Processing Large Data Streams

Gianluca Demartini

DATA7201 Data Analytics at Scale

Week 6

Last Week

- Zookeeper
- Map/Reduce jobs
- Hadoop, YARN, and Spark
- Python (PySpark), R

Week	Date	Lecture	Prac	Assessment
1	21-Feb	Introduction to DATA7201 - Data Analytics at Scale	-	
2	28-Feb	Supporting Infrastructures and Use Cases	-	
3	6-Mar	Storage Infrastructures for Large Data Volumes	Intro to Cluster and HDFS	
4	13-Mar	Analytics Queries for Large Data Volumes	PIG(1)	
5	20-Mar	Distributed Data Processing	PIG (2)	
6	27-Mar	Processing Large Data Streams	PySpark (1)	**Quiz 1 Due (5)**
		Semester Break		
7	10-Apr	Processing Large Graph Data (1) + use cases	PySpark (2)	
8	17-Apr	Processing Large Graph Data (2) + use cases	Project support	
9	24-Apr	Recommender Systems	Project support	**Quiz 2 Due (5)**
10	1-May	Opinion Mining + use cases	Project support	
11	8-May	Health Data Analytics (guest speaker)	Project support	
12	15-May	Large Language Models?	Project support	**Report Due (45)**
13	22-May	Course Revision	-	**Quiz 3 Due (5)**

Lecture Outline

- Data streams use cases
- Apache Storm
  - Data flows through computation nodes
- Apache Kafka
  - Pub/sub model
- SparkStreaming
  - Batching of data streams
  - Pyspark package

Data streams

- Not all datasets available on HDFS when we start
- Some data arrives over time (e.g., time series)

- Distributed architecture to process data streams
  - Still run over a cluster / HDFS

- Need for real-time processing rather than M/R batch processing
- Velocity (the 3 Vs of Big Data)

Stream processing - Big Data tools

- Batch vs Stream processing
  - Data is large but always available on disk
  - Data is arriving fast and cannot be stored / needs to be processed immediately
- Streams of data
  - Twitter
  - Internet of Things (Sensors)
    - Smart Cities
    - Power plants
  - Network monitoring, real-time fraud detection, algorithmic trading, risk management
  - Any use case from the book?

Apache Storm

- Distributed and fault-tolerant real-time computation system for processing limitless streaming data
- Built at Twitter
- Real-time analytics
  - Not batch data processing like Hadoop

- Define a topology: graph of computation
  - Consumes streams of data and processes those streams in arbitrarily complex ways, repartitioning the streams as needed

