**Apache Kafka**

**История**

Изначально Kafka был разработан в LinkedIn, а затем был открыт в начале 2011 года. Джей Крепс, Неха Нархеде и Джун Рао участвовали в создании Kafka. Выпуск из инкубатора Apache произошел 23 октября 2012 года. Джей Крепс решил назвать программное обеспечение в честь автора Франца Кафки, потому что это "система, оптимизированная для написания", и ему понравились работы Кафки.

**Предпосылки создания**

Основной причиной создания брокеров сообщения является осложнение инфраструктуры из-за более распределенной архитектуры приложений, которая начала набирать популярность за счёт своей отказоустойчивости. Из этого следует также и то, что части системы должны как-то общаться между собой, а это может приводить к проблемам работы системы, например: повышение времени обработки запроса. В связи c эти и появилась нужны в части системы, которая бы являлась промежуточным слоем между другими частями.

В дальнейшем, при развитии брокеров сообщений они также стали обеспечивать следующие функции:

* Увеличение производительности системы в целом за счет асинхронности.
* Обеспечение надёжности доставки сообщений: брокеры обеспечивают механизмы многократной отправки сообщений в тот же момент или через определённое время. Кроме того, обеспечивается соответствующая маршрутизация сообщений, которые не были доставлены.

**Описание и особенности**

Kafka — гибрид распределённой базы данных и брокера сообщений с возможностью горизонтального масштабирования. Kafka собирает у приложений данные, хранит их в своем распределённом хранилище, группируя по топикам, и отдаёт компонентам приложения по подписке. При этом сообщения хранятся на различных узлах-брокерах, что обеспечивает высокую доступность и отказоустойчивость.

Топик — это способ группировки потоков сообщений в хранилище по категориям. Сервисы публикуют сообщения определённой категории в топик, а потребители подписываются на топик и читают из него сообщения. Для каждого топика Apache Kafka ведёт лог сообщений, который может быть разбит на несколько разделов. Разделы — это последовательность сообщений топика в порядке поступления.

Сообщения сохраняются в так называемом журнале — долговременной упорядоченной структуре данных. Записи в журнал можно только добавлять, их нельзя ни изменять, ни удалять, а информация считывается слева направо, что гарантирует правильный порядок элементов.

Apache Kafka — это не СУБД в чистом виде, несмотря на то что она обеспечивает атомарность, согласованность, изолированность и долговечность хранимых данных, а также предоставляет возможность избирательного доступа к данным с помощью KSQL — SQL-движка на базе API Kafka Streams. Платформу используют как журнал фиксации и интеграционный центр для множества внешних СУБД и хранилищ.

Основное назначение — это централизованный сбор, обработка, безопасное хранение и передача большого количества сообщений от отделённых друг от друга сервисов. Эта распределённая, горизонтально масштабируемая платформа обычно применяется там, где очень много больших неструктурированных данных:

* Масштабные IoT/IIoT-системы, характеризующиеся архитектурой с множеством датчиков, сенсоров, контроллеров и других конечных устройств.
* Системы аналитики. Например, Kafka используется в компаниях IBM и DataSift в качестве коллектора для мониторинга событий и трекера потребления потоков данных пользователями в режиме реального времени.
* Финансовые системы. Используют финансовые организации: Сбербанк, Тинькофф, Альфа-Банк, ING Bank.
* Социальные сети. В Twitter Kafka — часть инфраструктуры потоковой обработки, а в LinkedIn используется для потоковой передачи данных о деятельности и операционных показателях приложений.
* Системы геопозиционирования. Foursquare — для передачи сообщений между онлайн- и офлайн-системами, а также для интеграции средств мониторинга в свою big data инфраструктуру на базе Hadoop.
* Телеком-операторы. ВымпелКом, МТС, Ростелеком и др.
* Онлайн-игры. Например, Demonware, подразделение Activision Blizzard — для обработки логов пользователей.

Самый простой пример: с помощью Apache Kafka можно собирать логи сеансов от клиентов в потоковом режиме или логи с физических файлов журналов с серверов, а затем помещать их в одном месте, например в HDFS — файловой системе Apache Hadoop. Также сервис позволяет построить конвейер данных, чтобы с помощью алгоритмов машинного обучения извлекать из сырой информации сведения, ценные для бизнеса.

**Отличие от других брокеров**

Kafka часто сравнивают с другим популярным программным брокером сообщений и системой управления очередями — RabbitMQ. Обе системы используются для обмена информацией между приложениями, работают по схеме «издатель — подписчик» и обеспечивают репликацию сообщений. Однако они реализуют принципиально разные модели доставки сообщений: Kafka — pull (получатели сами достают из топика сообщения), а RabbitMQ — push (отправляет сообщения получателям).

Также RabbitMQ удаляет сообщение после доставки, а Kafka хранит его до запланированной очистки журнала. Таким образом, Apache Kafka сохраняет текущее и все прежние состояния системы и может использоваться как достоверный источник исторических данных. Это позволяет множеству потребителей читать одни и те же данные независимо, и такой паттерн удобен, например, в event-driven-системах.

У RabbitMQ очень гибкое управление очередями сообщений (маршрутизация, шаблоны доставки, мониторинг получения), но при большой нагрузке это приводит к снижению производительности. Поэтому для сбора и агрегации событий из множества источников, метрик и логов лучше использовать Apache Kafka, а RabbitMQ подходит для быстрого обмена сообщениями между несколькими сервисами.