THE DATA SCIENCE LAB
Introduction to Data Stream Processing

COM 490 – Spring 2022

Week 8

THIS CLASS WILL BE RECORDED

Agenda Spring 2022

M1	23.02	• General introduction to Data Science • Python 3.x, Numpy, Scipy, Pandas, Matplotlib, Scikit-learn
	02.03	• Collaborative Data Science • Jupyter notebooks, Docker, Git(lab), interactive visualization
M2	09.03	• General introduction to big data, best practices and guidelines • HDFS, Hive, Loading and querying data, assessed project 1
	16.03	• Data wrangling with Hadoop • Parquet, ORC, HBase
M3	23.03	• Introduction to the Spark Runtime architecture • Python on Spark, Basic RDD manipulation; **assessed project 1 due**
	30.03	• Spark data Frames • Basic data analytics with Spark Data Frames
	06.04	• Advanced Spark, optimizations and partitioning • Data partitioning; **assessed project 2 is due**
M4	13.04	•Introduction to data stream processing •Stream processing with Apache Kafka
	20.04	•Spring break 15.04 – 25.04
	27.04	•Advanced data stream processing •Spark stream, data stream windows, Kafka-Spark; **assessed project 3 is due**
	04.05	•Analytics on data at rest and data in motion •**Final project is announced**
Rev	11.05	•Putting it all together - building scalable applications with real data •**Assessed project 4 is due**
Fin	18.05	•Final project: tips and hints
	25.05	•Final project: tips and hints
	01.06	•**Final project is due Sunday 29.05** : code and a short video presentation •Oral sessions on 01.06

*Details on Moodle

Stream Processing Module

- Objectives
  - Review concepts of stream processing
  - Experiment with typical tools for
    - Data ingestion and processing
- Week 8
  - Concepts
  - Experiments
- Week 9
  - Advanced topics
    - Operations on streaming data (joins)
    - Time constraints
- Week 10
  - Analytics on data at rest and data in motion

Why Stream Processing?

- **Reminder from module 2 (Big Data)**
  - Batch vs Stream

- Can wait until all information is available for a more accurate answer? batch
  - AKA: Data at rest
  - Operates on finite size data sets, and terminate when all data has been processed
- You want an updated answer as more information becomes available? streams
  - AKA: Data in motion, or Fast data
  - Continuous computations that never stop, process “infinite stream” of data on the fly
    - Designed to keep size of in-memory state bounded, regardless of how much data is processed
  - Update the answer as more data becomes available
    - Operate on small time windows

Why Stream Processing?

- **Relevance (vs batch)**
  - Insight more valuable shortly after events happen
    - (Near) real-time: from milliseconds to seconds, or minutes
  - It allows faster reaction
    - Detecting patterns, setting alerts
  - Some data is naturally unbounded (e.g. sensor data)
  - Resource constraints (storage and compute)
    - process large large volumes of data arriving at high velocities
    - Retain only what is useful
  - Continuous processing

