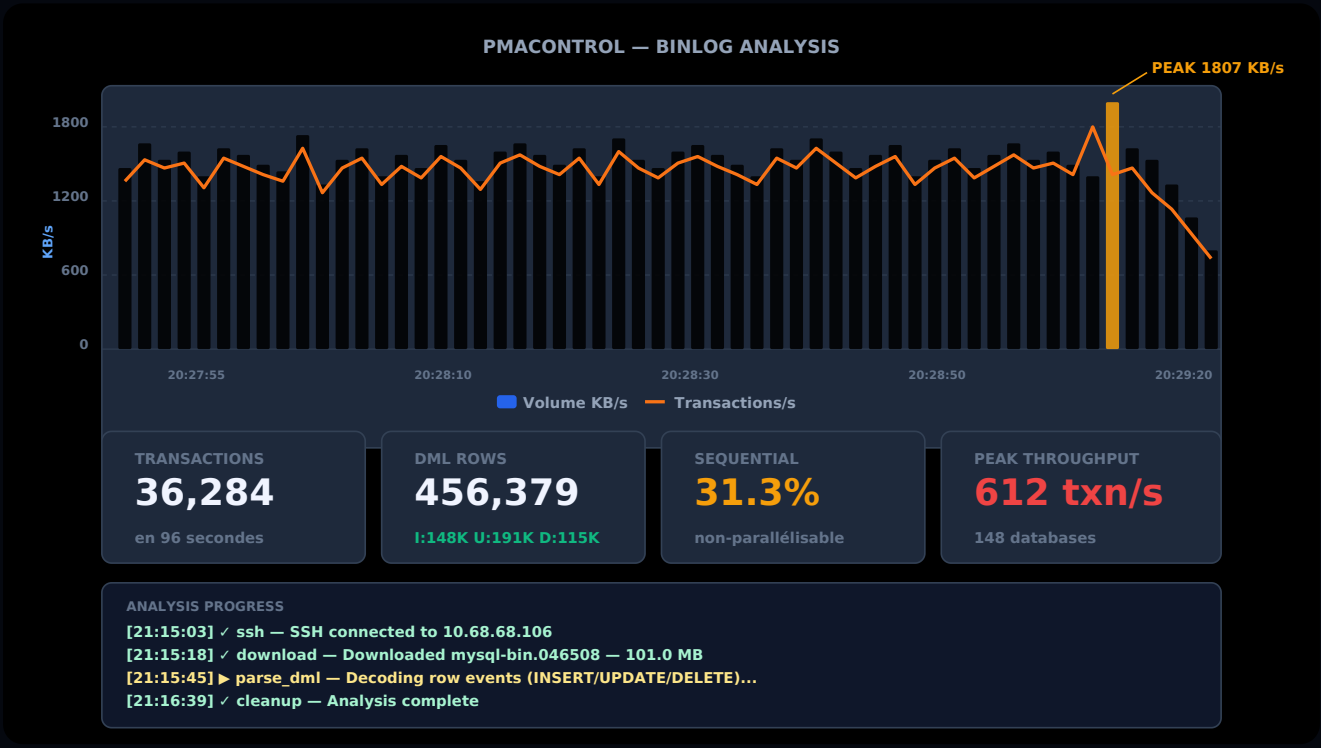
Analiza binlogów w celu zrozumienia opóźnień replikacji

Aurélien LEQUOY · April 13, 2026

PMACONTROL BINLOG REPLICATION MYSQL MARIADB PERCONA-SERVER PERFORMANCE LAG MYSQLBINLOG



Opóźnienie replikacji — cicha plaga

Każdy DBA przeżył ten moment: monitoring pokazuje 300 sekund opóźnienia na replice, alerty sypią się jeden za drugim, a pytanie jest zawsze to samo — **dlaczego?**

`Seconds_Behind_Master` (lub `Seconds_Behind_Source` w MySQL 8) to wskaźnik binarny: jest opóźnienie albo nie ma. Nie mówi nic o **przyczynie**. Czy to masowy batch DELETE? Brak paralelizmu? Transakcja 800 KB blokująca wątek SQL? Żeby się dowiedzieć, trzeba zdekompilować binlogi mastera — co wymaga dostępu SSH, odpowiedniego pliku binarnego `mysqlbinlog`, czasu i grepa.

