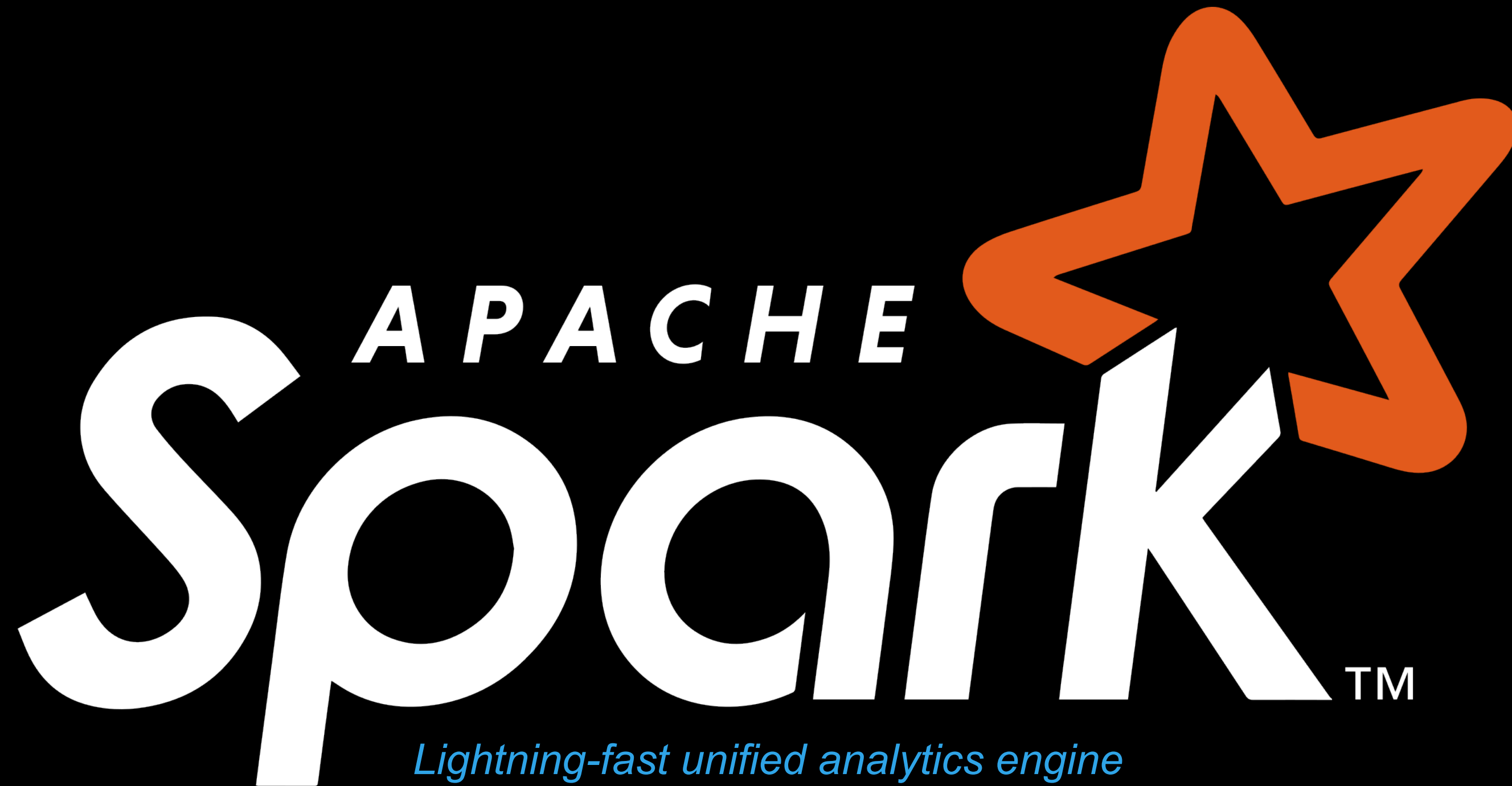


| | | | |
|--|-----------------------|---------------------------------|--|
|  <p>UTT – EBAM – PRO2</p> | Apache Spark | Environment configuration | Introduction to Apache Spark on Databricks |
| Spark Core – Theory | Spark Core – Practice | Introduction to Spark Libraries | Spark SQL |
| Spark MLlib | Spark Streaming | Questions | |



UTT – EBAM – PRO2

Présentation

Vincent

Data team leader – DIY Analytics
Ministère des Armées

Topics

Topics

- Apache Spark
- Data architecture

Prerequisites

- Basic python
- SQL

Le nouveau programme

- Qu'avez-vous appris jusqu'à maintenant ?
- Qu'est ce qui reste flou ?
- Qu'est-ce que vous attendez de ce cours ? Qu'est ce qui vous manque ?
- Ce qu'on recherche / les différents types de profil
 - Data Scientist
 - ML Engineer
 - Data Ops
 - Data Engineer
- Exemple d'un entretien de recrutement

Advice

- Take notes
- Be curious
- Explore links
- Ask questions
- Make it interactive
- Test everything

Slides to be sent by mail

Assessment

- Lectures 50 % (involvement, questions, presence)
- Exam 50 %
 - Multi-choice questions (most questions with one possible answer)
 - Around 25 questions in 30 minutes
 - Questions come from the lessons, the follow-up questions & the exercises

Timetable

- March 29 : Introduction & Spark core
- March 30 : Spark core, spark libraries

Mixed with lessons and practice

- You'll solve exercises with the help of the documentation
- You can share your code with others: <https://codeshare.io/5NAAP4>
- Breaks every 2 hours
- Free question time followed by the exam on Wednesday from 16 pm to 17 pm
- The exam is tomorrow so be sure to understand the concepts, ask questions and practice the exercises.

Environment configuration

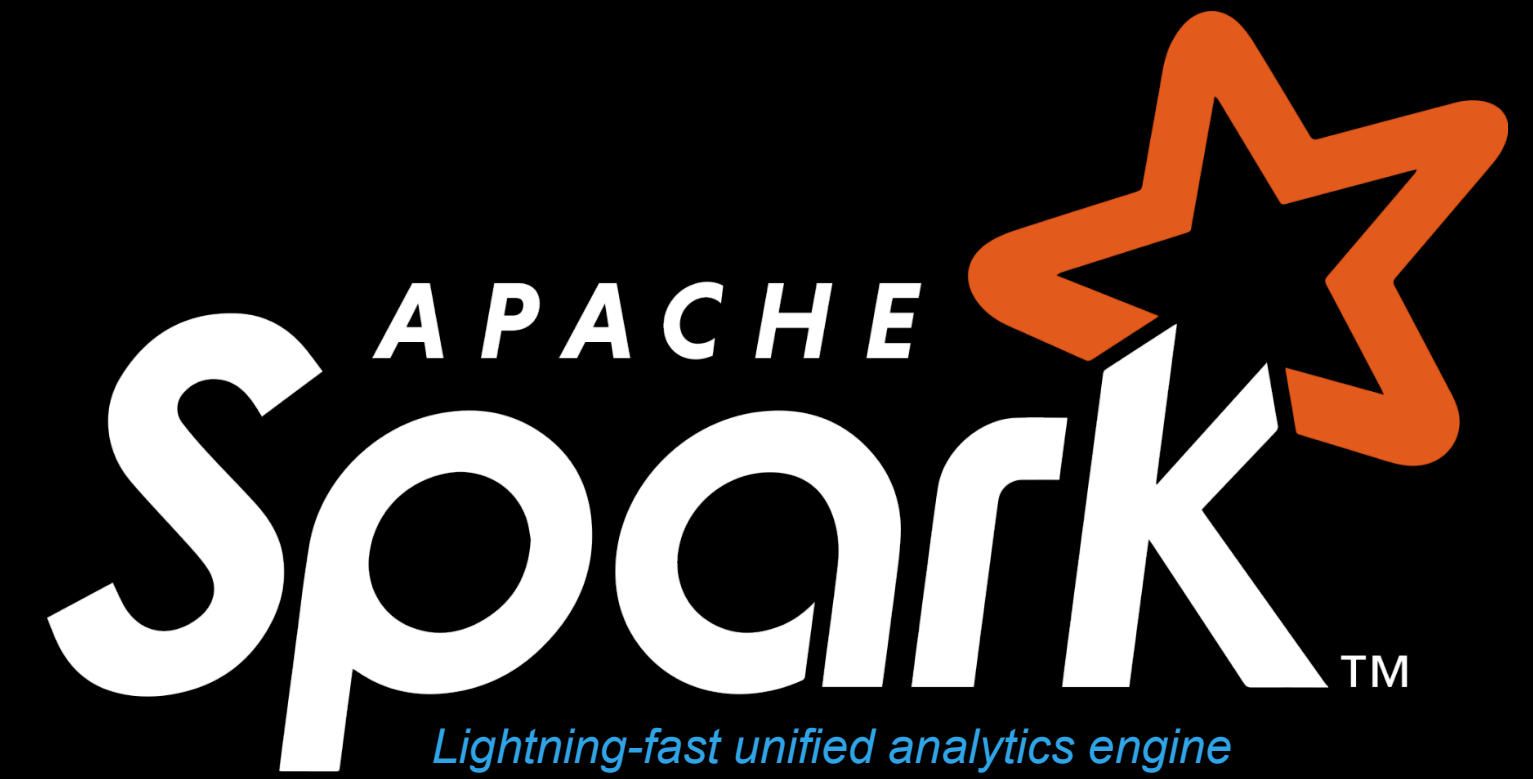
Get access to a cloud Spark cluster for free

1. Register to the Databricks Community Edition :

<https://databricks.com/try-databricks>

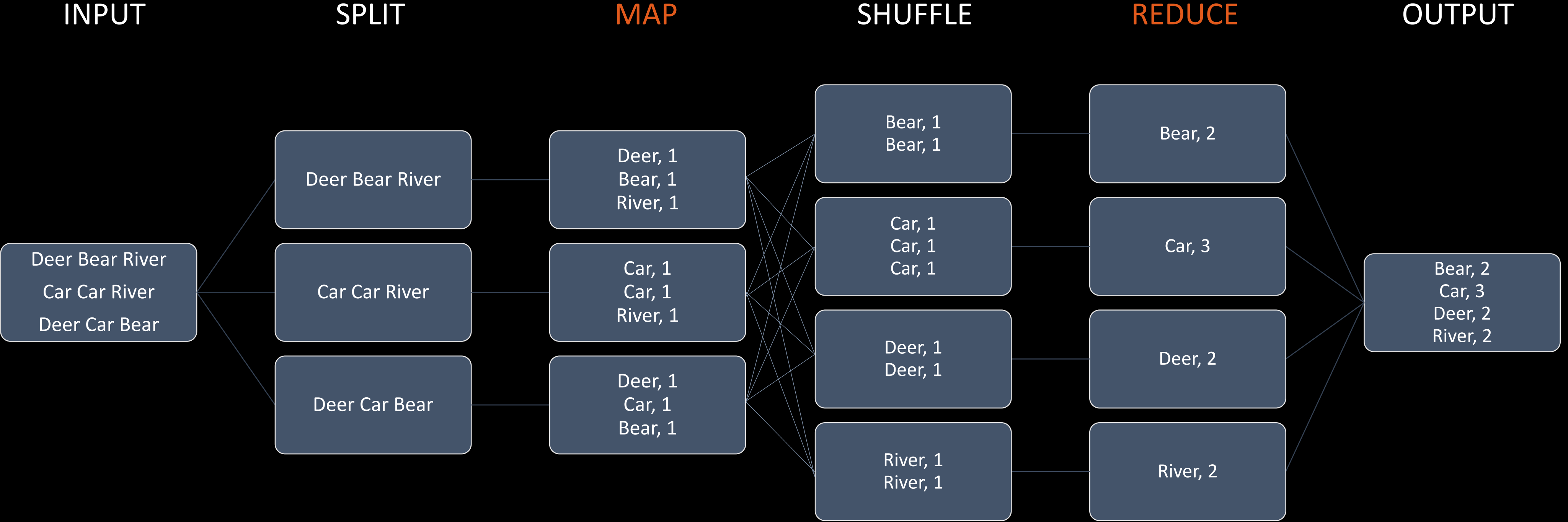
Use temp email : <https://temp-mail.org/fr/>

