Анализ инструментов быстрого захвата изменений данных

Мажара Е.Н., СПбГУ, Санкт-Петербург st087790@student.spbu.ru

Аннотация

В данной работе рассматривается задача быстрой репликации данных между хранилищем и аналитической системой посредством принципа захвата изменений (Change Data Capture, CDC). Для решения этой задачи рассмотрены инструменты Debezium и Airbyte, разработаны примеры конвейеров данных с их использованием. Для разработанных примеров проведена оценка функциональных и нефункциональных характеристик, таких как возможность захвата изменений разных типов и скорость доставки данных.

Введение

В современном мире хранение и обработка данных из различных источников стала важнейшей задачей практически в любой технологической сфере. Современные приложения часто используют несколько хранилищ данных для обслуживания различных задач: хранения информации и её анализа. Используются специализированные аналитические хранилища, способные повысить эффективность анализа данных и снизить нагрузку на основное хранилище [1].

Поскольку данные в хранилище-источнике могут меняться: добавляться, изменяться или удаляться – возникает проблема согласования данных между хранилищем-источником и аналитическим хранилищем. Согласно исследованию [2], принцип полного обновления хранилища для получения изменений неэффективен. Поэтому для решения проблемы синхронизации данных применяется инкрементальный паттерн ***CDC*** (*Change Data Capture, Захват изменений данных*), используемого для определения изменяемых данных, чтобы предпринять действия с использованием этих данных [3].

В данной работе будет проведён анализ существующих CDC-решений, выбраны наиболее подходящие кандидаты и на их основе построены конвейеры данных, решающие задачу репликации в аналитическое хранилище. Конвейеры затем будут протестированы на предмет скорости доставки изменений.

Анализ существующих решений

На данный момент на рынке представлено более десятка инструментов с необходимой функциональностью. В рамках работы была проанализирована документация и исходный код десяти CDC-инструментов, которые затем были отфильтрованы следующими критериями:

- Открытость исходного кода инструмента;
- Дата последнего обновления либо минорного изменения;
- Популярность и масштаб использования (на основе информации из репозитория с кодом приложения);
- Возможность отслеживания изменений данных в базе **PostgreSQL**;
- Публикация изменений в брокер сообщений **Apache Kafka**;
- Поддержка DML-операций **INSERT**, **UPDATE**, **DELETE**;
- Реализация *at-least-once* семантики.

В результате были выбраны две платформы: Debezium и Airbyte.

Debezium

Один из самых распространенных способов доставки изменений данных на сегодняшний день. Проект разрабатывается как полностью бесплатный open-source под лицензией Apache License v2.0 и спонсируется Red Hat. Отслеживание изменений происходит методом чтения журнала транзакций (WAL-журнал) [4].

№	Преимущества	Недостатки
1	Виртуализация	Вероятность потери данных
2	Широкий выбор базы-источника	Возможны дубликаты
3	Единая структура сообщений	Снапшот блокирует таблицы

Таблица 1: Особенности Debezium.

Работа Debezium построена на нескольких компонентах: сервер конфигурации Zookeeper, брокер сообщений Kafka и основной – Kafka Connect. Развертывание кластера Kafka Connect экспонирует REST API, которое позволяет получать информацию о существующих соединениях и создавать новые.

Для создания посылается HTTP POST-запроса с телом, содержащим информацию о создаваемом соединении: параметры БД-источника, список отслеживаемых таблиц. Изменения приходят в брокер в формате JSON и содержат данные о типе операции, моменте её фиксации, схеме и изменениях данных.

Airbyte

Платформа Airbyte предоставляет более широкую функциональность: это целая система хранения и обработки данных, которая предоставляет услуги облачного хранилища, а также позволяет настраивать соединения между различными источниками и получателями данных. Проект разрабатывается как частично бесплатный с открытым исходным кодом под лицензией Elastic License v2.0, его разработка поддерживается платными облачными функциями. Имеется возможность выбора метода отслеживания изменений, в частности, может быть выбран метод чтения журнала транзакций (WAL-журнал) [5].

№	Преимущества	Недостатки
1	Виртуализация	Ручная синхронизация
2	Выбор источника и приёмника	Возможны дубликаты
3	Удобный веб-интерфейс	Система логирования

Таблица 2: Особенности Airbyte.

Кластер Airbyte включает множество компонентов, которые запускаются совместно при помощи единого скрипта; в частности, создаётся локальный сервер, позволяющего управлять соединениями при помощи удобного интерфейса. Для создания CDC-соединения между хранилищем и брокером сообщений необходимо выбрать и настроить источник и приёмник, передав необходимые конфигурационные параметры, такие как JDBC-адрес базы данных и адрес брокера сообщений, название топика для записи, наименования таблиц для отслеживания и т.д., через веб-формы вместо файла. Сообщения также имеют формат JSON и содержат данные о моменте фиксации операции, а также об изменённых данных.