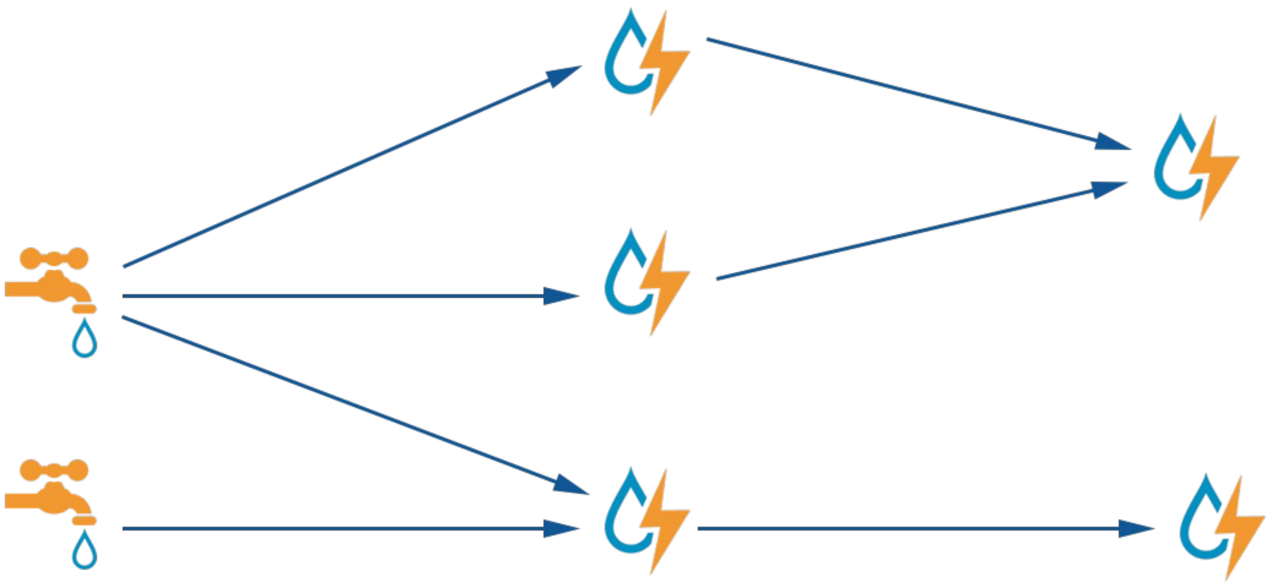
https://storm.apache.org/

Storm vs Hadoop

Storm	Hadoop
Real-time streams of data	Batch data processing
Stateless	Stateful (data stored on HDFS)
Zookeeper coordination	Zookeeper coordination
1K msg / sec processed	TB/PB processed in minutes/hours
Topology runs as more data arrives	M/R jobs completed and results written on HDFS

Apache Storm

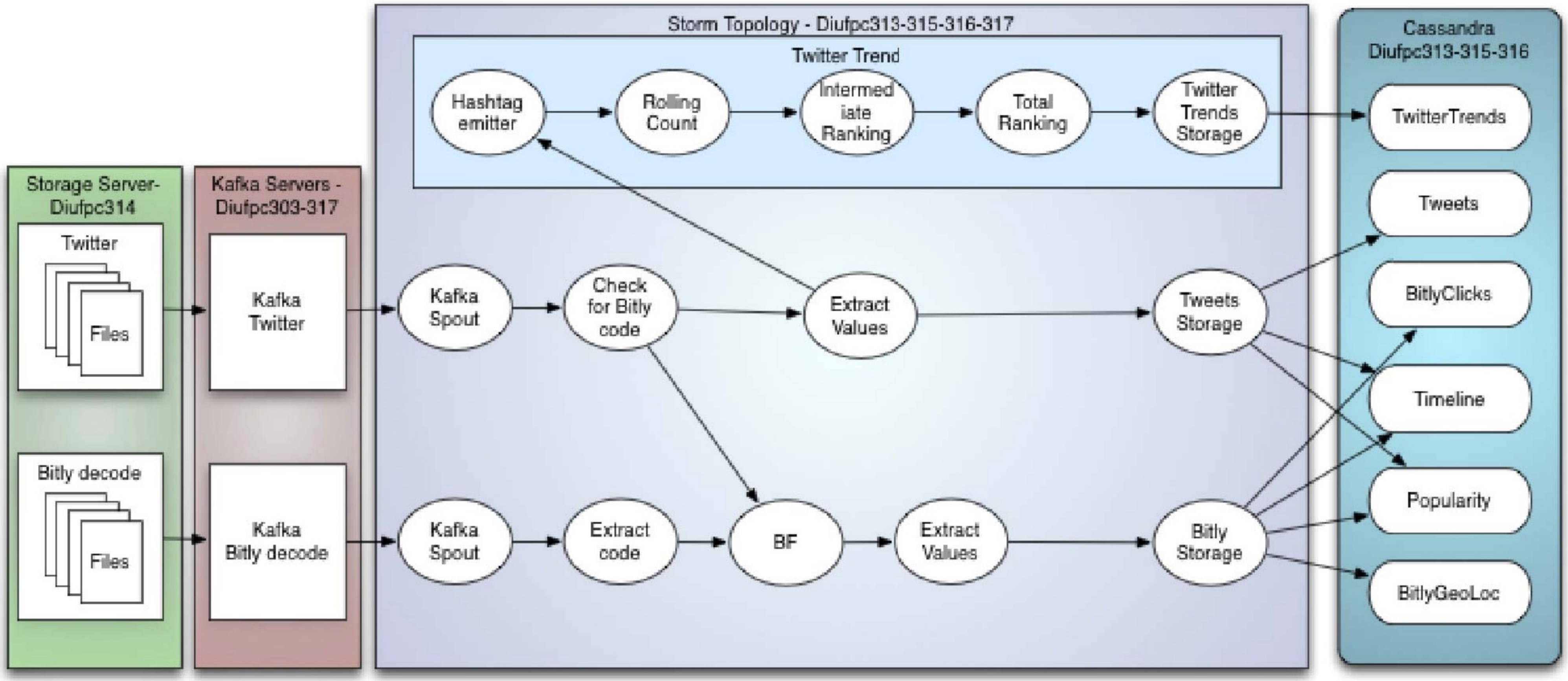
- Two kinds of nodes: spouts and bolts
  - **Spout**: source of data streams
  - **Bolt**: process input stream and outputs new stream
- Nodes execute in parallel



