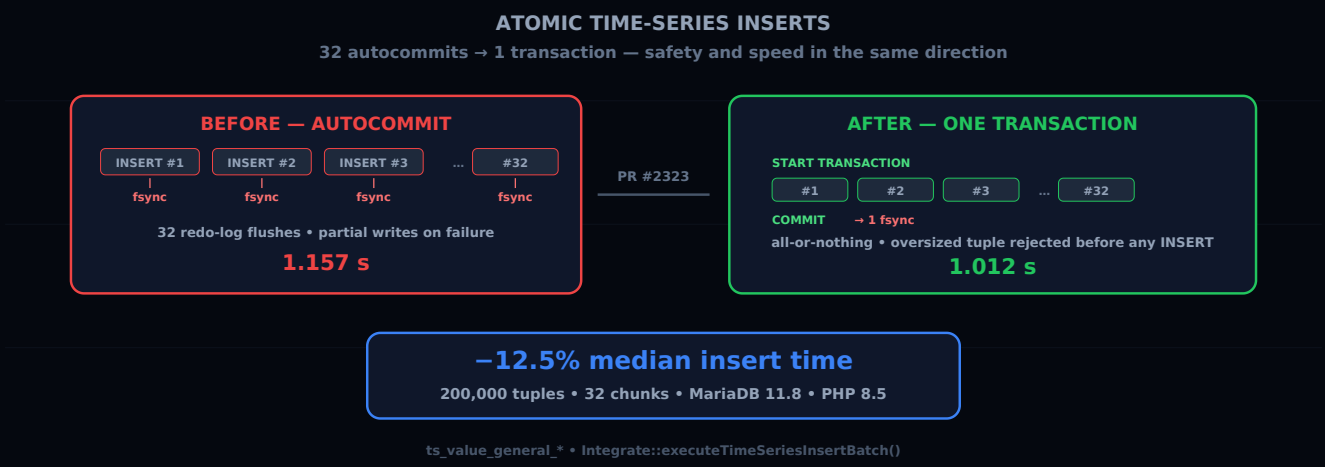
Gdy poprawka atomowości przyspiesza ingestię o 12%

Aurélien LEQUOY · July 25, 2026

MARIADB MYSQL INNODB TIME-SERIES PERFORMANCE TRANSACTIONS PMACONTROL



Kontekst: najgorętsza ścieżka zapisu w PmaControl

Co kilka sekund każdy agent PmaControl zbiera setki metryk z Twoich serwerów MariaDB / MySQL: zmienne statusu, liczniki InnoDB, stan replikacji, digesty zapytań. Wszystko to trafia do `Integrate::insert_value()`, które zapisuje pomiary do tabel `ts_value_general_*` — najbardziej obciążonej ścieżki zapisu w całej aplikacji.

Aby nigdy nie przekroczyć `max_allowed_packet`, krotki są grupowane w **chunki**: budżet SQL jest wyliczany dynamicznie na podstawie serwera (z marginesem bezpieczeństwa), a partia 200 000 pomiarów staje się np. 32 zapytaniami `INSERT INTO ... VALUES (...), (...), ...` po około 256 KiB każde.

Dotychczas te 32 zapytania wykonywały się w trybie **autocommit**: 32 INSERT-y, 32 niejawne COMMIT-y.

Błąd: ciche zapisy częściowe

Co się dzieje, gdy chunk numer 17 zawiedzie — pełna tabela, deadlock, zerwane połączenie?

Przed poprawką: chunki od 1 do 16 pozostawały zapisane, chunki od 17 do 32 przepadały, a dalsza księgowość (powiązanie serwer/zmienna, checkpoint ingestii) mogła wykonać się tak, jakby wszystko się powiodło. Logiczna partia zamieniała się w cichy zapis częściowy — najgorszy możliwy scenariusz dla danych monitoringu: wykresy z dziurami, których nic nie zgłasza.

Trzy odrębne zgłoszenia sprowadzały się do tej samej przyczyny:

- **#1467 / #1471** — zapisy częściowe przy awarii w środku partii;
- **#1468 / #1472** — pojedyncza krotka większa niż budżet i tak była wysyłana na serwer, gwarantując błąd `max_allowed_packet`;
- **#1469** — cichy fallback mógł wyłączyć dynamiczne wykrywanie budżetu.