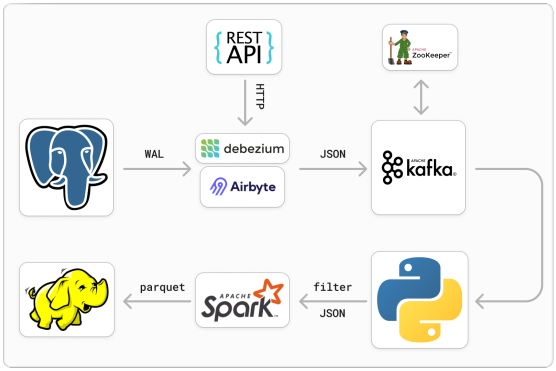


Рис. 1: Схема работы конвейеров данных.

Реализация конвейера данных

Источник

Источником, из которого происходит вычитывание данных для нотификации об изменениях, служит база данных на основе СУБД PostgreSQL [6]. Для отслеживания изменения данных используется одна таблица с несложной схемой данных (4 столбца). Для удобства сервер PostgreSQL с базой запущены в docker-контейнере.

Аналитическое хранилище

В качестве целевого аналитического хранилища используется распределённая файловая система HDFS (Hadoop Distributed File System) [7]. Изменения вносятся в неё в колоночном формате parquet. Запуск кластера HDFS происходит в docker-контейнерах, создаётся одна Datanode.

Доставка изменений

Попавшие в Kafka сообщения вычитываются и фильтруются python-скриптом, а трансформированная запись затем записывается в аналитическое хранилище при помощи Apache Spark [8], который преобразует JSON-объект в parquet-файл [9].

Тестирование

Методика тестирования

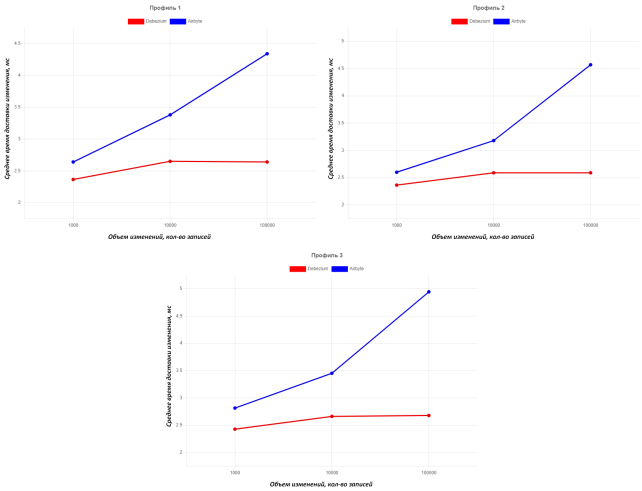
При тестировании измерялось среднее время доставки изменения при 1000, 10000 и 100000 изменяемых записях на трёх профилях, представленных в таблице 3.

№	Профиль 1	Профиль 2	Профиль 3
INS:UPD:DEL	1:1:1	1:0:0	1:1:8

Таблица 3: Профили тестирования.

Для снижения погрешности каждый замер проводился трижды на различных данных. Для вычисления среднего времени использовалась информация о времени доставки последнего сообщения, времени начала транзакции; получаемый интервал делился на количество изменений. Данные изменялись при помощи случайно сгенерированных python-программой SQL-скриптов.

Результаты



Заключение

В работе был проведён анализ и отбор CDC решений, в результате были выбраны инструменты Debezium и Airbyte. На их основе были созданы конвейеры данных, передающие изменения на источнике в целевое хранилище, которые были протестированы на предмет скорости доставки сообщений на нескольких профилях тестирования.

Результаты показали, что Debezium более стабилен при увеличении количества изменяемых записей. При этом оба инструмента показали схожие результаты на различных профилях, что говорит о несущественной зависимости времени доставки и типа отслеживаемой операции.

Список литературы

[1] Зайцев Г. Оптимизация применения CRUD операций для аналитического хранилища данных. — 2022. — 41С.

[2] Jorg T., Dessloch S. Formalizing ETL jobs for incremental loading of data warehouses // Proceedings der 13. GI-Fachtagung f ̈ur Datenbanksysteme in Business, Technologie und Web. Lecture Notes in Informatics. — 2009. — С. 327–-346.

[3] Imani F., Widyasari Y., Arifin S. Optimizing Extract, Transform, and Load Process Using Change Data Capture // 6th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI). — 2023. — С. 266–-269.

[4] Debezium Documentation // URL: https://debezium.io/documentation/reference/stable/index.html

[5] Airbyte Documentation // URL: https://airbyte.com/

[6] PostgreSQL Documentation // URL: https://www.postgresql.org/

[7] Shvachko K., Kuang H., Radia S., Chansler R. The Hadoop Distributed File System // IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). — 2010. — С. 1–-10.

[8] Spark Documentation // URL: https://spark.apache.org/

[9] Vohra D. Apache Parquet // Practical Hadoop Ecosystem: Apress, Berkeley, CA. — 2016. — С. 325–-335.