Тема 9.2
Приложения и микросервисы для работы в реальном времени

Apache Kafka

Apache Kafka - это быстрая, масштабируемая, долговечная и отказоустойчивая система обмена сообщениями с публикацией и подпиской. Она предназначена для предоставления всех необходимых компонентов управления потоками данных. Kafka динамически обрабатывает записи по мере их появления, предоставляет эффективный ввод-вывод, пакетирование, сжатие и многое другое.

Применения Kafka

- Потоковая обработка
- Отслеживание активности на сайте
- Сбор и мониторинг метрик
- Аналитика в реальном времени
- Захват и ввод данных в Spark / Hadoop
- CRQS, восстановление после ошибок
- Распределенное логирование состояний для вычислений в памяти

Базовые компоненты Kafka

- Кластер Kafka – это кластер, который состоит из Broker машин
- Kafka Brokers - это сервера Kafka, на которых хранятся сообщения
- Producers – клиент, который предоставляет поток сообщений поступающий на кластер
- Consumer – клиенты считывающие информацию с Kafka
- Topic – поток сообщений

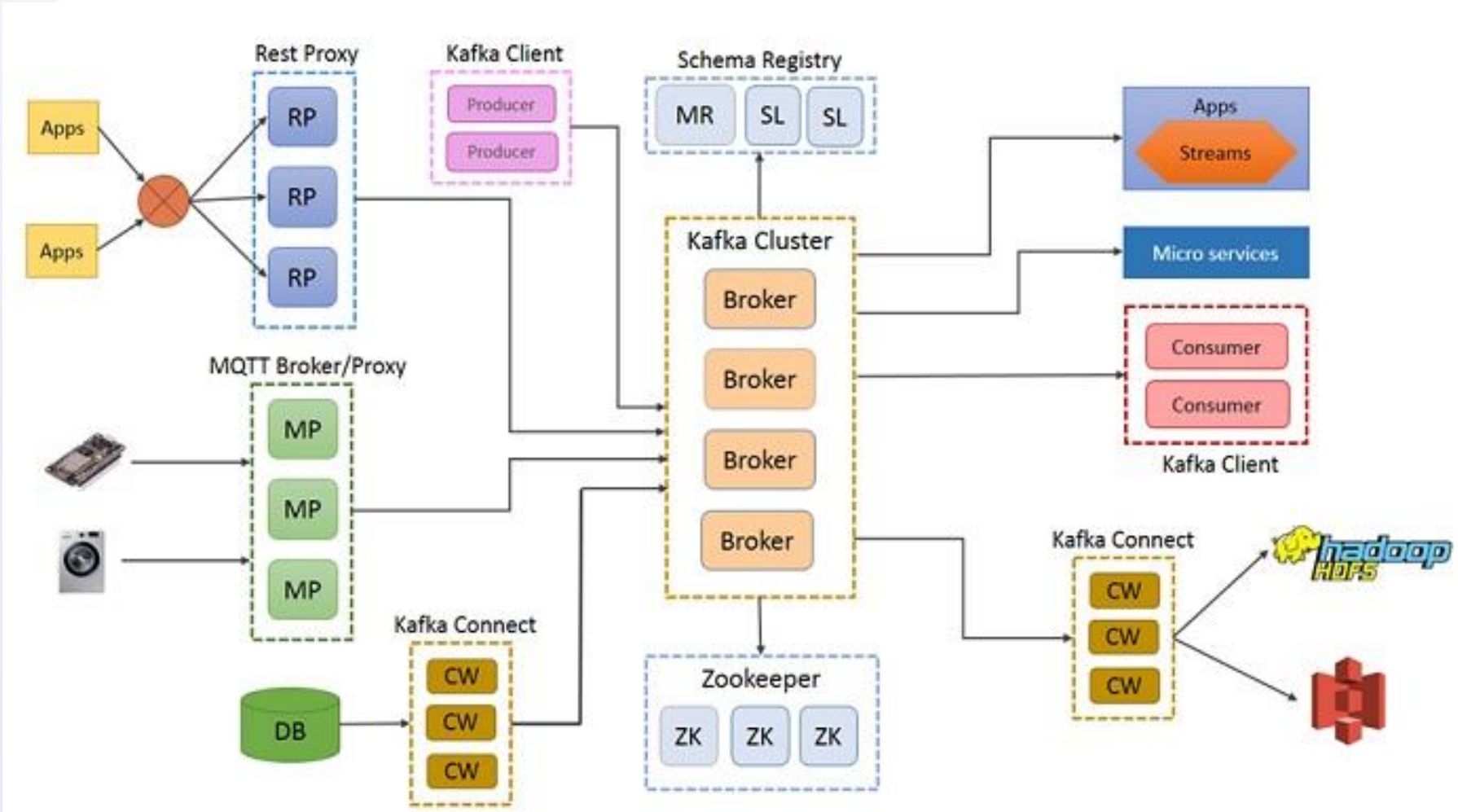
Принцип работы Kafka

Kafka разъединяет потоки данных, позволяя нескольким группам потребителей (consumers) контролировать, где они находятся в тематическом разделе. Производители (producer) не знают о потребителях. Поскольку брокер Kafka делегирует смещение раздела журнала (где потребитель находится в потоке записей) клиентам (потребителям), использование сообщений является гибким.

Принцип работы Kafka

