

Data Management in Large-Scale Distributed Systems

Stream processing

Thomas Ropars

`thomas.ropars@univ-grenoble-alpes.fr`

`http://tropars.github.io/`

2020

References

- The lecture notes of V. Leroy
- Designing Data-Intensive Applications by Martin Kleppmann
 - ▶ Chapter 11

In this lecture

- Introduction to Stream Processing
- Transmitting data streams
- An overview of Stream Processing engines

Agenda

Introduction

Transmitting event streams: Message brokers

Stream processing

The lambda architecture

An unbounded dataset

Batch processing

- Data are stored in files
- Process the whole data at once
- Examples: Hadoop MapReduce, Spark, etc.

In many use-cases, new data are generated continuously

- Data from sensors
- Data from social networks
- Web traffic
- Etc.

Near real-time processing

In many cases, data should be processed as *early* as possible:

- Detecting fraudulent behavior
- Identifying malfunctioning systems
- Monitoring trends (social networks, system load)

Adapting batch processing systems?

Near real-time processing

In many cases, data should be processed as *early* as possible:

- Detecting fraudulent behavior
- Identifying malfunctioning systems
- Monitoring trends (social networks, system load)

Adapting batch processing systems?

- Processing all the data of the day at the end of each day

Near real-time processing

In many cases, data should be processed as *early* as possible:

- Detecting fraudulent behavior
- Identifying malfunctioning systems
- Monitoring trends (social networks, system load)

Adapting batch processing systems?

- Processing all the data of the day at the end of each day
 - ▶ High latency 😞
- We need to process data more *frequently*
 - ▶ **This is what stream processing engines do**

Stream vs Batch processing

Batch processing

- Good for analyzing a static dataset
- Focuses on throughput
- Allows running complex analysis requiring multiple iterations on the data

Stream processing

- Good to analyze *live* data
- Continuously updates results based on new data
- Focuses on latency (between data production and update of the results)
- Processes data once

Stream processing computation

Stream analytics

- Measuring event rates
- Computing rolling statistics (average, histograms, etc)
- Comparing statistics to previous values (detecting trends)
- Sampling data
- Filtering data
- Applying basic machine learning algorithms

Questions related to stream processing

- How to transmit data from the producers to the consumers?
 - ▶ With multiple producers and/or consumers?
- How to process events in a distributed way?
- How to deal with failures?
- How to reason about time?
- How to maintain a state over time?

Agenda

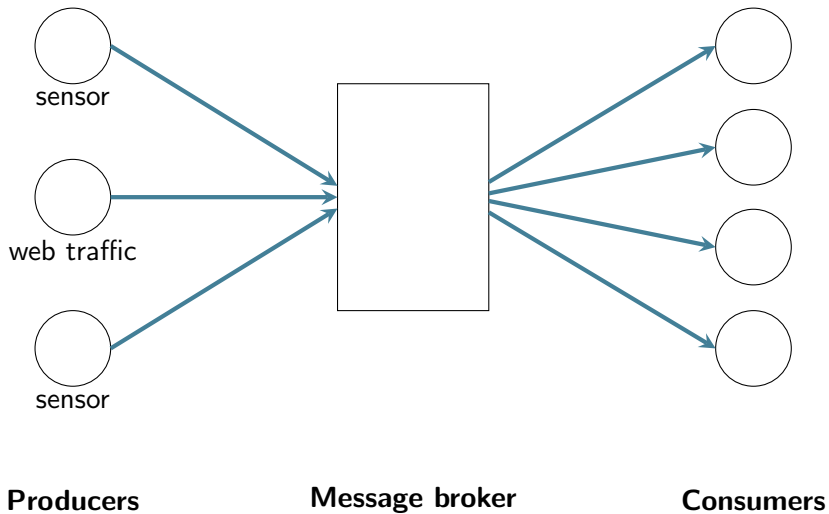
Introduction

Transmitting event streams: Message brokers

Stream processing

The lambda architecture

Motivation



Challenges

Routing messages

- Some consumers are only interested in some messages
- Some messages are useful for multiple consumers

Performance

- Amount of produced data might be huge
- Data might be produced faster than they are processed

