Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. Тихонова

Департамент электронной инженерии

РЕФЕРАТ

на тему

«Брокер Apache Kafka»

по курсу

«Принципы разработки ПО IoT/CPS»

Выполнила:

Студент группы

МИВ231

Ванин Кирилл

Преподаватель:

Алешин Д.В.

Москва, 2024

Содержание

1 Введение

Apache Kafka – open-source распределенная платформа потоковой передачи событий, используемая тысячами компаний для высокопроизводительных конвейеров данных, потоковой аналитики, интеграции данных и критически важных приложений, как указано на сайте компании. В данном реферате мы разберем что такое Apache Kafka, для каких случаев и в каких сценариях мы можем его применять и т.д. Брокер написан на языках Java и Scala. Изначально Kafka разрабатывалась в LinkedIn для внутреннего пользования, затем исходный код был открыт под лицензией Apache.

Потоковая передача событий, она же event streaming – это практика сбора данных в режиме реального времени из различных источников событий, например баз данных, датчиков, мобильных устройств, облачных сервисов в виде потоков этих данных. Потоки сохраняются для возможности их восстановления, изменения, обработки, перенаправления на соответствующие обработчики. Другими словами, задача event-streaming – обеспечение непрерывного потока и интерпретации данных.

Возможные сценарии приведем из примеров из официальной документации:

- Обработка платежей и финансовых транзакций в реальном времени;
- Отслеживание и контроль автомобилей, грузовиков и грузовых кораблей в реальном времени для транспортной логистики;
- Непрерывное считывание и обработка показаний IoT датчиков;
- Сбор и мгновенное реагирование на действия пользователей или их заказы, например в розничной торговле, отельном и туристическом бизнесе, мобильных приложениях;

- Мониторинг состояния пациентов и их жизненных показателей в больницах для мгновенного реагирования и предсказания изменений в состоянии;
- Соединение, хранение и предоставление доступа к данным, созданным различными подразделениями компании;
- Основа для платформ данных, событийно-ориентированных архитектур и микросервисов.

2 Брокер сообщений

Apache Kafka – это брокер сообщений, некий посредник между протоколами приложений-источников и приложений-потребителей, выполняющий обмен данными. Издатели публикуют данные в определенные топики, откуда подписчики этого топика забирают данные к себе.

В классическом варианте реализации сервиса очередей продюсеры отправляют сообщения в сконфигурированную на сервере очередь, а потребители считывают сообщения по мере их появления. Сервер с очередью является для отложенной обработки входящих сообщений или в качестве буфера для ограничения нагрузки на потребителей. Потребители могут получать данные с сервера или по собственном запросу (pull-модель), контролируя собственную нагрузку, или по запросу от сервера (push-модель), который сам выбирает момент для отправки данных. В таком случае снижается задержка обработки сообщений и распределение сообщений балансируется по потребителям. Каждое новое сообщение удаляется, как только подписчик подтверждает, что принял сообщение.

2.1 Продюсеры

Продюсеры в Kafka пушат свои данные в брокер лидеру партиции, о которых будет написано ниже. На серверах Kafka хранятся метаданных, позволяющие определить лидера каждой партиции. Продюсер определяет, в какой топик он отправляет данные, а дальше Kafka разбивает их по партициям. Для оптимизации отправки сообщений продюсеры стараются агрегировать данные и посылать их большими батчами по достижению заданного объема или же при превышении некоторого порога по времени сбора данных для отправки.

2.2 Консьюмеры

В Apache Kafka консьюмеры получают данные по модели pull, т.е. запрашивают данные самостоятельно. К преимуществам такой системы можно отнести агрессивную группировку данных, отправляемых подписчику на топик. В push-based системе брокер должен как то определить, отправить ли ему сообщение консьюмеру сразу при получении или накопить их и отправить пачкой. При этом брокер не знает о готовности консьюмера принимать эти сообщения, что приводит к буферизации сообщений где-то процессе, увеличивая задержку. Pull-based брокер же отдает сразу все доступные сообщения пакетом, исключая при этом задержки на буферизацию, оптимизируя тем самым время доставки.

В Kafka присутсвуют настройки, позволяющие оптимизировать pull запросы. Чтобы консьюмер не зависал в бесконечном цикле опроса, ожидая появления новых данных, Kafka может заблокировать запрос до тех пор, пока в брокере не появится заданный объем данных, чтобы передать большой объем за раз.

