Apache kafka with real-time data streaming

Apache Kafka and real-time data streaming

Abstract— A real-time data streaming is a most popular model of Big data analytics. In the current era, Apache Kafka is one of the finest framework used for real-time data streaming which is scalable, durable, distributed platform with low latency and High throughput. In Kafka, the real-time streaming of data builds a data stream pipelines between systems or applications and designed to serve huge data for large organizations. In this paper, I will be discussing Apache Kafka architecture and real-time data streaming in Kafka.

Introduction—

Real-time data analytics has become a challenge with the continuously rising amount of data, which is generated by systems or applications. Stream processing in big data technology, which enables users to input continuous data stream and detect conditions fast within a minor period. It is also called as real-time analytics and streaming analytics. In the current era, Apache Kafka is most popular architecture used for processing the stream data with aim to provide a unified, high throughput, low-latency platform. It is an open source stream framework for real-time data processing, which is written in Java and Scala. Kafka process streams of records as they occur and is better for applications of two broad classes, one is structuring a real time streaming of data pipelines between systems or applications and another is to build applications streaming for real time that reacts to the record streams. Kafka can do these things as it run as a cluster on any number of servers. The Kafka cluster accumulates streams of records in categories called topics while every record has a key, a value, and a timestamp.

Apache Kafka Architecture—

Kafka is a distributed streaming platform, which is developed by Apache Software Foundation with the aim to make it scalable, fast, reliable & durable. Kafka publishes and subscribe the records, which are in the form of streams and work as a messaging system of an enterprise. It is used in storing streams of records without any fault and helps in processing record streams. Kafka runs like cluster, which is compatible for more than one host. The record consists of key, timestamp and a value. Apache Kafka efficiently processes the real-time, streaming data when implemented along with Apache Storm, Apache HBase and Apache Spark. The major terms of Kafka's architecture

are topics, records, and brokers. Topics consist of stream of records while brokers are responsible for replicating the messages.

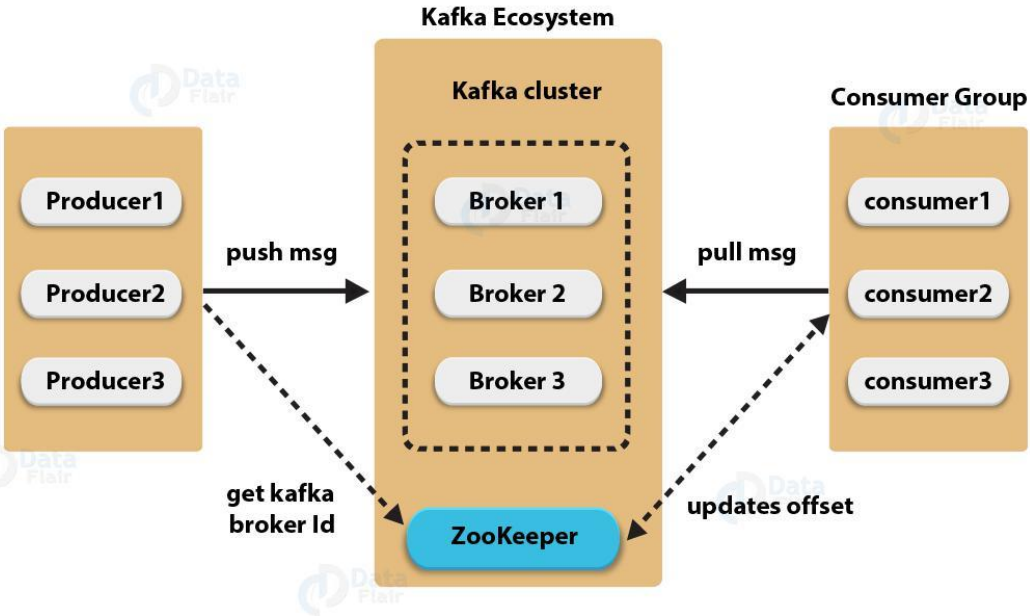

Figure 1.1 Apache Kafka Architecture

Kafka is deployed as a cluster on multiple servers, so Kafka handles it's publish and subscribe messaging system with the help of four APIs i.e., producer API, consumer API, streams API and connector API.

Producer API: It allows application to publish streams of records.

Consumer API: Processing of record streams takes place and the subscription of any number of topics of Kafka.

Streams API: Effectively processing of input streams takes place, which ultimately produces output stream to any number of output topics.

