Streaming items through a cluster with Spark Streaming

Tathagata “TD” Das

@tathadas

CME 323: Distributed Algorithms and Optimization
Stanford, May 6, 2015

Who am I?

- Project Management Committee (PMC) member of Apache Spark
- Lead developer of Spark Streaming
- Formerly in AMPLab, UC Berkeley
- Software developer at Databricks
- Databricks was started by creators of Spark to provide Spark-as-a-service in the cloud

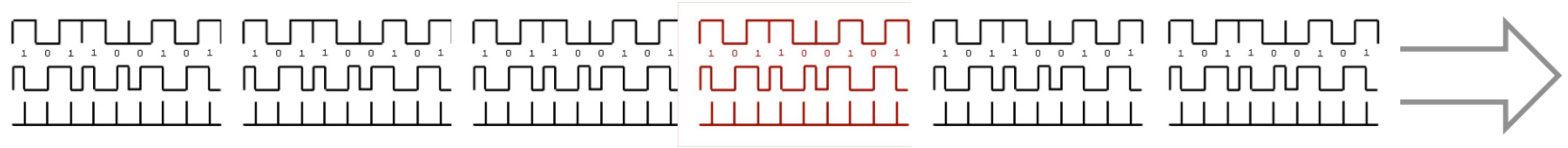
Big Data

Big *Streaming* Data

Why process Big *Streaming* Data?

Fraud detection in bank transactions

Anomalies in sensor data

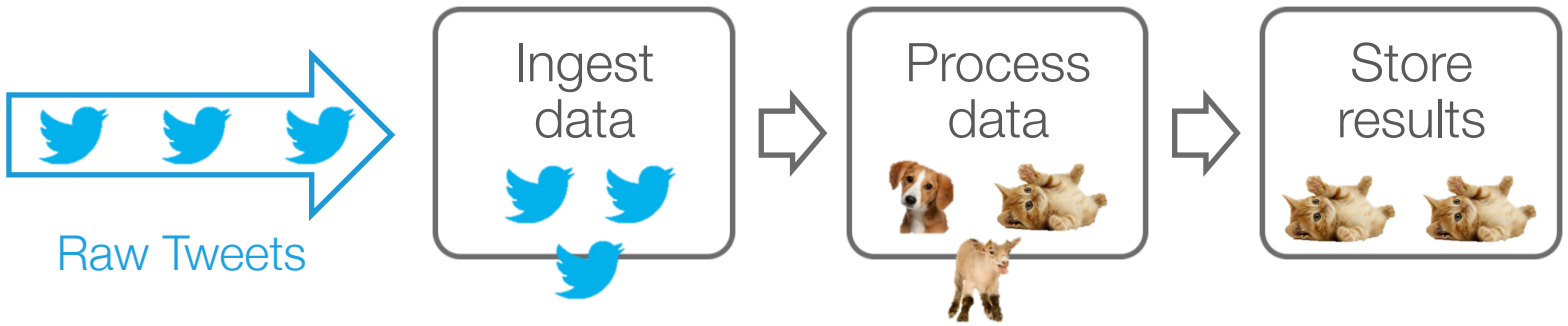


Cat videos in tweets

databricks

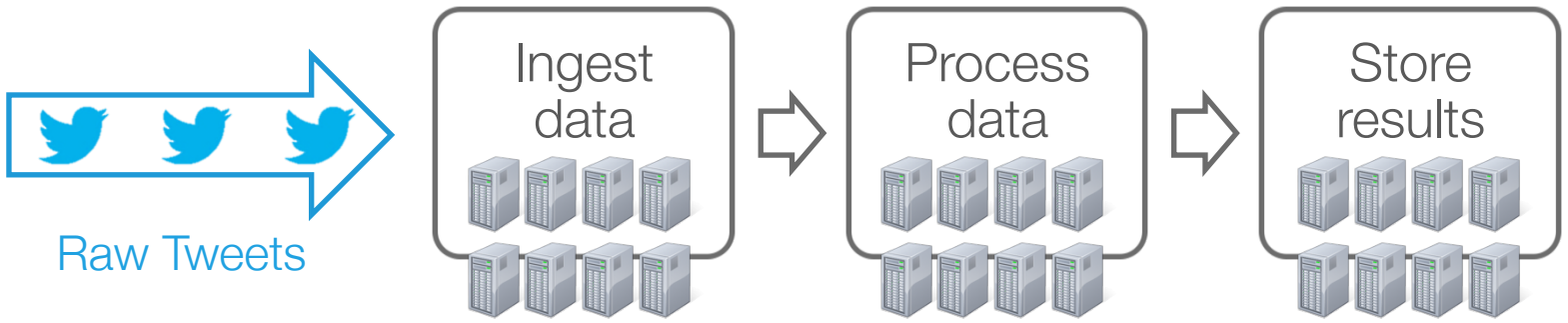
Spark

How to Process Big *Streaming* Data



- Ingest – Receive and buffer the streaming data
- Process – Clean, extract, transform the data
- Store – Store transformed data for consumption