Apache Storm

- Spout
  - Data sources like Twitter Streaming API
  - Kafka queue (see later)
  - Read from datasources
- Bolt
  - Filtering, aggregation, joining operations
  - Interact with other datasources, e.g., databases
- Topology
  - Directed graph where vertices are computation and edges are streams of data
  - Distributed over multiple worker nodes running all the time and waiting for jobs to process
  - Multiple nodes can execute one bolt and take a share of the data
  - Topology is run by the master node (called Nimbus) assigning tasks to nodes

Storm Topology Example



Thibaud Chardonnens, Philippe Cudré-Mauroux, Martin Grund, Benoit Perroud: Big data analytics on high Velocity streams: A case study. Big Data Conference 2013: 784-787

Storm – use cases

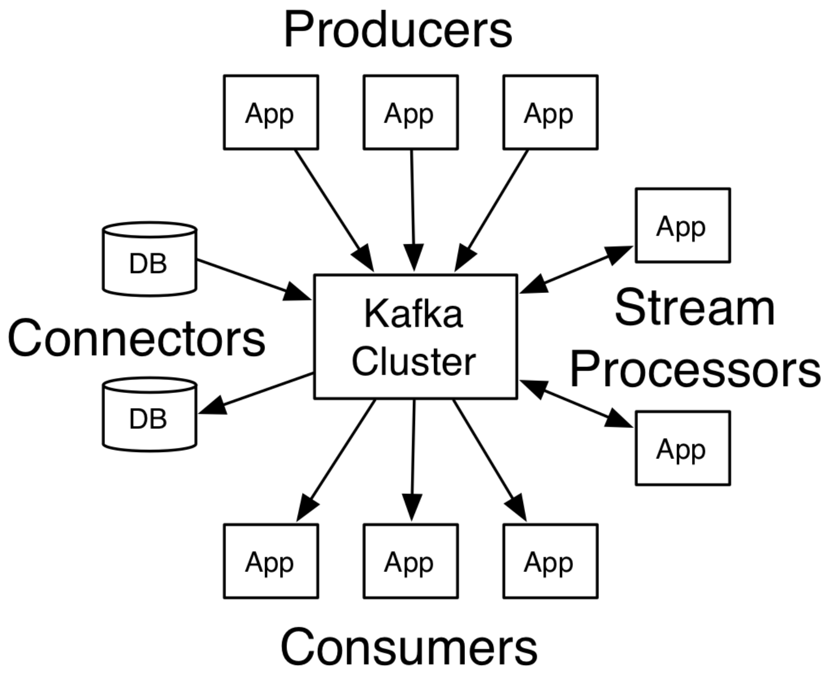
- The Weather Channel uses Storm topologies to ingest weather data
- Telecom companies
  - (Swisscom)
  - Phone calls data
  - Identify patterns that indicate problems in the network
- Wego
  - Metasearch engine combining data from different sources