Connector API: Running and making use of consumers and producers, which can be reused again, which helps in connecting Kafka topics to the application or data systems, which is already available.

Kafka continuously receives requests for receiving and sending messages. The data is transferred through broker, which are coordinated and managed by Zookeeper. Kafka with ZooKeeper is used to manage service discovery for Kafka brokers of the Kafka cluster. The broker is nothing more than a Kafka server, which is a meaningful name to refer to it. The notion of broker leads directly to that of a cluster. Indeed, brokers run on a Kafka cluster, which is a group of machines, each executing one instance of Kafka broker.

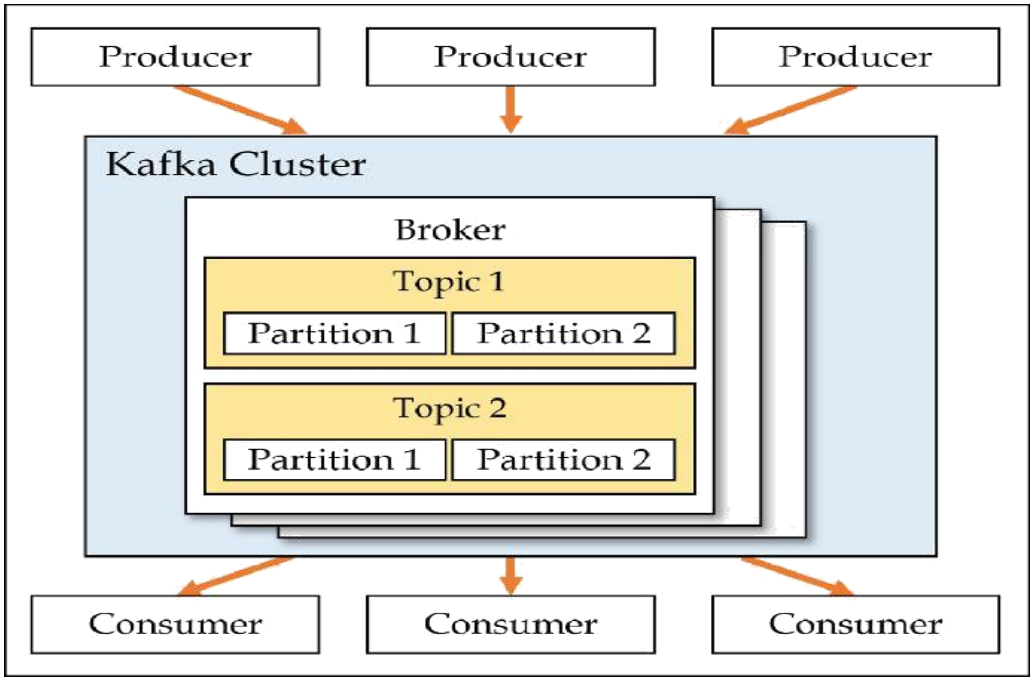

Figure 1.2 Apache Kafka Architecture

Broker: A Kafka server running on a cluster. It manages the messages received and send them to the consumer. Each topic is generally divided into a number of partitions (saved on peer-to-peer nodes, known as brokers).

Cluster: Kafka cluster typically consists of multiple brokers to maintain load balance. Kafka brokers are stateless, so they use ZooKeeper for maintaining their cluster state.

Topics: A topic (similar to a Table in RDMBS, but here each record is immutable) is a category or feed name to which records are published.

Zookeeper: It has a distributed key-value store. It is highly optimized for reads but writes are slower. It stores metadata and handles the mechanics of clustering (distributing updates/configurations, etc). Zookeeper itself is allowing multiple clients to perform simultaneous reads, writes, and acts as a shared configuration service within the system.

Real-time streaming in Kafka—

Kafka streaming platform can handle billions of messages per day. Streams of records are stored in a fault-tolerant durable way. It process streams of records as they occur in real-time with low latency, publish and subscribe to streams of records, similar to a message queue or enterprise messaging system. The main applications of the Kafka streaming is to built real-time streaming data pipelines that reliably get data between systems or

applications and built real-time streaming applications that transform or react to the streams of data.

Kafka allows us to have a huge amount of messages go through a centralized medium and store them without worrying about things like performance or data loss. This means that it is perfect for use as the heart of your system's architecture, acting as a centralized medium that connects different applications. It simplifies the application development by building on the producer and consumer libraries that are in Kafka to leverage the Kafka native capabilities, making it more straightforward and swift. Apache Kafka has a light but powerful streaming library called Kafka Stream to perform data processing.