How to Process Big *Streaming* Data



> For big streams, every step requires a cluster

> Every step requires a system that is designed for it

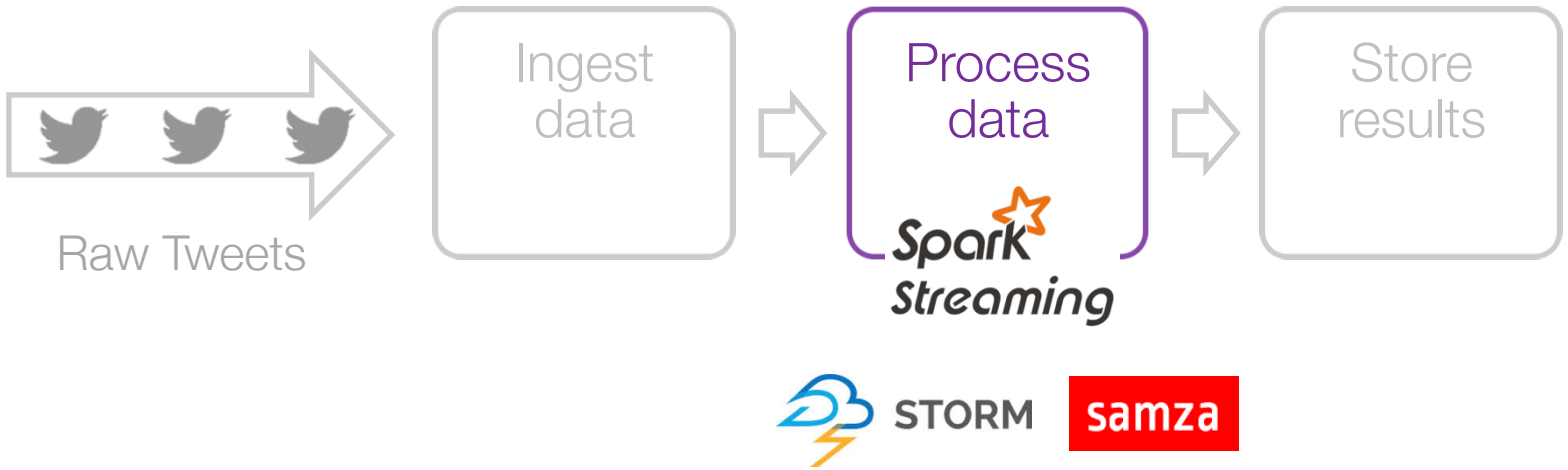
Stream Ingestion Systems

> Kafka – popular distributed pub-sub system

> Kinesis – Amazon managed distributed pub-sub

> Flume – like a distributed data pipe

Stream Ingestion Systems

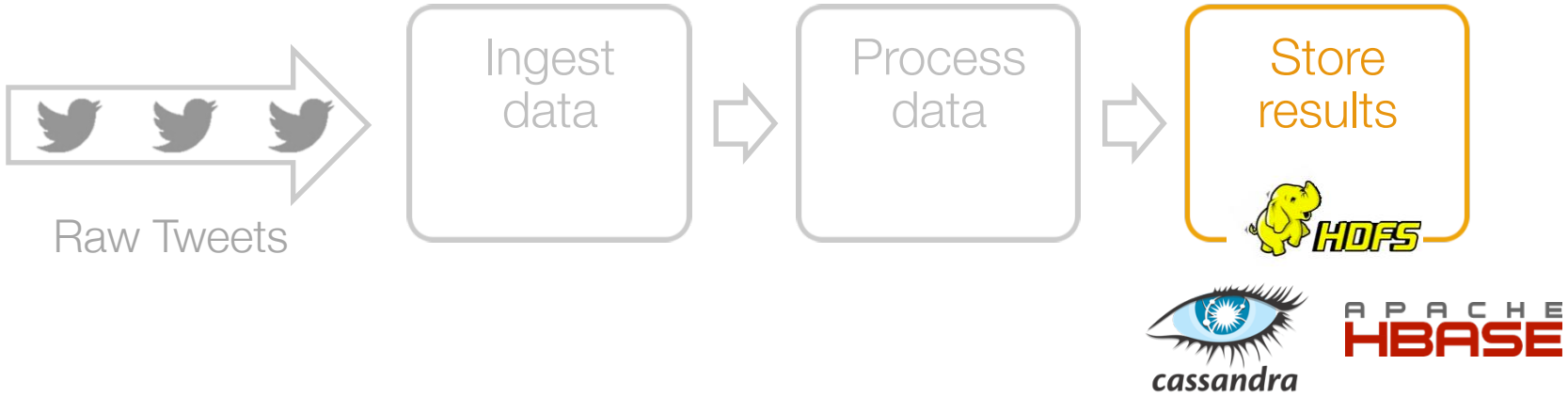


> Spark Streaming – most demanded

> Storm – most widely deployed (as of now ;))

> Samza – gaining popularity in certain scenarios

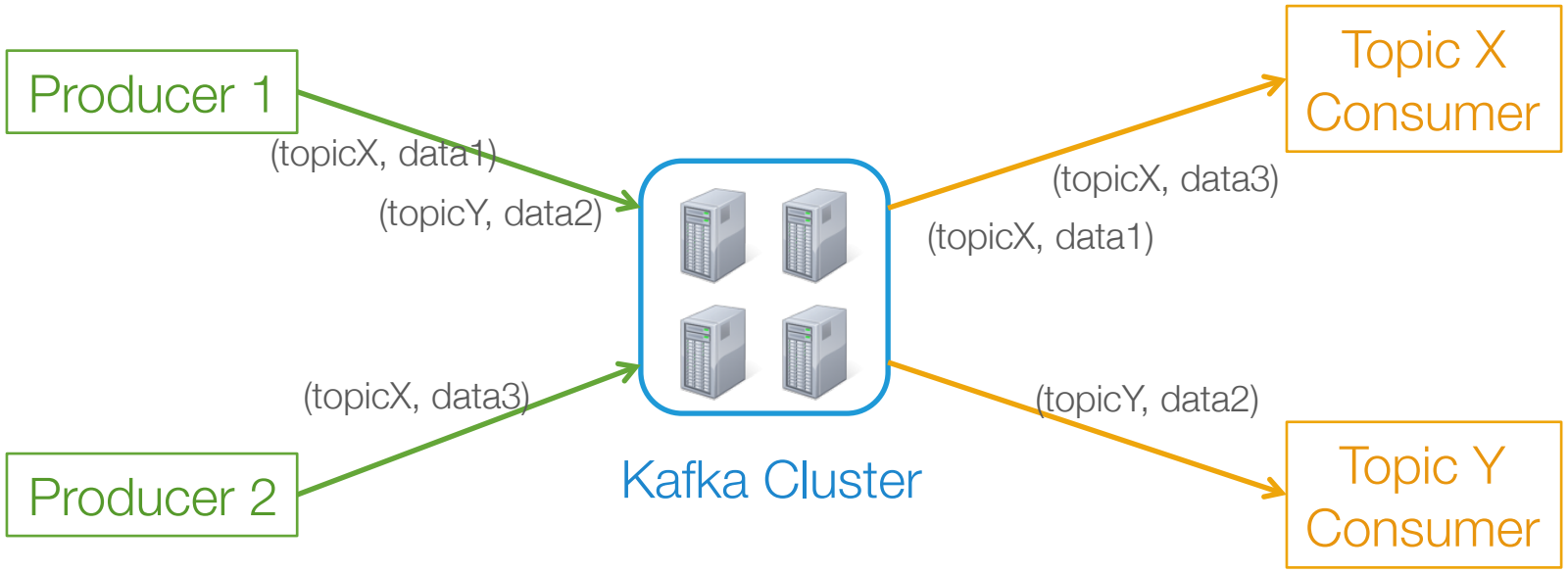
Stream Ingestion Systems



- File systems – HDFS, Amazon S3, etc.
- Key-value stores – HBase, Cassandra, etc.
- Databases – MongoDB, MemSQL, etc.


kafka

kafka Producers and Consumers

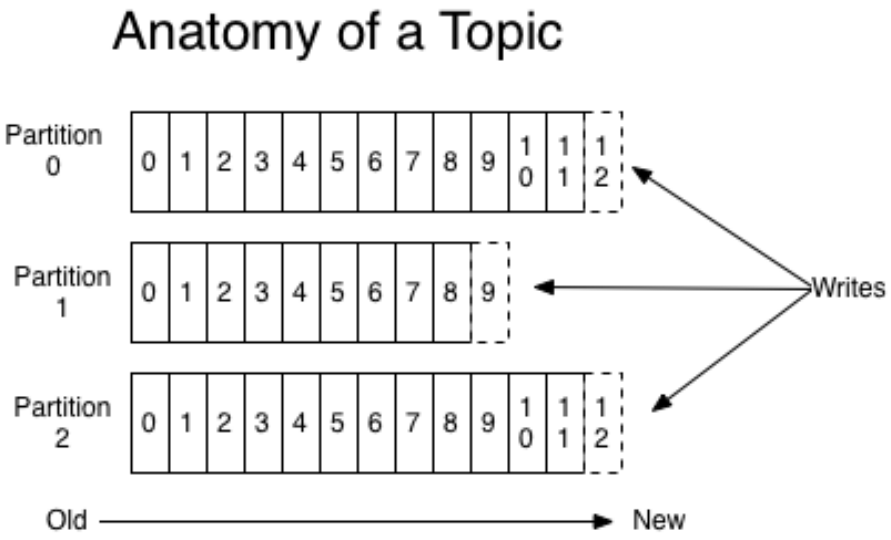


> Producers publish data tagged by “topic”