Storm – use cases

	"Prevent" Use Cases	"Optimize" Use Cases
Financial Services	Securities fraud Operational risks & compliance violations	Order routing Pricing
Telecom	Security breaches Network outages	Bandwidth allocation Customer service
Retail	Shrinkage Stock outs	Offers Pricing
Manufacturing	Preventative maintenance Quality assurance	Supply chain optimization Reduced plant downtime
Transportation	Driver monitoring Predictive maintenance	Routes Pricing

https://youtu.be/_Q2uzRIkTd8 Storm developments

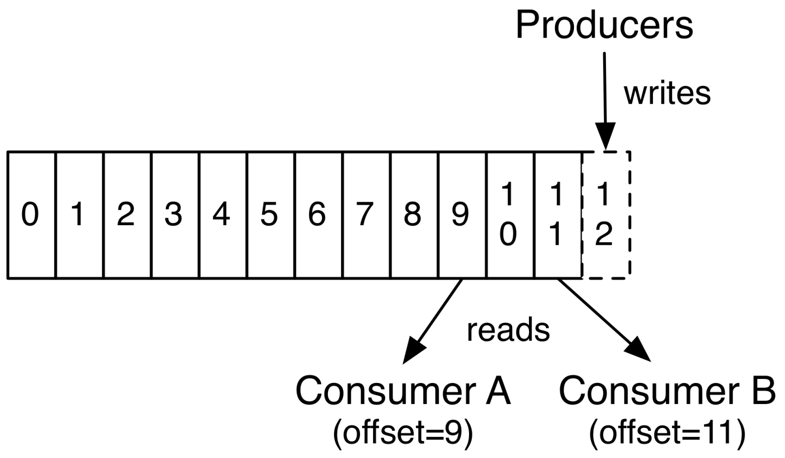
Apache Kafka



- Designed for transaction logs
- **Publish/subscribe** model
  - Broadcast data to multiple processes
  - Producers: publish a stream of data
  - Consumers: subscribe to a stream
  - Stream processors: consume and produce a new stream
- Originally built by LinkedIn (open-sourced in 2011)

https://kafka.apache.org/intro.html

Apache Kafka



- Producers write into a sequence of records that is continually appended
  - Sequences are partitioned and distributed in the cluster to scale
  - Partitions are replicated for fault tolerance
  - Order only maintained within a partition!
- Kafka cluster retains all published records for a predefined retention period (e.g., 2 days)
- Consumers read content using offset information

https://sookocheff.com/post/kafka/kafka-in-a-nutshell/

Example Kafka applications

- A retail application
  - takes in input streams of **sales** and **shipment** data
  - outputs a stream of **reorders** and **price adjustments** based on this data
  - See Walmart use case from week 1
- Usage at LinkedIn
  - https://engineering.linkedin.com/kafka/kafka-linkedin-current-and-future
    - Page views, clicks
  - https://engineering.linkedin.com/kafka/running-kafka-scale
    - Multiple datacenters
    - Mirroring data across Kafka clusters
    - (2015) 650TB of messages /day
    - 13M msg/sec 2.75GB/sec

Spark Streaming

- An extension of the core Spark
- Latest solution for big data streams
- Scalable, high-throughput, fault-tolerant stream processing of live data streams
- Different data sources

