Apache Kafka

Riccardo Scotti

Università di Bologna
Distributed Software Systems
Prof. Paolo Ciancarini

Agenda

1. Introduction
2. Context
3. Architectural drivers
4. Structure
5. Behavior
6. Rationale
7. Similar or competing middlewares
8. Conclusions

Introduction

Introduction

- Distributed platform for event streaming
- Designed to handle real-time data
- Based on pub/sub
- Originally developed at LinkedIn

Context

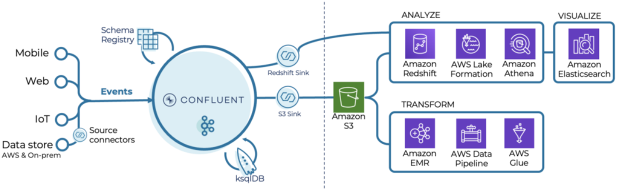
Use cases and scenarios

- What are Kafka's main use cases?

Use cases and scenarios

- What are Kafka's main use cases?
- High flexibility ⇒ many scenarios:
  - Real-time data analytics
  - Aggregated logs
  - Data broker

Real-time data analytics



LinkedIn

- Activity data
- Operational metrics

DataSift

- User consumption tracker

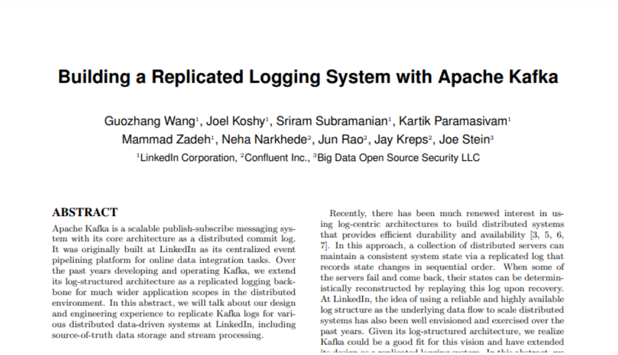
Twitter

- Part of Twitter Storm

Image source: www.kai-waehner.de

Aggregated logs

- Exploits Kafka's distributed commit log architecture
- Already employed by Kafka's development team [6]



Building a Replicated Logging System with Apache Kafka

Guozhang Wang[1], Joel Koshy[1], Sriram Subramanian[1], Kartik Paramasivam[1]
Mammad Zadeh[1], Neha Narkhede[2], Jun Rao[2], Jay Kreps[2], Joe Stein[3]
[1]LinkedIn Corporation, [2]Confluent Inc., [3]Big Data Open Source Security LLC

ABSTRACT

Apache Kafka is a scalable publish-subscribe messaging system with its core architecture as a distributed commit log. It was originally built at LinkedIn as its centralized event pipelining platform for online data integration tasks. Over the past years developing and operating Kafka, we extend its log-structured architecture as a replicated logging backbone for much wider application scopes in the distributed environment. In this abstract, we will talk about our design and engineering experience to replicate Kafka logs for various distributed data-driven systems at LinkedIn, including source-of-truth data storage and stream processing.

Recently, there has been much renewed interest in using log-centric architectures to build distributed systems that provides efficient durability and availability [3, 5, 6, 7]. In this approach, a collection of distributed servers can maintain a consistent system state via a replicated log that records state changes in sequential order. When some of the servers fail and come back, their states can be deterministically reconstructed by replaying this log upon recovery. At LinkedIn, the idea of using a reliable and highly available log structure as the underlying data flow to scale distributed systems has also been well envisioned and exercised over the past years. Given its log-structured architecture, we realize Kafka could be a good fit for this vision and have extended its design as a replicated logging system. In this abstract, we

Data broker

- Publisher/subscriber communication paradigm
- Fine tuned data exchange policies

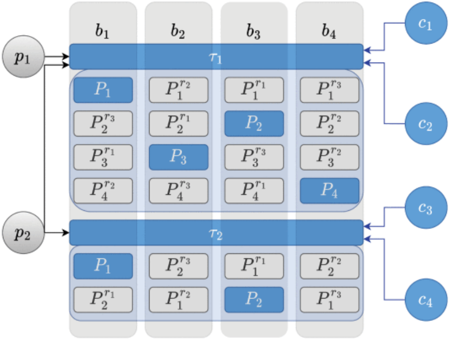


Image source: Raptis and Passarella [4]

Architectural drivers

Architectural drivers

Most of them are a direct consequence of being developed for LinkedIn [1]:

- Low latency & high throughput
- Scalability
- Fault tolerance
- Consistency
- Interoperability

Structure

Components

The basic unit of information in Kafka is called **message**, a key-value pair

- Topics and partitions
- Producers and consumers
- Brokers and clusters
- ZooKeeper (almost completely replaced by KRaft [5])
- Connectors
  - Kafka Connect
  - Source and sink connectors
  - Connect workers

Architectural patterns

Kafka is a pub/sub, distributed commit log [5]

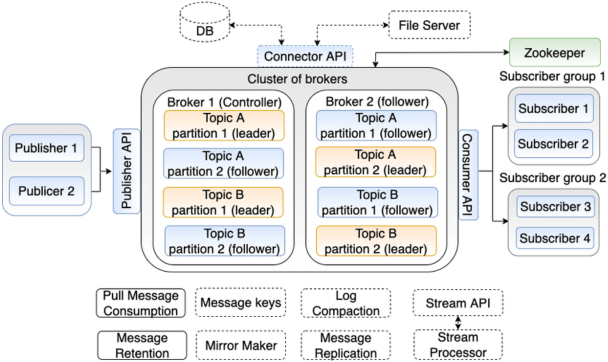


Image source: Lazidis et al. [3]