> Consumers subscribe to data of a particular “topic”

kafka Topics and Partitions

> Topic = category of message, divided into partitions
> Partition = ordered, numbered stream of messages
> Producer decides which (topic, partition) to put each message in



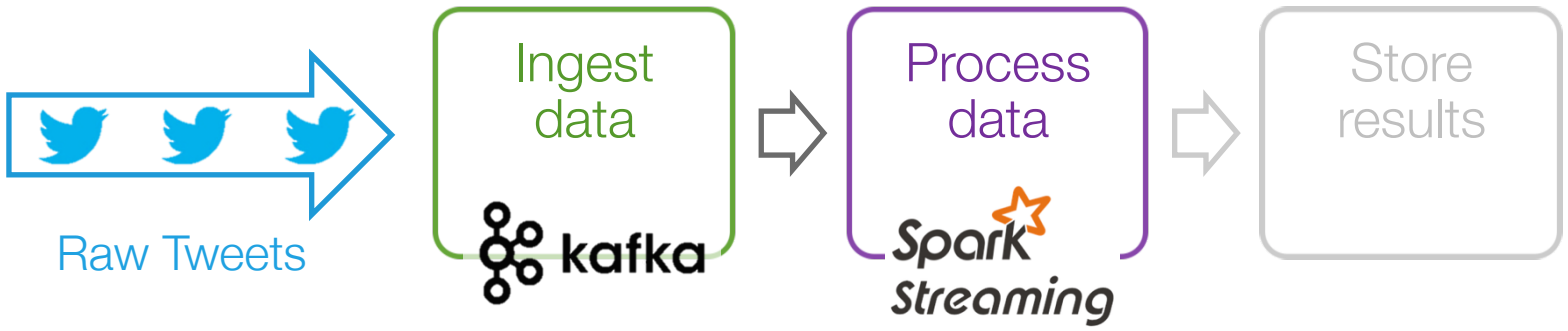
databricks

Spark

Topics and Partitions

- Topic = category of message, divided into partitions
- Partition = ordered, numbered stream of messages
- Producer decides which (topic, partition) to put each message in
- Consumer decides which (topic, partition) to pull messages from
  - High-level consumer – handles fault-recovery with Zookeeper
  - Simple consumer – low-level API for greater control

How to process Kafka messages?



- Incoming tweets received in distributed manner and buffered in Kafka
- How to process them?

Streaming

What is Spark Streaming?

Scalable, fault-tolerant stream processing system

High-level API

joins, windows, …
often 5x less code

Fault-tolerant

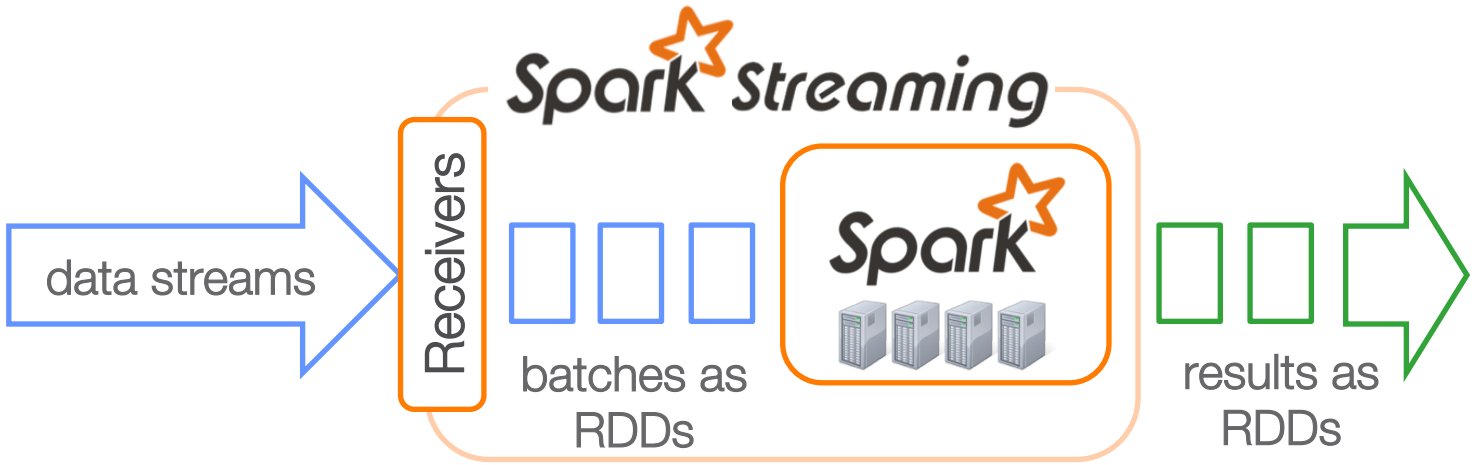
Exactly-once semantics,
even for stateful ops

Integration

Integrate with MLlib, SQL,
DataFrames, GraphX

How does Spark Streaming work?

- Receivers chop up data streams into batches of few seconds
- Spark processing engine processes each batch and pushes out the results to external data stores



databricks

Spark

Spark Programming Model

> *Resilient distributed datasets (RDDs)*

- Distributed, partitioned collection of objects
- Manipulated through parallel *transformations*

  (map, filter, reduceByKey, …)
- All transformations are lazy, execution forced by *actions*

  (count, reduce, take, …)
- Can be *cached* in memory across cluster
- Automatically rebuilt on failure

Spark Streaming Programming Model

> *Discretized Stream (DStream)*

- Represents a stream of data
- Implemented as a infinite sequence of RDDs

> DStreams API very similar to RDD API

- Functional APIs in Scala Java python
- Create input DStreams from Kafka, Flume, Kinesis, HDFS, …
- Apply transformations

Example – Get hashtags from Twitter

```
val ssc = new StreamingContext(conf, Seconds(1))
```

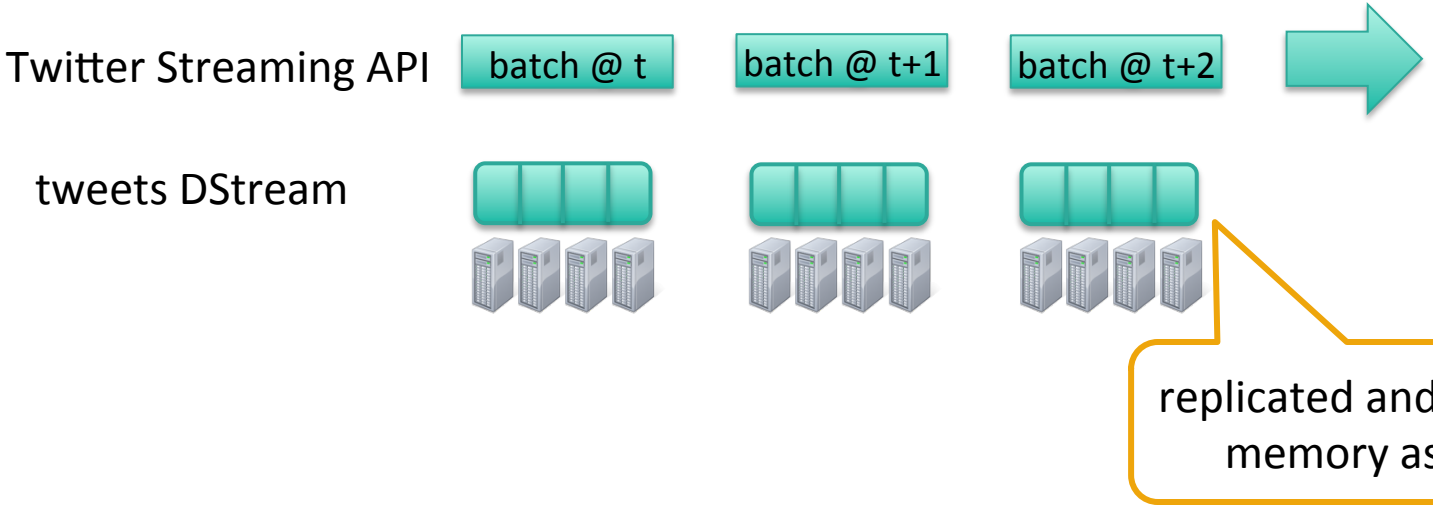
StreamingContext is the starting point of all streaming functionality

