Optimization

Лучше вставлять большими кусками (10 000 -100 000 строк), т.к. на каждую операцию вставки идут доп метаданные. <https://clickhouse.com/docs/ru/guides/inserting-data?utm_source=chatgpt.com#insert-in-large-batch-sizes>

Оптимизация вставок в Clickhouse через Kaffka SinkConnector

<https://habr.com/ru/companies/magnit/articles/926834/>

Кратко:

**Первый этап оптимизации: настройка fetch и poll параметров**

Чтобы коннектор формировал крупные батчи, мы скорректировали параметры потребителя Kafka (Kafka Consumer), используемого Sink-коннектором. Ниже в таблице приведены ключевые параметры **до** и **после** оптимизации и их роли:

|  |  |  |  |
| --- | --- | --- | --- |
| **Параметр** | **Дефолтное значение** | **Измененное значение** | **Описание параметра** |
| fetch.min.bytes | 1 байт | 2 621 440 байт (2,5 Мб) | увеличивает размер пачки и снижает частоту обращений |
| fetch.max.wait.ms | 500 мс | 72 000 мс (72 с) | Максимальное время, которое брокер будет ждать, чтобы набрать пачку размера fetch.min.bytes. Дефоолт – 0,5 с |
| max.poll.records | 500 сообщений | 50 000 сообщений | Максимальное число сообщений, возвращаемых за один вызов poll(). По умолчанию 500, мы увеличили до 50 000, |
| [max.poll.interval.ms](http://max.poll.interval.ms/) | 300 000 мс (5 мин) | 120 000 мс (120 с) | Максимальная задержка между вызовами poll() при работе в составе группы потребителей. По умолчанию 5 минут: если потребитель не вызывает poll дольше, он считается «зависшим» [и](https://docs.confluent.io/platform/current/installation/configuration/consumer-configs.html#max-poll-interval-ms) [исключается из группы](https://docs.confluent.io/platform/current/installation/configuration/consumer-configs.html#max-poll-interval-ms). |

**Проблема таймаутов**

После применения новых настроек данные действительно стали накапливаться в большие пачки. Однако поначалу возникли побочные эффекты, связанные с таймаутами. Мы получили разрывы подключения к Kafka: в логах Kafka Client было сообщение о отключении от узла. Брокер разъединялся, потому что коннектор ждал ответ дольше дефолтных 30 с (при новом fetch.max.wait.ms = 72 секунд).

**Решение**: повысить request.timeout.ms на клиенте до 100 с ("request.timeout.ms" = "100000"), чтобы покрыть максимальное время ожидания батча + возможные сетевые задержки. После увеличения этого параметра разрывы соединения прекратились.

**Второй этап оптимизации: настройка ClickHouse-коннектора с помощью опции ignorePartitionsWhenBatching**

После оптимизации настроек потребителя Kafka на первом этапе (увеличен объём данных, забираемых за один запрос к брокеру), мы столкнулись с ограничением: по умолчанию коннектор Kafka → ClickHouse формирует INSERT-запросы, разбивая полученные сообщения по отдельным батчам согласно партициям Kafka-топика.

Это приводит к тому, что даже при достаточно большом объёме данных, который коннектор получает за один вызов метода poll(), данные всё равно отправляются в ClickHouse множеством мелких транзакций, так как каждый переход к новой партиции топика вынуждает отправить текущий батч и начать формирование нового. В итоге снижается эффективность передачи данных в ClickHouse, так как кластер наиболее эффективен именно при крупных пакетных вставках.

Чтобы устранить это ограничение, мы применили дополнительную оптимизацию на стороне самого коннектора:

**ignorePartitionsWhenBatching = true**