Distributed systems for stream processing

Apache Kafka and Spark Structured Streaming

Alena Hall
Alena Hall - twitter lenadroid

- Large-scale data processing
- Distributed Systems
- Functional Programming
- Data Science & Machine Learning
Ever-increasing Data
Direct result of some action
Produced as a side effect
Continuous indicators
Reaction

**urgent**  **not-so-urgent**  **flexible**
Reaction

urgent    not-so-urgent    flexible

real-time    near-real-time    batch
            ~ sub milliseconds    ~ minutes, hours, days, weeks
Event Ingestion

Processing & Reaction

- real-time
- micro-batch
- batch
Data Producers and Consumers

Are data workflows flexible enough?
Challenges

Simplicity. Scalability. Reliability
Meet Apache Kafka
Apache Kafka is an open-source stream-processing software platform developed by the Apache Software Foundation written in Scala and Java.
Inside of a Kafka Topic

Partition 0

0 1 2 3 4 5

Partition 1

0 1 2 3 4

Partition 2

0 1 2 3 4 5 6 7 8

Old ------ New
Kafka Topic Partition

Producers

writes

reads

Consumer X

Consumer Y

lenadroid
Kafka Producers and Consumers
Systems for stream processing

Kafka Streams

Spark

Storm

Flink
Meet Apache Spark
Apache Spark is a unified analytics engine for large-scale data processing: batch, streaming, machine learning, graph computation with access to data in hundreds of sources.
Spark SQL and batch processing
Stream processing with Spark Streaming and Structured Streaming
* Continuous processing
Machine Learning with Mllib
Graph computations with GraphX

* Experimental
How does Spark work?
Spark application (Driver) 

Cluster Manager 

Spark workers have executors of tasks
Apache Kafka + Apache Spark
Existing infrastructure and resources

- Kafka cluster (HDInsight or other)
- Spark cluster (Azure Databricks workspace, or other)
- Peered Kafka and Spark Virtual Networks
- Sources of data: Twitter & Slack & Nomics APIs
Databricks: Interactive Environment
Processing crypto currency trading data

Example
markets | exchanges | trades

ETH / BTC
BTC / USDT

Bitfinex
Binance

{ ..., ... }
markets  exchanges  trades

```json
{
    "volume":"5",
    "price":"3.0871",
    "id":"123456",
    "timestamp":"2018-07-17T17:00:00.00Z"
}
```
Indicators to watch and act on

✓ Price spikes (all-time high, all-time low)
✓ Significant changes in price or volume of trades
✓ Profitability of potential trade at current moment
✓ Price or volume of trades crossing given threshold during the past X minutes
✓ More
Getting trades data from API

- Market and exchange data
- Trades data for given market and base/quote currencies
- Sending data to Kafka
Processing trades

✓ Consuming data coming from Kafka topics
✓ Watching relevant indicators
More examples?

Processing streams of events from multiple sources with Apache Kafka and Spark
Data sources: external, internal, ...

- Big number of data sources
- Most of the data sources are independent
- Sources of data used for many processing tasks & end-goals
Feedback from Slack

✓ Sending messages to Slack
Listener for new Slack messages

- Messages under specific channels
- Focused on a particular topic
- Sent to a specific Kafka topic
Receiving events in Kafka topic

- Spark consumer for Kafka topics
- Sending only topic related messages to Kafka
Sending Twitter feedback to Kafka

✓ Getting latest tweets about specific topic to Kafka
✓ Receiving those events from Kafka in Spark
Analyzing feedback in real-time

- Kafka is receiving events from many sources
- Sentiment analysis on incoming Kafka events
- Sentiment $\leq 0.3 \rightarrow \#negative-feedback$ for review
- Sentiment $\geq 0.9 \rightarrow \#positive-feedback$ channel
Kafka + Spark = Reliable, scalable, durable event ingestion and efficient stream processing
Bonus Topics
Continuous Processing

\texttt{trigger(Trigger.\texttt{Continuous("1 second")})}

Low (~1 ms) end-to-end latency

At-least-once fault-tolerance guarantees

Not nearly all operations are supported yet

No automatic retries of failed tasks

Needs enough cluster power to operate
Micro-batch

Check-pointing epoch

When event is at source

When event is processed to sink

Every X seconds

Every X seconds

Every X seconds
Continuous

Check-pointing epoch

When event is at source

Every X seconds

When event is processed to sink

Every X seconds

~ 1ms

When event is processed to sink

Every X seconds
Kafka API for Event Hubs

aka.ms/eventhubs-kafka
Confluent/Kafka **Operator**

and other **Operators** ...
Thank you!

Apache Kafka: aka.ms/apache-kafka
Apache Spark: aka.ms/apache-spark
Event stream processing architecture on Azure with Apache Kafka and Spark: aka.ms/kafka-spark-azure and aka.ms/oscon-18
Create HDInsight Kafka cluster using ARM: aka.ms/hdi-kafka-arm
Create Kafka topics in HDInsight: aka.ms/hdi-kafka-topic
Alena Hall - 🦃 lenadroid

- Works on Azure at Microsoft
- Lives in Seattle
- F# Software Foundation Board of Trustees
- Organizes @ML4ALL 🐙
- Program Committee for Lambda World
- Has a channel: YouTube/c/AlenaHall