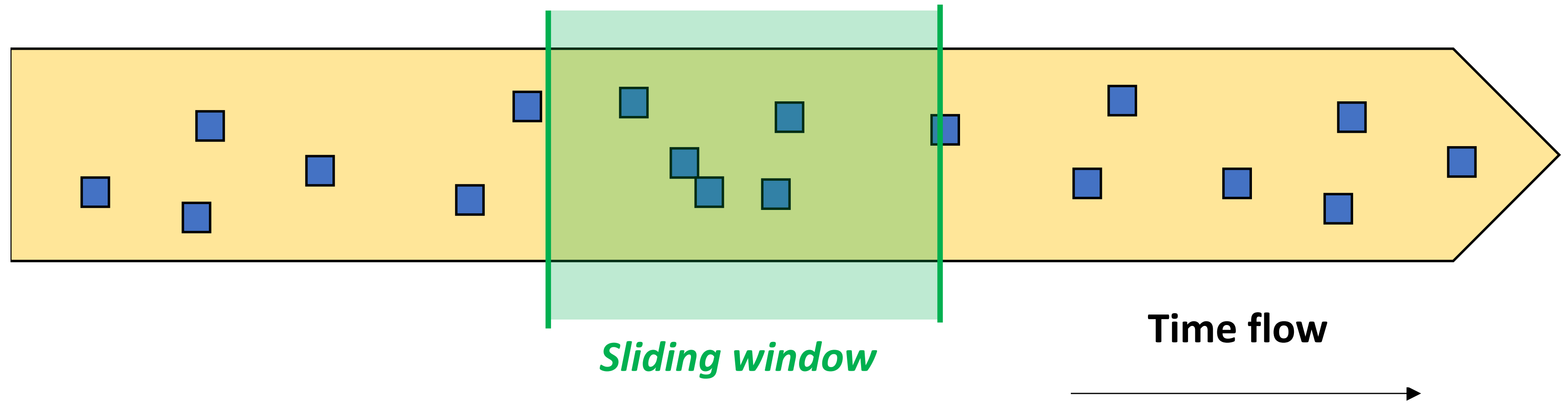
Applications of Stream Processing

- Computing
  - Log analysis,
  - Detection of DoS attacks,
  - Scaling service capacities
- Real-time monitoring
  - Fraud detection (credit cards),
  - Intrusion detection (surveillance)
- Sensor data processing
  - Weather,
  - Transportation
  - Traffic
  - Patient health
- Social media
  - Trend analysis
- Industry
  - Process optimization
  - Predictive maintenance
  - Logistics
- Advertising and promotions
  - Contextualized to user behavior or geolocation
- Financial trading
  - Algorithmic trading
  - Risk analysis
- …

Constraints and challenges

- **Inputs**
  - **Time constraints**
  - **Data elements**
    - **Unbound**
    - **Unordered**
    - **Uncomplete**
- **Outputs**
  - **Approximate answers**

