Apache Kafka Paper

Aranya Aryaman

April 9, 2025

1 Problem Faced

Modern web-scale applications, such as LinkedIn, required reliable handling of massive volumes of real-time event data (e.g., page views, user activity tracking, log aggregations) across distributed systems. Existing messaging systems struggled with:

- High throughput demands.
- Durability and reliable delivery guarantees.
- Scalability across servers and data centers.
- Support for both real-time and batch consumers.
- Operational simplicity and fault tolerance.

Traditional message brokers were either too heavyweight (e.g., requiring expensive delivery guarantees that limited throughput) or too lightweight (sacrificing durability and consistency).

2 Solution Proposed and Major Architecture Decisions

Kafka was introduced as a distributed messaging system designed for high-throughput, fault-tolerant, publish-subscribe messaging.

Key architecture decisions included:

- **Log-Centric Design**: Kafka treats topics as partitioned, immutable logs where records are appended sequentially.
- **Consumer Pull Model**: Consumers control the pace of reading messages, enabling different processing speeds and batch reads.
- **Persistent Storage**: All published messages are immediately written to disk, leveraging OS page cache for high performance.
- **Horizontal Scalability**: Topics are partitioned and distributed across multiple brokers; producers and consumers can parallelize across partitions.

- **At-Least-Once Delivery**: Simplified delivery guarantees with consumer responsibility for tracking offsets.
- **Built-in Replication**: Kafka ensures fault tolerance by replicating partitions across multiple brokers.
- **Minimal Broker State**: Brokers remain stateless regarding consumer positions, reducing complexity.

3 Industry Impact

Kafka has become a critical infrastructure component across industries for building real-time data pipelines and streaming applications.

Notable impacts include:

- **Widespread Adoption**: Used by major tech companies like LinkedIn, Netflix, Uber, and Airbnb for event streaming.
- **Ecosystem Growth**: Sparked the creation of the Kafka ecosystem (e.g., Kafka Connect, Kafka Streams, ksqlDB).
- **Shift to Event-Driven Architectures**: Enabled modern, loosely coupled microservices architectures.
- **Standard for Data Integration**: Kafka has become a common backbone for connecting heterogeneous data systems.
- **Open Source and Community**: Kafka's success led to the foundation of Confluent and an active open-source community driving continuous innovation.

Reference:
Jay Kreps, Neha Narkhede, Jun Rao. *Kafka: A Distributed Messaging System for Log Processing*, LinkedIn, 2011.