PmaControl automatyzuje teraz to wszystko.

Nowa funkcja: Binlog Analysis

Zasada działania

Z poziomu strony **slave/show** dowolnej repliki możesz teraz:

1. **Wybrać zakres czasowy** bezpośrednio na wykresie opóźnień (kliknięcie i przeciągnięcie myszą) lub przez pola datetime
2. **Uruchomić analizę** jednym kliknięciem — wszystko dzieje się w tle
3. **Śledzić postęp w czasie rzeczywistym** krok po kroku
4. **Przeglądać raport** z interaktywnym wykresem i szczegółowymi metrykami
5. **Otrzymać powiadomienie Telegram** z podsumowaniem

Co dzieje się za kulisami

Gdy uruchamiasz analizę, PmaControl wykonuje pipeline w 15 fazach:

```
[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
```

Każdy etap jest widoczny w czasie rzeczywistym w interfejsie, ze szczegółami tego, co się dzieje. Nie trzeba już zgadywać, czy transfer jest w trakcie, czy parsowanie się zacięło. Uwaga: pobieranie używa protokołu MySQL (`--read-from-remote-server`), nie SSH — żadnego klucza do wdrażania na masterze.

Właściwy plik binarny dla właściwej wersji

Jedna z klasycznych pułapek: użycie `mysqlbinlog` MariaDB do odczytu binlogu MySQL 8 lub odwrotnie. Zdarzenia są oznaczane jako "Ignorable", a dane row-based stają się nieczytelne.

PmaControl zawiera **8 statycznych plików binarnych mysqlbinlog** (x86_64 i aarch64):

Plik binarny	Obsługiwane wersje
`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 (ogólny fallback)

Wybór jest automatyczny: PmaControl odczytuje zmienną `version` mastera ze swoich metryk i wybiera najbliższy kompatybilny plik binarny (zob. `MysqlbinlogBinaryResolver`). Dla MySQL Oracle strategia polega na wyborze **najniższej wersji ≥ wersji mastera** (kompatybilność w przód jest pewniejsza niż wstecz). Dla MariaDB resolver obsługuje również bardziej szczegółowe binaria nazwane `mysqlbinlog-mariadb-10.11`, `mysqlbinlog-mariadb-11.8` itd. — wdrażane na żądanie, gdy host eksponuje format binlogu specyficzny dla swojej wersji minor.

Raport analizy

Wykres wolumenu sekundę po sekundzie

Główny wykres wyświetla dwie nałożone serie:

- **Niebieskie słupki**: wolumen w KB/s (suma `transaction_length` na sekundę)
- **Pomarańczowa linia**: liczba transakcji na sekundę

To pierwszy odruch DBA: czy opóźnienie pochodzi od punktowego piku, czy od utrzymywanego przepływu? Wykres odpowiada natychmiast.

Metryki paralelizmu MTS

Tu analiza staje się naprawdę przydatna. PmaControl parsuje pola `last_committed` i `sequence_number` każdego zdarzenia GTID, by obliczyć:

- **% transakcji sekwencyjnych** — te, gdzie `sequence_number = last_committed + 1`, których Multi-Threaded Slave nie może zrównoleglić
- **Maksymalny rozmiar grup paralelizmu** — ile transakcji może rzeczywiście wykonywać się równolegle
- **Rozkład grup** — ile grup po 1, 2, 3... transakcje

Konkretny przykład: jeśli 31% transakcji jest sekwencyjnych, a maksimum paralelizmu wynosi 7, to nawet przy 16 `replica_parallel_workers` większość będzie bezczynna. Rekomendacja jest jasna: włączyć `binlog_transaction_dependency_tracking = WRITESET` na masterze.