2. Quickstart notebook



Apache Spark

Hadoop MapReduce



Hadoop MapReduce strengths

Unlimited scale

- Multiple data sources
- Multiple applications
- Multiple users

Enterprise platform

- Reliability
- Resiliency
- Security

Wide range data formats

- Files
- Semi-structured
- Databases

Hadoop MapReduce Weaknesses

Not so easy

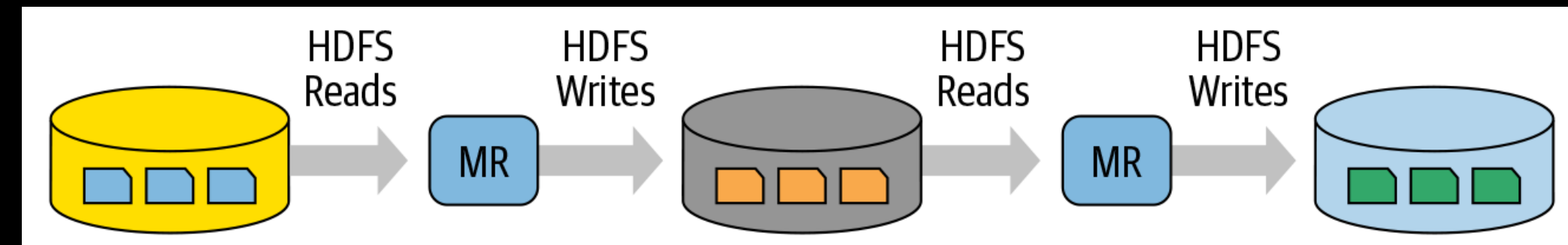
- Need deep Java skills
- Few abstractions available for analysts

Not so efficient

- Not in memory framework
- Application tasks write to disk with each cycle

Not so flexible

- Only suitable for batch workloads
- Rigid processing model



Apache Spark

Ease of development

- Easier API : Python, Scala, Java, R, SQL
- Runs everywhere (Hadoop, Kubernetes, Cloud)

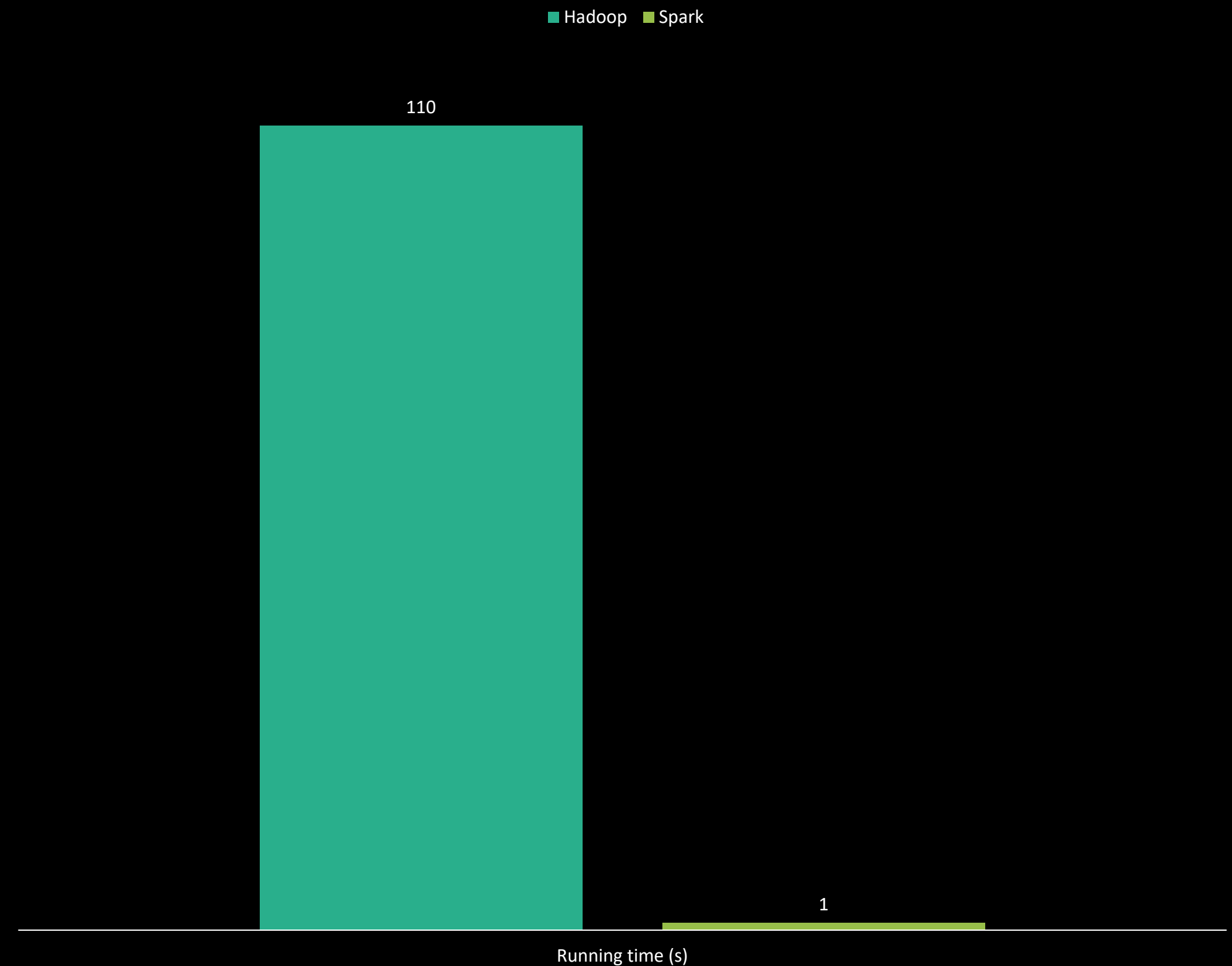
Performance

- RDD
- In memory
- DAGs

Combine workflows

- Micro batch
- Batch
- Interactive
- True streaming (experimental with continuous processing)

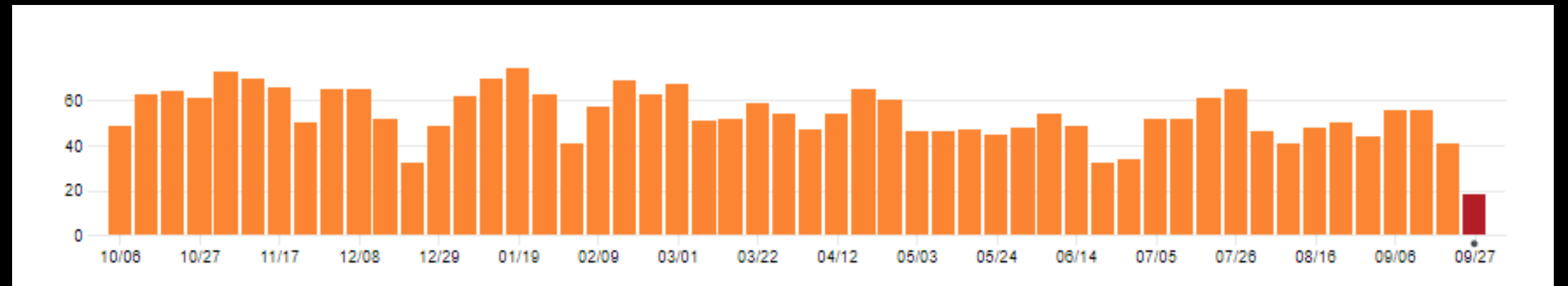
Logistic Regression in Hadoop and Spark



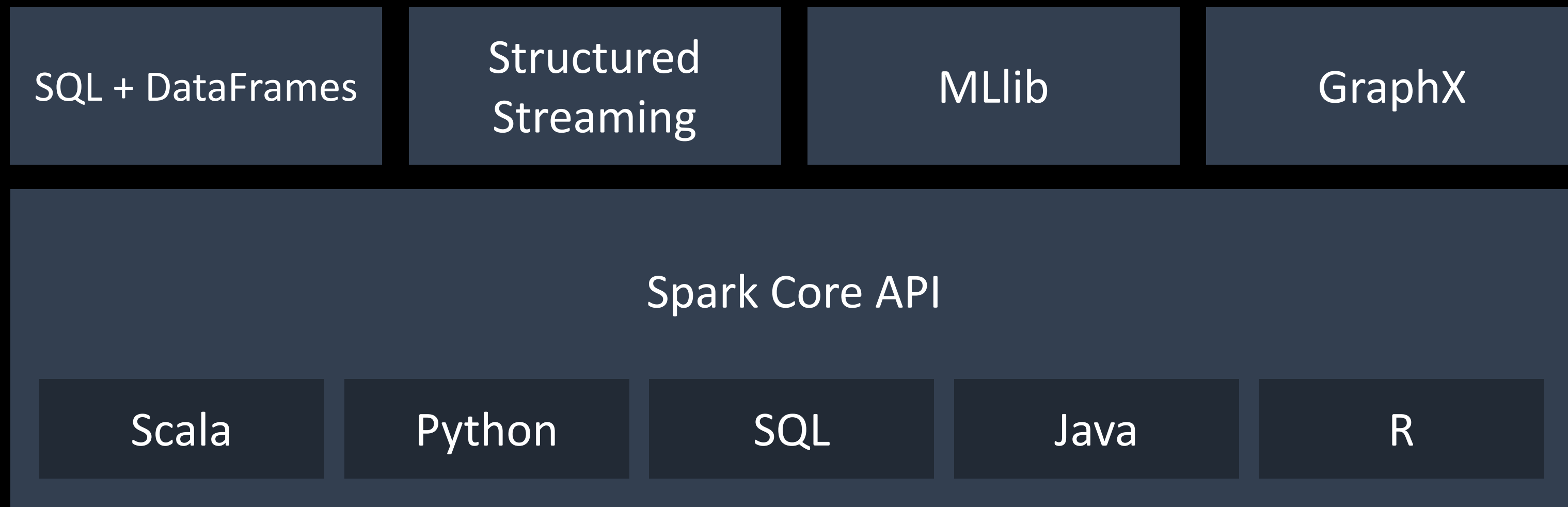
Apache Spark is a unified analytics engine for large-scale data processing.

Spark History: one of the most active open-source projects

- 2004 – MapReduce paper
 - 2010 – Spark paper
 - 2014 – Apache Spark v1
 - 2022 – 3.2.1
-
- Most active project in Hadoop ecosystem
 - Databricks founded by the creators of Spark



Apache Spark ecosystem



Spark Programming Languages

Java

- Too verbose
- The Scala lambda expressions are far more powerful than in Java
- New features released in spark will come a bit late in java

Scala

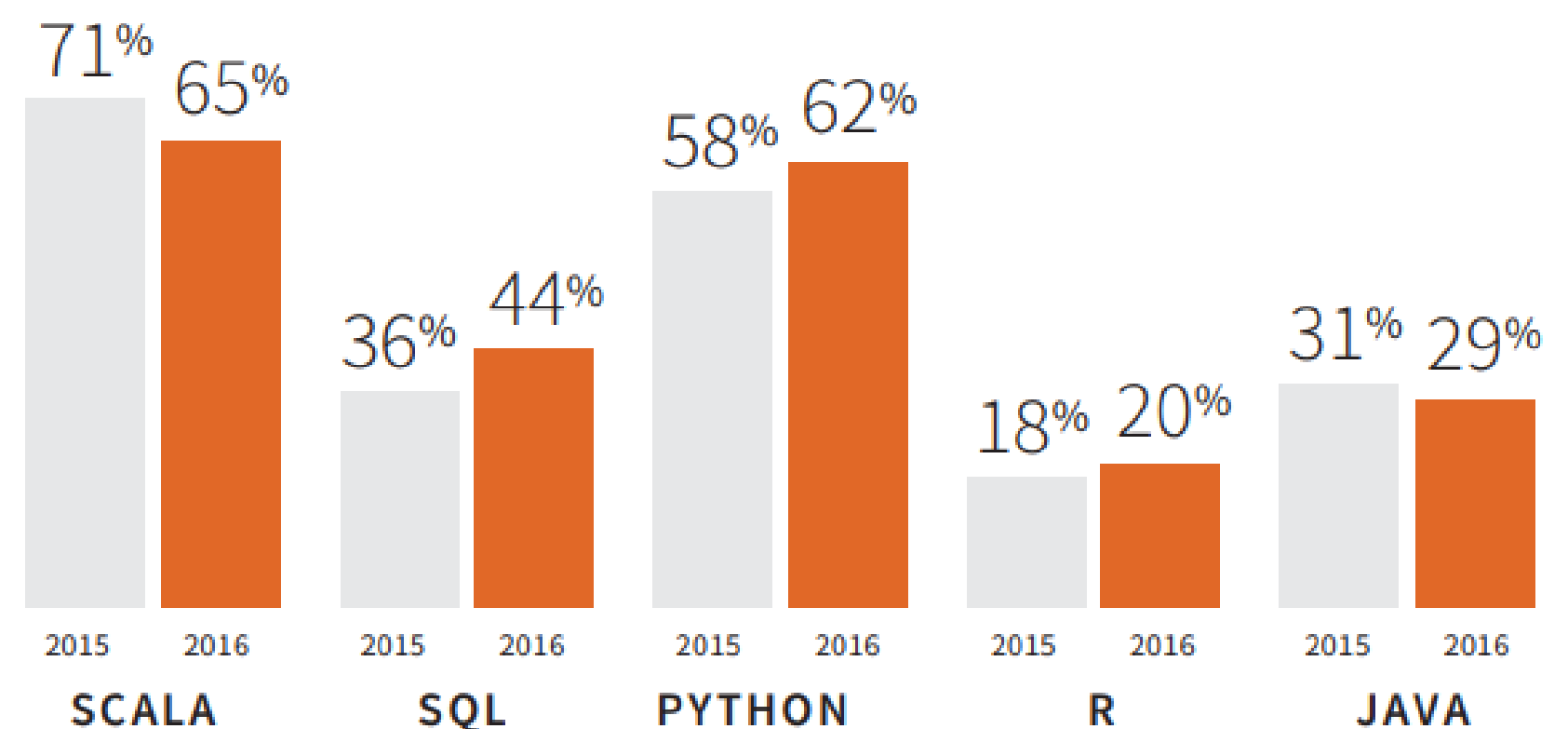
- Functional programming
- Spark written in Scala
- Statically typed
- Hard to understand at beginning

