ORES
Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.75

УНИВЕРСАЛЬНЫЙ МОДУЛЬ ПАКЕТНОЙ ОБРАБОТКИ СООБЩЕНИЙ APACHE KAFKA ПРИ ЗАГРУЗКЕ ДАННЫХ В CLICKHOUSE

Пономарев Максим Дмитриевич,
Национальный исследовательский ядерный университет «МИФИ», Москва, Российская Федерация
Институт интеллектуальных кибернетических систем, кафедра «Компьютерные системы и технологии»
e-mail: pmd005@campus.mephi.ru;

Красникова Светлана Анатольевна,
Национальный исследовательский ядерный университет «МИФИ», Москва, Российская Федерация
Институт интеллектуальных кибернетических систем, кафедра «Компьютерные системы и технологии»
e-mail: sakrasnikova@mephi.ru

Аннотация

В статье рассматривается задача повышения устойчивости ETL-конвейера, передающего потоковые данные из Apache Kafka в аналитическое хранилище ClickHouse. Исходная проблема связана с тем, что потребители сообщений выполняли частые мелкие вставки, не соответствующие модели хранения MergeTree и приводившие к накоплению частей таблиц. Предложен универсальный модуль-обертка над новой версией клиентской библиотеки Kafka, который скрывает сложность низкоуровневого API и вводит управляемую пакетную обработку, повторные попытки с экспоненциальной задержкой и добавлением случайной компоненты, корректное завершение работы и идемпотентную запись в ClickHouse. Практическая проверка показала, что внедрение модуля позволило снизить частоту INSERT-запросов в несколько раз, устранить проявления ошибки too many parts и сохранить контролируемую семантику обработки сообщений при сбоях.

Ключевые слова: Apache Kafka, ClickHouse, ETL, пакетная обработка, идемпотентность, дедупликация, распределенные системы.

UNIVERSAL MODULE FOR BATCH PROCESSING OF APACHE KAFKA MESSAGES DURING DATA LOADING INTO CLICKHOUSE

Ponomarev Maksim Dmitrievich,
National Research Nuclear University MEPhI, Moscow, Russian Federation
Institute of Cyber Intelligent Systems, Department of Computer Systems and Technologies
e-mail: pmd005@campus.mephi.ru

Krasnikova Svetlana Anatolevna,

National Research Nuclear University MEPhI, Moscow, Russian Federation
Institute of Cyber Intelligent Systems, Department of Computer Systems and Technologies
e-mail: sakrasnikova@mephi.ru

ABSTRACT

The paper studies the reliability and performance problem of an ETL pipeline that transfers streaming data from Apache Kafka to the ClickHouse analytical database. The initial implementation produced frequent small inserts, which contradicted the MergeTree storage model and caused excessive table part creation. A universal wrapper module over a new Kafka client library is proposed. The module isolates application code from the low-level API and introduces controlled batch processing, retries with exponential backoff and jitter, graceful shutdown, and idempotent writes to ClickHouse. Practical evaluation showed that the module reduced the frequency of INSERT queries severalfold, removed the too many parts failure mode, and preserved controlled message-processing semantics under transient failures.

Keywords: Apache Kafka, ClickHouse, ETL, batch processing, idempotency, deduplication, distributed systems.

Введение

Потоковая аналитика в прикладных информационных системах строится на взаимодействии нескольких разнородных компонентов: брокера сообщений, слоя преобразования данных и аналитического хранилища. Apache Kafka проектировалась как распределенная система доставки больших объемов событий с малой задержкой, где данные организуются в топики и партиции, а потребители могут независимо считывать последовательный журнал сообщений [1]. Такая модель удобна для ETL-сервисов, поскольку позволяет обрабатывать поток данных почти в реальном времени, масштабировать потребление по партициям и повторно считывать сообщения при сбоях. Однако свойства брокера сообщений сами по себе не определяют эффективность всего конвейера. Производительность конечной системы зависит от того, насколько согласованы режимы потребления событий, обработки ошибок и записи данных в целевое хранилище.

