(s) probed
[21:15:45] ✓ parse_gtid — Parsed 36284 transactions, total 97.2 MB, 123 >100KB, 3 >500KB
[21:16:12] ✓ parse_dml — I:148,659 U:191,867 D:115,853 = 456,379 rows across 148 databases
[21:16:38] ✓ parse_volume — 97 seconds of data, peak 612 txn/s, peak 1807 KB/s
[21:16:39] ✓ parse_ddl — No DDL — 100% DML row-based
[21:16:39] ✓ metadata — 1 metadata file(s) written
[21:16:39] ✓ store — Report stored successfully
[21:16:39] ✓ cleanup — Cleanup done — analysis complete
```

Каждый этап отображается в реальном времени в интерфейсе с детализацией происходящего. Больше не нужно гадать — идёт ли передача или парсинг завис.

Примечание: получение использует протокол MySQL (`--read-from-remote-server`), а не SSH — никаких ключей разворачивать на мастере не нужно.

Правильный бинарник для правильной версии

Одна из классических ловушек: использование `mysqlbinlog` от MariaDB для чтения binlog MySQL 8, или наоборот. События помечаются как "Ignorable", и row-based данные становятся нечитаемыми.

PmaControl содержит **8 статических бинарников mysqlbinlog** (x86_64 и aarch64):

Бинарник	Поддерживаемые версии
`mysqlbinlog-5.6`	MySQL 5.5, 5.6, Percona 5.6
`mysqlbinlog-5.7`	MySQL 5.7, Percona 5.7
`mysqlbinlog-8.0`	MySQL 8.0
`mysqlbinlog-8.4`	MySQL 8.4 LTS
`mysqlbinlog-9.7`	MySQL 9.7 LTS
`mysqlbinlog-mariadb`	MariaDB 10.x, 11.x, 12.x (общий fallback)

Выбор автоматический: PmaControl считывает переменную `version` мастера из своих метрик и выбирает наиболее совместимый бинарник (см. `MysqlbinlogBinaryResolver`). Для MySQL Oracle стратегия — выбрать **минимальную версию ≥ версии мастера** (совместимость вперёд надёжнее обратной). Для MariaDB резолвер также поддерживает более точные бинарники с именами `mysqlbinlog-mariadb-10.11`, `mysqlbinlog-mariadb-11.8` и т.д. — разворачиваются по требованию, когда хост экспонирует формат binlog, специфичный для его минорной версии.

Отчёт анализа

График объёма посекундно

Главный график отображает два наложенных ряда:

- **Синие столбцы**: объём в КБ/с (сумма `transaction_length` за секунду)
- **Оранжевая линия**: количество транзакций в секунду

Это первый рефлекс DBA: лаг вызван разовым всплеском или устойчивой нагрузкой? График отвечает мгновенно.

Метрики параллелизма MTS

Здесь анализ становится по-настоящему полезным. PmaControl парсит поля `last_committed` и `sequence_number` каждого GTID-события для расчёта:

- **% последовательных транзакций** — те, где `sequence_number = last_committed + 1`, которые не могут быть параллелизированы Multi-Threaded Slave
- **Максимальный размер групп параллелизма** — сколько транзакций реально могут выполняться параллельно
- **Распределение групп** — сколько групп из 1, 2, 3... транзакций

Конкретный пример: если 31% транзакций последовательные, а максимум параллелизма — 7, то даже с 16 `replica_parallel_workers` большинство будут простаивать. Рекомендация ясна: включить `binlog_transaction_dependency_tracking = WRITESET` на мастере.

Обнаружение крупных транзакций

Объёмные транзакции — убийца репликации номер один. Одна транзакция в 834 КБ, затрагивающая 6171 строку, блокирует SQL applier thread на всё время её выполнения — остальные транзакции ждут в очереди.