Poprawka: jedna partia = jedna transakcja

Poprawka (PR #2323) wprowadza `executeTimeSeriesInsertBatch()`: wszystkie chunki danego typu metryki są teraz opakowane w **jedną transakcję**:

```
START TRANSACTION;
INSERT INTO ts_value_general_int (...) VALUES (...), (...), ...;  -- chunk 1
INSERT INTO ts_value_general_int (...) VALUES (...), (...), ...;  -- chunk 2
-- ... 30 kolejnych chunków ...
COMMIT;
```

Każde niepowodzenie — wynik falsy, niełagodne ostrzeżenie, wyjątek — wywołuje `ROLLBACK` i zgłasza typowany wyjątek z bezpiecznymi metadanymi diagnostycznymi (tabela, chunk, szacowane bajty, budżet). Plik pomiarów jest zachowywany do ponowienia: albo cała partia zostaje utrwalona, albo nic.

Dodatkowo pojedyncza krotka, której szacowany rozmiar już przekracza budżet, jest wykrywana **zanim** transakcja w ogóle się otworzy: żaden skazany na porażkę INSERT nie jest wysyłany na serwer.

Pytanie za 12%: ile to kosztuje?

Transakcja obejmująca 32 zapytania to dodatkowa księgowość po stronie InnoDB: dłuższy undo log, dłużej trzymane blokady. Spodziewaliśmy się niewielkiego narzutu, który uznaliśmy za w pełni wart gwarancji atomowości.

Zrobiliśmy benchmark przed merge'em. Protokół:

- **dokładne** odtworzenie gorącej pętli `insert_value()` (chunkowanie → budowa SQL → wykonanie), a nie syntetyczny mikro-benchmark;
- 200 000 deterministycznych krotek do `ts_value_general_int` (unikalne klucze główne), budżet 256 KiB → 32 chunki;
- MariaDB 11.8, PHP 8.5.8, tabela InnoDB, `TRUNCATE` między przebiegami, 1 rozgrzewka + 5 mierzonych przebiegów;
- ten sam kontener LXC, te same dane, między pomiarami zmienia się tylko kod aplikacji.

Wyniki:

Przebieg	Przed (autocommit ×32)	Po (1 transakcja)
1	1,127 s	0,927 s
2	1,199 s	0,947 s
3	1,155 s	1,012 s
4	1,168 s	1,019 s
5	1,157 s	1,026 s
Mediana	**1,157 s**	**1,012 s**

Mediana szybsza o 12,5%. Poprawka atomowości nic nie kosztuje: oszczędza czas. Przepustowość ingestii rośnie z około 173 000 do 198 000 pomiarów na sekundę.

Dlaczego jest szybciej: ukryta cena COMMIT-a

Odpowiedź tkwi w tym, co `COMMIT` naprawdę robi w InnoDB.

Przy każdym zatwierdzeniu InnoDB musi uczynić transakcję **trwałą**: zapisać strony redo loga i — przy `innodb_flush_log_at_trx_commit = 1`, wartości domyślnej i jedynej naprawdę bezpiecznej — wymusić `fsync()` pliku loga. Ten flush to najdroższa operacja w cyklu życia transakcji: fizycznie czeka na nośnik.

W trybie autocommit każdy `INSERT` jest osobną transakcją:

- **przed**: 32 chunki = 32 COMMIT-y = 32 flushe redo loga;

- **po**: 32 chunki = 1 COMMIT = 1 flush redo loga.

Zaoszczędzone 31 synchronizacji waży znacznie więcej niż nieco dłuższy undo log. To ten sam mechanizm, dzięki któremu import SQL jest dramatycznie szybszy w transakcji, albo dzięki któremu *group commit* w MariaDB amortyzuje flushe między równoległymi transakcjami — tutaj zastosowany wewnątrz jednej logicznej partii.

Zysk zależy od sprzętu: na nośniku, gdzie `fsync()` jest bardzo wolny (dyski talerzowe, wirtualizacja bez bezpiecznego cache'u zapisu), przewaga rośnie. Na szybkim NVMe maleje. Ale znak się nie zmienia: pojedyncza transakcja jest zawsze co najmniej równie szybka.

Wnioski

- **Atomowość to nie luksus, za który się płaci — to często optymalizacja.** Grupowanie powiązanych zapisów w jedną transakcję eliminuje flushe redo loga — bezpieczeństwo i wydajność idą w tę samą stronę.
- **Benchmarkuj prawdziwą ścieżkę kodu.** To odtworzenie realnej gorącej pętli na prawdziwej bazie zamieniło „akceptujemy narzut" w „narzutu nie ma".
- **Awaria musi być głośna i całkowita.** Partia metryk zapisana w połowie jest gorsza niż partia ponowiona: od tej poprawki PmaControl gwarantuje wszystko-albo-nic przy ingestii time-series.

Poprawka jest dostępna na gałęzi `master` PmaControl od 25 lipca 2026.