Behavior

Production

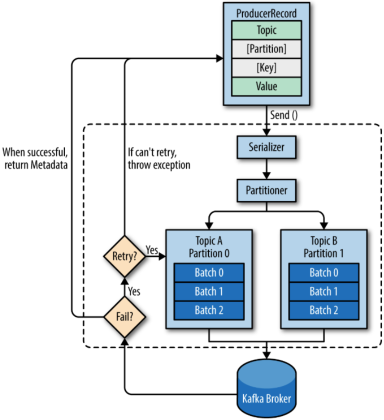


Image source: Shapira et al. [5]

Consumption

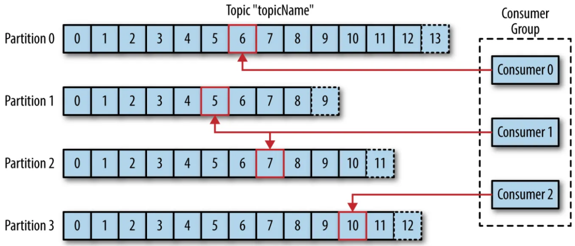


Image source: Shapira et al. [5]

Brokers and clusters

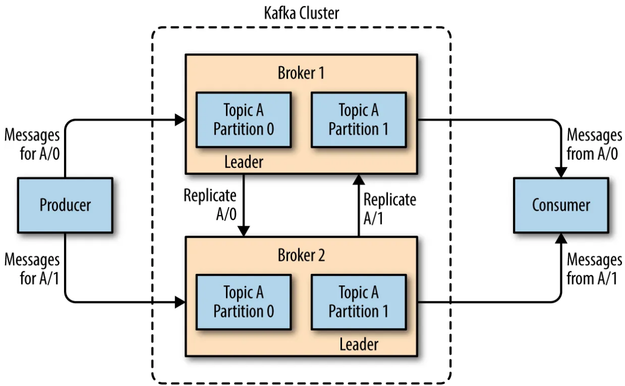


Image source: Shapira et al. [5]

ZooKeeper

- Part of the Kafka environment, but a separate Apache system (introduced one year before Kafka in [2])
- Takes care of coordination
  - Storing metadata
  - Elects controller brokers
  - Assigns partitions
- No fixed role, community decides best practices
- Currently being removed, will be replaced by KRaft

Rationale

Rationale

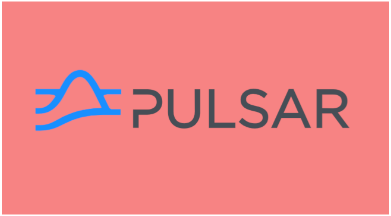
- Publisher/subscriber model
  - Seamless communication between producers and consumer
  - Space decoupling
- Partitions and distributed commit log
  - Horizontal scalability
  - High availability and parallel reads
  - Fault tolerance
- Consumer groups
  - Handling high volumes of data
- Log-based architecture
  - Ensure ordering of events
  - Easily rebuilding state after a failure
- Retention policy
  - Data durability for aggregated logs or monitoring

Similar or competing middlewares

RabbitMQ

- Message broker with load balancing
- Based on point-to-point message queues
- No message retention policy
- Priority messages

Apache Pulsar



- Cloud native, publisher/subscriber platform for messaging, streaming and queueing
- Similar architecture to Kafka, more complex
- BookKeeper and RocksDB to handle long-term persistence
- Less efficient
- Not as mature and widespread as Kafka

NATS

- Lightweight, message-oriented middleware
- Designed for message exchange with low latency and high performances
- Minimal additional functionalities
- Well suited for IoT contexts
- Less versatile than Kafka

Conclusions

Conclusions

To sum up Kafka:

- Pub/sub platform
- Distributed commit log
- Real-time data
- Analytics and brokering
- Availability, scalability and fault tolerance
- Big presence on the market (LinkedIn, Netflix, Twitter)
- Large open-source community

References I

GARG, N.
Apache kafka.
Packt Publishing Birmingham, UK, 2013.

HUNT, P., KONAR, M., JUNQUEIRA, F. P., AND REED, B.
{ZooKeeper}: Wait-free coordination for internet-scale systems.
In *2010 USENIX Annual Technical Conference (USENIX ATC 10)* (2010).

LAZIDIS, A., TSAKOS, K., AND PETRAKIS, E.
Publish-subscribe approaches for the iot and the cloud: Functional and performance evaluation of open-source systems.
Internet of Things 19 (05 2022), 100538.

References II

- Raptis, T. P., and Passarella, A.
  A survey on networked data streaming with apache kafka.
  IEEE Access 11 (2023), 85333–85350.

- Shapira, G., Palino, T., and Sivaram, R.
  Kafka - the definitive guide, 2 ed.
  O'Reilly Media, Sebastopol, CA, Nov. 2021.

- Wang, G., Koshy, J., Subramanian, S., Paramasivam, K., Zadeh, M., Narkhede, N., Rao, J., Kreps, J., and Stein, J.
  Building a replicated logging system with apache kafka.
  Proc. VLDB Endow. 8, 12 (aug 2015), 1654–1655.