Apache Kafka...

...”a system optimized for writing”

Bernhard Hopfenmüller

18. Oktober 2018
Bernhard Hopfenmüller
IT Consultant @ ATIX AG

IRC: Fobhep
github.com/Fobhep
The Linux & Open Source Company
Garching @ München

over 15 years
datacenter automation, Linux Consulting, Engineering, Support, Training
What is the relation between Kafka, the writer, and Apache Kafka, the distributed messaging system?

Jay Kreps: I thought that since Kafka was a system optimized for writing using a writer’s name would make sense. I had taken a lot of lit classes in colleague and liked Franz Kafka. Plus the name sounded cool for an OS project.
- developed by LinkedIn, Open Source since 2011
- 2014 foundation of Confluent
Why do we need a messaging system?
Why do we need a messaging system?

- Challenge 1: Sender not available
- Challenge 2: Sending too much (DoS)
- Challenge 3: Receiver crash upon processing
Supermarket vs Television

Supermarket: Wait until it's your turn

Television: Choose what you want to receive

#atix #osad2018
Kafka-Basic structure
Use Cases

- Messaging (ActiveMQ or RabbitMQ)
- Website Activity Tracking
- Metrics
- Log Aggregation
- Stream Processing
- Apache Storm and Apache Samza.
- Commit Log
• core component of Kafka
• is filled by producer
• consists of one or more partitions
Topics II

- producer can choose partition
- partition has running offset
- message is identified by offset
Topics III

- Messages are stored physically!
- Key-value principle
- Clean-Up policies:
Topics IV

Clean-Up policies:

- default: Retention-time (delete old data after $x$ days)
- Retention-size (delete old data if data memory $> x$)
Clean-Up policies:

- default: Retention-time (delete old data after \( x \) days)
- Retention-size (delete old data if data memory > \( x \))
- Log-Compaction (replace old value to key with new)
Topic consumption

- topics are pulled! (no DoS)
- any existing data can be pulled
Consumer Groups

- parallelism allows high throughput
- never more consumers than partitions
- Kafka features exactly-once-semantics!
Wait but who knows what’s read?

- Consumer commit their offset
- Upon failure re-processing possible
Replication implemented on partition level

Source [3]
Did somebody hear my message?

Producer decides if message was successfully sent
Configuration possibilities:

- as soon as sent
- as soon as received by first broker
- as soon as desired number of replica exist
Broker and ZooKeeper

- Brokers are stateless!
- Which Broker is alive?
- Broker communication?
- → ZooKeeper!
ZooKeeper

- distributed, hierarchical file system
- management of znodes()
- HA via ensemble (=ZooKeeper cluster)
Talk to Kafka - Kafka Connect

- I/O for Kafka
- Connect with external systems
- Source → Kafka Topic
- Kafka Topic → Sink
Talk to Kafka - Schema Registry

- define standards
- version and store them

source: confluent

#atix #osad2018
Who likes Kafka?

- zalando - microservices
- Cisco Systems - security
- Airbnb - event pipeline
- Netflix (Monitoring!)
- The New York Times (Kafka as data storage! Super awesome blog post) [5][6]
- PayPal
- Spotify
- Twitter
- Uber (Kafka = Backbone!!!)
- https://kafka.apache.org/powered-by
Sources

Kafka vs MQ

- Kafka has no P2P model!
- Messages are Persistent!
- Topic Partitioning!
- Message Sequencing: for one partition (send order=received order)
- Message reading: Choose where to read, Rewind, no FIFO!
- Loadbalancing: automatic distribution easier with metadata
- HA and failover implemented very easily