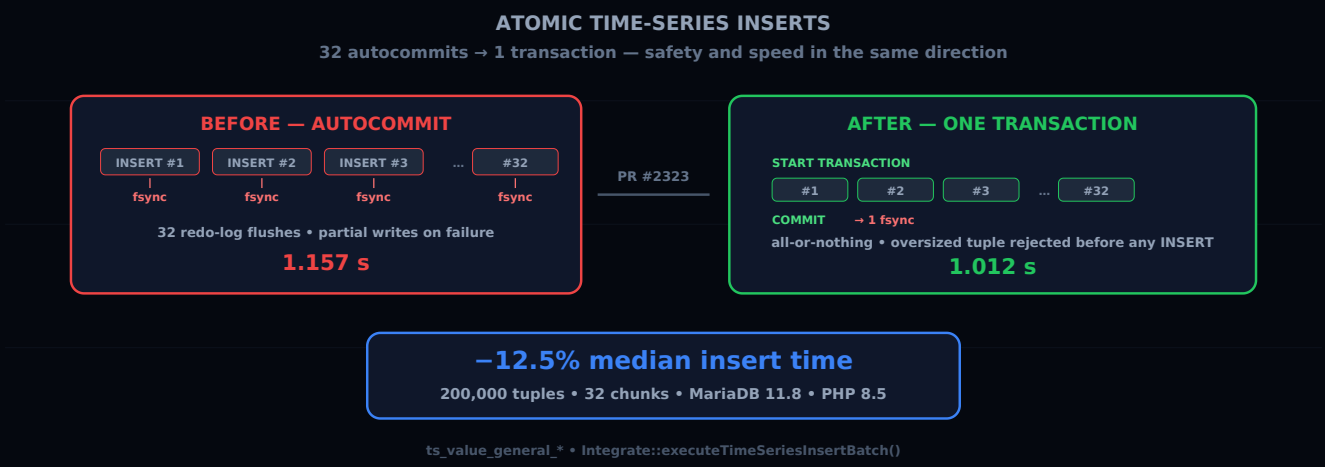
Когда фикс атомарности ускоряет ингестию на 12%

Aurélien LEQUOY · July 25, 2026

MARIADB MYSQL INNODB TIME-SERIES PERFORMANCE TRANSACTIONS PMACONTROL



Контекст: самый горячий путь записи в PmaControl

Каждые несколько секунд каждый агент PmaControl собирает сотни метрик с ваших серверов MariaDB / MySQL: статусные переменные, счётчики InnoDB, состояние репликации, дайджесты запросов. Всё это стекается в `Integrate::insert_value()`, который пишет измерения в таблицы `ts_value_general_*` — самый нагруженный путь записи в приложении.

Чтобы никогда не превышать `max_allowed_packet`, кортежи группируются в **чанки**: SQL-бюджет вычисляется динамически по данным сервера (с запасом), и партия из 200 000 измерений превращается, например, в 32 запроса `INSERT INTO ... VALUES (...), (...), ...` примерно по 256 КиБ каждый.

До сих пор эти 32 запроса выполнялись в режиме **autocommit**: 32 INSERT, 32 неявных COMMIT.

Баг: тихие частичные записи

Что происходит, если чанк № 17 падает — переполненная таблица, дедлок, разрыв соединения?

До фикса: чанки с 1 по 16 оставались записанными, чанки с 17 по 32 терялись, а последующий учёт (связка сервер/переменная, чекпоинт ингестии) мог выполниться так, будто всё прошло успешно. Логическая партия превращалась в тихую частичную запись — худший сценарий для данных мониторинга: графики с дырами, о которых никто не сообщает.

Три отдельных задачи сходились к одной и той же первопричине:

- **#1467 / #1471** — частичные записи при сбое посреди партии;
- **#1468 / #1472** — одиночный кортеж крупнее бюджета всё равно отправлялся на сервер, гарантированно падая на `max_allowed_packet`;
- **#1469** — тихий фолбэк мог отключить динамическое определение бюджета.