Batch interval, by which streams will be chopped up

Example – Get hashtags from Twitter

```
val ssc = new StreamingContext(conf, Seconds(1))
val tweets = TwitterUtils.createStream(ssc, auth)
```

Input DStream

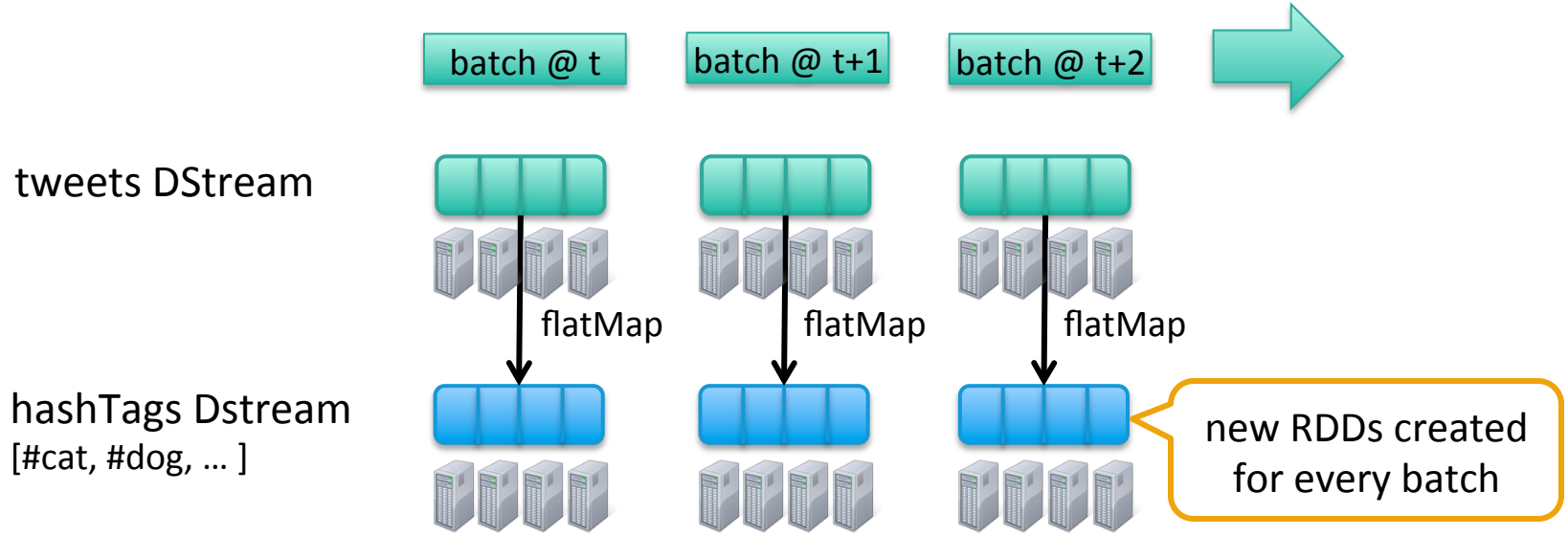


Example – Get hashtags from Twitter

```
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
```

transformed DStream

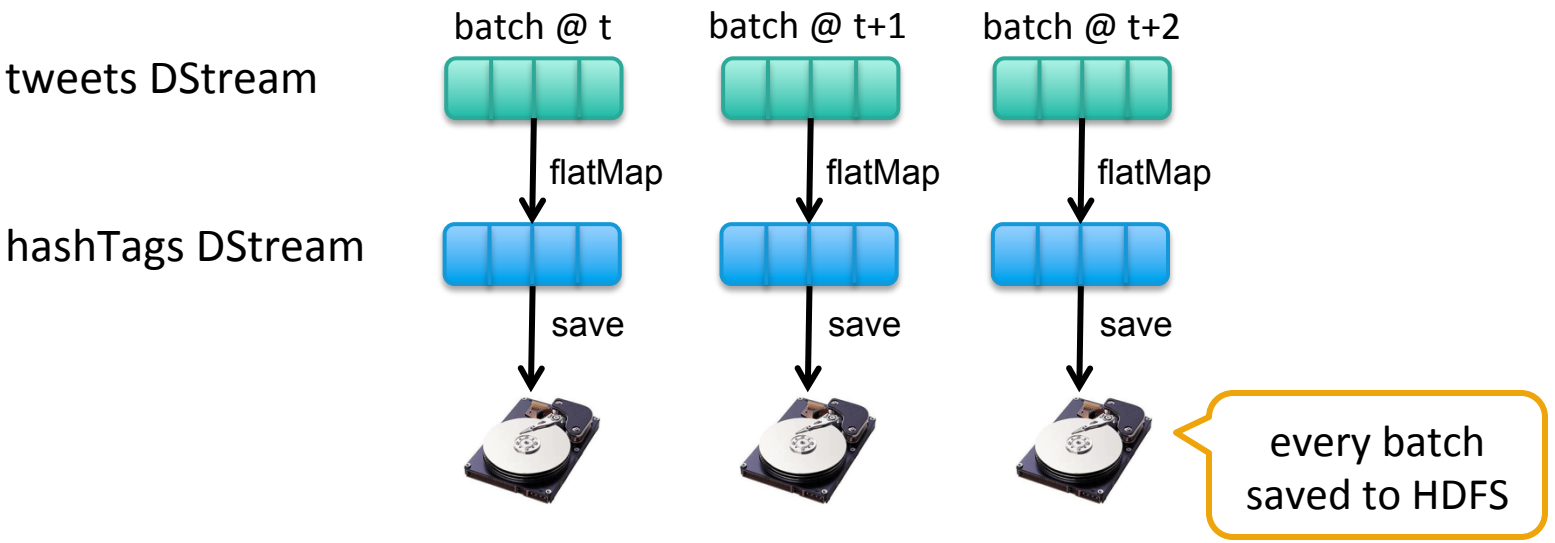
transformation: modify data in one DStream to create another DStream



Example – Get hashtags from Twitter

```
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
hashTags.saveAsTextFiles("hdfs://...")
```