The main API in Kafka Streaming is a stream processing Domain Specific Language (DSL) offering multiple high-level operators. With the Stream API, it is easier than ever to write business logic, which enriches Kafka topic data for service consumption. Some of the companies that uses Apache Kafka are: LinkedIn, Netflix, yahoo and Uber.

Kafka as messaging System:

Kafka builds on publish-subscribe model with the advantage of a messaging queue system. It achieves this with the use of consumer-groups and message retention by brokers.

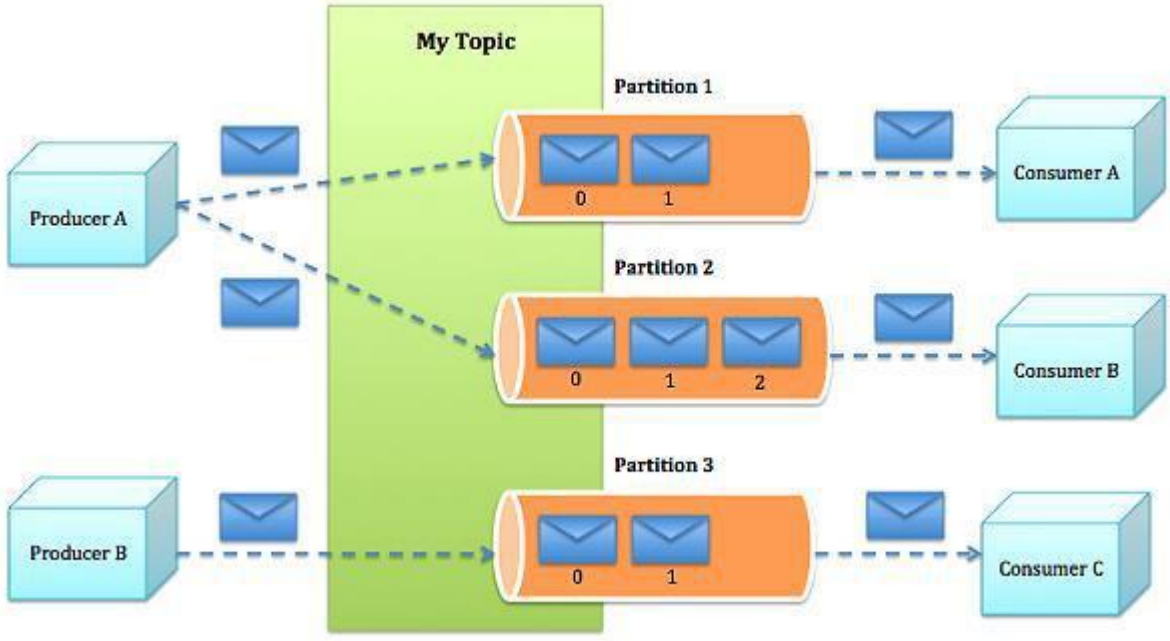

Figure 1.3 Apache Kafka Architecture

Within a consumer-group, for a subscribed topic, only one consumer from the group actually consumes the message from a given partition of the topic,

and so only one of the consumers within a consumer-group reads each particular message. The brokers in their topic partitions, unlike traditional message queues, also retain the messages.

Multiple consumer-groups can read from the same set of topics, and at different times catering to different logical application domains. Thus, Kafka provides both the advantage of high scalability via consumers belonging to the same consumer group and the ability to serve multiple independent downstream applications simultaneously.

Conclusion—

As Kafka is a highly reliable and expandable enterprise messaging system to connect multiple systems, and offers partitioning and high throughput message delivery. Apache Kafka has a light but powerful streaming library called Kafka Stream to perform data processing.

With the Kafka widespread integration into enterprise-level infrastructures, monitoring and Kafka performance at scale has become an increasingly important concern. Kafka monitoring and end-to-end performance requires tracking metrics from brokers, consumer, and producers, in addition to monitoring ZooKeeper, which is used by Kafka for coordination among consumers. Kafka is used in number of different ways with different cases. Number of well-reputed companies adopted Kafka streaming platforms to build mission critical, real-time applications that power their core business from small to large-scale use cases that handle millions of events per second. Nowadays Kafka is indeed a great platform to serve millions within less amount of time and can be used with Apache spark to make real-time data processing more reliable.