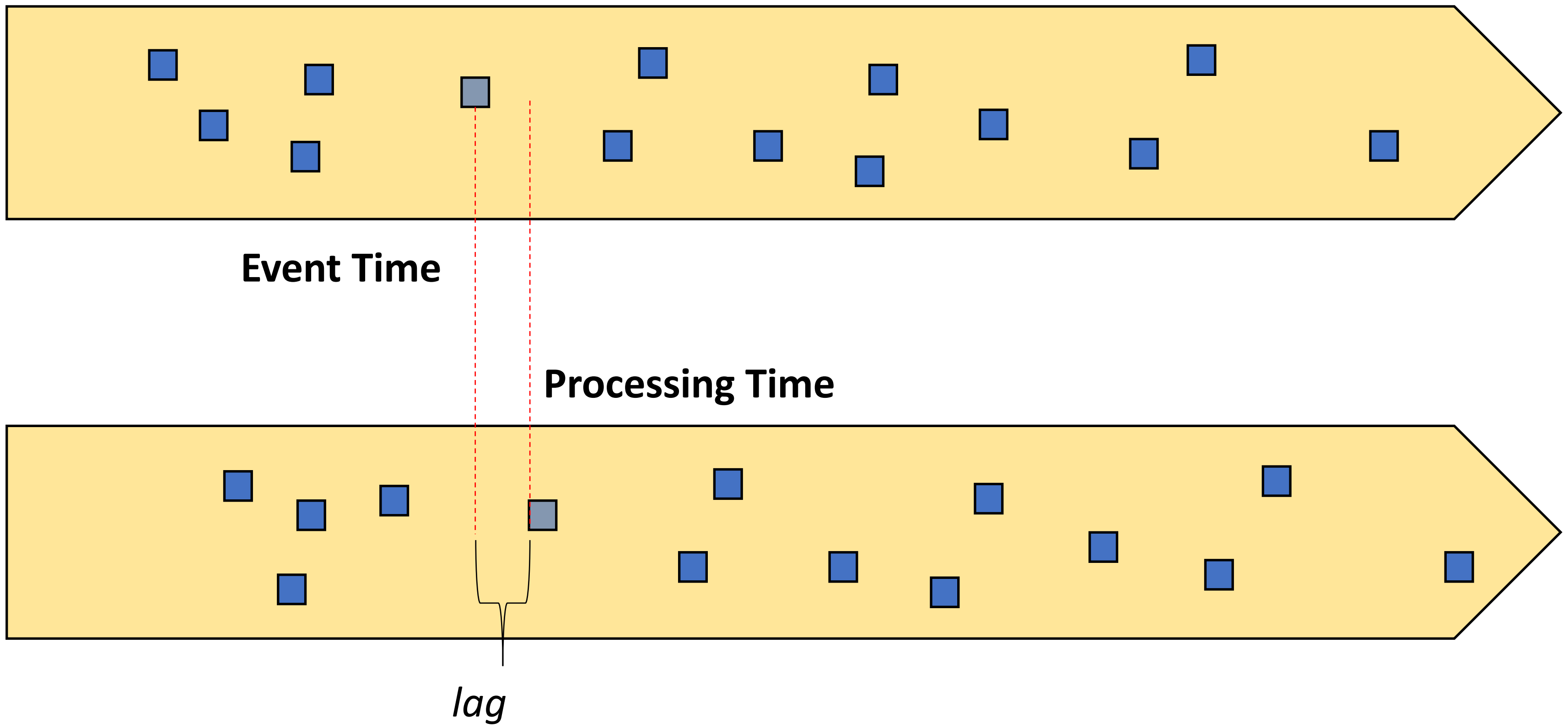
Sliding Window



Related Concepts

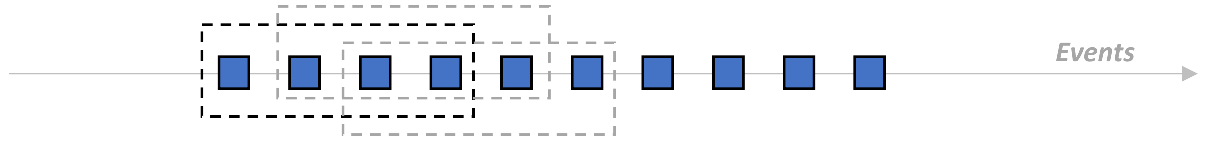
- **Event Time vs Processing Time**
- **Types of Windows**
  - **Sliding**
  - **Tambling**
  - **Time-based vs count-based**
- **Window Operations (transformations)**
- **Stateful / Stateless Operations (transformations)**

Related Concepts

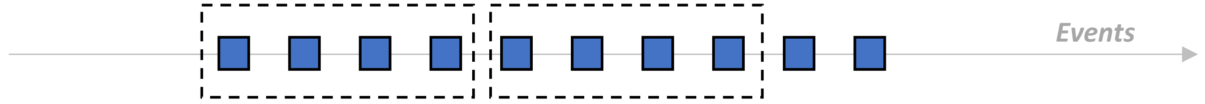


Related Concepts

Sliding Windows

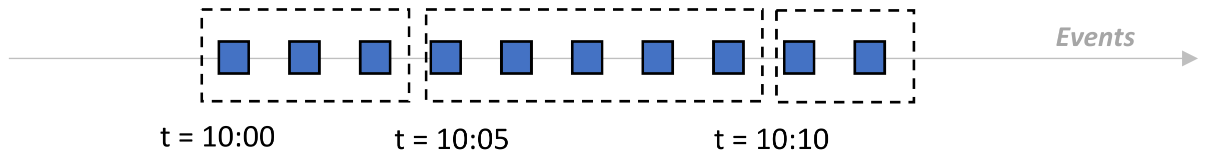


Tumbling windows

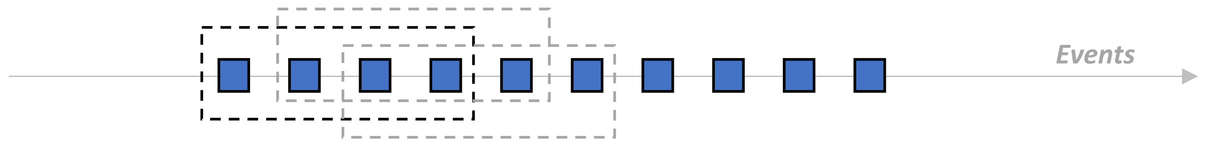


Related Concepts

Time-based windows



Count-based windows



Related Concepts

- **Window Operations (transformations)**
  - **Aggregations**
    - **Sums, averages, counts, maximum, …**
  - **Filtering**
    - **By type, IDs, …**
  - **…**

Related Concepts

- **Stateful vs Stateless Operations (Transformations)**
  - **Stateful: need to memorize records or partial results**
    - **e.g. Min, Max and average temperature of a sensor**
  - **Stateless: rely only on information within the window**
    - **e.g. Average temperature of sensor over last 5 minutes**