ClickHouse относится к классу высокопроизводительных колоночных OLAP-СУБД и ориентирован на аналитические нагрузки с большими объемами данных. Его архитектура предполагает, что вставляемые данные преобразуются в части таблиц, которые затем асинхронно объединяются фоновыми merge-процессами [7]. Поэтому частые мелкие вставки создают непропорциональную служебную нагрузку: число частей растет быстрее, чем система успевает их объединять, повышается давление на файловую систему и ClickHouse Keeper, а при превышении внутренних порогов возникают задержки или отказ INSERT-запросов. Официальные рекомендации ClickHouse прямо связывают устойчивую загрузку с крупными пакетами данных и ограничением частоты INSERT-операций [4, 5].

Практическая проблема, рассмотренная в настоящей работе, возникла в ETL-сервисе, который принимал множество потоков Kafka и сохранял преобразованные события в кластер ClickHouse. Используемая версия клиентской библиотеки не поддерживала накопление сообщений в батчи по времени или размеру, вследствие чего приложение выполняло вставки почти по мере поступления событий. В обычном режиме интервал между вставками составлял несколько секунд, а в пиковые периоды сокращался до порядка 100 мс. Такое поведение хорошо согласовывалось с низколатентной природой потребителя

Kafka, но вступало в противоречие с режимом, в котором ClickHouse достигает устойчивой производительности.

Новая версия клиентской библиотеки предоставляла необходимые механизмы пакетной обработки, однако ее API был несовместим со старым интерфейсом и перенос существующих потребителей напрямую повышал риск ошибок. В связи с этим возникла исследовательская задача: спроектировать промежуточный слой, который позволит использовать возможности новой библиотеки, но сохранит для прикладного кода простой и единообразный контракт. Научно-практический интерес здесь состоит не только в замене одной версии библиотеки другой, а в построении архитектурного способа согласования семантики Kafka-потребления, retry-обработки и ограничений MergeTree-хранилища.

Цель исследования

Цель исследования заключается в разработке и проверке универсального модуля пакетного потребления сообщений Apache Kafka, предназначенного для идемпотентной загрузки данных в ClickHouse и снижения нагрузки, вызванной частыми мелкими вставками. Для достижения цели необходимо было выделить устойчивый контракт между инфраструктурным слоем и бизнес-логикой, реализовать управляемое накопление сообщений, обеспечить корректную обработку временных отказов и подтвердить, что внедрение решения устраняет эксплуатационную проблему без потери управляемости обработки сообщений.

Материалы и методы исследования

Материалом исследования послужил ETL-сервис, реализующий типовой конвейер «Kafka – обработчик событий – ClickHouse». Входной поток представлен сообщениями из нескольких топиков Kafka; каждое сообщение после десериализации и преобразования сохраняется в аналитические таблицы ClickHouse. В исходной реализации прикладные потребители зависели от устаревшей клиентской библиотеки, которая не давала разработчику достаточного контроля над размером пакета и временем накопления. Поэтому даже при умеренном общем объеме данных число операций вставки становилось высоким, а нагрузка переносилась с полезной обработки на создание и последующее объединение множества малых частей таблиц.

Метод исследования включал архитектурный анализ потокового конвейера, проектирование фасадного слоя над новой версией клиентской библиотеки, реализацию модуля в языке Go и последующую проверку на уровне модульных и системных тестов. При проектировании учитывались две группы ограничений. Первая группа связана с Kafka: потребитель должен обрабатывать сообщения в пределах назначенной партиции последовательно, фиксировать смещение только после успешной обработки и корректно реагировать на ребалансировку группы потребителей. Официальная модель Kafka различает позицию чтения и зафиксированное смещение; именно зафиксированное смещение определяет точку восстановления после перезапуска [2, 3]. Вторая группа ограничений связана с ClickHouse: целевое хранилище эффективно при редких крупных вставках, а также поддерживает дедупликацию повторных insert-блоков для таблиц семейства MergeTree [4, 6].

В качестве основного архитектурного приема выбран фасад над низкоуровневой клиентской библиотекой. Модуль предоставляет прикладному коду минимальный интерфейс: разработчик передает функцию обработки батча и параметры накопления, а инфраструктурный слой берет на себя запуск потребителя, управление жизненным циклом, реакцию на ошибки, повторную обработку и закрытие ресурсов. Такой подход снижает связанность бизнес-логики с деталями клиента Kafka и позволяет применять одинаковую модель обработки для разных топиков без копирования инфраструктурного кода.