Fault tolerance

- Clients might connect/disconnect at any time

Log-based message broker

Main principles

- Maintain a log of all the messages received
 - ▶ Append-only sequence of records on disk
- Each record is identified with a sequence number
- The offset of each client in the log can be stored

Existing systems

- Apache Kafka
- Amazon Kinesis Data Streams

Kafka

<https://kafka.apache.org/>



- Originally developed at LinkedIn
- Open-source
- Used by many companies

Kafka main principles

A partitioned log

The log is divided into multiple partitions

- Each partition has its own monotonically increasing sequence number
- Partitions can be hosted on different machines

Advantages of logs

- Old records can be replayed
 - ▶ Clients can arrive late or disconnect
 - ▶ Question: How to do garbage collection?
 - Note that filling a 6TB disk takes half a day
- Data are buffered in the log
 - ▶ Deal with the case where the consumers are slower than the producers

Kafka communication abstractions

Topics

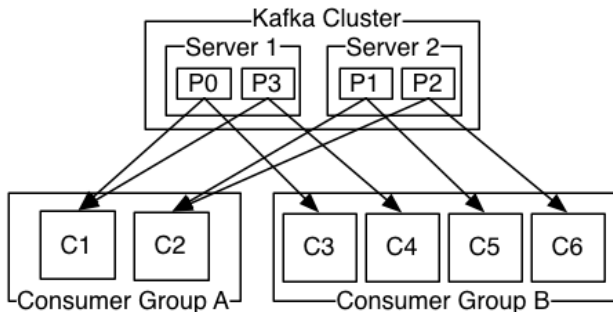
- Partitions are grouped into logical topics
- A producer can send to multiple topics

Consumer groups

- Consumers gather themselves into consumer groups
- A consumer group can register to one or several topics
- Multiple groups can register to the same topic
- Each record is delivered to one consumer in each registered group
 - ▶ Records from one partition are sent to exactly one consumer in a group
 - ▶ Total-order delivery is only ensured inside partitions

Kafka consumer groups

source <https://kafka.apache.org/documentation/#gettingStarted>



2 types of communication patterns

- Load balancing
- Broadcasting

Kafka fault tolerance

Data availability

- A Kafka cluster spans multiple nodes
- Partitions are replicated on multiple nodes

Dealing with consumer disconnections/failures

- Offset of the consumer in the log partition is recorded permanently
- The same/another consumer can start processing records from this point
- Provided delivery semantics:
 - ▶ *At-least-once*
 - ▶ *At-most-once*

Agenda

Introduction

Transmitting event streams: Message brokers

Stream processing

The lambda architecture

Stream processing engines

Description

- A set of transformations is applied to a stream of records
 - ▶ A program is a graph of transformations (Directed acyclic graph)
 - ▶ Transformations are the same operations as in batch processing systems

Examples

- Storm
- Flink
- Samza
- Spark streaming

Graph of transformations (Flink)

source <https://ci.apache.org/projects/flink/flink-docs-release-1.6/concepts/programming-model.html>

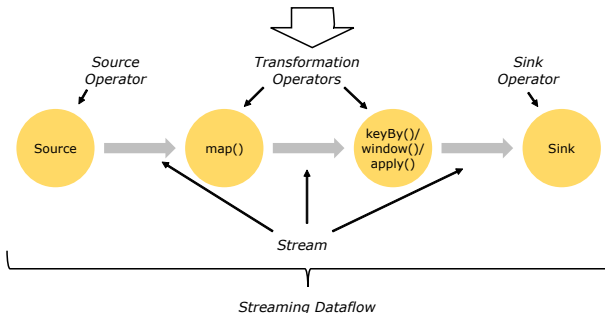
```
DataStream<String> lines = env.addSource(  
    new FlinkKafkaConsumer<>(...));  
  
DataStream<Event> events = lines.map((line) -> parse(line));  
  
DataStream<Statistics> stats = events  
    .keyBy("id")  
    .timeWindow(Time.seconds(10))  
    .apply(new MyWindowAggregationFunction());  
  
stats.addSink(new RollingSink(path));
```