Stream Processing - Tools

APACHE STORM™
Distributed • Resilient • Real-time

Flink

APACHE kafka®
A distributed streaming platform

APACHE Spark™ Streaming

RabbitMQ

ØMQ

FLUME

akka

samza

druid

EPFL

Stream Processing - Tools

APACHE STORM™
Distributed • Resilient • Real-time

Flink

APACHE kafka®
A distributed streaming platform

APACHE Spark™ Streaming

RabbitMQ

ØMQ

FLUME

akka

samza

druid

Stream Processing - Tools

APACHE STORM

Flink

APACHE kafka

A distributed streaming platform

APACHE Spark Streaming

RabbitMQ

ØMQ

FLUME

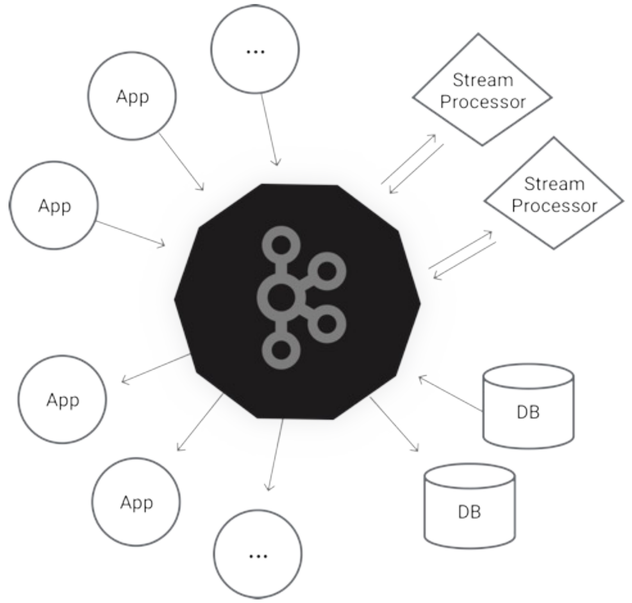
akka

samza

druid

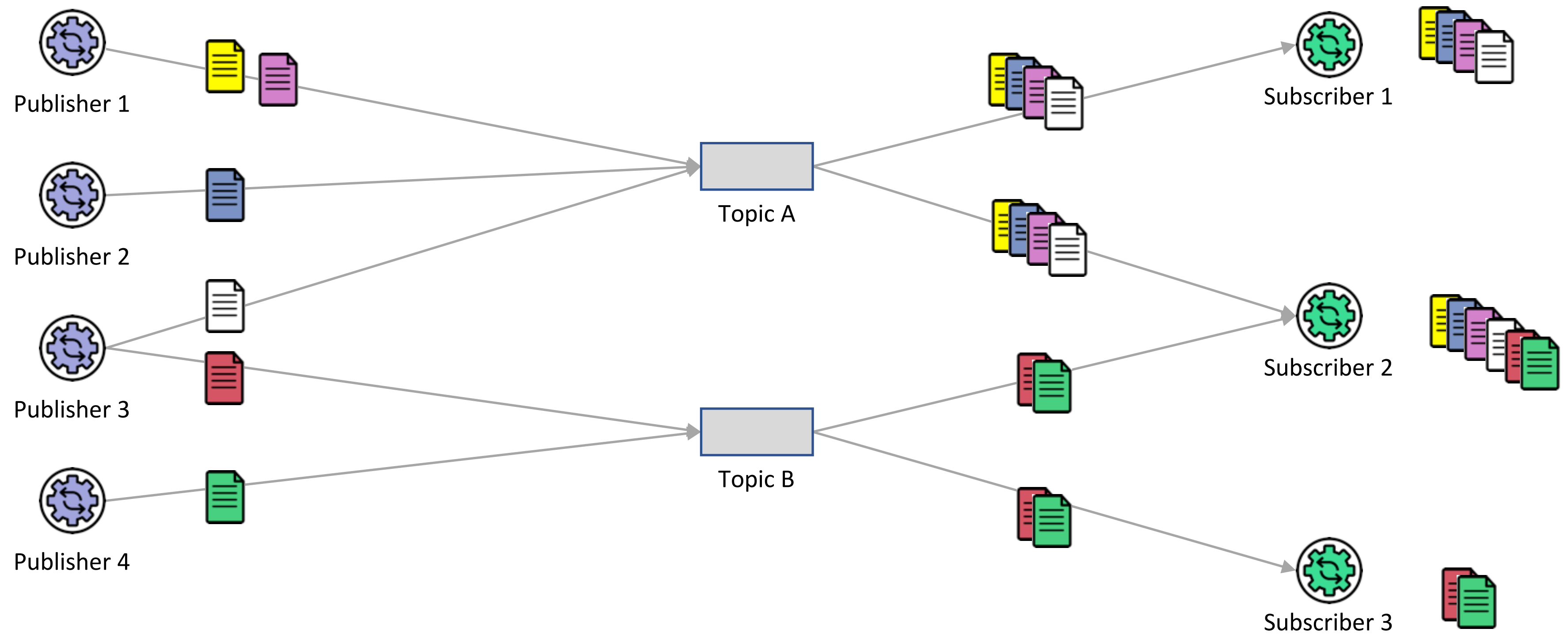
Kafka

- Messaging system
  - Publish & Subscribe
- Distributed
- Fault tolerant
- Scalable (large data volumes)
- Real-time
- Low latency