Связь Kafka с клиентами и серверами использует проводной протокол по TCP, который является версионным и документированным. Kafka имеет обратную совместимость со старыми клиентами, поддерживая множество языков. Экосистема Kafka также предоставляет REST-прокси, что упрощает интеграцию через HTTP и JSON. Kafka также поддерживает Avro через Confluent Schema Registry для Kafka. Avro и Schema Registry позволяют клиентам создавать и считывать сложные записи на многих языках программирования и позволяют развивать документацию.

Архитектура Kafka



https://www.learningjournal.guru/article/kafka/kafka-enterprise-architecture/

Kafka Connect

Kafka Connect построен поверх основных компонентов Kafka. Kafka Connect включает в себя готовые к использованию коннекторов Kafka, которые можно использовать для перемещения данных между брокером Kafka и другими приложениями. Kafka Connect также предлагает платформу, которая позволяет разрабатывать собственные пользовательские Source и Sink Connect.

Kafka Clients

Kafka Clients используются в приложениях, которые генерируют и потребляют сообщения. Apache Kafka предоставляет набор API производителей (producers) и потребителей (consumer) для возможности приложениям отправлять и получать непрерывные потоки данных с использованием брокеров Kafka. Данные API доступны на языках: Java, C++, Python, Node.js и Go.

Kafka Brokers

Kafka Broker - это сервер Kafka, работающий в кластере Kafka. Kafka Brokers являются основными компонентами хранения и обмена сообщениями Kafka. Kafka Broker использует ZooKeeper, который является обязательным компонентом в каждом кластере Kafka.

