Apache Kafka: A High-Throughput Distributed Messaging System

Khin Me Me Thein, Thi Thi Soe Nyunt

University of Computer Studies, Yangon

Abstract

•Apache Kafka is publish-subscribe messaging implemented as a distributed commit log, suitable for both offline and online message consumption.

•It is a messaging system initially developed at LinkedIn for collecting and delivering high volumes of event and log data with low latency.

•Message publishing is a mechanism for connecting various applications with the help of messages that are routed between them, for example, by a message broker such as Kafka.

• It acts as a kind of write-ahead log that records messages to a persistent store and allows subscribers to read and apply these changes to their own stores in a system appropriate time-frame.

•Common subscribers include live services that do message aggregation or other processing streams, as well as Hadoop and data warehousing pipelines which load virtually all feeds for batch-oriented processing.

Introduction

•In today's world, real-time information is continuously getting generated by applications (business, social, or any other type), and this information needs easy ways to be reliably and quickly routed to multiple types of receivers.
•Most of the time, applications that are producing information and applications that are consuming this information are well apart and inaccessible to each other.
•This leads to redevelopment of information producers or consumers to provide an integration point between them.
•Therefore, a mechanism is required for seamless integration of information of producers and consumers to avoid any kind of rewriting of an application at either end .
•In the present big data era, the very first challenge is to collect the data as it is a huge amount of data and the second challenge is to analyze it.
•This analysis typically includes following type of data and much more:
- User behavior data
- Application performance tracing
- Activity data in the form of logs
- Event messages

•Kafka is a solution to the real-time problems of any software solution, that is, to deal with real-time volumes of information and route it to multiple consumers quickly.
• Kafka provides seamless integration between information of producers and consumers without blocking the producers of the information and without letting producers know who the final consumers are.

Related Work

•In a modern data architecture that is built on YARN-enabled Apache Hadoop NextGen MapReduce) Apache Hadoop, Kafka works in combination with Apache Storm, Apache Hbase and Apache Spark for real-time analysis and rendering of streaming data.
• Kafka can message geospatial data from fleet of long-haul trucks or sensor data from heating and cooling equipment in office buildings.
•Kafka is fully supported and included in HDP (*Hortonworks Data Platform)* today.
•Some of the companies that are using Apache Kafka in their respective use cases are as follow:
•**LinkedIn** (www.linkedin.com): Apache Kafka is used at LinkedIn for the streaming of activity data and operational metrics. This data powers various products such as LinkedIn news feed and LinkedIn Today in addition to offline analytics systems such as Hadoop.
•**DataSift** (www.datasift.com/): At DataSift, Kafka is used as a collector for monitoring events and as a tracker of users' consumption of data streams in real time.
•**Twitter** (www.twitter.com/): Twitter uses Kafka as a part of its Storm— a stream-processing infrastructure.
• **Foursquare** (www.foursquare.com/): Kafka powers online-to-online and online-to-offline messaging at Foursquare. It is used to integrate foursquare monitoring and production systems with Foursquare, Hadoop-based offline infrastructures.
• **Square** (www.squareup.com/): Square uses Kafka as a *bus* to move all system events through Square's various datacenters. This includes metrics, logs, custom events, and so on. On the consumer side, it outputs into Splunk, Graphite, or Esper-like real-time alerting.

Kafka Architecture and Design

•Kafka is a distributed, partitioned, replicated commit log service. It maintains feeds of messages in categories called *topics*.
•Processes that publish messages to a Kafka topic are called *producers* and processes that subscribe to topics and process the feed of published messages are called *consumers*.
•Kafka is run as a cluster comprised of one or more servers each of which is called a broker.
•Producers publish messages to Kafka topics, and consumers subscribe to these topics and consume the messages.
•A server in a Kafka cluster is called a broker. For each topic, the Kafka cluster maintains a partition for scaling, parallelism and fault-tolerance. Each partition is an ordered, immutable sequence of messages that is continually appended to a commit log.
•The messages in the partitions are each assigned a sequential id number called the offset.
•Kafka uses ZooKeeper. It serves as the coordination interface between the Kafka broker and consumers.
•At a high level, producers send messages over the network to the Kafka cluster which in turn serves them up to consumers like this in Figure 1:

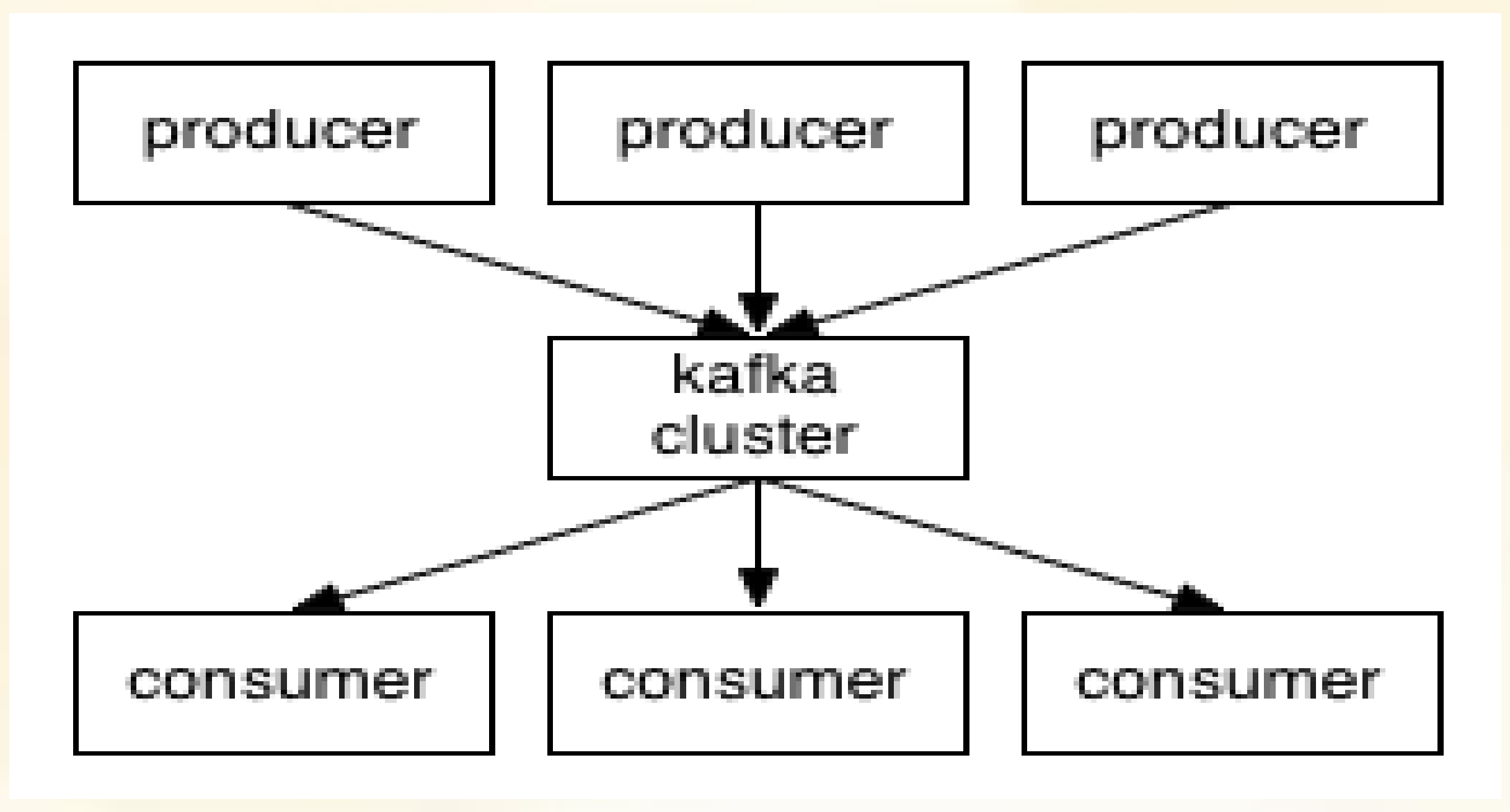

Figure 1: Basic architecture of Kafka

Testing Kafka from Console in Ubuntu

•Some of the commands that are tested in command prompt in Ubuntu by using Kafka 0.8.1.1.tgz

•[root@ubuntu:/opt/kafka-2.8.0-0.8.1.1] #bin/zookeeper-server-start.sh config/zookeeper.properties

•[root@ubuntu:/opt/kafka-2.8.0-0.8.1.1] #bin/kafka-server-start.sh config/server.properties

•[root@ubuntu:/opt/kafka-2.8.0-0.8.1.1] # bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic kafkatopic

•[root@ubuntu:/opt/kafka-2.8.0-0.8.1.1] #bin/kafka-topics.sh --list --zookeeper localhost:2181

•[root@ubuntu:/opt/kafka-2.8.0-0.8.1.1] #bin/kafka-console-producer.sh --broker-list localhost:9092 --topic kafkatopic

•[root@ubuntu:/opt/kafka-2.8.0-0.8.1.1] #bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic kafkatopic --from-beginning

Conclusion and Future Work

•Kafka employs a pull-based consumption model that allows an application to consume data at its own rate and rewind the consumption whenever needed.
• It achieves much higher throughput than conventional messaging systems.
•It also provides integrated distributed support and can scale out.
•Later, Kafka in API for both producer and consumer by using Scala Programming Language'll be tested.