Kafka

- Concept of Publish/Subscribe messaging



Icons made by Freepik from www.flaticon.com

Spark Streaming

- Extension to Spark
  - Integrated with Spark API
- Scalable, fault tolerant
- Can read from multiple sources
- Apply ML algorithms to data streams

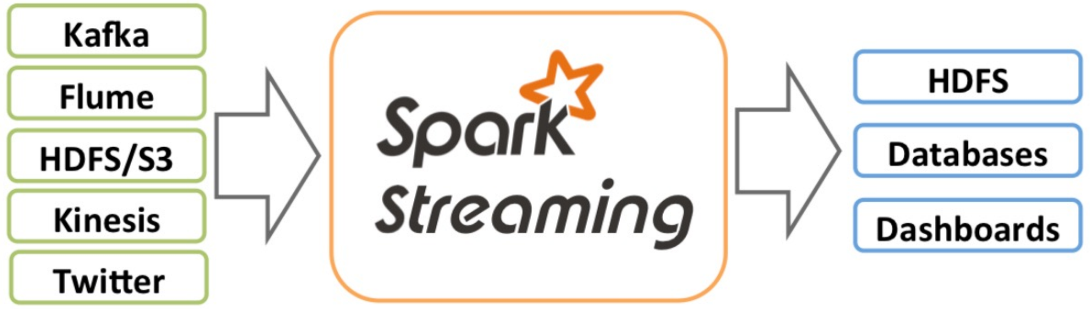
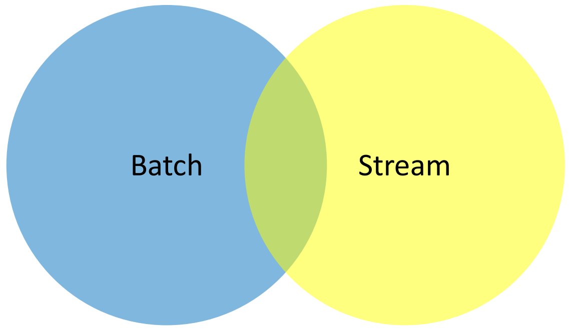


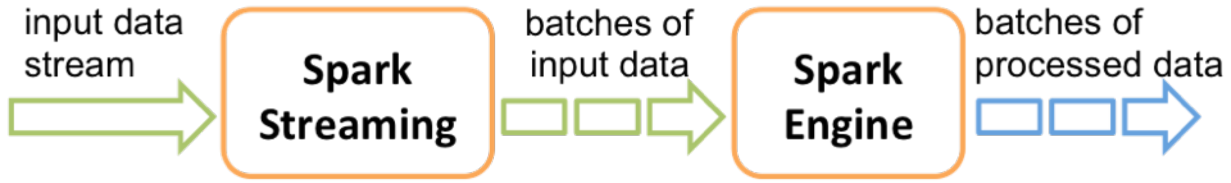
Image credits: https://spark.apache.org/docs/latest/streaming-programming-guide.html

Spark Streaming

- How it works
  - Micro-batches processing



Micro-batch



- DStream: continuous stream of data
  - Created from inputs (e.g. Kafka) or derived from other Dstreams
  - Continuous series of RDDs
  - Supports (many) transformations similar to RDDs
    - (map, count, join, etc)

Image credits: https://spark.apache.org/docs/latest/streaming-programming-guide.html

Exercises

- Documentation and Resources
  - Spark Streaming Programming Guide [1]
  - Kafka Documentation [2]

- Practical Exercise
  - https://dslab2022-renku.epfl.ch/projects/com490/lab-course

[1] https://spark.apache.org/docs/latest/streaming-programming-guide.html

[2] https://kafka.apache.org/documentation/

Exercises

1. Message queue

- Introduction to Apache Kafka
- Topics
  - Creation
  - Publish
  - Subscribe
- Synthetic example

Exercises

2. Next week

Stream Processing with Spark Streaming and Kafka

- How to properly setup Spark Streaming
- Resume synthetic exercise
- Connect to Kafka and consume stream
- Window operations
- Use actual live public stream

Start your engines

https://dslab2022-renku.epfl.ch/projects/com490/lab-course - week 9

Useful references

[1] https://spark.apache.org/docs/latest/streaming-programming-guide.html

[2] https://kafka.apache.org/documentation/