Python

- Always a bit behind Scala in functionality
- Better for data science
- Easy

LANGUAGES USED IN APACHE SPARK

Respondents were allowed to select more than one language.



What Spark is not!

- Not only for Hadoop – Spark can work with Hadoop (especially HDFS), but Spark is a standalone system
- Not a data store – Spark attaches to other data stores but does not provide its own
- Not only for machine learning – Spark includes machine learning and does it very well, but it can handle much broader tasks equally well
- Not a complete streaming engine – Spark Streaming is micro-batching, not true streaming (going to change with continuous processing ?)
- Not a language !

Who uses Spark?

Build models quickly. Iterate faster. Apply intelligence everywhere.

Data Engineer

Put right data to work for the job at hand

Abstract data access complexity

Enable real-time solutions

Machine Learning Engineer

Identify patterns, trends, and risks

Discover new actionable insights

Build new models

Spark Common Use Cases

- Interactive Query
 - Enterprise-scale data volumes accessible to interactive query for business intelligence (BI)
 - Faster time to job completion allows analysts to ask the “next” question about their data & business
- Large-Scale batch
 - Data cleaning to improve data quality (missing data, entity resolution, unit mismatch, etc.)
 - Nightly ETL processing from production systems
- Complex Analytics
 - Forecasting vs. “Nowcasting” (e.g. Google Search queries analyzed for Google Flu Trends to predict outbreaks)
 - Data mining across various types of data
- Event Processing
 - Web server log file analysis (human-readable file formats that are rarely read by humans) in near-real time
 - Responsive monitoring of RFID-tagged devices
- Model Building
 - Predictive modeling answers questions of "what will happen?"
 - Self-tuning machine learning, continually updating algorithms, and predictive modeling

How to develop and run a Spark job?

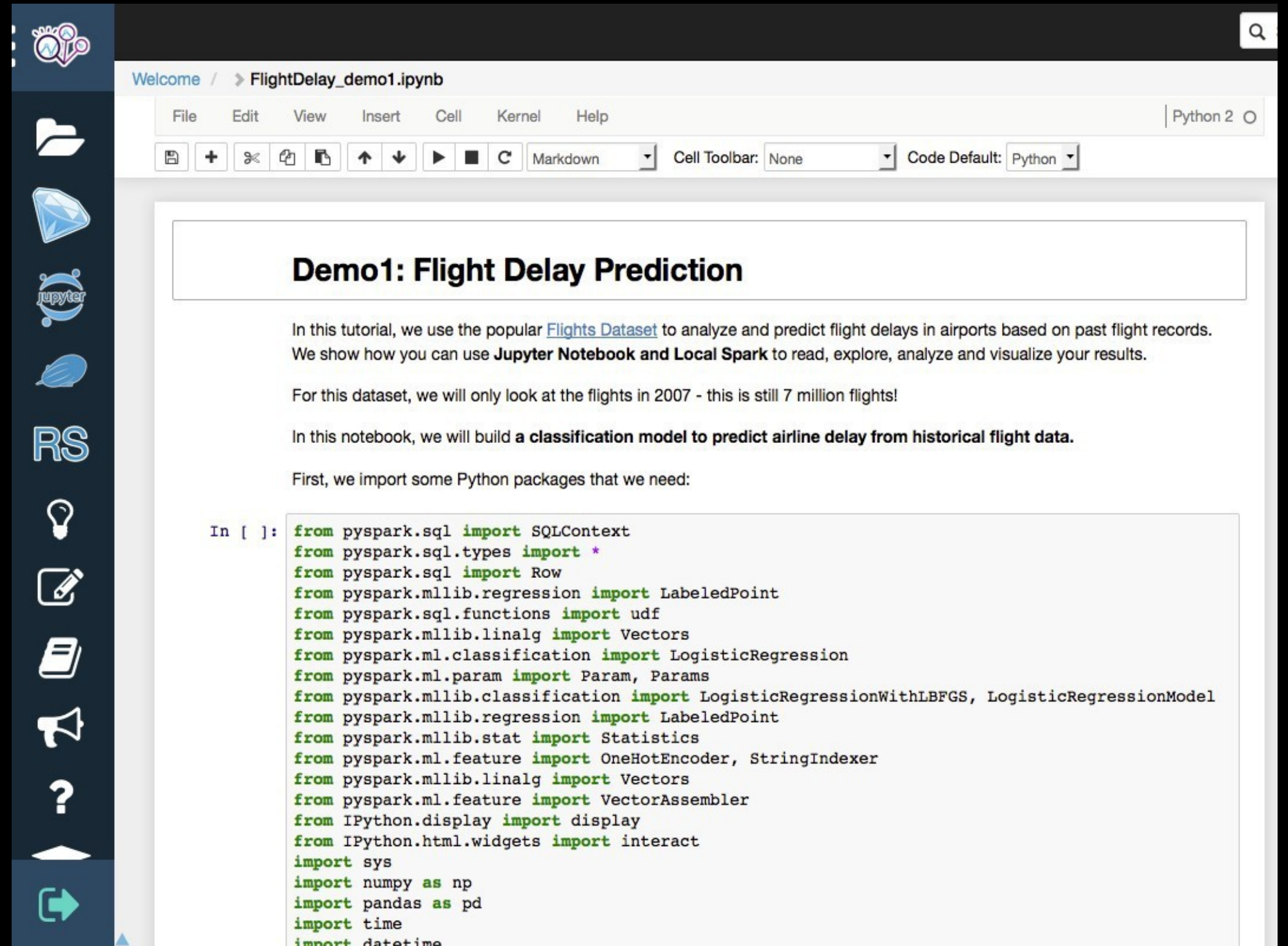
- Interactive analysis
 - spark shell : interactive Scala
 - pyspark : interactive Python
- Batch or streaming analysis
 - Compiled Scala code in a jar file
 - Pyspark scripts
- Notebooks
 - Jupyter, Zeppelin, Databricks



Spark Notebooks

Collaborative web-based environment for

- Data Exploration
- Visualization



The screenshot displays a web-based Spark Notebook interface. On the left is a dark sidebar with various icons for file management, execution, and help. The main area has a light background with a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for saving, adding cells, and running code. The notebook title is 'FlightDelay_demo1.ipynb'. The content includes a title 'Demo1: Flight Delay Prediction', an introductory paragraph about the 'Flights Dataset', and a code cell with Python imports for pyspark and other libraries.

Welcome / > FlightDelay_demo1.ipynb

File Edit View Insert Cell Kernel Help Python 2

File Edit View Insert Cell Kernel Help Python 2

File Edit View Insert Cell Kernel Help Python 2

Demo1: Flight Delay Prediction

In this tutorial, we use the popular [Flights Dataset](#) to analyze and predict flight delays in airports based on past flight records. We show how you can use **Jupyter Notebook** and **Local Spark** to read, explore, analyze and visualize your results.

For this dataset, we will only look at the flights in 2007 - this is still 7 million flights!

In this notebook, we will build a **classification model to predict airline delay from historical flight data**.

First, we import some Python packages that we need:

```
In [ ]: from pyspark.sql import SQLContext
from pyspark.sql.types import *
from pyspark.sql import Row
from pyspark.mllib.regression import LabeledPoint
from pyspark.sql.functions import udf
from pyspark.mllib.linalg import Vectors
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.param import Param, Params
from pyspark.mllib.classification import LogisticRegressionWithLBFGS, LogisticRegressionModel
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.stat import Statistics
from pyspark.ml.feature import OneHotEncoder, StringIndexer
from pyspark.mllib.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from IPython.display import display
from IPython.html.widgets import interact
import sys
import numpy as np
import pandas as pd
import time
import datetime
```


Web-Based Notebooks

Zeppelin

- More native connectors
- Possibility to use different languages in same notebook

Jupyter

- More mature
- More used
- More documentation
- More language interpreters
- More exports formats
- Auto-completion
- Open-source collaborative hub (JupyterHub)

Introduction to Apache Spark on Databricks

Introduction to Apache Spark on Databricks

1. Gentle introduction databricks notebook
2. Introduction notebook: <https://tinyurl.com/ebam-spark-intro>

Spark Core – Theory

RDDs, DataFrames, DataSets

- DataSets is the new default object to interact with Spark from Spark 2.0
- DataFrames were introduced in release 1.3 with a more structured format to enable SQL and Spark optimizations
- RDD (Resilient Distributed Dataset) is the primary and only way to interact with earlier Spark versions

DataFrames vs DataSets

The 2 APIs are merged

DataFrames

- Data organized into named columns
- Untyped
- Python and R

DataSets

- Typed
- Scala and Java

Spark Transformations

| Transformation | Meaning |
|---|---|
| map(func) | Return a new dataset formed by passing each element of the source through a function func . |
| filter(func) | Return a new dataset formed by selecting those elements of the source on which func returns true. |
| flatMap(func) | Similar to map, but each input item can be mapped to 0 or more output items. So func should return a Seq rather than a single item. |
| join(otherDataset, [numTasks]) | When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. |
| reduceByKey(func) | When called on a dataset of (K, V) pairs, returns a dataset of (K,V) pairs where the values for each key are aggregated using the given reduce function func . |
| sortByKey([ascending], [numTasks]) | When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K,V) pairs sorted by keys in ascending or descending order. |

```

val arrayStructureData = Seq(
  Row("James,,Smith",List("Java","Scala","C++"),"CA"),
  Row("Michael,Rose,",List("Spark","Java","C++"),"NJ"),
  Row("Robert,,Williams",List("CSharp","VB","R"),"NV")
)

val arrayStructureSchema = new StructType()
  .add("name",StringType)
  .add("languagesAtSchool", ArrayType(StringType))
  .add("currentState", StringType)

val df = spark.createDataFrame(
  spark.sparkContext.parallelize(arrayStructureData),arrayStructureSchema)
import spark.implicits._

//flatMap() Usage
val df2=df.flatMap(f=> f.getSeq[String](1).map((f.getString(0),_,f.getString(2))))
  .toDF("Name","language","State")

df2.show(false)

```


| <i>Name</i> | <i>Language</i> | <i>State</i> |
|-------------------------|-----------------|--------------|
| <i>James,,Smith</i> | <i>Java</i> | <i>CA</i> |
| <i>James,,Smith</i> | <i>Scala</i> | <i>CA</i> |
| <i>James,,Smith</i> | <i>C++</i> | <i>CA</i> |
| <i>Michael,Rose,</i> | <i>Spark</i> | <i>NJ</i> |
| <i>Michael,Rose,</i> | <i>Java</i> | <i>NJ</i> |
| <i>Michael,Rose,</i> | <i>C++</i> | <i>NJ</i> |
| <i>Robert,,Williams</i> | <i>CSharp</i> | <i>NV</i> |
| <i>Robert,,Williams</i> | <i>VB</i> | <i>NV</i> |
| <i>Robert,,Williams</i> | <i>R</i> | <i>NV</i> |