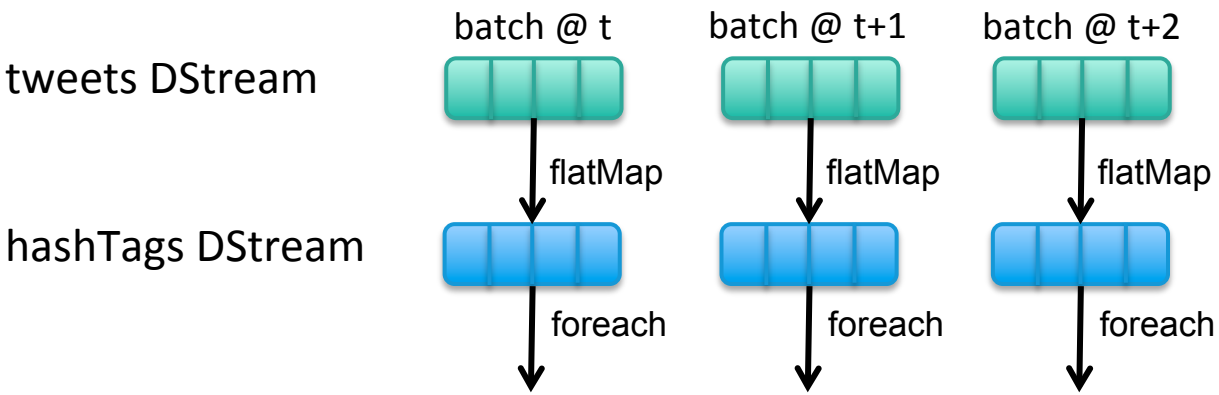
output operation: to push data to external storage



Example – Get hashtags from Twitter

```
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
hashTags.foreachRDD(hashTagRDD => { ... })
```

foreachRDD: do whatever you want with the processed data



Write to a database, update analytics UI, do whatever you want

Example – Get hashtags from Twitter

```
val tweets = TwitterUtils.createStream(ssc, None)
val hashTags = tweets.flatMap(status => getTags(status))
hashTags.foreachRDD(hashTagRDD => { ... })
```

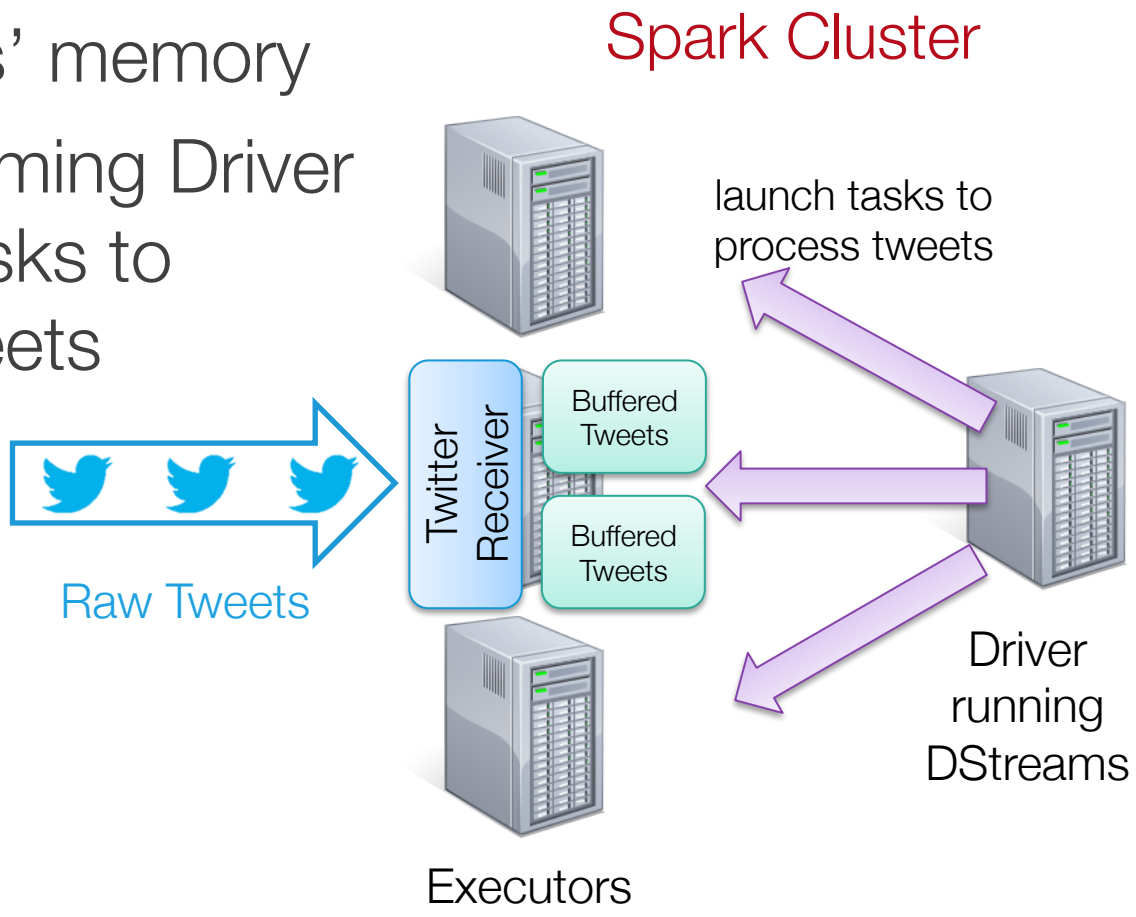
all of this was just setup for what to do when streaming data is receiver

```
ssc.start()
```

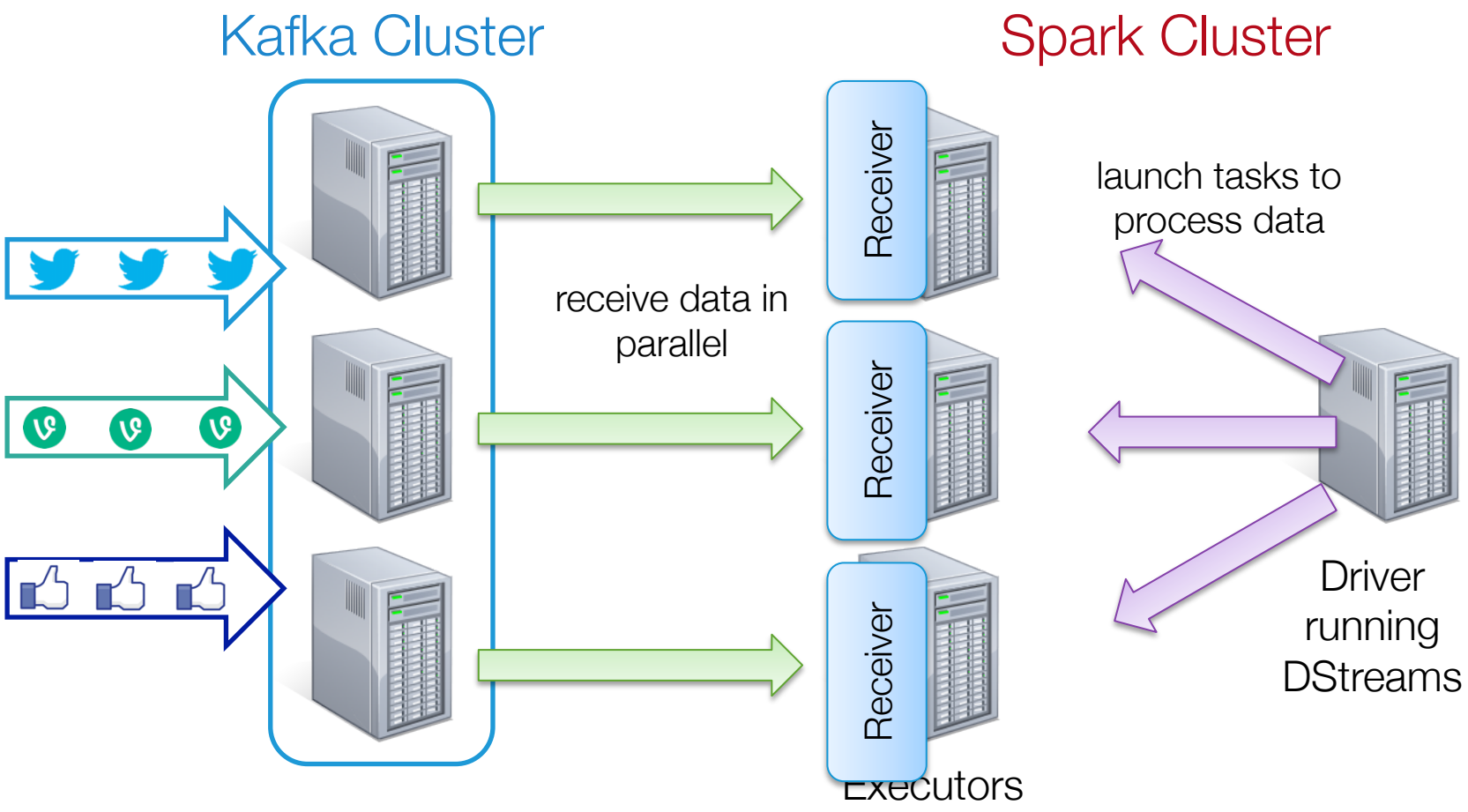
this actually starts the receiving and processing