PmaControl считает:

- Транзакции > 100 КБ
- Транзакции > 500 КБ
- Самую крупную транзакцию (с её содержимым: какие таблицы, сколько строк)

Топ таблиц

Отчёт перечисляет 30 наиболее изменяемых таблиц с детализацией INSERT/UPDATE/DELETE. Именно здесь часто обнаруживается токсичный паттерн: пакетные операции `DELETE FROM table WHERE ...; INSERT INTO table ...` для «обновления» данных вместо использования `INSERT ... ON DUPLICATE KEY UPDATE` или более мелких транзакций.

Автоматические рекомендации

На основе метрик PmaControl генерирует конкретные рекомендации:

- "Sequential transaction ratio is 31.3%. Consider `binlog_transaction_dependency_tracking = WRITESET`."

- "3 transaction(s) > 500 KB. Large transactions block the SQL applier thread."
- "148 databases modified. Multi-tenant pattern may cause replication hot-spots."

Это не общие советы — они рассчитаны на основе реальных данных ваших binlog.

Уведомление в Telegram

Когда анализ завершён, резюме автоматически отправляется через Telegram:

```
Binlog Analysis Complete
Slave: replica-prod-01
Master: master-prod-01 (8.0.44)
Range: 2026-04-13 20:27:51 → 2026-04-13 20:29:27
──────────────────────
Size: 101.0 MB | Txn: 36,284 | Duration: 96s
DML: I:148,659 U:191,867 D:115,853 = 456,379 rows
Peak: 612 txn/s | Avg: 378.0 txn/s
Sequential: 31.3% | Max parallel: 7
Large txn: 123 >100K, 3 >500K (max 815 KB)
Databases: 148
──────────────────────
• High write throughput (peak 612 txn/s). Ensure replica_parallel_workers >= 8.
• Sequential transaction ratio is 31.3%. Consider WRITESET.
• 3 transaction(s) > 500 KB. Large transactions block the SQL applier thread.
```

DBA на дежурстве получает всё необходимое для действий прямо в кармане.

Техническая архитектура

Получение binlog: как IO thread

PmaControl не использует SSH для получения binlog. Он использует `mysqlbinlog --read-from-remote-server`, который подключается к мастеру по протоколу MySQL, точно так же, как IO thread реплики. Учётных данных MySQL от PmaControl достаточно — SSH-ключ на мастере не нужен.

```
mysqlbinlog --read-from-remote-server \
  --host=10.105.1.11 --port=3306 \
  --user=pmacontrol --password=*** \
```

```
    --raw --result-file=/tmp/analysis/ \
    mysql-bin.1054495
```

Это означает, что если PmaControl может подключиться к MySQL мастера (что он уже делает для мониторинга), он может получить binlog. Никакой дополнительной настройки.

Интеллектуальное обнаружение файлов binlog

Мастер в продакшне может иметь **тысячи** файлов binlog (мы встречали 2712 файлов при разработке). Сканирование каждого файла для поиска временного диапазона было бы слишком затратным.

Стратегия в 2 этапа:

1. **Привязка через позицию слейва** — считываем `Master_Log_File` из `SHOW SLAVE STATUS`, чтобы узнать текущий binlog. Также считываем `SHOW MASTER STATUS` на мастере для учёта лага. Это даёт узкое окно вместо сканирования всех 2712 файлов.
2. **Бинарный поиск** — в этом окне мы проверяем первую метку времени каждого файла через `mysqlbinlog --stop-position=8192` (читает только заголовок FORMAT_DESCRIPTION + первое событие). С бинарным поиском нахождение нужного файла среди 100 кандидатов требует всего 7 проб вместо 100.

Результат: обнаружение binlog на мастере с 2712 файлами занимает **2 секунды** вместо нескольких минут.

Постоянный кеш + JSON sidecar для ИИ

Скачанные binlog хранятся в `data/binlog_analysis/<id_мастера>/` с **TTL 30 дней**. Прямое следствие: повторный анализ того же временного диапазона (или соседнего диапазона, попадающего на те же файлы) бесплатен по сети и времени — вы сразу получаете `Cache hit: mysql-bin.046508`.