Spark Actions

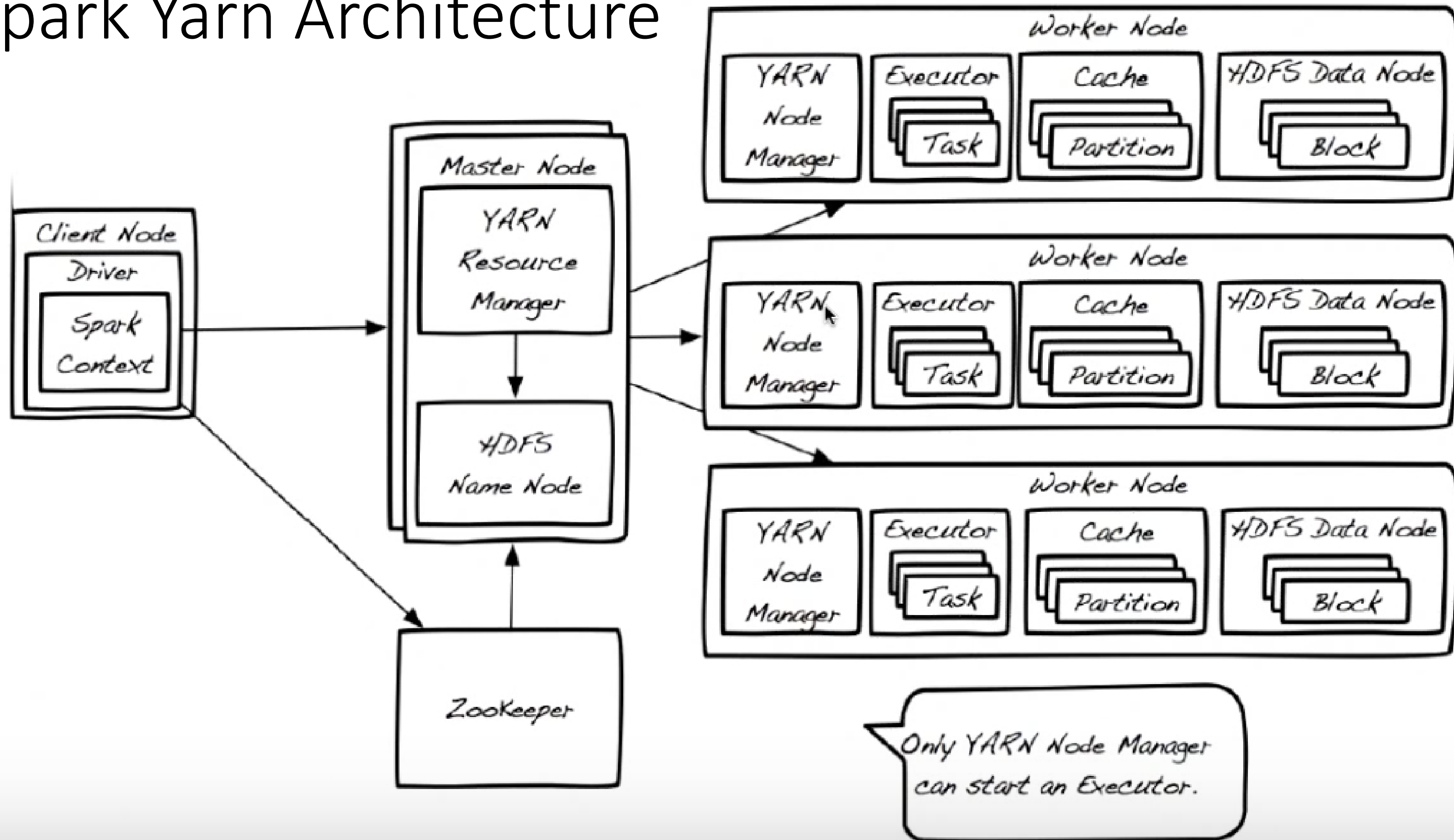
Actions return values or save DataFrames to disks

| Action | Meaning |
|-----------------------------------|--|
| <code>collect()</code> | Return all the elements of the dataset as an array of the driver program. This is usually useful after a filter or another operation that returns a sufficiently small subset of data. |
| <code>count()</code> | Return the number of elements in a dataset. |
| <code>first()</code> | Return the first element of the dataset. |
| <code>take(n)</code> | Return an array with the first <code>n</code> elements of the dataset. |
| <code>foreach(func)</code> | Run a function <code>func</code> on each element of the dataset. Does not return any RDD. |
| <code>saveAsTextFile(path)</code> | Save the RDD into a TextFile at the given <code>path</code> . |

Spark cluster manager

| Mode | Spark driver | Spark executor | Cluster manager |
|----------------|--|---|---|
| Local | Runs on a single JVM, like a laptop or single node | Runs on the same JVM as the driver | Runs on the same host |
| Standalone | Can run on any node in the cluster | Each node in the cluster will launch its own executor JVM | Can be allocated arbitrarily to any host in the cluster |
| YARN (cluster) | Runs with the YARN Application Master | YARN's NodeManager's container | YARN's Resource Manager works with YARN's Application Master to allocate the containers on NodeManagers for executors |
| Kubernetes | Runs in a Kubernetes pod | Each worker runs within its own pod | Kubernetes Master |

Spark Yarn Architecture



Pipelining & Dags

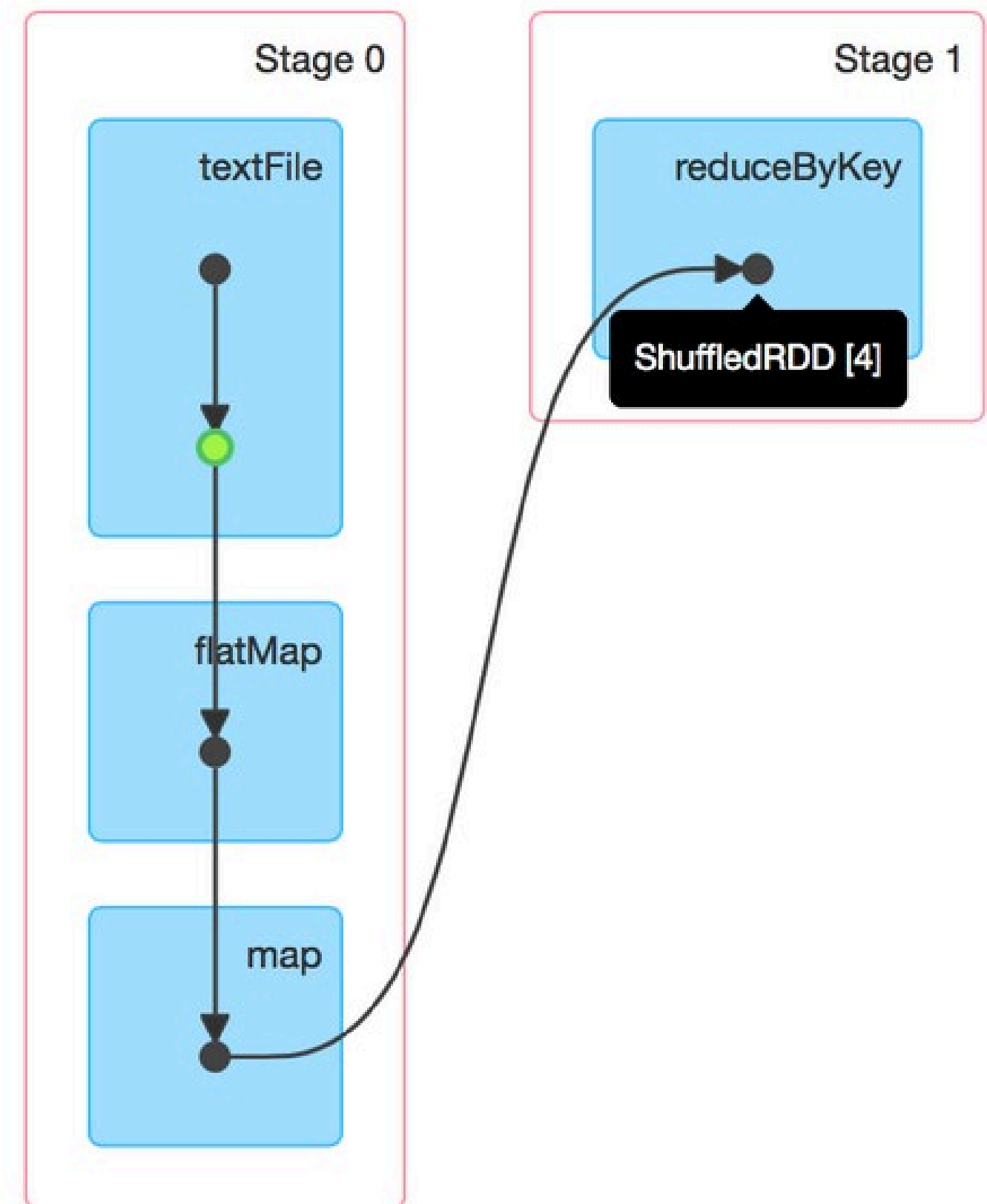
Details for Job 0

Status: SUCCEEDED

Completed Stages: 2

► Event Timeline

▼ DAG Visualization



Details for Job 4

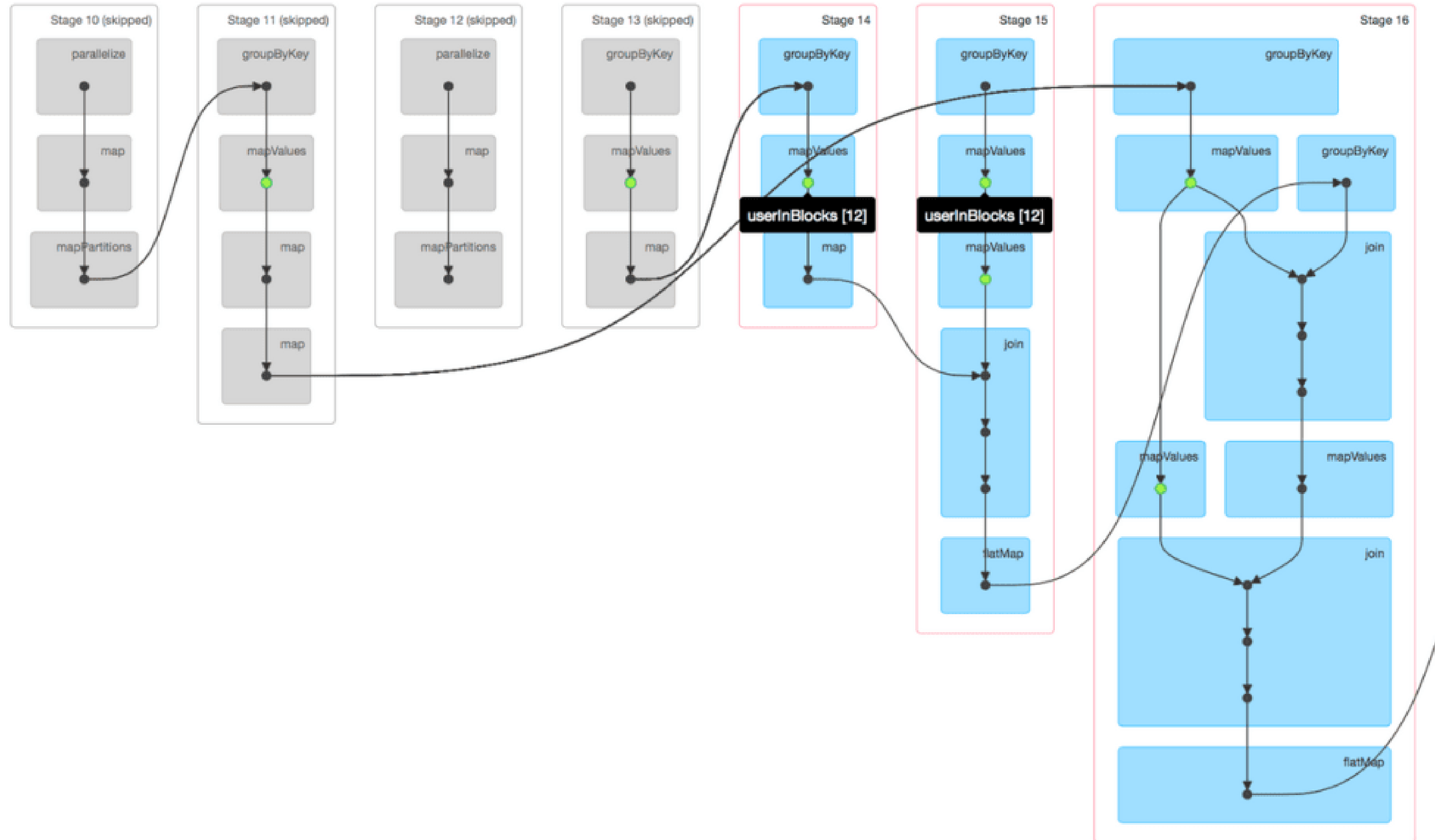
Status: SUCCEEDED

Completed Stages: 22

Skipped Stages: 4

▸ Event Timeline

▾ DAG Visualization



Dataframe Persistence

- Each node stores any partitions of the cache that it computes in memory
- Reuses them in other actions on that dataset (or datasets derived from it)

| Storage Level | Meaning |
|---|---|
| MEMORY_ONLY | Store as deserialized Java objects in the JVM. If the RDD does not fit in memory, part of it will be cached. The other will be recomputed as needed. This is the default. The cache() method uses this. |
| MEMORY_AND_DISK | Same except also store on disk if it doesn't fit in memory. Read from memory and disk when needed. |
| MEMORY_ONLY_SER | Store as serialized Java objects (one byte array per partition). Space efficient, but more CPU intensive to read. |
| MEMORY_AND_DISK_SER | Similar to MEMORY_AND_DISK but stored as serialized objects. |
| DISK_ONLY | Store only on disk. |
| MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc. | Same as above, but replicate each partition on two cluster nodes. |

