«Why Kafka Data Streaming is the Future»

Patrick P. Bucher

Summary

Apache Kafka is a horizontally scalable, fault-tolerant distributed platform for building real-time data pipelines and streaming applications (Apache Foundation, 2019). Kafka is more than a event stream or messaging system, even though it uses that terminology. Its built-in distributed commit log keeps track of all messages being processed, turning Kafka into an *event ledger*, which allows going back to an earlier state. Kafka is commonly used for search indices, machine learning, buffered ingestion of big data sets, and common *ETL* (extract, transform, load) or *CDC* (change, data, capture) use cases.

A *record*, representing an event or a message, is the basic entity in Kafka. Every record consists of a key, a value and a timestamp. Records are immutable. Multiple cecords are grouped together to *partitions*, which are logically grouped to named *topics*. Kafka ledgers typically consist of multiple nodes. A *producer* sends a record for a topic to a *broker* (a node inside a Kafka cluster), and a *consumer* reads a record from a broker. Messages are *not* pushed to the consumers, but must be polled. Kafka applies *back pressure* to restrain producers from drowning nodes with too many messages. Kafka does not store the records in memory, but on the disk, harnessing fast disk caches and recent developments in storage hardware (solid state drives).

Records are ordered within a partition. Kafka replicates and balances data on the partition level from a *leader* node to multiple *followers*. Within a partition, every record has its unique *offset*. A consumer typically starts reading at offset zero. After an interruption, the consumer can continue at the offset where it left off. Multiple consumers can work together by splitting up the partition at specific offsets. Consumers within a *consumer group* are guaranteed to not process the same record twice. Different message delivery guarantees are being offered. Producers can send messages asynchronously (without delivery guarantee), or expect commit confirmation from either the leader or from the entire quorum (group of followers). Consumers can process messages at least once, at most once, or effectiverly once. (Ward, 2018)

Sources

- James Ward, Introduction to Apache Kafka, Devoxx Converence 2017, YouTube
- Kafka Website (Apache Foundation), kafka.apache.org

Questions

1. Are there companies in Switzerland using Kafka productively?
2. What use cases is Kafka commonly used for?
3. Is Kafka the ultimate interface for interconnecting microservices?
4. Is Kafka suitable to process big media payloads (audio, video, pictures)?
5. Does Kafka work well in a Kubernetes environment with a network file system?