Wykrywanie dużych transakcji

Duże transakcje to zabójca replikacji nr 1. Pojedyncza transakcja 834 KB dotykająca 6171 wierszy blokuje wątek SQL applier na cały czas trwania — inne transakcje czekają za nią.

PmaControl liczy:

- Transakcje > 100 KB
- Transakcje > 500 KB
- Największą transakcję (z jej zawartością: które tabele, ile wierszy)

Top tabele

Raport wymienia 30 najczęściej modyfikowanych tabel ze szczegółami INSERT/UPDATE/DELETE. To często tutaj znajduje się toksyczny wzorzec: batche wykonujące `DELETE FROM table WHERE ...; INSERT INTO table ...` do "odświeżania" danych zamiast używania `INSERT ... ON DUPLICATE KEY UPDATE` lub mniejszych transakcji.

Automatyczne rekomendacje

Na podstawie metryk PmaControl generuje konkretne rekomendacje:

- "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."

To nie są ogólne rady — są obliczane na podstawie rzeczywistych danych z twoich binlogów.

Powiadomienie Telegram

Gdy analiza jest zakończona, podsumowanie jest automatycznie wysyłane przez 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 dyżurny ma wszystko, czego potrzebuje do działania, bezpośrednio w kieszeni.

Architektura techniczna

Pobieranie binlogów: jak wątek IO

PmaControl nie używa SSH do pobierania binlogów. Używa `mysqlbinlog --read-from-remote-server`, który łączy się z masterem przez protokół MySQL, dokładnie jak wątek IO repliki. Dane uwierzytelniające MySQL PmaControl wystarczają — nie potrzeba klucza SSH na masterze.

```
mysqlbinlog --read-from-remote-server \
  --host=10.105.1.11 --port=3306 \
  --user=pmacontrol --password=*** \
  --raw --result-file=/tmp/analysis/ \
  mysql-bin.1054495
```

Oznacza to, że jeśli PmaControl może połączyć się z MySQL mastera (co już robi do monitoringu), może pobrać binlogi. Żadna dodatkowa konfiguracja.

Inteligentne odkrywanie plików binlog

Master produkcyjny może mieć **tysiące** plików binlog (napotkaliśmy 2712 plików podczas rozwoju). Skanowanie każdego pliku w poszukiwaniu zakresu czasowego byłoby zbyt kosztowne.

Strategia w 2 etapach:

1. **Zakotwiczenie przez pozycję slave'a** — odczytujemy `Master_Log_File` z `SHOW SLAVE STATUS`, by poznać bieżący binlog. Odczytujemy też `SHOW MASTER STATUS` na masterze, by uwzględnić opóźnienie. Daje to wąskie okno zamiast skanowania wszystkich 2712 plików.
2. **Wyszukiwanie binarne** — w tym oknie badamy pierwszy znacznik czasu każdego pliku przez `mysqlbinlog --stop-position=8192` (odczytuje tylko nagłówek FORMAT_DESCRIPTION + pierwsze zdarzenie). Z wyszukiwaniem binarnym znalezienie właściwego pliku wśród 100 kandydatów wymaga tylko 7 prób zamiast 100.

Rezultat: odkrywanie binlogów na masterze z 2712 plikami trwa **2 sekundy** zamiast kilku minut.

Persystentny cache + sidecar JSON dla AI

Pobrane binlogi są przechowywane w `data/binlog_analysis/<id_mastera>/` z **TTL 30 dni**. Bezpośrednia konsekwencja: ponowna analiza tego samego zakresu czasowego (lub przyległego zakresu trafiającego w te same pliki) jest darmowa pod względem sieci i czasu — natychmiast widzisz `Cache hit: mysql-bin.046508`.