Hands-on

Databricks Data engineering notebook

Spark Word Count

```
1 import java.io.BufferedReader;
2 import java.io.FileReader;
3 import java.io.IOException;
4 import java.net.URI;
5 import java.util.ArrayList;
6 import java.util.HashSet;
7 import java.util.List;
8 import java.util.Set;
9 import java.util.StringTokenizer;
10
11 import org.apache.hadoop.conf.Configuration;
12 import org.apache.hadoop.fs.Path;
13 import org.apache.hadoop.io.IntWritable;
14 import org.apache.hadoop.io.Text;
15 import org.apache.hadoop.mapreduce.Job;
16 import org.apache.hadoop.mapreduce.Mapper;
17 import org.apache.hadoop.mapreduce.Reducer;
18 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
19 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
20 import org.apache.hadoop.mapreduce.Counter;
21 import org.apache.hadoop.util.GenericOptionsParser;
22 import org.apache.hadoop.util.StringUtils;
23
24 public class WordCount2 {
25
26     public static class TokenizerMapper
27         extends Mapper<Object, Text, Text, IntWritable>{
28
29         static enum CountersEnum { INPUT_WORDS }
30
31         private final static IntWritable one = new IntWritable(1);
32         private Text word = new Text();
33
34         private boolean caseSensitive;
35         private Set<String> patternsToSkip = new HashSet<String>();
36
37         private Configuration conf;
38         private BufferedReader fis;
39
40         @Override
41         public void setup(Context context) throws IOException,
42             InterruptedException {
43             conf = context.getConfiguration();
44             caseSensitive = conf.getBoolean("wordcount.case.sensitive", true);
45             if (conf.getBoolean("wordcount.skip.patterns", true)) {
46                 URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
47                 for (URI patternsURI : patternsURIs) {
48                     Path patternsPath = new Path(patternsURI.getPath());
49                     String patternsFileName = patternsPath.getName().toString();
50                     parseSkipFile(patternsFileName);
51                 }
52             }
53         }
54     }
```

```
55     private void parseSkipFile(String fileName) {
56         try {
57             fis = new BufferedReader(new FileReader(fileName));
58             String pattern = null;
59             while ((pattern = fis.readLine()) != null) {
60                 patternsToSkip.add(pattern);
61             }
62         } catch (IOException ioe) {
63             System.err.println("Caught exception while parsing the cached file '"
64                 + StringUtils.stringifyException(ioe));
65         }
66     }
67
68     @Override
69     public void map(Object key, Text value, Context context
70         ) throws IOException, InterruptedException {
71         String line = (caseSensitive) ?
72             value.toString() : value.toString().toLowerCase();
73         for (String pattern : patternsToSkip) {
74             line = line.replaceAll(pattern, "");
75         }
76         StringTokenizer itr = new StringTokenizer(line);
77         while (itr.hasMoreTokens()) {
78             word.set(itr.nextToken());
79             context.write(word, one);
80             Counter counter = context.getCounter(CountersEnum.class.getName(),
81                 CountersEnum.INPUT_WORDS.toString());
82             counter.increment(1);
83         }
84     }
85
86     public static class IntSumReducer
87         extends Reducer<Text, IntWritable, Text, IntWritable> {
88         private IntWritable result = new IntWritable();
89
90         public void reduce(Text key, Iterable<IntWritable> values,
91             Context context
92         ) throws IOException, InterruptedException {
93             int sum = 0;
94             for (IntWritable val : values) {
95                 sum += val.get();
96             }
97             result.set(sum);
98             context.write(key, result);
99         }
100     }
101 }
```

```
1 val textFile = sc.textFile("hdfs://...")
2 val counts = textFile.flatMap(line => line.split(" "))
3                       .map(word => (word, 1))
4                       .reduceByKey(_ + _)
5 counts.saveAsTextFile("hdfs://...")
```

```
103 public static void main(String[] args) throws Exception {
104     Configuration conf = new Configuration();
105     GenericOptionsParser optionParser = new GenericOptionsParser(conf, args);
106     String[] remainingArgs = optionParser.getRemainingArgs();
107     if (!(remainingArgs.length != 2 || remainingArgs.length != 4)) {
108         System.err.println("Usage: wordcount <in> <out> [-skip skipPatternFile]");
109         System.exit(2);
110     }
111     Job job = Job.getInstance(conf, "word count");
112     job.setJarByClass(WordCount2.class);
113     job.setMapperClass(TokenizerMapper.class);
114     job.setCombinerClass(IntSumReducer.class);
115     job.setReducerClass(IntSumReducer.class);
116     job.setOutputKeyClass(Text.class);
117     job.setOutputValueClass(IntWritable.class);
118
119     List<String> otherArgs = new ArrayList<String>();
120     for (int i=0; i < remainingArgs.length; ++i) {
121         if ("-skip".equals(remainingArgs[i])) {
122             job.addCacheFile(new Path(remainingArgs[++i]).toUri());
123             job.getConfiguration().setBoolean("wordcount.skip.patterns", true);
124         } else {
125             otherArgs.add(remainingArgs[i]);
126         }
127     }
128     FileInputFormat.addInputPath(job, new Path(otherArgs.get(0)));
129     FileOutputFormat.setOutputPath(job, new Path(otherArgs.get(1)));
130
131     System.exit(job.waitForCompletion(true) ? 0 : 1);
132 }
133 }
```

Spark Properties

Inside job →

```
[13] from pyspark.sql import SparkSession

      spark = SparkSession.builder \
        .config("spark.executor.memory", "1g") \
        .appName('abc') \
        .master('local[*']) \
        .getOrCreate()
```

Dynamically setting Spark properties :

Supply the configuration values during runtime:

```
[ ] ./bin/spark-submit --name "My app" --master local[4] --conf spark.shuffle.spill=false myApp.jar conf/spark-defaults.conf
```

- Application web UI: <http://<driver>:4040>
- Spark History web UI: <http://<driver>:18080>

Executing a Spark job

- Pyspark : Python REPL
- spark-shell : Scala REPL
- sparkR : R REPL
- spark-submit: execute a Scala/Java compiled .jar file, a Python .py script or a R .r script

Example: `./bin/spark-submit examples/src/main/python/pi.py 10`

Deploy mode:

- Standalone Mode
- Apache Hadoop YARN
- Kubernetes

Spark Final Thoughts

- Spark is very fast
 - Easy to use framework
 - Powerful abstractions : DataFrames
 - Big higher level libraries: SparkSQL, SparkML, Streaming, GraphX
 - Huge ecosystem adoption
 - Great community and enterprise support
-
- This is a very fast paced environment, so keep up !
 - Lot of new features at each new release (major release every 3 months)
 - Spark has the latest / best offer but things may change again

Spark Core – Practice

Notebook

1. Go through the notebook [0 spark rdd pyspark.ipynb](#) on spark rdd
2. When you are down, go to the movielens-exercises.ipynb notebook

MovieLens

- <http://grouplens.org/datasets/movielens/latest/>
- MovieLens 20M dataset
- 138k users (Netflix announced 69M users)
- 27k movies (Netflix probably cast between 10k and 100k different movies)
- 20M ratings (Sparse Matrix)

More exercises

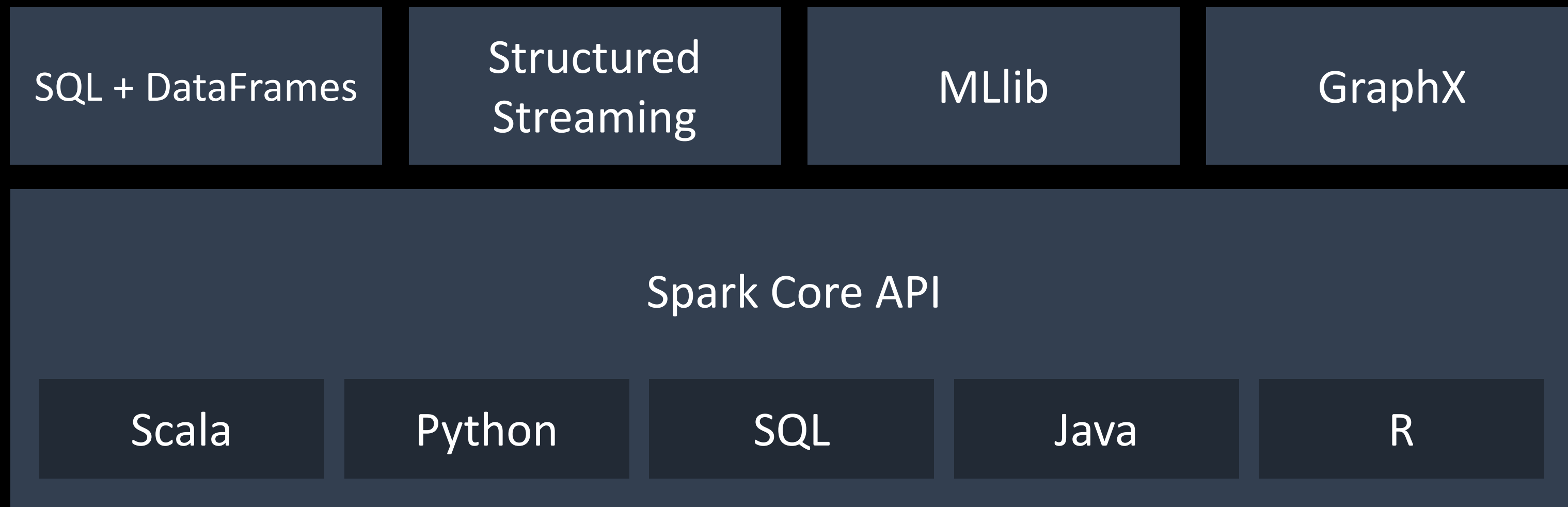
- Use the timestamp column from the ratings
 - Convert it as a human-readable time/date
 - Check the evolution of ratings versus time?
-
- Is that something related to time or users with the .5 notation?

Explode the genres column and compute statistics such as :

- Distribution
- Average rating by genre

Introduction to Spark Libraries

Apache Spark ecosystem



Spark SQL

What are structured / semi-structured / not structured data?

- Structured has a schema with type that is usually control on read/write
- Semi-structured can have an easy schema on read JSON/XML or implicit structure with no type
- Not structure is plain text or image
- Less structured data over time with IoT, Big Data, etc.

What is SparkSQL?