Фикс: одна партия = одна транзакция

Исправление (PR #2323) вводит `executeTimeSeriesInsertBatch()`: все чанки одного типа метрики теперь обёрнуты в **одну транзакцию**:

```
START TRANSACTION;
INSERT INTO ts_value_general_int (...) VALUES (...), (...), ...;  -- чанк 1
INSERT INTO ts_value_general_int (...) VALUES (...), (...), ...;  -- чанк 2
-- ... ещё 30 чанков ...
COMMIT;
```

Любой сбой — falsy-результат, небезобидный warning, исключение — вызывает `ROLLBACK` и поднимает типизированное исключение с безопасными диагностическими метаданными (таблица, чанк, оценка байтов, бюджет). Файл измерений сохраняется для повтора: либо вся партия записана, либо ничего.

Бонус: одиночный кортеж, чей оценочный размер уже превышает бюджет, обнаруживается **до** открытия транзакции — заведомо обречённый INSERT вообще не отправляется на сервер.

Вопрос на 12%: сколько это стоит?

Транзакция на 32 запроса — это дополнительная бухгалтерия на стороне InnoDB: более длинный undo log, дольше удерживаемые блокировки. Мы ожидали небольших накладных расходов и считали их справедливой ценой за гарантию атомарности.

Перед мержем мы сделали бенчмарк. Протокол:

- **точное** воспроизведение горячего цикла `insert_value()` (чанкование → сборка SQL → выполнение), а не синтетический микробенчмарк;
- 200 000 детерминированных кортежей в `ts_value_general_int` (уникальные первичные ключи), бюджет 256 КиБ → 32 чанка;
- MariaDB 11.8, PHP 8.5.8, таблица InnoDB, `TRUNCATE` между прогонами, 1 прогрев + 5 измеряемых прогонов;
- один и тот же LXC-контейнер, одни и те же данные, между замерами меняется только код приложения.

Результаты:

Прогон	До (autocommit ×32)	После (1 транзакция)
1	1,127 с	0,927 с
2	1,199 с	0,947 с
3	1,155 с	1,012 с
4	1,168 с	1,019 с
5	1,157 с	1,026 с
Медиана	**1,157 с**	**1,012 с**

Медиана быстрее на 12,5%. Фикс атомарности ничего не стоит — он экономит время. Пропускная способность ингестии выросла примерно со 173 000 до 198 000 измерений в секунду.

Почему быстрее: скрытая цена COMMIT

Ответ кроется в том, что `COMMIT` на самом деле делает в InnoDB.

При каждой фиксации InnoDB обязан сделать транзакцию **долговечной**: записать страницы redo-лога и — при `innodb_flush_log_at_trx_commit = 1`, значении по умолчанию и

единственном по-настоящему безопасном — принудительно выполнить `fsync()` файла лога. Этот flush — самая дорогая операция в жизненном цикле транзакции: она физически ждёт хранилище.

В режиме autocommit каждый `INSERT` — отдельная транзакция:

- **до**: 32 чанка = 32 COMMIT = 32 сброса redo-лога;
- **после**: 32 чанка = 1 COMMIT = 1 сброс redo-лога.

Сэкономленная 31 синхронизация весит куда больше, чем чуть более длинный undo log. Это тот же механизм, из-за которого SQL-импорт в транзакции работает радикально быстрее, а *group commit* в MariaDB амортизирует сбросы между конкурентными транзакциями — здесь он применён внутри одной логической партии.

Выигрыш зависит от железа: на хранилище с очень медленным `fsync()` (механические диски, виртуализация без безопасного кэша записи) разрыв ещё больше. На быстром NVMe он сокращается. Но знак не меняется: одна транзакция всегда как минимум не медленнее.

Выводы

- **Атомарность — не роскошь, за которую платят; часто это оптимизация.** Группировка связанных записей в одну транзакцию устраняет сбросы redo-лога — надёжность и производительность работают в одну сторону.
- **Бенчмаркайте реальный путь кода.** Именно воспроизведение настоящего горячего цикла на настоящей базе превратило «мы принимаем накладные расходы» в «накладных расходов нет».
- **Сбой должен быть громким и полным.** Наполовину записанная партия метрик хуже, чем партия, записанная повторно: с этим фиксом PmaControl гарантирует «всё или ничего» для ингестии time-series.

Фикс доступен в ветке `master` PmaControl с 25 июля 2026 года.