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

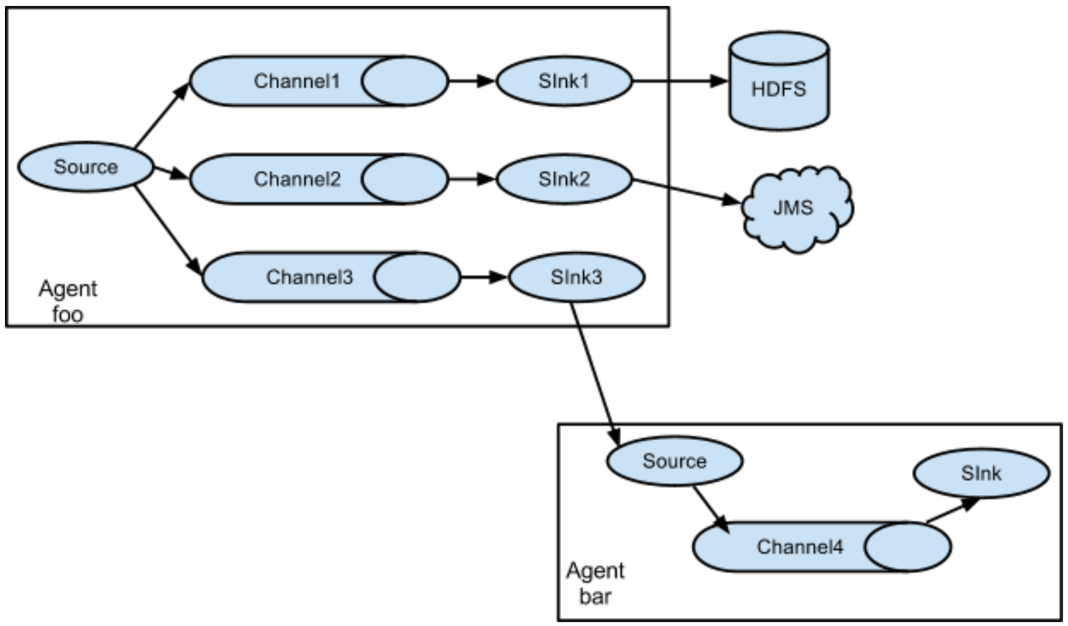
Spark Streaming - Data Sources

- Kafka
  - Most popular (and older)
  - High throughput (20k msg/sec)
- Apache Flume
  - Streaming event log data (web page visits, clicks) from a web server
  - Distributed / high availability
- Kinesis
  - Amazon AWS solution (since 2013) https://aws.amazon.com/kinesis/

- Streams are represented as a sequence of **RDDs**.

Kafka vs Flume

- Kafka
  - Subscribe to streams of data (pull)
  - Higher throughput
  - General purpose
- Flume
  - Push data to clients
  - Data goes to HDFS
  - Several built-in sources of data
  - No replication (but still reliable)
  - https://flume.apache.org/FlumeUserGuide.html

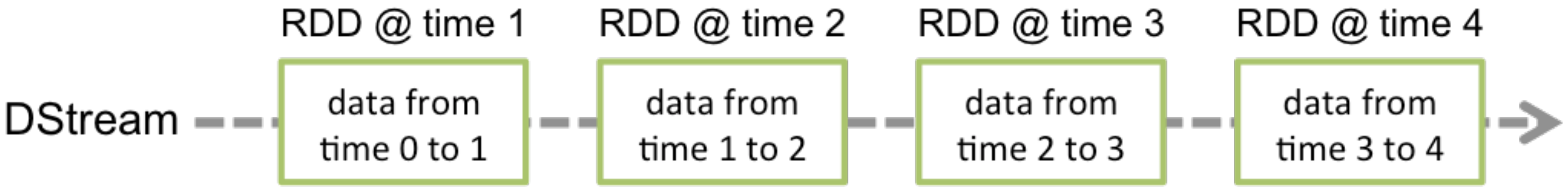


Spark - Stream processing

- Series of batch computations on small time intervals (windows over the stream)
- Spark Streaming receives live input data streams
- Divides the data into batches
- Spark engine processes batches

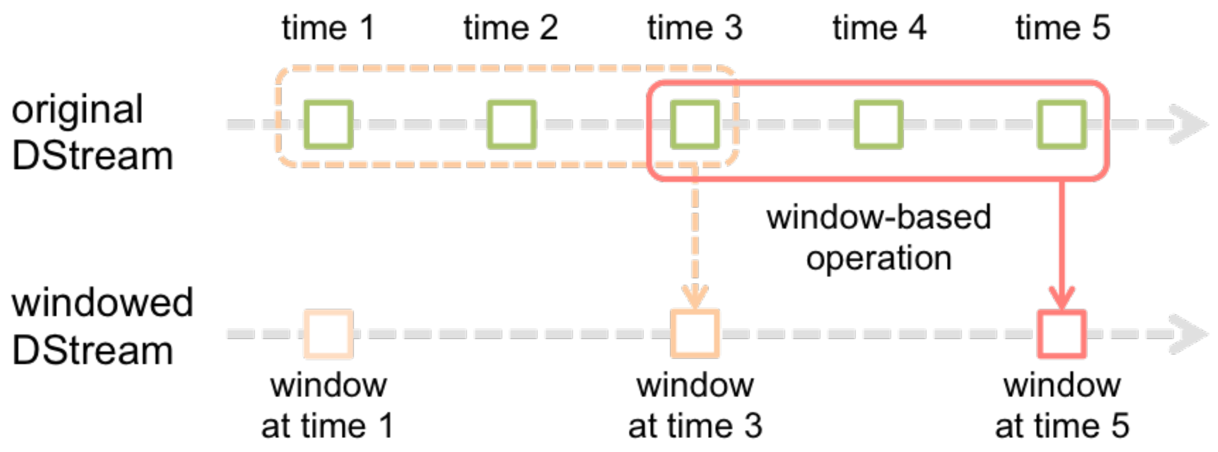
Discretized Streams (DStreams)