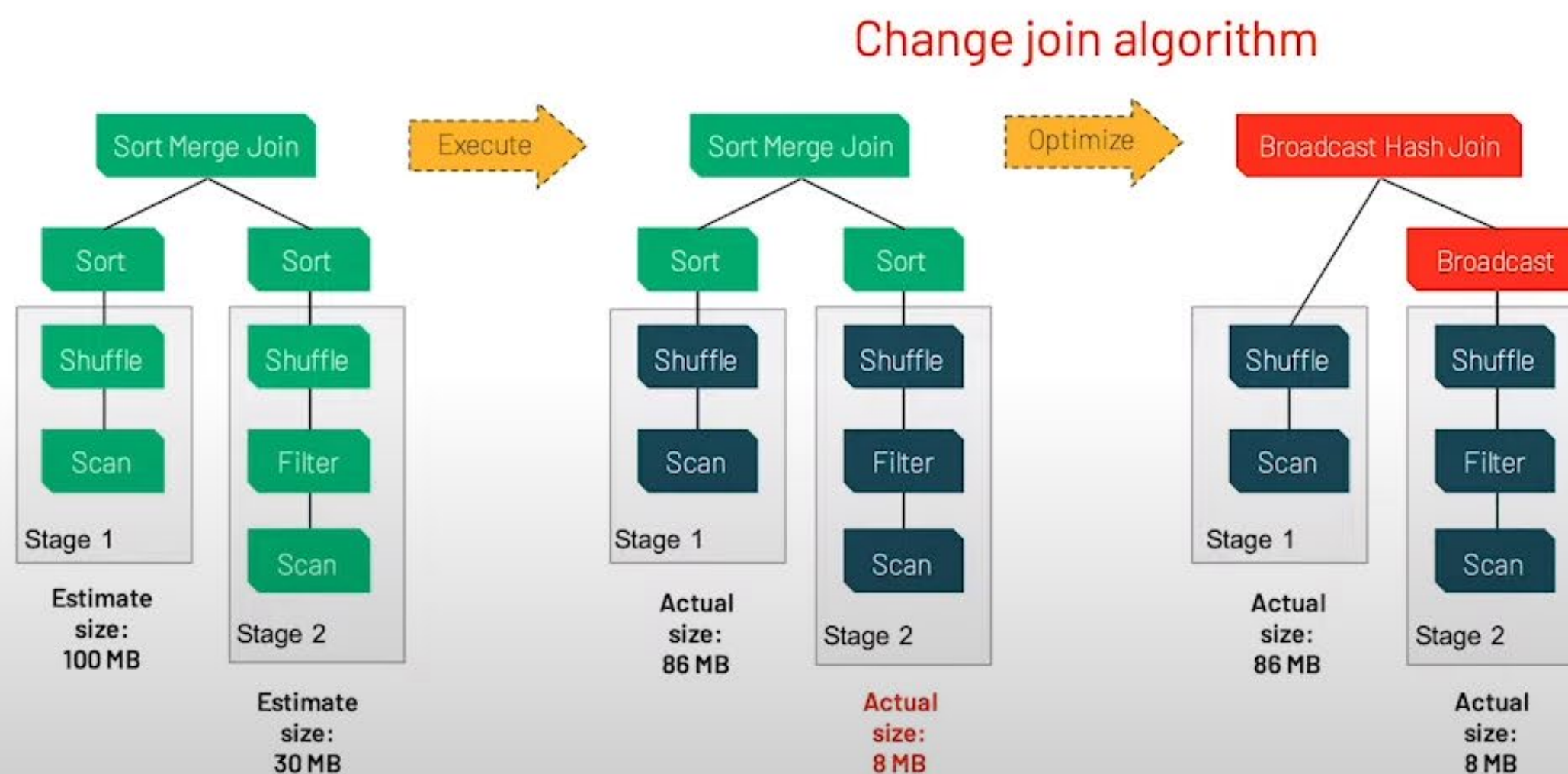
Spark SQL is Apache Spark's module for working with structured data

- Define a DataFrame object for structured data
- Let users seamlessly mix SQL queries with Spark programs
- Use either SQL or a familiar DataFrame API
- Connect to multiple data sources and data formats from an unified API
 - Sources include Swift, S3, HDFS, Hive or any JDBC database
 - Formats include ORC, JSON, CSV or Parquet
 - Reuse Hive data, queries, UDFs or SerDes

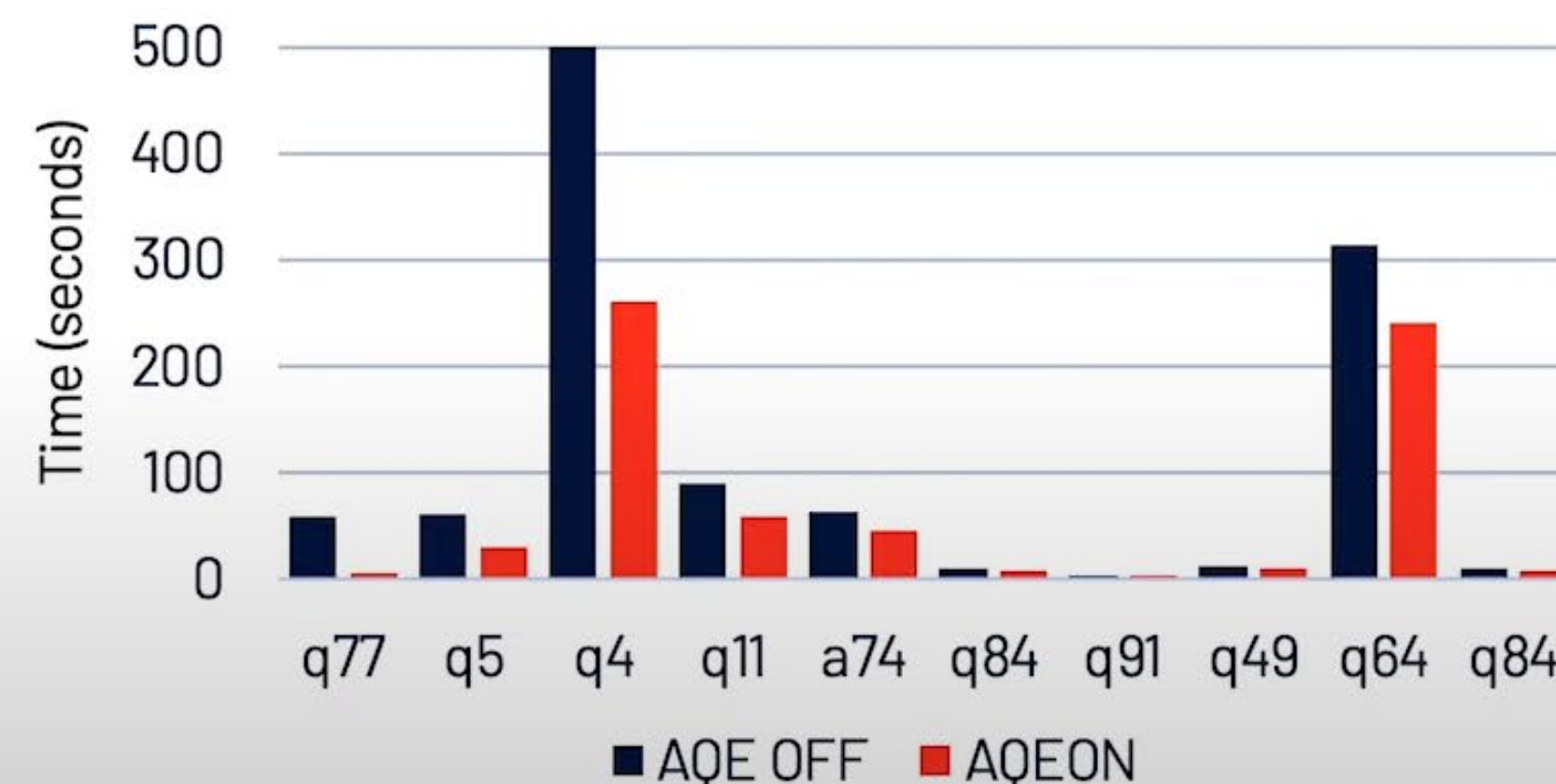


3.0: SQL Engine

Adaptive Query Execution (AQE): change execution plan at runtime to automatically set # of reducers and join algorithms



TPC-DS 1 TB No-Stats with and without AQE



Accelerates TPC-DS queries up to 8x

PLUS DE VIDÉOS

SPARK+AI SUMMIT

29/03/2022

13:03 / 28:59

Spark

#Datateams #SparkAISummit



YouTube



Exercises

Notebook Spark SQL

Spark MLlib

Spark MLlib

Make machine learning scalable

- ML Algorithms : common learning algorithms such as classification, regression, clustering
- Featurization : feature extraction, transformation, dimensionality reduction
- Pipelines : tools for constructing, evaluating, and tuning ML Pipelines
- Persistence : saving and load algorithms, models, and Pipelines
- Utilities : linear algebra, statistics, data handling, etc.

SparkML Pipelines

A pipeline defines a Machine Learning Workflow

Combination of :

- DataFrame : same as before
- Transformer : an algorithm to transform an input DataFrame into an other DataFrame, for instance with predictions. (Ex : ML model)
- Estimator : an algorithm which can be fit on a DataFrame to produce a Transformer, for instance a learning algorithm that can be fitted on a dataset and produce a model (which is a Transformer) (ex : ML algorithm)
- Pipeline : chains multiple Transformers and Estimators together to build an ML workflow
- Parameter : a common API for specifying parameters to Transformers and Estimators

Transformers

Transformers can be :

- Feature transformers
 - e.g. a transformer that read a column (e.g. text) and map it into a new column (e.g. feature vectors), returning the complete DataFrame
- Learned models
 - e.g. a learning model that read the feature vectors, predict the label for each feature vector and output the complete DataFrame with predicted labels
- Both transform a DataFrame into an other DataFrame, appending one or more columns

Estimators

- Estimators abstract the concept of a learning algorithm that fits or trains on a dataset
- Estimators implement the method `.fit()` which accept a `DataFrame` and produce a `Model`, which is a `Transformer`

e.g. `LogisticRegression` is an Estimator, `LogisticRegression.fit()` trains a `LogisticRegressionModel` which is a `Model` and hence a `Transformer`

Pipeline

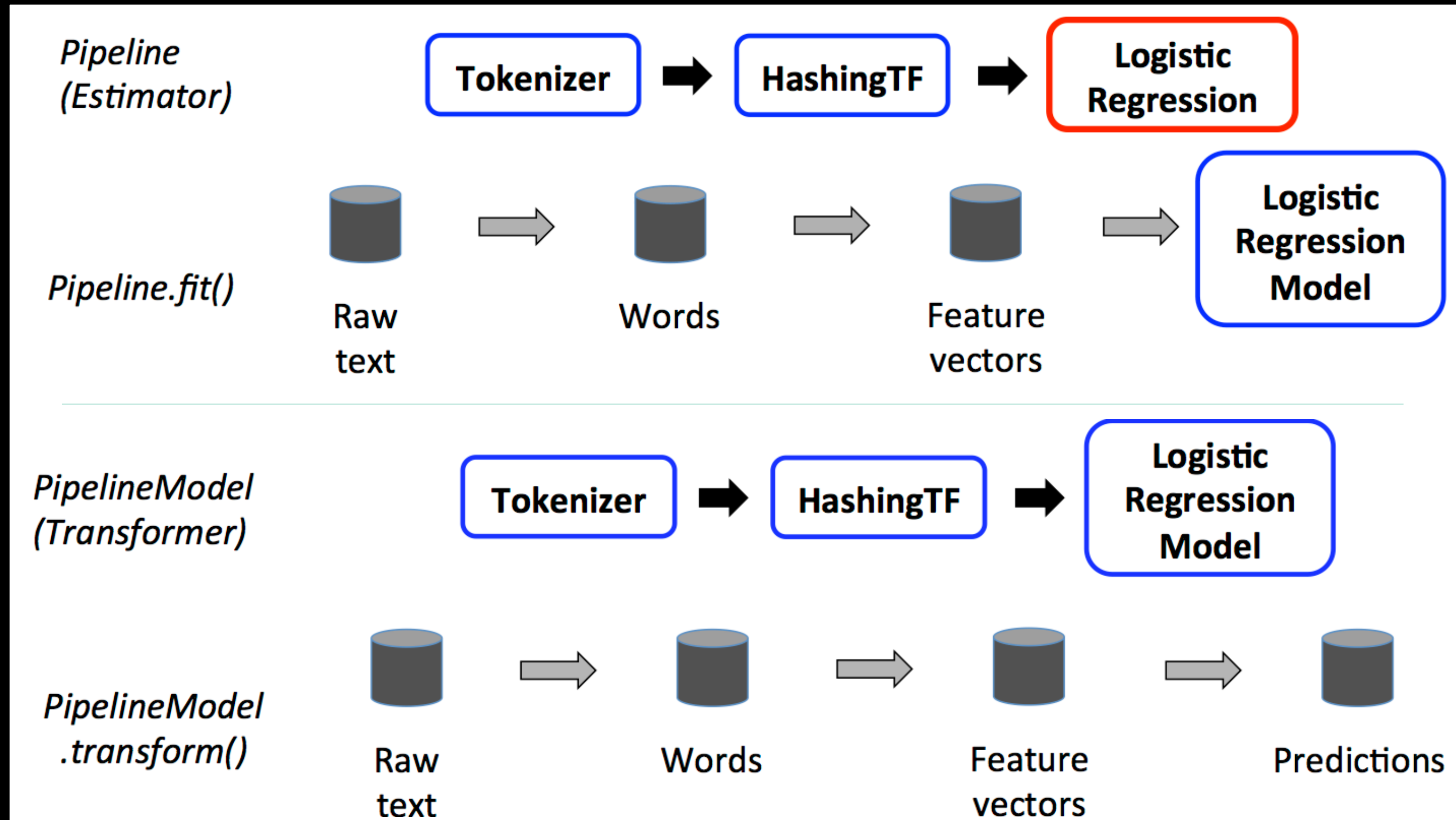
- A Machine Learning workflow usually follows a pipeline of transformations and learning algorithms going through features engineering to model training
- In SparkML, this is represented as a Pipeline which is a sequence of PipelineStages (Transformers and Estimators) that are run in order
- For every Transformer, the `.transform()` method is called outputting a new DataFrame
- For every Estimator, the `fit()` method is called, outputting a new Model (Transformer)