Особое внимание уделено семантике повторной обработки. Kafka предоставляет механизмы, необходимые для надежной доставки, однако при записи во внешнюю систему достижение эффекта «ровно один раз» требует согласования состояния потребителя и результата записи [2]. В рассматриваемом решении это согласование достигается на прикладном уровне: смещение Kafka фиксируется только после успешного завершения обработки батча, а повторная запись того же набора данных в ClickHouse нейтрализуется механизмом дедупликации insert-блоков. Если обработчик успешно записал данные, но сбой произошел до подтверждения смещения, батч может быть получен повторно; при неизменном содержимом блока ClickHouse определяет повторную вставку как дубликат и не создает вторую копию данных [6].

Для временных отказов нижестоящих систем реализован retry-механизм с экспоненциальной задержкой и случайной добавкой. Экспоненциальное увеличение интервала между попытками (exponential backoff) уменьшает давление на перегруженную систему, а случайная добавка (jitter) рассеивает моменты повторных запросов между несколькими потребителями. Такой принцип согласуется с общими исследованиями алгоритмов exponential backoff, где задержка рассматривается как способ стабилизировать конкурирующие запросы при росте нагрузки [9]. В модуле retry-цикл остается управляемым: при отмене контекста из-за остановки приложения или потери партиции повторные попытки прекращаются, что предотвращает обработку данных вне актуального владения партицией.

Результаты и их обсуждение

Разработанный модуль представляет собой потребительский слой между Kafka-клиентом и прикладной бизнес-логикой. Его работа начинается с инициализации подписчика для заданного топика и группы потребителей, после чего чтение запускается асинхронно. Каждая назначенная партиция обрабатывается как отдельная последовательность сообщений; это сохраняет порядок внутри партиции и одновременно позволяет использовать параллелизм между партициями. Батч считается успешно обработанным только после возврата прикладного обработчика без ошибки, а переход к следующей итерации инициирует подтверждение последнего безопасно обработанного смещения.

Внутренняя структура решения разделяет три ответственности. Первый уровень отвечает за жизненный цикл подписчика: запуск, перехват критических ошибок, логирование контекста и закрытие соединения. Второй уровень формирует обработчик партиции, который получает накопленный батч, передает его в бизнес-логику и принимает решение о повторной попытке. Третий уровень реализует стратегию управления повторными попытками операций при временных ошибках. Благодаря такому разделению прикладной разработчик не управляет смещениями, не реализует повторные попытки и не должен знать детали протокола новой библиотеки; его код ограничивается десериализацией сообщения, преобразованием в доменную модель и записью подготовленного набора строк в хранилище.