What's going on inside?

- Receiver buffers tweets in Executors' memory
- Spark Streaming Driver launches tasks to process tweets



What's going on inside?

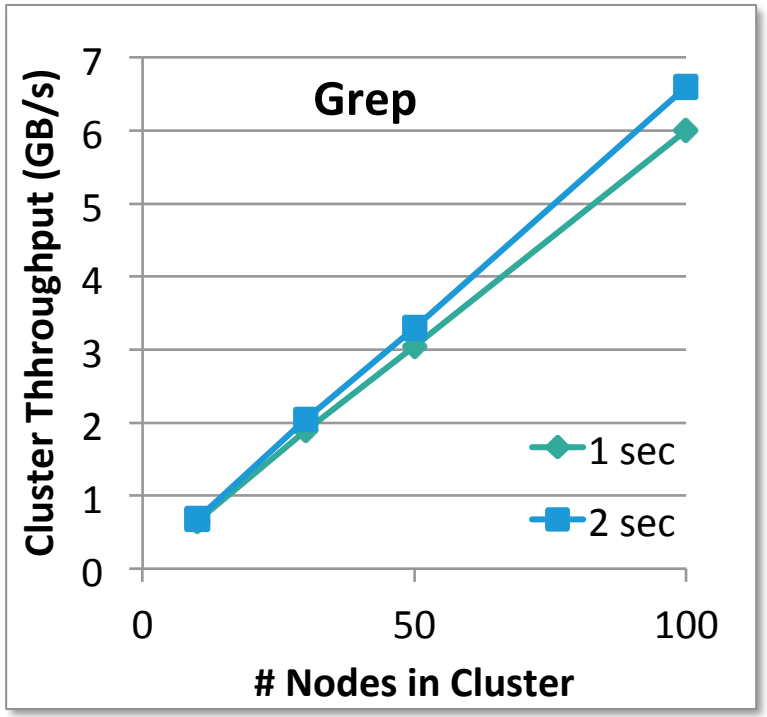


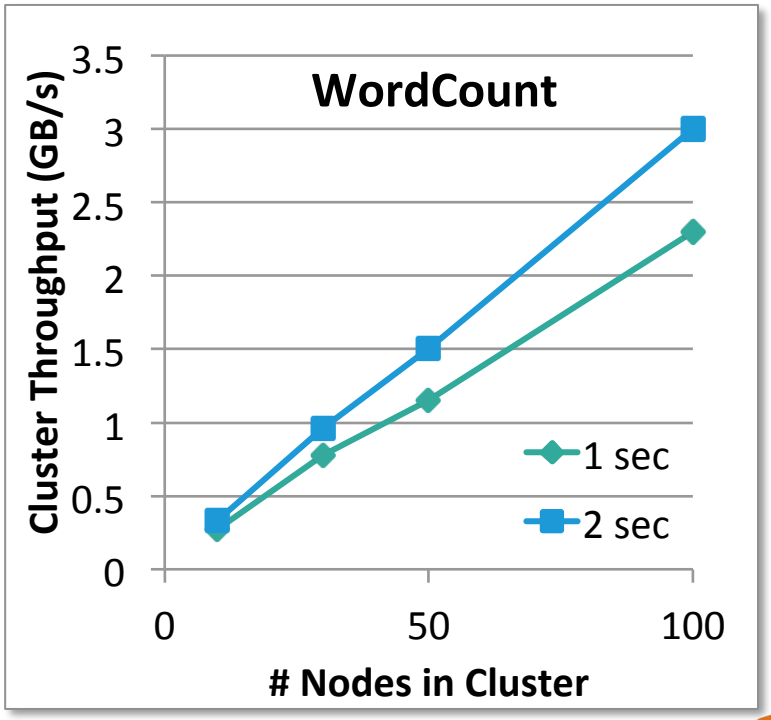
databricks

Spark

Performance

Can process 60M records/sec (6 GB/sec) on 100 nodes at sub-second latency





databricks

Spark

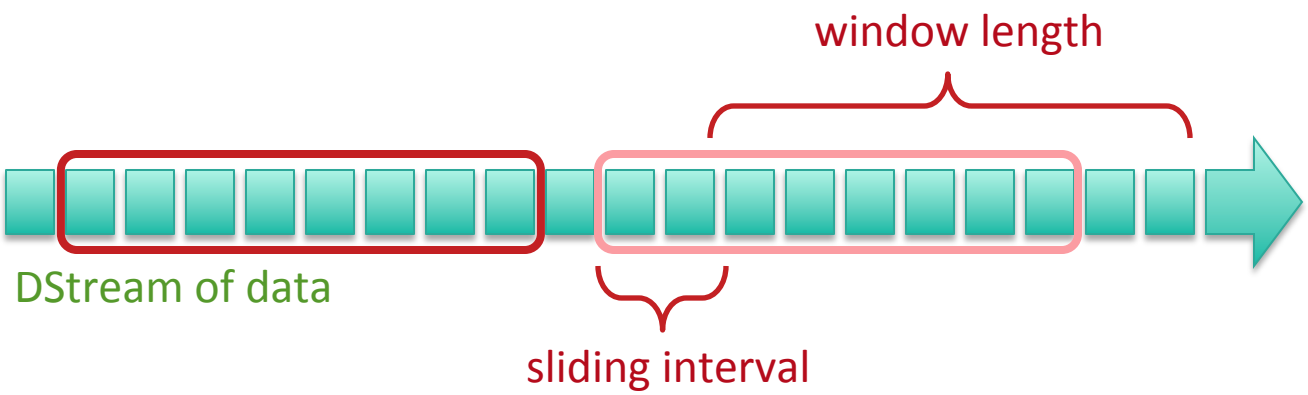
Window-based Transformations

```
val tweets = TwitterUtils.createStream(ssc, auth)
val hashTags = tweets.flatMap(status => getTags(status))
val tagCounts = hashTags.window(Minutes(1), Seconds(5)).countByValue()
```