Pipeline Example

Document
Classification

Transformers

Estimator



Pipeline Usage

- The built Pipeline is an Estimator and thus, can be fitted on a DataFrame (test data), resulting in a Model that can be then applied as a Transformer to a DataFrame (to-predict data)
- Pipeline.fit() produces a PipelineModel, which is a Transformer and can be used at test time
- The PipelineModel has the same numbers of steps as the Pipeline but every Estimator from the Pipeline is replaced in the PipelineModel by a Transformer (see following example)
- This unified Pipeline process helps ensure that training and test data go through the exact same steps

Model Selection: Hyperparameter Optimization

Hyperparameter Optimization or Tuning is the process of choosing the Estimator or Pipeline parameters that produce the best model / results

To tune your Estimator or Pipeline, you need:

- a parameter grid: a set of ParamMaps to search over
- a metric to measure how well a fitted Model is performing: an Evaluator
- Model Selection Tools such as CrossValidator and TrainValidationSplit

Use the following process to measure Model performance:

1. Split the input data between training and test datasets
2. For each (training, test) pair and for each ParamMap, fit the Estimator or Pipeline and evaluate the fitted Model using the specified Evaluator
3. Select the Model that shows the best results

Announcing Koalas 1.0!



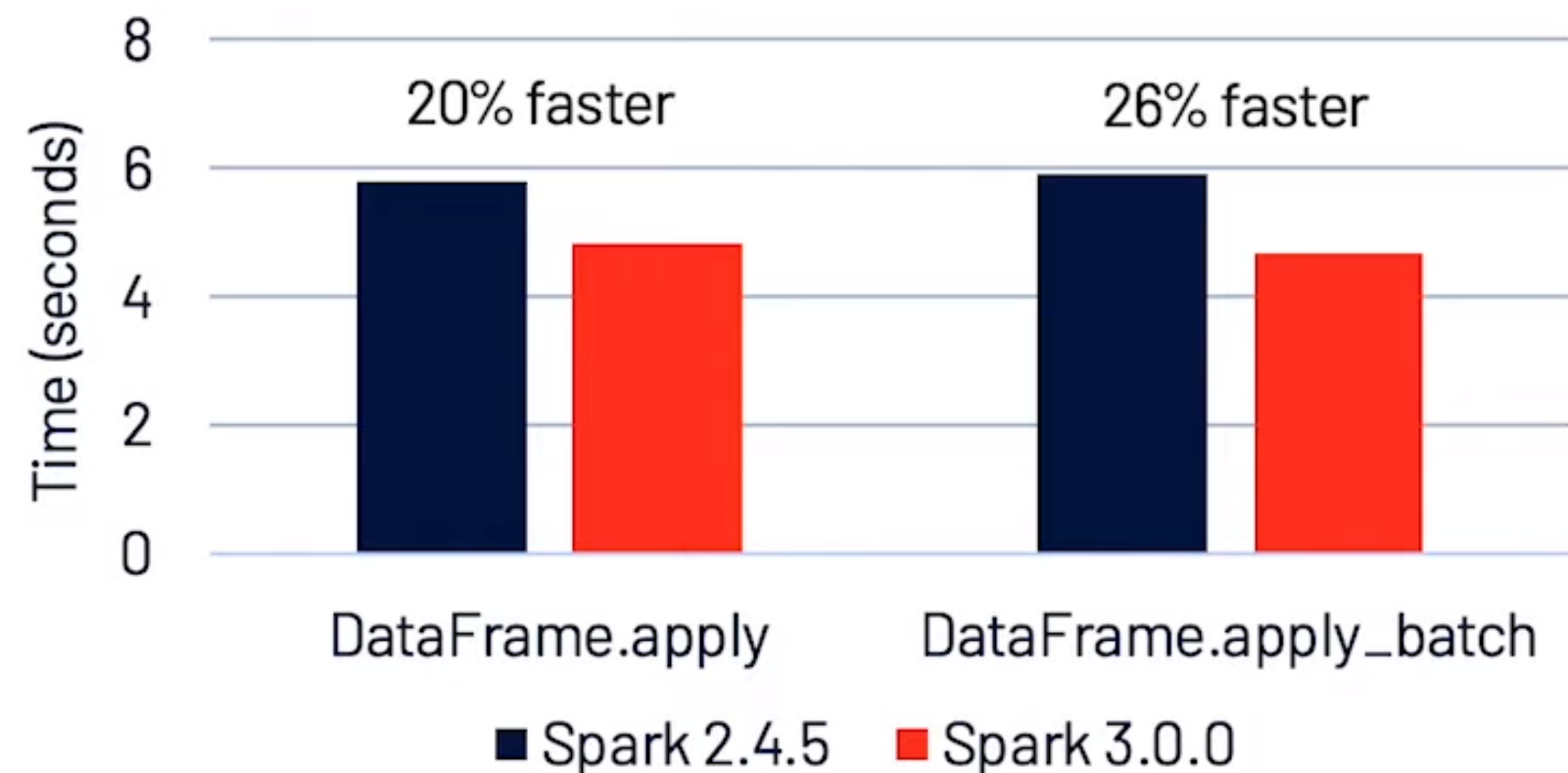
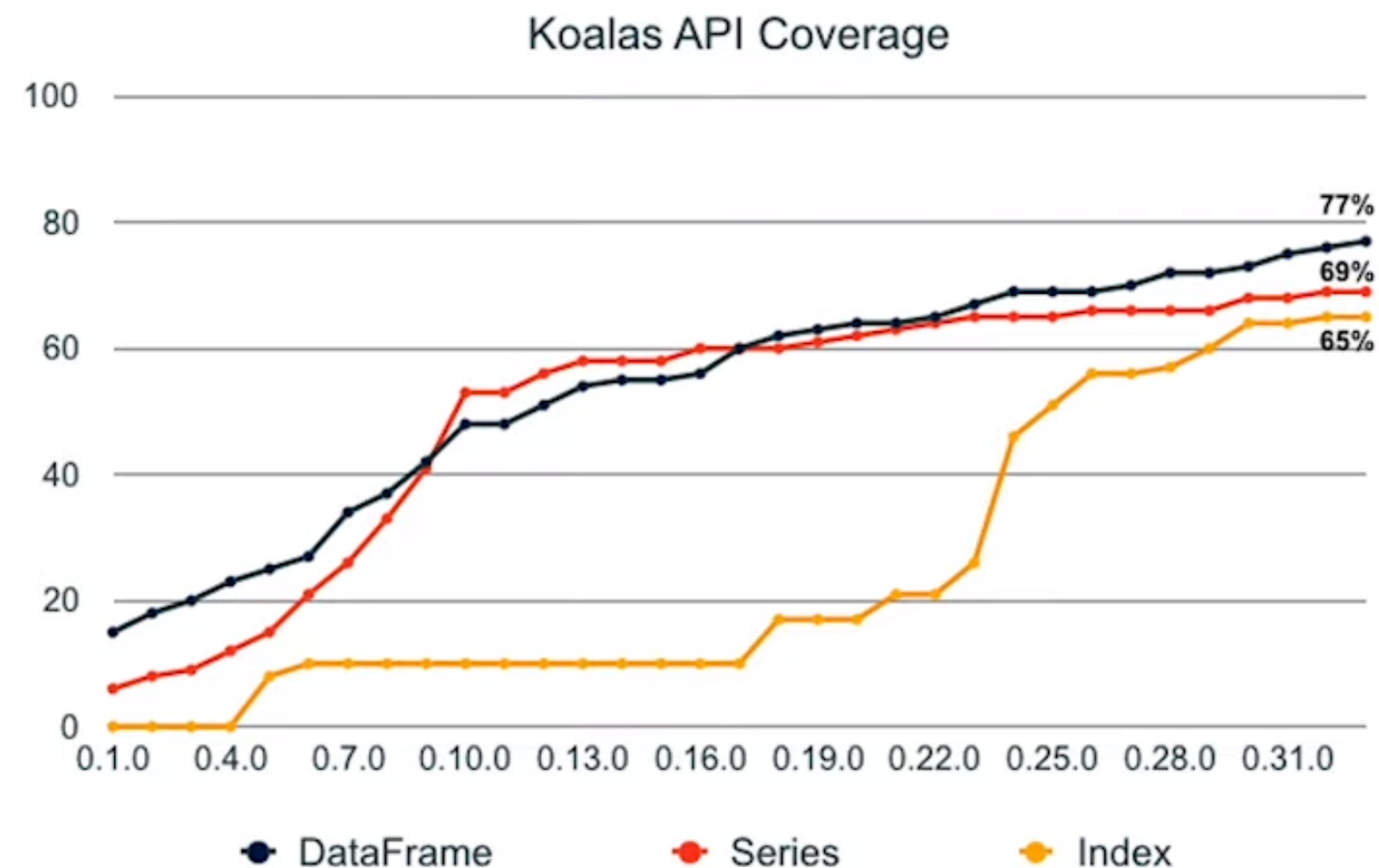
Close to 80% API coverage

Faster performance with Spark 3.0 APIs

More support for missing values, NA, and in-place updates

Faster distributed index type

`pip install koalas`
to get started!



Spark Streaming

What is Spark Streaming?

- Spark Streaming makes it applications easy to build scalable fault-tolerant streaming
- Spark Streaming brings Apache Spark's API to stream processing letting you write streaming jobs the same way you write batch jobs
- Spark Streaming provide fault-tolerance semantics out of the box (checkpointing)
- Spark Streaming is based on Spark and let batch to join streams against architecture historical data or build a so called lambda with stateful exactly-once user combine streaming with

Spark Streaming

- Scalable, high-throughput, fault-tolerant stream processing of live data streams
- Write Spark streaming applications like Spark applications
- Recovers lost work and operator state (sliding windows) out-of-the-box
- Uses HDFS and Zookeeper for high availability

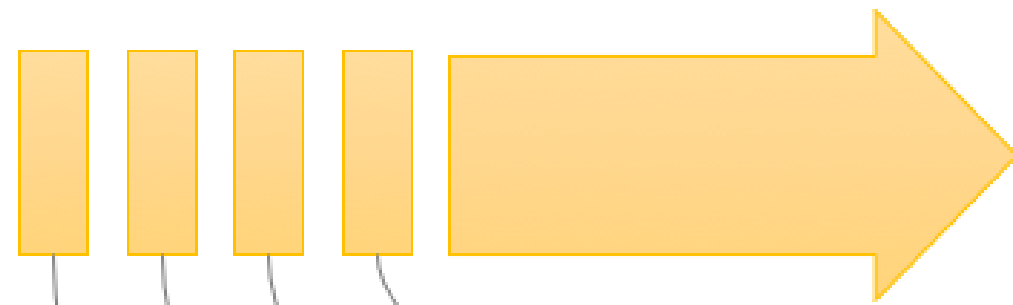
Spark Structured Streaming

Spark Structured Streaming

“Structured Streaming provides fast, scalable, fault-tolerant, end-to-end exactly-once stream processing without the user having to reason about streaming.”