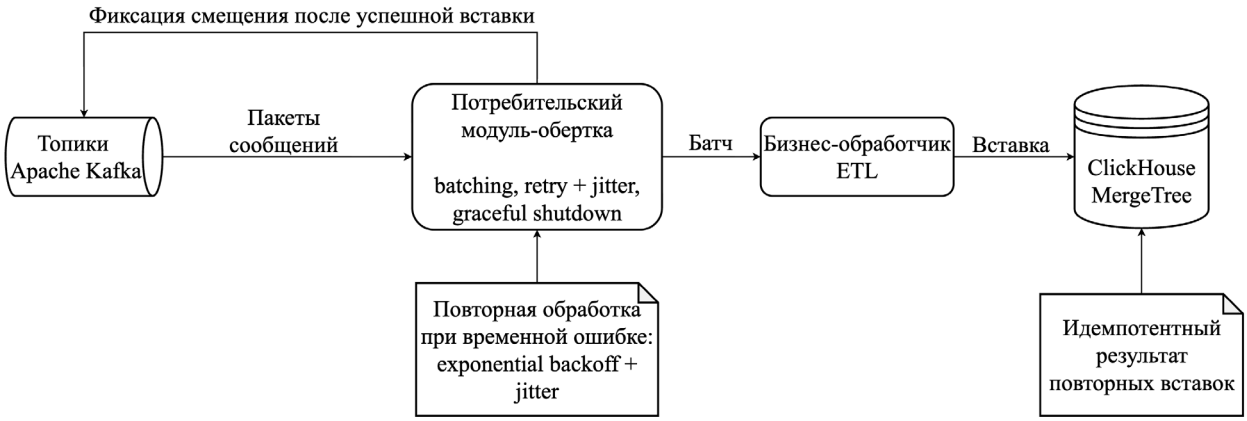

Рисунок 1 – Обобщенная архитектура пакетного потребительского слоя

На рисунке 1 показана обобщенная схема решения. Модуль-обертка расположен до прикладного обработчика и поэтому способен регулировать размер батча еще до обращения к ClickHouse. Это принципиально отличает предложенный подход от попытки компенсировать проблему только настройками асинхронных вставок на стороне ClickHouse. Асинхронные вставки действительно могут переносить буферизацию на сервер и уменьшать число создаваемых частей [8], однако они не устраняют избыточную активность клиента, если потребитель продолжает порождать большое число мелких операций. В рассматриваемой архитектуре поток сглаживается раньше, на стороне потребителя Kafka, что снижает как количество запросов к хранилищу, так и число ситуаций, требующих повторной записи.

Практическая проверка включала модульное и системное тестирование. Unit-тесты покрыли запуск подписчика, обработку ошибок подписки, штатную отмену контекста, восстановление после panic, закрытие подписчика, повторную обработку батча после первичной ошибки и завершение обработки при отмене контекста. Покрытие модуля unit-тестами составило 100 %, что важно именно для инфраструктурного компонента: ошибка в нем тиражировалась бы на все потребители топиков. Системная проверка выполнялась после внедрения модуля в ETL-сервис и включала контроль приема сообщений, устойчивость при ребалансировках, корректность graceful shutdown и отсутствие накопления критического отставания потребителей.

После внедрения модуля частота INSERT-запросов к ClickHouse снизилась в несколько раз по сравнению с исходным режимом. В мониторинге наблюдался переход от нестабильного высокочастотного профиля вставок к более редким пакетным операциям; при этом ошибка too many parts перестала проявляться как эксплуатационный отказ. Полученный эффект объясняется не отдельной оптимизацией запроса, а изменением формы нагрузки: вместо множества малых вставок ClickHouse получает более крупные блоки данных, которые лучше соответствуют рекомендациям по bulk inserts и уменьшают давление на фоновые merge-процессы [4, 5].

С точки зрения надежности основным результатом является согласованная обработка сбоя между Kafka и ClickHouse. Если ошибка возникает до записи данных, батч повторяется после задержки. Если запись была выполнена, но подтверждение смещения не зафиксировано, Kafka может выдать тот же батч повторно; при этом ClickHouse использует журнал дедупликации и не создает дубликат блока [6]. Следовательно, модуль не пытается заменить транзакционный протокол между Kafka и внешней СУБД, а использует более прикладной механизм идемпотентности, приемлемый для аналитической загрузки данных. В отличие от строгой транзакции с внешним хранилищем, такое решение проще внедряется в существующий ETL-сервис и не требует изменения схемы всех потребителей.

Ограничением предложенного подхода является зависимость от устойчивости формы вставляемого блока. Дедупликация ClickHouse корректно работает тогда, когда при повторной попытке сохраняется тот же набор строк и порядок данных в блоке, а параметры дедупликации не отключены. Поэтому бизнес-обработчик должен быть детерминированным относительно входного батча, а операции обогащения данных из внешних источников не должны менять результат между повторными попытками без явного учета этого факта. Другое ограничение связано с окном дедупликации: если число других вставок или время между попытками выходит за заданные параметры журнала, повторная вставка может перестать распознаваться как дубликат [6]. Эти ограничения не отменяют применимость решения, но задают эксплуатационные требования к конфигурации ClickHouse и к дисциплине разработки обработчиков.

Заключение

В результате исследования разработан и внедрен универсальный модуль пакетной обработки сообщений Apache Kafka для ETL-конвейера, выполняющего загрузку данных в ClickHouse. Предложенное решение устраняет несоответствие между низколатентным режимом потребления сообщений и пакетной природой эффективных вставок в MergeTree-хранилище. Архитектурная новизна работы заключается в выделении отдельного потребительского слоя, который объединяет батчирование, управляемые повторные попытки, контролируемое завершение работы, безопасный момент фиксации смещения и идемпотентную запись данных во внешнее аналитическое хранилище.