Для каждого закешированного binlog PmaControl также записывает sidecar `mysql-bin.046508.meta.json`, содержащий структурированную сводку: общее количество транзакций, параллелизм MTS, топ таблиц DML, DDL, пик txn/s. Этот JSON предназначен для потребления LLM-агентом (см. страницу ИИ-агенты сайта), которому нужно рассуждать об инциденте без повторного парсинга binlog.

8 статических бинарников и почему

PmaControl содержит предкомпилированные бинарники `mysqlbinlog` для каждой мажорной версии. Этот выбор обусловлен техническим наблюдением: **несовместимый `mysqlbinlog` не выдаёт ошибку, а выдаёт бесшумно некорректные результаты**.

Реальный пример: `mysqlbinlog` от MariaDB, читающий binlog MySQL 8.0, помечает row-based события как "Ignorable" и пропускает их. Счётчик транзакций падает с 36284 до 0. Никакой ошибки не отображается.

Бинарники извлечены из официальных архивов:

- MySQL 5.6 и 5.7: с `dev.mysql.com/get/Downloads/MySQL-5.x/`
- MySQL 8.0 и 8.4: из минимальных glibc2.17
- MySQL 9.2: релиз Innovation
- MariaDB: `mariadb-binlog` из системного дистрибутива

Баг mysqlbinlog 5.6/5.7: `--raw` игнорирует `--stop-datetime`

Ловушка, обнаруженная при разработке: в режиме `--raw --read-from-remote-server` версии 5.6 и 5.7 `mysqlbinlog` **бесшумно игнорируют** опции `--start-datetime` и `--stop-datetime`. Вместо остановки на запрошенной дате процесс ведёт себя как IO thread и **ожидает бесконечно** новые события.

Решение в PmaControl: использовать `--raw` без временного фильтра (скачивает файл целиком, затем останавливается) и применять фильтрацию по временному диапазону только при локальном парсинге. Для безопасности добавлен таймаут 120 секунд на файл.

Асинхронный конвейер с прогрессом в реальном времени

Анализ выполняется в фоновом режиме через CLI Glial (`php App/Webroot/index.php slave runBinlogAnalysisCli <id>`). Фронтенд опрашивает статус каждые 2 секунды и отображает прогресс в стилизованном терминале. Поле `progress` в JSON хранит каждый этап с меткой времени, фазой, сообщением и статусом — 15 фаз от инициализации до очистки.

Хранение результатов

Всё сохраняется в таблице `binlog_analysis` :

- Объём посекундно (JSON)

- Топ таблиц (JSON)
- Распределение параллелизма (JSON)
- Рекомендации (JSON)

Результаты доступны в любое время из истории анализов.

Практические примеры использования

«Лаг реплики в 3 часа ночи»

Мониторинг показывает пик лага в 3 часа. На следующее утро DBA выбирает диапазон 02:55-03:10 на графике и запускает анализ. Результат: пакетное обновление `port_flux`, которое делает DELETE + INSERT 15000 строк одной транзакцией в 148 базах. Решение: разбить на транзакции по 500 строк.

«Почему реплика не догоняет?»

Лаг постепенно растёт, несмотря на 8 parallel workers. Анализ показывает 35% последовательных транзакций и максимум параллелизма 5. Решение: включить WRITESET на мастере и перейти на 16 workers.

«Каково влияние этого пакета?»

Перед запуском пакетной миграции в продакшне можно проанализировать аналогичный binlog из среды тестирования, чтобы оценить влияние на репликацию.

Как использовать

1. Откройте PmaControl → Slave → Show на вашей реплике
2. Кликните-перетащите на графике лага для выбора диапазона
3. Нажмите "Analyze Binlogs"
4. Наблюдайте за этапами в реальном времени
5. Просмотрите отчёт, проверьте рекомендации
6. Получите резюме в Telegram

Вот и всё. Никакого ручного SSH, никакого `mysqlbinlog | grep | awk | sort`, никаких скриптов для поддержки. Диагностика лага репликации в один клик.