Source

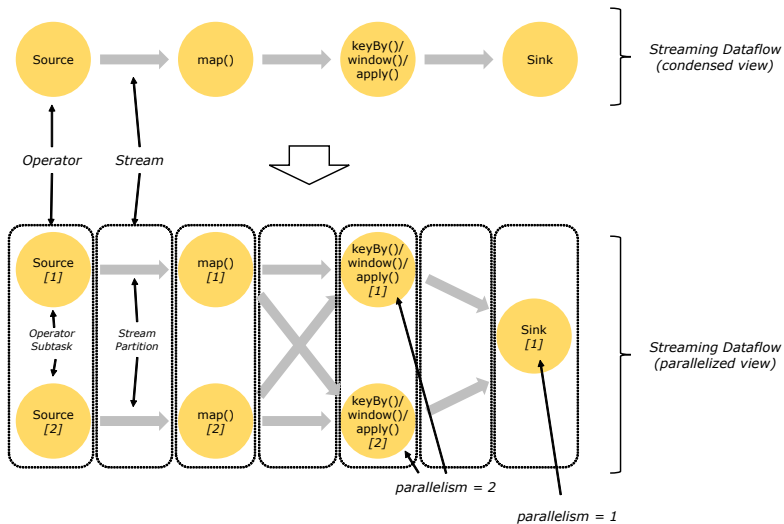
Transformation

Transformation

Sink



Parallel Dataflow (Flink)



About the notion of time

To run computations on a continuous stream, it has to be split into windows.

Size of the windows

- 1 event: Each event is processed separately (Storm)
- Windows limits is defined based on:
 - ▶ Amount of data received
 - ▶ Time
 - ▶ Activity (concept of sessions)

2 reference times co-exists in the system

- **Event time**: time at which the events happened
- **Processing time**: time at which the events are processed
 - ▶ Most systems build windows based on the processing time

About the notion of time

Window type

- **Tumbling window**: Fixed size window, each event belong to one window
- **Hopping window**: Fixed size window, windows overlap
 - ▶ $\text{hop size} = \text{time between the generation of two windows}$
 - ▶ $\text{hop size} < \text{window size}$
- **Sliding window**: Fixed size window
 - ▶ A new window is considered at each time step
- **Session window**: No fix size, group together events that happened closely together in time

Spark Streaming

Based on micro-batches

- The data stream is divided into micro-batches
 - ▶ Tumbling windows
 - ▶ Typically 1 to 4 seconds
- Each micro-batch is a **RDD**
- Multiple receivers can be created to manipulate multiple data streams in parallel
 - ▶ The receiver tasks are distributed over the workers



Agenda

Introduction

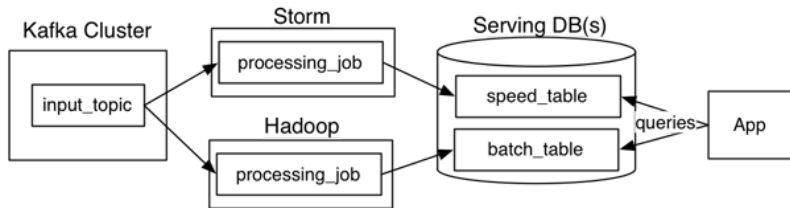
Transmitting event streams: Message brokers

Stream processing

The lambda architecture

The Lambda architecture

source <https://www.oreilly.com/ideas/questioning-the-lambda-architecture>



The Lambda architecture

Motivations

- Combination of batch processing and stream processing in a single architecture
 - ▶ Stream processing allows building fast (approximate) views of the data
 - ▶ Batch processing is used for more complex (and accurate) data analysis

Limits

Architecture becoming less popular (lambda-less architecture)

- Maintaining two code bases is costly
- Processing engines start allowing doing both (Spark, Flink)
 - ▶ Stream processing engines are becoming more mature, they allow running more complex computations
 - ▶ Log-based message brokers allow processing the same record multiple times

Additional references

Mandatory reading

- <https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101>, T. Akidau, 2015.

Suggested reading

- *Apache Flink: Stream and Batch Processing in a Single Engine.*, P. Carbone et al., IEEE, 2015.
- <https://www.oreilly.com/ideas/questioning-the-lambda-architecture>, J. Kreps, 2014.
- <https://data-artisans.com/blog/batch-is-a-special-case-of-streaming>