Практическая проверка показала, что модуль уменьшает частоту INSERT-запросов в ClickHouse в несколько раз, устраняет проявление ошибки too many parts и упрощает разработку новых Kafka-потребителей за счет минимального API. Прикладной разработчик реализует только преобразование сообщения и доменную операцию записи, тогда как инфраструктурные аспекты обработки сосредоточены в едином компоненте. Дальнейшее развитие решения может быть связано с автоматической адаптацией параметров батчирования к текущей нагрузке и с дополнительным уровнем дедупликации на стороне приложения для сценариев, где свойства внешнего хранилища недостаточны.

Список литературы:

1. Kreps J., Narkhede N., Rao J. Kafka: a Distributed Messaging System for Log Processing // NetDB'11. Athens, 2011. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/09/Kafka.pdf (дата обращения: 03.05.2026).
2. Apache Kafka. Design: Message Delivery Semantics [Электронный ресурс]. URL: https://kafka.apache.org/43/design/design/ (дата обращения: 08.05.2026).
3. Apache Kafka. Distribution: Consumer Offset Tracking [Электронный ресурс]. URL: https://kafka.apache.org/42/implementation/distribution/ (дата обращения: 13.05.2026).
4. ClickHouse. Bulk inserts [Электронный ресурс]. URL: https://clickhouse.com/docs/optimize/bulk-inserts (дата обращения: 04.05.2026).
5. ClickHouse. Resolving «Too Many Parts» Error in ClickHouse [Электронный ресурс]. URL: https://clickhouse.com/docs/knowledgebase/exception-too-many-parts (дата обращения: 04.05.2026).
6. ClickHouse. Deduplicating inserts on retries [Электронный ресурс]. URL: https://clickhouse.com/docs/guides/developer/deduplicating-inserts-on-retries (дата обращения: 15.05.2026).

7. Schulze R., Schreiber T., Yatsishin I., Dahimene R., Milovidov A. ClickHouse — Lightning Fast Analytics for Everyone // Proceedings of the VLDB Endowment. 2024. Vol. 17, № 12. P. 3731–3744. DOI: 10.14778/3685800.3685802.
8. ClickHouse. Asynchronous inserts [Электронный ресурс]. URL: https://clickhouse.com/docs/optimize/asynchronous-inserts (дата обращения: 19.05.2026).
9. Song N.-O., Kwak B.-J., Miller L. E. On the Stability of Exponential Backoff // Journal of Research of the National Institute of Standards and Technology. 2003. Vol. 108, № 4. DOI: 10.6028/jres.108.027. URL: https://www.nist.gov/publications/stability-exponential-backoff (дата обращения: 18.05.2026).

References:

1. Kreps J., Narkhede N., Rao J. Kafka: a Distributed Messaging System for Log Processing. NetDB'11, Athens, 2011. Available at: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/09/Kafka.pdf (accessed 03.05.2026).
2. Apache Kafka. Design: Message Delivery Semantics. Available at: https://kafka.apache.org/43/design/design/ (accessed 08.05.2026).
3. Apache Kafka. Distribution: Consumer Offset Tracking. Available at: https://kafka.apache.org/42/implementation/distribution/ (accessed 13.05.2026).
4. ClickHouse. Bulk inserts. Available at: https://clickhouse.com/docs/optimize/bulk-inserts (accessed 04.05.2026).
5. ClickHouse. Resolving “Too Many Parts” Error in ClickHouse. Available at: https://clickhouse.com/docs/knowledgebase/exception-too-many-parts (accessed 04.05.2026).
6. ClickHouse. Deduplicating inserts on retries. Available at: https://clickhouse.com/docs/guides/developer/deduplicating-inserts-on-retries (accessed 15.05.2026).
7. Schulze R., Schreiber T., Yatsishin I., Dahimene R., Milovidov A. ClickHouse — Lightning Fast Analytics for Everyone. Proceedings of the VLDB Endowment, 2024, vol. 17, no. 12, pp. 3731–3744. DOI: 10.14778/3685800.3685802.
8. ClickHouse. Asynchronous inserts. Available at: https://clickhouse.com/docs/optimize/asynchronous-inserts (accessed 19.05.2026).
9. Song N.-O., Kwak B.-J., Miller L. E. On the Stability of Exponential Backoff. Journal of Research of the National Institute of Standards and Technology, 2003, vol. 108, no. 4. DOI: 10.6028/jres.108.027. Available at: https://www.nist.gov/publications/stability-exponential-backoff (accessed 18.05.2026).