Why Kafka data streaming is the future

https://github.com/christopherchristensen

20. März 2019

Inhaltsverzeichnis

Summary

There are many problems still today when exchanging data within existing IT-Systems and they remains hard ones to solve. Some of these problems are low-latency ingestion of vast amouts of event data, use cases where real-time systems that have great delivery guarantees and do support things such as transactions and message consumption tracking are just overkill (Posta, 2019) or as Christoph Suter showed us where simply too much redundancy gets created when scaling existing databases with further applications. In Christoph Suter's presentation he talks about how Kafka can solve at least some of these problems in some specific use cases.
Apache Kafka is sold as a distributed streaming platform that positions itself as a viable solution for building real-time data pipelines and streaming apps. Its strong suit lies in its horizontal scalability, fault-tolerance, performance (high-throughput with low latency) and durability just to mention a few advantages according to the Apache Foundation. Kafka can be run on one server or load balanced across a distributed network. In both scenarios it is run as a cluster. (Apache Foundation, 2019) Suter showed us an example of an IT-System for insurances where they improved the backend by implementing such a cluster to eradicate their many redundancies spread across many databases plus a datawarehouse with a single source of truth. (Suter, 2019) Kafka consists of four core APIs which are the Producer (publish streams of records topics), the Consumer (subscribe to one or more topics and process them), Streams (consumes input from one or more output topics) and the Connector API (build and run reusable producers or consumers that connect Kafka topics to applications or data systems). Records get topics, that are categories and always multi-subscriber, assigned to them. These topics get stored in a partitioned log which the Kafka cluster persists (True Storage). This of the difference between Kafka and other services that don't always offer durable data persistence. (Suter, 2019)
Suter did, putting all of the great benefits of Kafka aside, mention some dangers or pitfalls within Kafka. One of the main pitfalls is that once you decide to implement a Kafka cluster you are immediate confronted with much higher levels of complexity. Where Kafka is maybe great for scaling larger service or applications it doesn't make sense in very small application that maybe never need any scalability. He mentioned that the complexity lies in the components, the endless possibilities of configurations of these components, the scaling which brings parallelism such as dealing with many message on different nodes and transaction that tend to be more complex than usual transaction in other services. Another issue can be Data Governance where in a case of replication you have to assign which message has priority over another which can also lead to higher complexity. (Suter, 2019)

Sources

- Suter, C. (2019, March). Live Talk at HSLU.
- Posta, C. (2019), What is Apache Kafka? Why is it so popular? Should you use it TechBeacon.com
- Kafka Website (Apache Foundation), `kafka.apache.org`

Questions & Answers

1. Are there any serious downsides to not having individual message IDs?

- Messages can land in dead letter storage which means harder error handling and lost messages

2. When should you use Kafka and when not?

- Use:
  - For streaming applications
  - Generally for large scale products

  - For real-time data pipelines
  - An example given by Christoph: Credit Card Defrauding

- Dont use:

  - In a very simple application where no scaling is needed (overkill)
  - Where synchronized messagin is required, since Kafka is asynchronous

3. Are there any dangers / pitfalls to Kafka?

- Is a single point of failure which could also be a risk since the data is all available there as a single source of truth. Although Christoph insisted on very secure access rights and a strong system that uses private and public keys with many more additional security features. On top of the actual data there is also an abstraction layer so that not all data is directly visible and accessible.
- "Everytime you see an Apache Kafka component you should be scare" - Christoph Suter, speeking on the complexity of Kafka
- Timestamps are used for the ordering of the records, therefore using slow hardware could cause timestamp issue
- You obviously need hardware and that means that if your hardware is slow Kafka is slow
- Again, complexity and data governance
- No Transaction Support
- No Error Handling (messages stored in dead letter storage)
- No Slim Broker

4. What are the main advantages of Apache Kafka?

- High-Throughput
- Low Latency
- Fault-Tolerant
- Durability
- Scalability