Kafka At Linkedin

Kafka was built with a few key design principles in mind: a simple API for both producers and consumers, designed for high throughput, and a scaled-out architecture from the beginning.

**What is Kafka today at LinkedIn?**

Kafka became a universal pipeline, built around the concept of a commit log, and was built with speed and scalability in mind. Our early Kafka use cases encompassed both the online and offline worlds, both feeding systems that consume events in real-time and those that perform batch analysis. Some common ways we used Kafka included traditional messaging (publishing data from our content feeds and relevance systems to our online serving stores), to provide metrics for system health (used in dashboards and alerts), and to better understand how members use our products (user activity tracking and feeding data to Hadoop grid for analysis and report generation). In 2011 we open sourced Kafka via the Apache Software Foundation, providing the world with a powerful open source solution for managing streams of information.

Today we run several clusters of Kafka brokers for different purposes in each data center. We generally run off the open source Apache Kafka trunk and put out a new internal release a few times a year. However, as our Kafka usage continued to rapidly grow, we had to solve some significant problems to make all of this happen at scale. In the years since we released Kafka as open source, the Engineering team at LinkedIn has developed an entire ecosystem around Kafka.

As pointed out in this blog post by Todd Palino, a key problem for an operation as big as LinkedIn’s is the need for message consistency across multiple datacenters. Many applications, such as those maintaining the indices that enable search, need a view of what is going on in all of our datacenters around the world. At LinkedIn, we use the Kafka MirrorMaker to make copies of of our clusters. There are multiple mirroring pipelines that run both within data centers and across data centers and are laid out to keep network costs and latency to a minimum.

<https://engineering.linkedin.com/kafka/running-kafka-scale>

<https://cwiki.apache.org/confluence/pages/viewpage.action?pageId=27846330>

<https://github.com/linkedin/Burrow>

<https://engineering.linkedin.com/apache-kafka/how-we_re-improving-and-advancing-kafka-linkedin>

<https://engineering.linkedin.com/blog/2016/01/whats-new-samza>

<https://engineering.linkedin.com/espresso/introducing-espresso-linkedins-hot-new-distributed-document-store>

<https://github.com/linkedin/databus>

<https://engineering.linkedin.com/blog/2016/04/kafka-ecosystem-at-linkedin>

**The Kafka ecosystem**

A key innovation that has allowed Kafka to maintain a mostly self-service model has been our integration with Nuage, the self-service portal for online data-infrastructure resources at LinkedIn. This service offers a convenient place for users to manage their topics and associated metadata, abstracting some of the nuances of Kafka’s administrative utilities and making the process easier for topic owners.

Another open source project, Burrow, is our answer to the tricky problem of monitoring Kafka consumer health. It provides a comprehensive view of consumer status, and consumer lag checking as a service without the need to specify thresholds. It monitors committed offsets for all consumers at topic-partition granularity and calculates the status of those consumers on demand.

**Scaling Kafka in a time of rapid growth**

The scale of Kafka at LinkedIn continues to grow in terms of data transferred, clusters and the number of applications it powers. As a result we face unique challenges in terms of reliability, availability and cost of our heavily multi-tenant clusters. In this blog post, Kartik Paramasivam explains the various things that we have improved in Kafka and its ecosystem at LinkedIn to address these issues.

Samza is LinkedIn’s stream processing platform that empowers users to get their stream processing jobs up and running in production as quickly as possible. Unlike other stream processing systems that focus on a very broad feature set, we concentrated on making Samza reliable, performant and operable at the scale of LinkedIn. Now that we have a lot of production workloads up and running, we can turn our attention to broadening the feature set. You can read about our use-cases for relevance, analytics, site-monitoring, security, etc., here.

Kafka’s strong durability, low latency, and recently improved security have enabled us to use Kafka to power a number of newer mission-critical use cases. These include replacing MySQL replication with Kafka-based replication in Espresso, our distributed document store. We also plan to support the next generation of Databus, our source-agnostic distributed change data capture system, using Kafka. We are continuing to invest in Kafka to ensure that our messaging backbone stays healthy as we ask more and more from it.