Data stream

Unbounded Table



| | | |
|--|--|--|
| | | |
| | | |
| | | |
| | | |

| | | |
|--|--|--|
| | | |
|--|--|--|

| | | |
|--|--|--|
| | | |
|--|--|--|

| | | |
|--|--|--|
| | | |
|--|--|--|

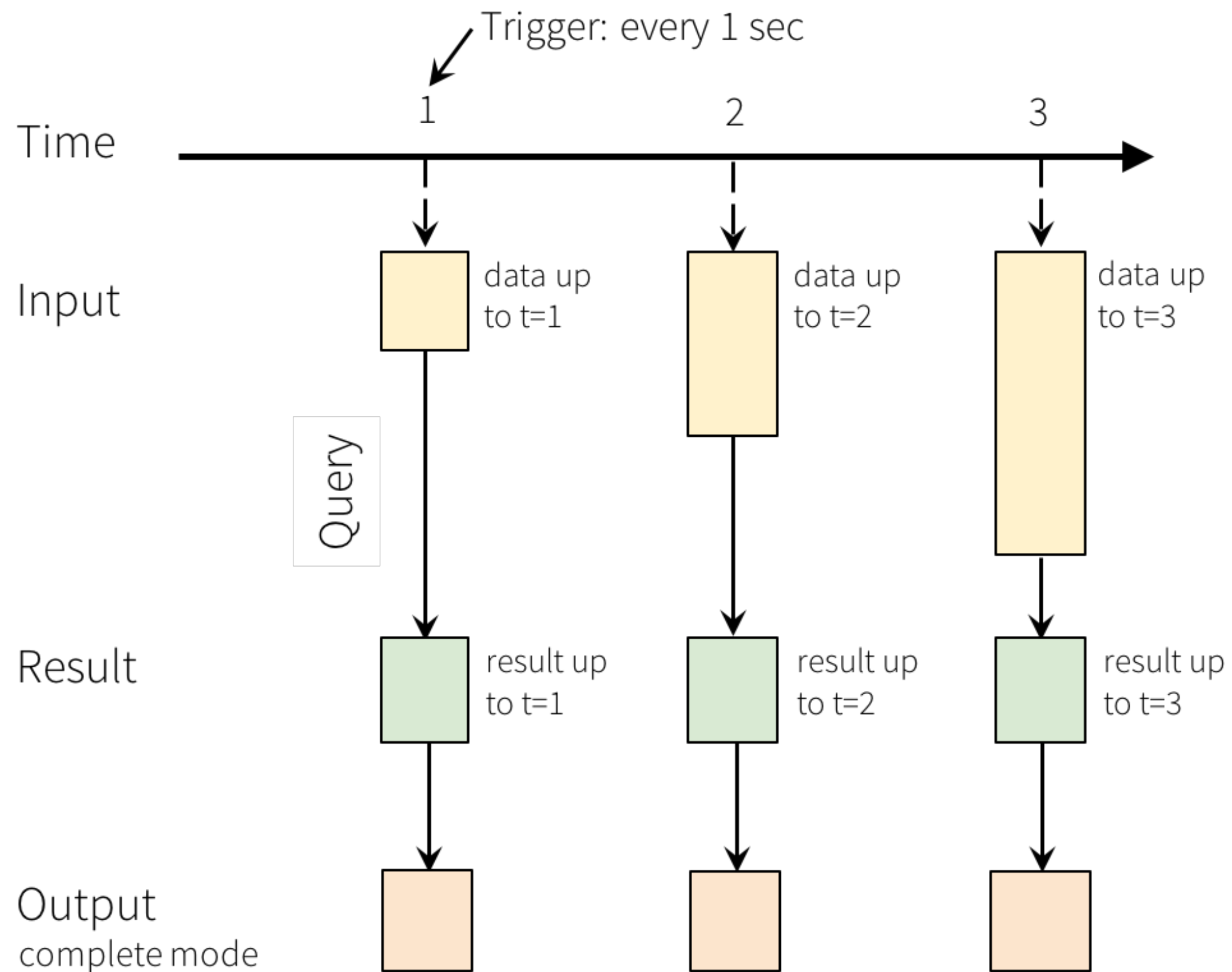
| | | |
|--|--|--|
| | | |
|--|--|--|

new data in the
data stream

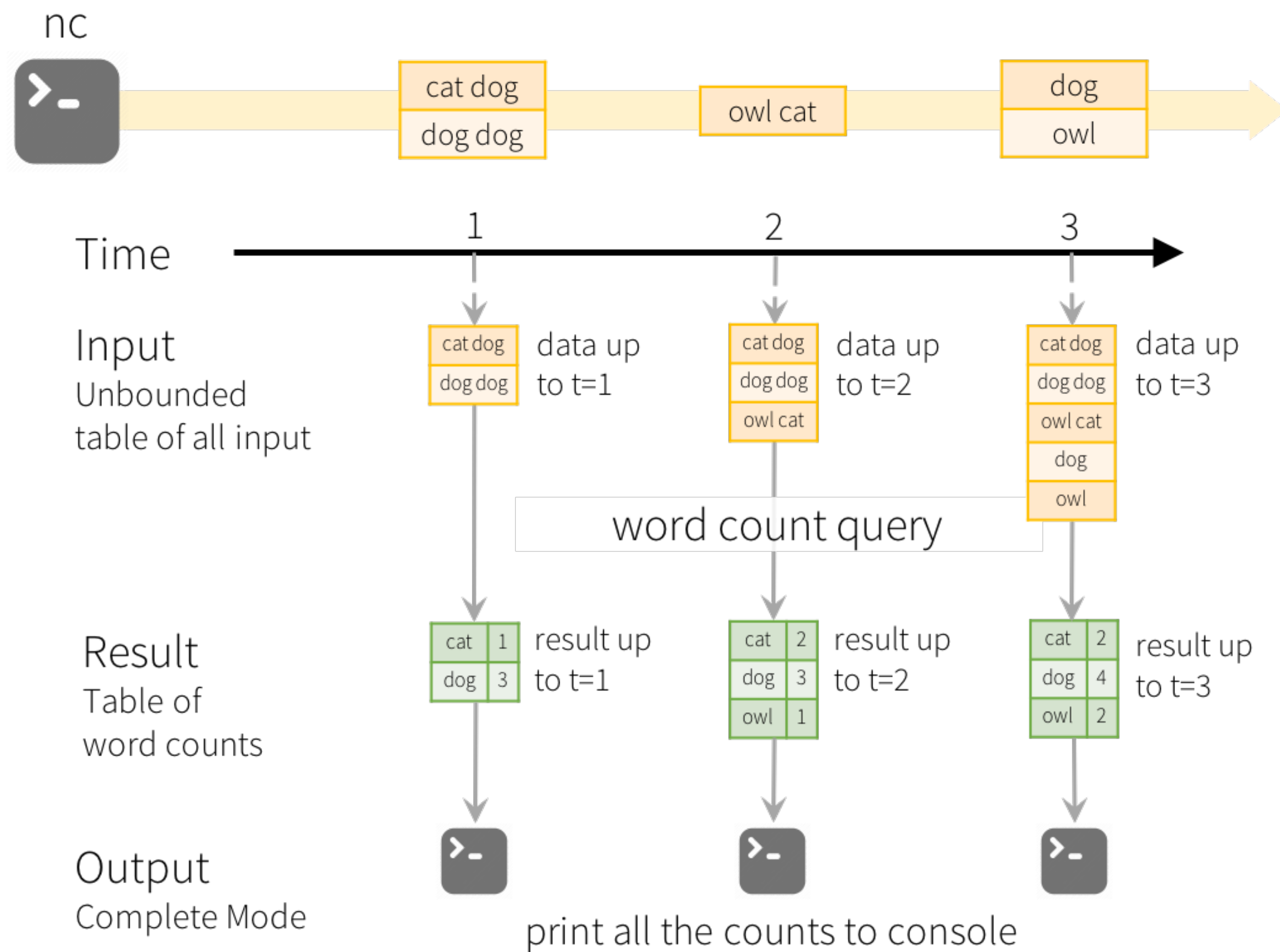
=

new rows appended
to a unbounded table

Data stream as an unbounded table



Programming Model for Structured Streaming



Model of the Quick Example

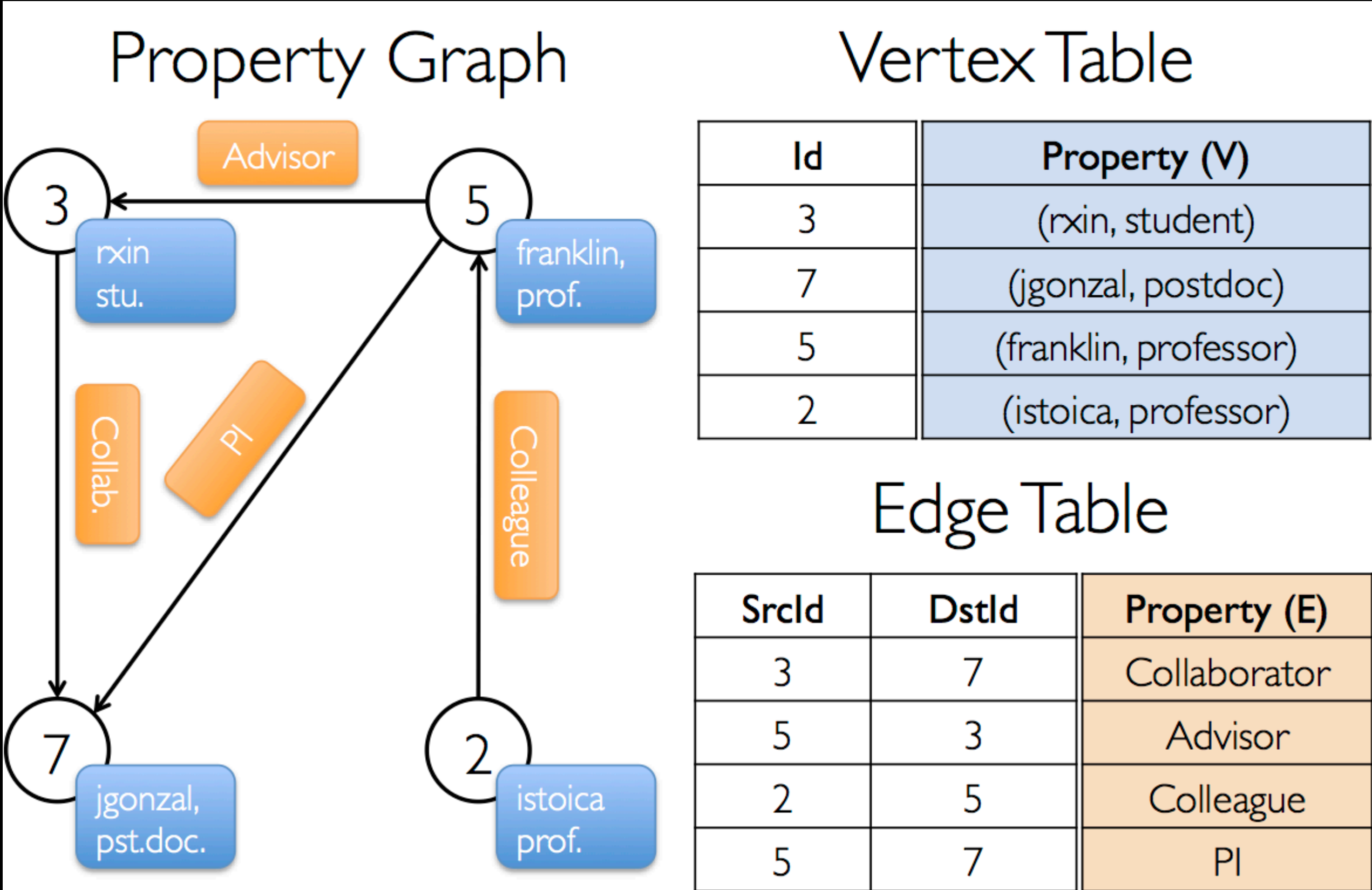
Spark GraphX

Spark GraphX

New graph abstraction

Add operators (subgraph, joinVertices, ...)

Graph algorithms (pagerank)



Questions

- Spark core
- Spark libraries
- Any other topic regarding data
- (or anything you might want to know)