Фундаментальное отличие Kafka от очередей заключается в том, что хранятся на брокере продолжительное время, а не удаляются сразу же после после обработки консьюмерами. Это позволяет обрабатывать одно и то же сообщение разными потребителями в разных контекстах и в разное время. Это позволяет, в отличие от других сервисов очередей, не конфигурировать очередь «разгребания» для нового подключённого сервиса, поскольку данные будут храниться в Kafka заданное время и каждый потребитель сможет считать его по мере необходимости. Кроме того, поскольку данные сохраняются на какой-то срок, мы можем откатить историю, например, для восстановления сбоя, или для передачи контекста новым потребителям.

Консьюмеры обычно объединяются в группы. Каждая группа имеет уникальное название и регистрируется брокерами в кластере Kafka. Для

распределения нагрузки консьюмеры из одной группы читают данные из разных партиций топика. При этом на один топик могут быть подписаны несколько групп.

Если внутри группы всего один потребитель, он читает информацию из всех партиций. При увеличении числа потребителей партиции разделяются между ними, а идеальное распределение нагрузки достигается при совпадении числа партиций и числа потребителей внутри группы. Тем самым, увеличивая число партиций мы можем масштабировать Kafka, а наличие группы потребителей позволяет продолжить работу сервисов даже если одни потребитель отваливается.

Поскольку сообщения хранятся продолжительное время, нужно как-то определять, читал ли конкретный консьюмер определенное сообщение или нет. Как уже говорилось, каждое сообщение в партиции имеет свое значение offset. Перед чтением сообщений консьюмер делает запрос к брокеру, совершая offset-commit, указывая свою группу, идентификатор топика-партиции и offset, который должен быть отмечен обработанным. У брокера имеется отдельный топик, где хранятся эти данные. Соответственно, если у консьюмера теряется значение offset, он может запросить информацию по последнему offset-commit для нужного топика и партиции, тем самым продолжив читать сообщения с того места, на котором он остановился. Соответственно, указание значения offset позволяет нам читать необходимые сообщения, двигаясь по времени происхождения при необходимости.

3 Как хранятся данные

Поскольку Kafka хранит данные в топиках какое-то время, необходимо решить предотвратить повторное чтение одних и тех же данных потребителями. Топики разделены на партиции, расположенные по разным брокерам внутри одного кластера. Такое расположение данных – ключевая особенность, позволяющая масштабировать систему горизонтально. Это позволяет консьюмерам и издателям записывать и считывать данные в брокеры параллельно. При этом сообщения записываются в партицию топика по уникальному ключу, что гарантирует правильный порядок хранения информации для каждого из событий. Более того, каждый топик может быть реплицирован для защиты от возможных сбоев.

У каждой партиции существует «лидер» - брокер, ответсвенный непосредственно за передачу сообщений. К нему обращаются остальные брокеры (Follower), которые уже хранят реплики данных из партиций. Каждый из брокеров хранит метаданные, которые позволяют определить лидера партиции.

По сути, партиция – это распределенный реплицированный лог, которых хранится где-то на диске. Каждое новое сообщение сохраняется в «голову» этого лога и получает свое значение offset (64-битное число, которое назначает брокер). Длительность хранения каждого сообщения ограничивается настройками Kafka и по сути ограничено лишь памятью на диске, при этом это не влияет на производительность системы.

4 Заключение

К преимуществам Apache Kafka можно отнести высокую пропускную способность и низкую задержку за счет встроенного механизма партицирования брокеров, что отлично подходит для задач Big Data. Система эффективно работает параллельно с различными источниками и потребителями, а также хорошо масштабируется горизонтально. Отличием Kafka от, допустим, RabbitMQ, является то, что сообщения хранятся в виде распределенного лога, причем порядок сообщений в этом логе гарантированно отражает порядок поступления этих сообщений, а сами сообщения могут храниться продолжительное время.

Такими образом, брокер Apache Kafka больше предназначен для построения высоконагруженных систем сферы bigdata, так как сама его парадигма параллельной обработки, репликации позволяет создавать достаточно надёжные системы и обеспечивать неограниченные возможности по масштабированию. Высокая пропускная способность, а также возможности извлечения сообщений из очереди за определённый период времени являются мощным инструментом для анализа происходящего в историческом разрезе.

5 Список литературы

1. Документация Apache Kafka – kafka.apache.org/documentation
2. Apache Kafka: основы технологии - habr.com/ru/companies/slurm/articles/550934/