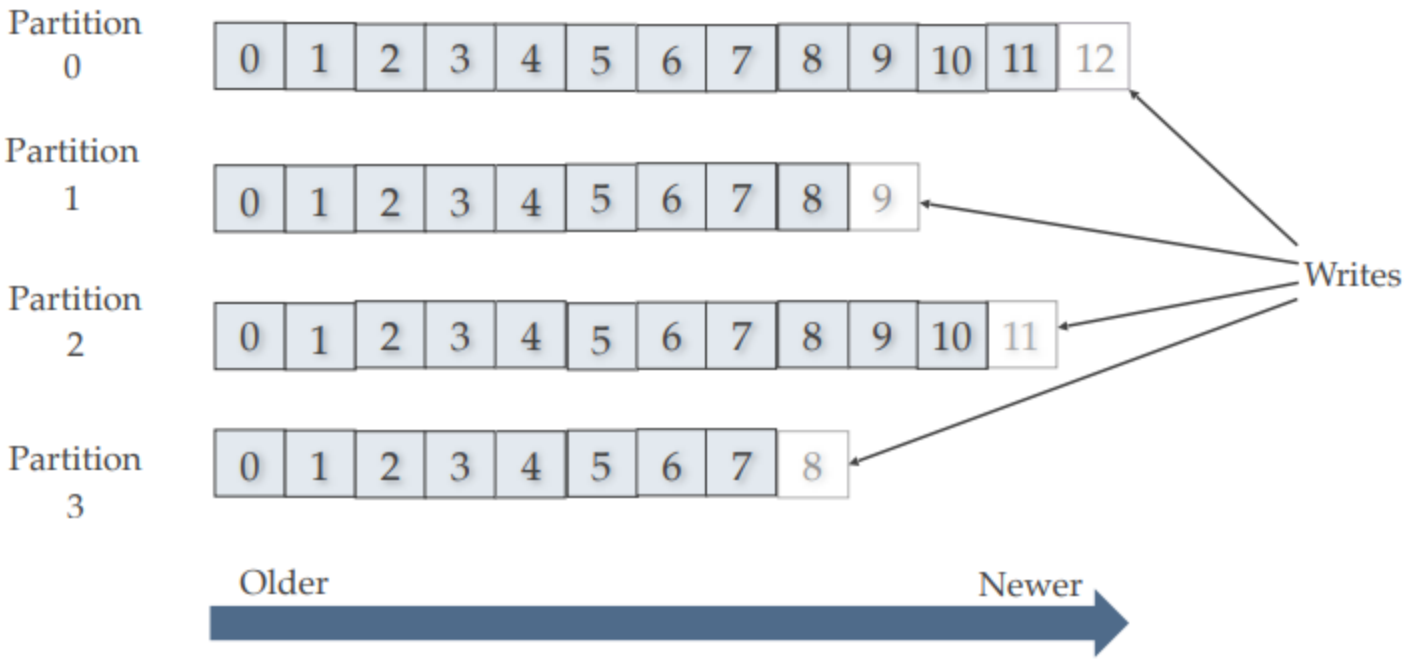
ZooKeeper

ZooKeeper - это централизованный сервис для поддержки информации о конфигурации, наименования, предоставления распределенной синхронизации и обеспечения работоспособности групповых сервисов. ZooKeeper стремится объединить сущность этих различных сервисов в очень простой интерфейс с централизованным сервисом координации. Служба реализует протоколы согласования, группового управления и присутствия.

Partition

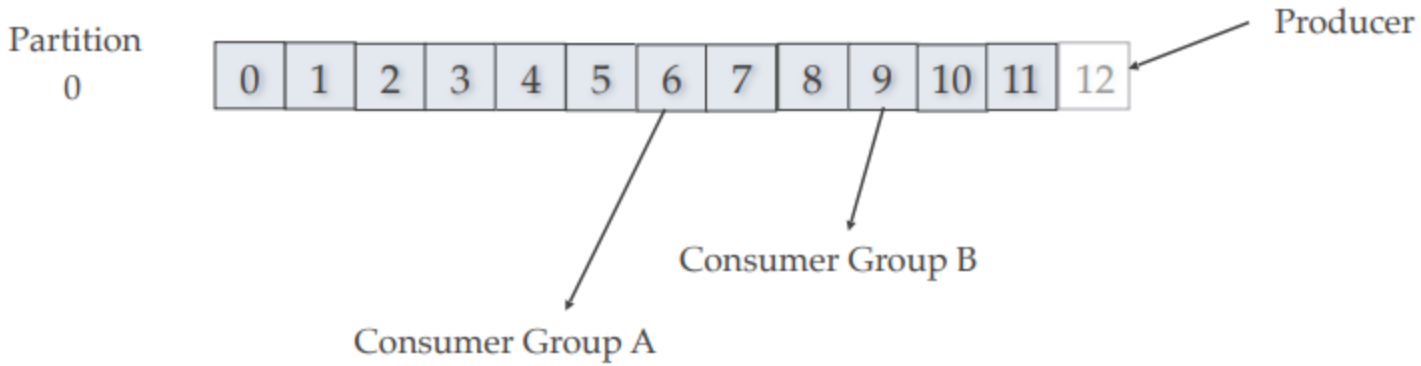
Topic - это поток данных, который может достигать больших размеров. В этом случае у брокера может возникнуть проблема с хранением этих данных, поэтому Topic разбивается на несколько Partitions. Partitions - это набор пронумерованных сообщений (список). Новое сообщение помещается в конце данного списка. Каждый Partition можно хранится только на одной машине для обеспечения высокой производительности и горизонтальной масштабируемости.