sliding window operation

window length

sliding interval



databricks

Spark

Arbitrary Stateful Computations

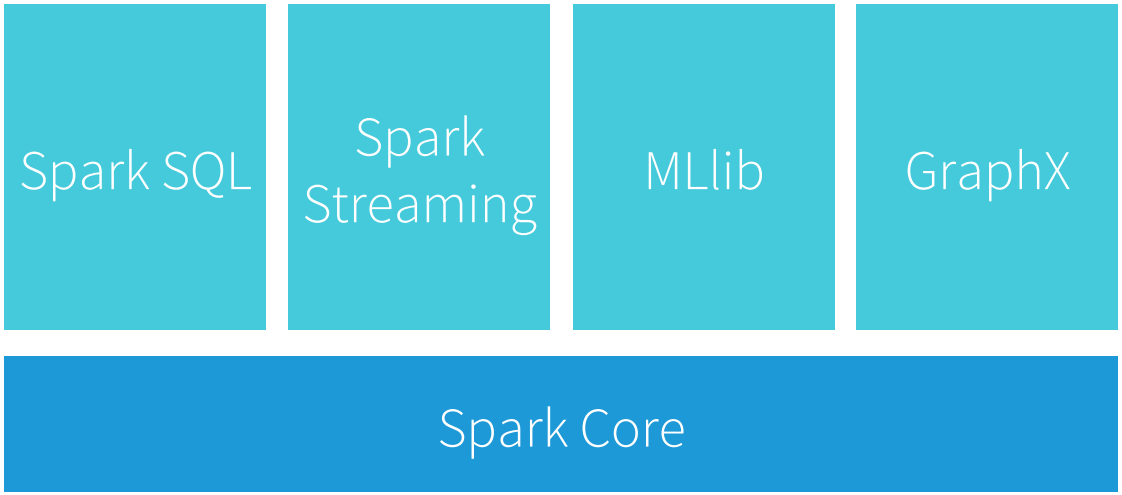
Specify function to generate new state based on previous state and new data

- Example: Maintain per-user mood as state, and update it with their tweets

```
def updateMood(newTweets, lastMood) => newMood

val moods = tweetsByUser.updateStateByKey(updateMood _)
```

Integrates with Spark Ecosystem

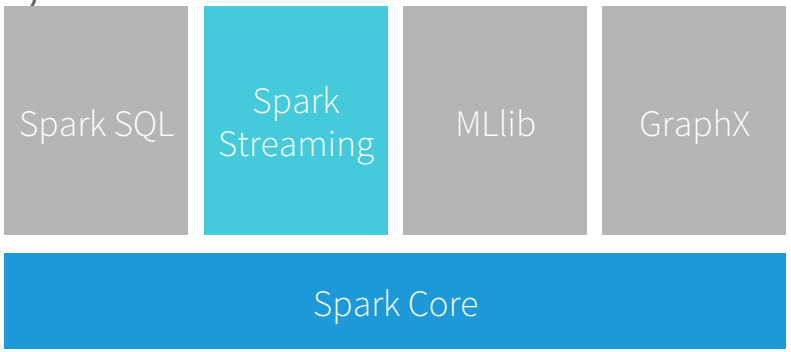


Combine batch and streaming processing

> Join data streams with static data sets

```
// Create data set from Hadoop file
val dataset = sparkContext.hadoopFile("file")

// Join each batch in stream with dataset
kafkaStream.transform { batchRDD =>
      batchRDD.join(dataset)filter(...)
}
```

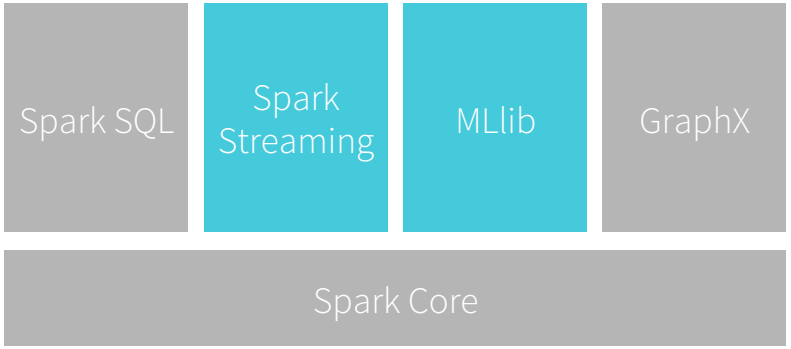


Combine machine learning with streaming

> Learn models offline, apply them online

```
// Learn model offline
val model = KMeans.train(dataset, ...)

// Apply model online on stream
kafkaStream.map { event =>
    model.predict(event.feature)
}
```

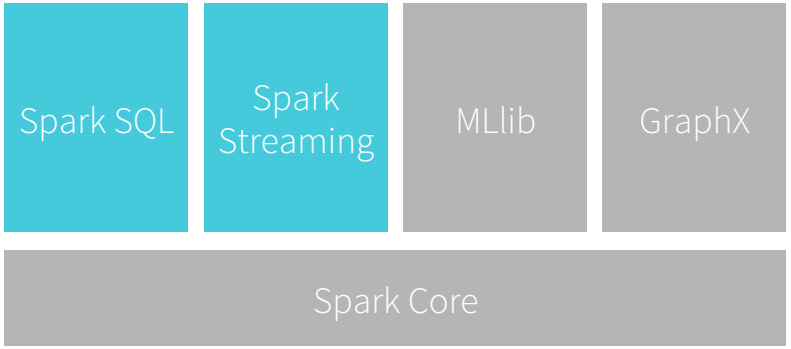


Combine SQL with streaming

> Interactively query streaming data with SQL

```
// Register each batch in stream as table
kafkaStream.map { batchRDD =>
  batchRDD.registerTempTable("latestEvents")
}

// Interactively query table
sqlContext.sql("select * from latestEvents")
```



databricks

Spark

100+ known industry deployments

Why are they adopting Spark Streaming?

