Kafka: a Distributed Messaging System for Log Processing

Summary by: Natasha Kubiak

ECE 531

SUMMER 2022

Introduction

Kafka will be introduced in this summary as a distributed messaging system, developed for collecting and delivering high volumes of log data with low latency. This distributed messaging system is capable of online and offline messaging. Log data is created daily by internet companies which can include: user activity events, pageviews, clicks, likes, sharing, comments, search queries, operational metrics, call latency, errors, system metrics, memory, network or disk utilization, etc. Log analytics is a key component in being able to track and log user engagement and activity. Using this data is key in things such as improving search relevance, recommendations, ad targeting, and security. The real-time influx of all this log data can prove to be a challenge because of the sheer volume of data. In order to be able to find a novel improvement to this message service, Kafka was created. Throughout this summary it will be made clear that Kafka has superior performance and is capable of processing hundreds of gigabytes of data.

Kafka Efficiency

Simple Storage:

An advantage of Kafka is the simple storage layout. Each topic partition corresponds to a log, where a log is a set of segment files. When a message is stored in kafka, it does not contain a message ID, and instead messages are addressed by its logical offset. This is done to avoid maintaining random-access index structures that map out message ID's which can be labor intensive.

Efficient Transfer:

Kafka is cautious when it comes to data transfer. Kafka avoids caching messages in the memory at the Kafka layer, and instead uses the file system page cache. This is an added benefit as it avoids double-buffering and also retains a warm cache. Since Kafka avoids caching messages it has little overhead and makes it more efficient. Kafka is also a multi-subscriber system, which means that a message may be consumed multiple times by multiple consumer applications.

Stateless Broker:

In Kafka, the broker will not maintain the information on consumer habits, instead the consumer will know how much it has consumed. This reduces the load and overhead on the broker.

Distributed Coordination

In a Kafka distributed setting each producer will publish a message to a random partition or a semantically determined partition using a partition key and function. In Kafka there are consumer groups, which consist of one or many consumers that will consume a set of topics all together. When a group has multiple consumers, they will be notified of a broker or consumer change. When a new group is created, consumers can begin either with the smallest or largest offset

Delivery Guarantees:

Kafka guarantees at-least-once delivery. Kafka guarantees that messages from a partition are delivered to the consumer in order, but there is not a guarantee on the ordering of the messages that come from different partitions.

Conclusions

In this summary we went over a system called Kafka, which aids and abets processing large amounts of log data. Kafka focuses on log processing which allows it to achieve a higher throughput than competing systems. The system is distributable, open source, and scalable. Kafka is already being implemented at a wide scale and primarily LinkedIn. It has a lot more development and work to be done and the developers have ideas on how to improve it. Some mentioned research topics around Kafka include: supporting both asynchronous and synchronous replication models, stream processing capability, and increasing durability from machine failures.