Пример Partition Topic



Partition Offset

Каждая группа потребителей отслеживает смещение (offset) в partition от того места, где они прекратили чтение.



В данном примере:
Продюсер пишет со смещение 12.
Потребительская группа A читает со смещения 6.
Потребительская группа B читает со смещения 9.

Partition Replicated

Partition Kafka имеют свойство репликации (replicated) для поддержки отработки отказа. Topic можно разделить на несколько узлов для восстановления после отказа. Topic должна иметь коэффициент репликации больше 1 (2 или 3). Если один из брокеров Kafka выходит из строя, то брокер Kafka, может обслуживать данные (синхронная реплика).

Partition Leader

Kafka выбирает реплики раздела одного брокера в качестве лидера, используя ZooKeeper. Посредник с лидером раздела обрабатывает все операции чтения и записи для раздела. Kafka реплицирует записи в раздел лидера для последователей (пара узел / раздел). Последователь, который находится в синхронизации, называется ISR (синхронная реплика). В случае неудачи лидера раздела Кафка выбирает новый ISR в качестве нового лидера.

Kafka Streams

Это библиотека потоковой обработки, которая позволяет расширять приложения с возможностью использования, обработки и создания новых потоков данных. Библиотека Kafka Streams способна автоматически обрабатывать распределение нагрузки и восстановление после отказа. Для поддержания своего состояния приложения Kafka Streams использует встроенную базу данных Rocks DB.

Kafka Streams

Kafka Stream API решает сложные проблемы с записями не по порядку, агрегируя по нескольким потокам, объединяя данные из нескольких потоков, обеспечивая вычисления с учетом состояния.

Stream Processor

Kafka Streams поддерживает потоковый процессор. Потоковый процессор принимает непрерывные потоки записей из входных разделов, выполняет некоторую обработку, преобразование, агрегирование входных данных и создает один или несколько выходных потоков.

Spark Streaming

Apache Kafka позволяет собирать и агрегировать непрерывные потоки Big Data от разных источников. Однако, чтобы эти большие данные были полезны в прикладном смысле, их необходимо обрабатывать и анализировать. Apache Kafka не предназначена для интеллектуальной обработки данных, поэтому необходимы другие средства Big Data. В таком случае целесообразно использовать Spark Streaming, который будет обрабатывать потоки сообщений Кафка и записывать результат обработки в облачное хранилище, базу данных или распределенную файловую систему типа HDFS.

Spark Streaming

Spark Streaming - это надстройка фреймворка с открытым исходным кодом Apache Spark для обработки потоковых данных. Apache Spark входит в экосистему проектов Hadoop и позволяет осуществлять распределённую обработку неструктурированной и слабоструктурированной информации.

Примеры использования Kafka

- Хранение и распространение в реальном времени опубликованного контента среди различных приложений и систем;
- Масштабирование систем бюджетирования рекламной инфраструктуры в режиме реального времени, повысив точность прогнозов по финансовым расходам;
- Оповещения клиентов в режиме реального времени о финансовых событиях;
- Обеспечение свободного доступа к корпоративным данным для всех сотрудников компании.

Применения Kafka для Smart City

Для организации умного города используются множество инфраструктур, работающее с определенными Big Data. Для взаимодействия общения данных систем и организации управления потоками данных в основе можно использовать Broker Kafka.

Применения Kafka для Smart City