- Continuous stream of data
  - From source
  - Transforming an input file
- DStream is represented by a continuous series of RDDs
  - Each RDD has data from a certain interval
- Resilient Distributed Datasets (RDDs)
  - Keep data in memory
  - Can recover it without replication (track the lineage graph of operations that were used to build it)



Window Operations

- Apply **transformations** (map, flatMap, etc) over a sliding window of data
- RDDs that fall within the window are combined and operated upon
  - Parameters: *window length, sliding interval*
  - Custom window-based transformations



Spark Streaming Fault-tolerance

- Streams arrive 24/7
- Storage able to recover from failures (HDFS)
  - Store computation metadata
  - Store data from streams
- When a node fails, each node in the cluster works to recompute part of the lost node's RDDs

- Batch interval needs to be set such that the expected data rate in production can be sustained

Spark Streaming – Use Cases

- Uber
  - Data from mobile users
  - Kafka as data source
  - Event data to structured data into HDFS
  - Analytics as M/R
- Pinterest
  - Real-time user interaction analysis
  - Use this for recommendations (products to buy, places to visit)

https://www.datanami.com/2015/11/30/spark-streaming-what-is-it-and-whos-using-it/

Stream of data **from HDFS** / word count - Python

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

if __name__ == "__main__":
    if len(sys.argv) != 2:
        print("Usage: hdfs_wordcount.py <directory>", file=sys.stderr)
        exit(-1)

    sc = SparkContext(appName="PythonStreamingHDFSWordCount")
    ssc = StreamingContext(sc, 1)

    lines = ssc.textFileStream(sys.argv[1])
    counts = lines.flatMap(lambda line: line.split(" "))\
                  .map(lambda x: (x, 1))\
                  .reduceByKey(lambda a, b: a+b)
    counts.pprint()

    ssc.start()
    ssc.awaitTermination()
```

A StreamingContext represents the connection to a Spark cluster,

lines is a DStream

Wordcount M/R

It keeps running and waits for data

https://github.com/apache/spark/blob/master/examples/src/main/python/streaming/hdfs_wordcount.py

Stream of data from **Kafka stream** / word count - Python

```
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: direct_kafka_wordcount.py <broker_list> <topic>", file=sys.stderr)
        exit(-1)

    sc = SparkContext(appName="PythonStreamingDirectKafkaWordCount")
    ssc = StreamingContext(sc, 2)

    brokers, topic = sys.argv[1:]
    kvs = KafkaUtils.createDirectStream(ssc, [topic], {"metadata.broker.list": brokers})
    lines = kvs.map(lambda x: x[1])
    counts = lines.flatMap(lambda line: line.split(" ")) \
        .map(lambda word: (word, 1)) \
        .reduceByKey(lambda a, b: a+b)
    counts.pprint()

    ssc.start()
    ssc.awaitTermination()
```

Batch duration: time interval to divide streams into batches (in ms)

Creates data stream
Topic: set of records
Brokers: nodes providing data

Summary

- Data streams use cases
- Apache Storm
  - Data flows through computation nodes
- Apache Kafka
  - Pub/sub model
- SparkStreaming
  - Batching of data streams
  - Pyspark package

What’s next (Part II)

- Data Streams (week 6)
  - Apache Storm and Apache Kafka
  - Spark Streaming
- **Graph data / network data (weeks 7-8)**
  - (Social) Network data analytics at scale
  - Modularity, community detection
  - Link Analysis
  - Systems