Kafka – Review

Yixing Chen, yc3094

The paper *Kafka: a Distributed Messaging System for Log Processing* introduced Kafka – a distributed messaging system for log processing. Kafka is better than the traditional solutions, it combines the benefits of traditional log aggregators and messaging systems. On the one hand, Kafka is distributed and scalable, and offers high throughput. On the other hand, Kafka provides an API similar to a messaging system and allows applications to consume log events in real time.

Kafka is distributed in nature, an Kafka cluster typically consists of multiple brokers. Multiple producers and consumers can publish and retrieve messages at the same time. For each broker, it stores messages from publishers and wait for consumers to process. Kafka is simple and efficient. It uses the simple storage, efficient transfer and stateless broker. Unlike the most message system, the brokers are stateless. Such a design reduces a lot of the complexity and the overhead on the broker. And also make it possible for consumers and rewind back and resume the data deliberately.

The producers and consumers behave in a distributed setting. Each producer can publish a message to either a randomly selected partition or a partition semantically determined by a partitioning key and a partitioning function. The goal is to divide the messages store in the brokers evenly among the consumers, without introducing too much coordination overhead. The first decision is to make a partition with a topic the smallest unit of parallelism, this makes the consuming processing only needs to coordinate when the consumer rebalance the load, and save a lot locking and maintenance overhead. The second decision is to not have a central “master” node, but instead let consumers coordinate among themselves in a decentralized fashion, which saves the necessity of having a master node which would cause the further worries about master failure.

In the test part, the writers use the producer test and consumer test. In the producer performance, we can know that the Kafka (batch 50) has the best behavior, and in the consumer performance, Kafka is better than activemq and rabbitmq.

In conclusion, the writers present a novel system called Kafka for processing huge volume of log data streams. From this paper, we know that Kafka employs a pull-based consumption model that allows an application to consume data at its own rate and rewind the consumption whenever needed. By focusing on log processing applications, Kafka achieves much higher throughput than traditional messaging systems. It also provides integrated distributed support and can scale out. Besides, the Kafka is successfully used in many online or offline application or websites such as Linkedin.