Dla każdego buforowanego binlogu PmaControl zapisuje też sidecar `mysql-bin.046508.meta.json` zawierający ustrukturyzowane podsumowanie: łączna liczba transakcji, paralelizm MTS, top tabele DML, DDL, szczyt txn/s. Ten JSON jest zaprojektowany do konsumowania przez agenta LLM (zob. stronę Agenty AI serwisu), który musi rozumować o incydencie bez ponownego parsowania binlogów.

8 statycznych plików binarnych i dlaczego

PmaControl zawiera wstępnie skompilowane pliki binarne `mysqlbinlog` dla każdej głównej wersji. Ten wybór wynika z technicznego spostrzeżenia: **niekompatybilny `mysqlbinlog` nie produkuje błędu — produkuje cicho fałszywe wyniki**.

Doświadczony przykład: `mysqlbinlog` MariaDB odczytujący binlog MySQL 8.0 oznacza zdarzenia row-based jako "Ignorable" i je pomija. Liczba transakcji spada z 36284 do 0. Żaden błąd nie jest wyświetlany.

Bug mysqlbinlog 5.6/5.7: `--raw` ignoruje `--stop-datetime`

Pułapka odkryta podczas rozwoju: w trybie `--raw --read-from-remote-server` wersje 5.6 i 5.7 `mysqlbinlog` **cicho ignorują** opcje `--start-datetime` i `--stop-datetime`. Zamiast zatrzymać się na żądanej dacie, proces zachowuje się jak wątek IO i **czeka w nieskończoność** na nowe zdarzenia.

Rozwiązanie w PmaControl: używanie `--raw` bez filtra czasowego (pobiera cały plik, potem się zatrzymuje) i stosowanie filtrowania po zakresie czasowym tylko podczas lokalnego parsowania. Timeout 120 sekund na plik jest dodany dla bezpieczeństwa.

Asynchroniczny pipeline z postępem w czasie rzeczywistym

Analiza działa w tle przez CLI Glial (`php App/Webroot/index.php slave runBinlogAnalysisCli <id>`). Frontend odpytuje status co 2 sekundy i wyświetla postęp w stylizowanym terminalu. Pole `progress` w JSON przechowuje każdy etap ze znacznikiem czasu, fazą, komunikatem i statusem — 15 faz od inicjalizacji do czyszczenia.

Przechowywanie wyników

Wszystko jest persystowane w tabeli `binlog_analysis` :

- Wolumen na sekundę (JSON)
- Top tabele (JSON)
- Rozkład paralelizmu (JSON)
- Rekomendacje (JSON)

Wyniki są dostępne w każdej chwili z historii analiz.

Przypadki użycia

"Opóźnienie repliki o 3 rano"

Monitoring pokazuje pik opóźnienia o 3:00. Rano DBA wybiera zakres 02:55-03:10 na wykresie i uruchamia analizę. Wynik: batch odświeżania `port_flux` wykonujący DELETE + INSERT 15000 wierszy w jednej transakcji na 148 bazach. Rozwiązanie: podzielić na transakcje po 500 wierszy.

"Dlaczego replika nie nadrabia?"

Opóźnienie rośnie stopniowo mimo 8 parallel workers. Analiza pokazuje 35% transakcji sekwencyjnych i maksimum paralelizmu 5. Rozwiązanie: włączyć WRITESET na masterze i przejść na 16 workerów.

"Jaki jest wpływ tego batcha?"

Przed uruchomieniem batcha migracji na produkcji można przeanalizować podobny binlog ze środowiska staging, by oszacować wpływ na replikację.

Jak tego używać

1. Otwórz PmaControl → Slave → Show na swojej replice
2. Kliknij i przeciągnij na wykresie opóźnień, by wybrać zakres
3. Kliknij "Analyze Binlogs"
4. Obserwuj etapy pojawiające się w czasie rzeczywistym
5. Przeglądaj raport, sprawdź rekomendacje
6. Odbierz podsumowanie na Telegram

To wszystko. Bez ręcznego SSH, bez `mysqlbinlog | grep | awk | sort`, bez skryptów do utrzymywania. Diagnoza opóźnień replikacji jednym kliknięciem.