Easy, high-level API

Unified API across batch and streaming

Integration with Spark SQL and MLlib

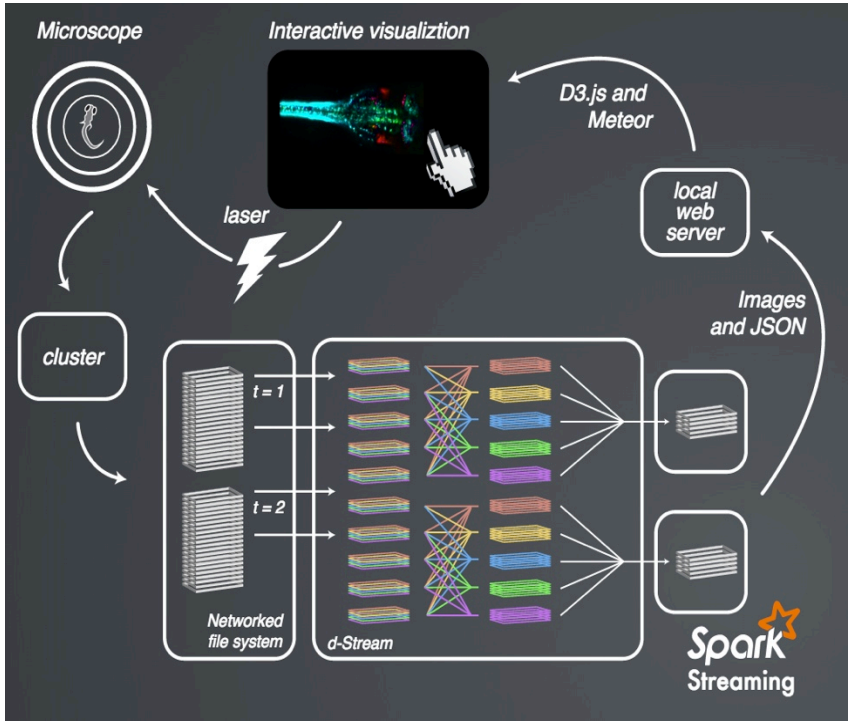
Ease of operations

Neuroscience @ Freeman Lab, Janelia Farm

Spark Streaming and MLlib to analyze neural activities

Laser microscope scans Zebrafish brain→ Spark Streaming → interactive visualization →

laser *ZAP* to kill neurons!



http://www.jeremyfreeman.net/share/talks/spark-summit-2014/

databricks

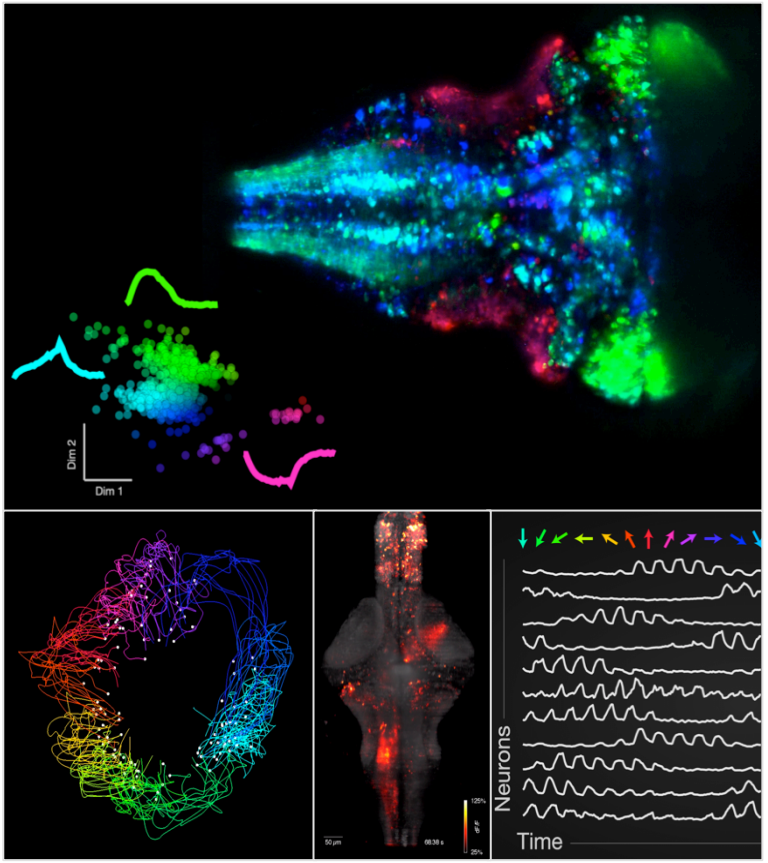
Spark

Neuroscience @ Freeman Lab, Janelia Farm

Streaming machine learning algorithms on time series data of every neuron

Upto 2TB/hour and increasing with brain size

Upto 80 HPC nodes



http://www.jeremyfreeman.net/share/talks/spark-summit-2014/

databricks

Spark

Streaming Machine Learning Algos

- Streaming Linear Regression
- Streaming Logistic Regression
- Streaming KMeans

http://www.jeremyfreeman.net/share/talks/spark-summit-east-2015/#/algorithms-repeat

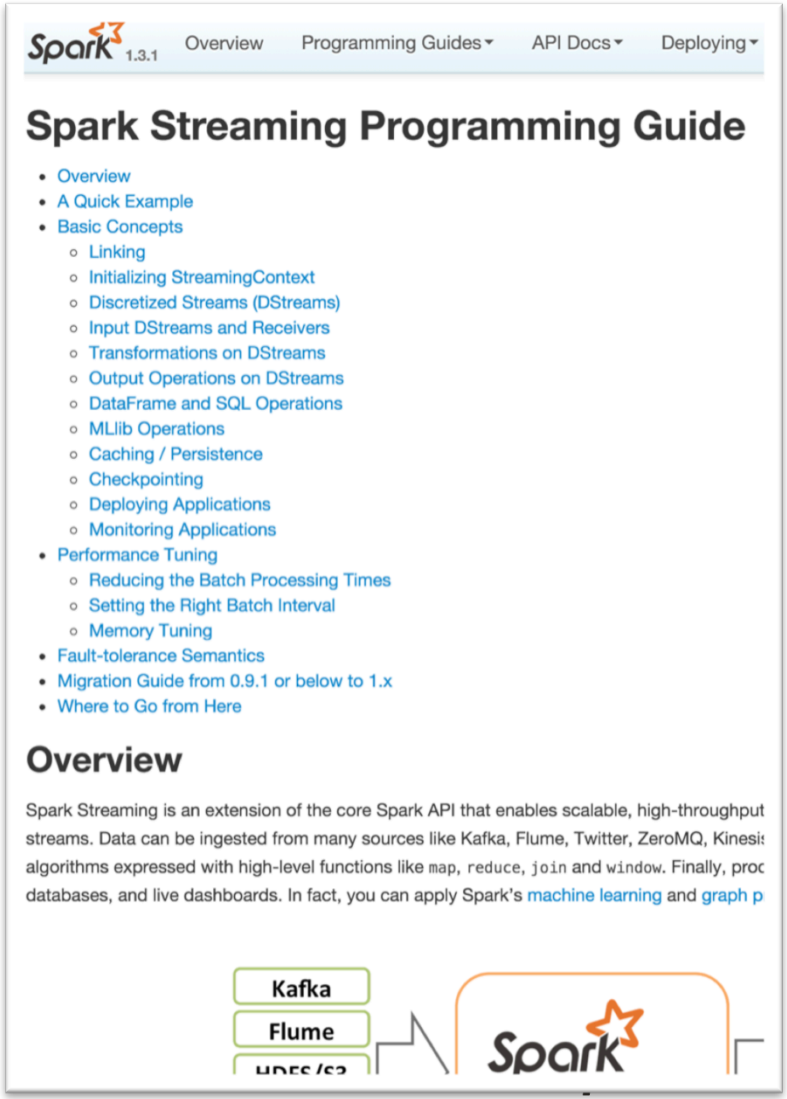
Okay okay, how do I start off?

> Online Streaming Programming Guide

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

> Streaming examples

https://github.com/apache/spark/tree/master/examples/src/main/scala/org/apache/spark/examples/streaming



databricks