Intro to Apache Spark

Agenda

- Bio
- IBM
- Intro to Spark
- Jupyter Notebook in Data Science Experience

Speaker

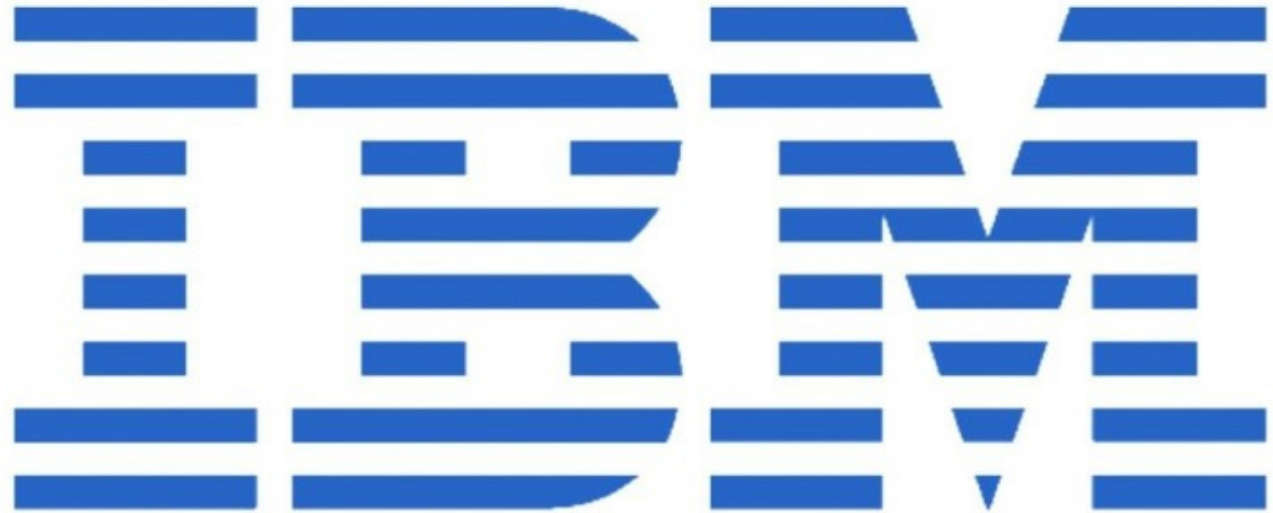
Joseph Kambourakis

- Open Source Analytics Technical Evangelist at IBM
  joseph.kambourakis@ibm.com

Previously
- Cloudera
- EMCDell

Education
- MBA from Bentley University
- BS from WPI

What do you think of?



Founding Member

Linux

Platinum Member

IBM is All-in on Spark

"It's like Spark just got blessed by the enterprise rabbi."

Ben Horowitz,
Cofounder of Venture Capital firm Andreessen Horowitz

Contribute to the Core

Launch Spark Technology Center (STC), 300 engineers

Open source SystemML

Partner with Databricks

Infuse the Portfolio

Integrate Spark throughout portfolio

3,500 employees working on Spark-related topics

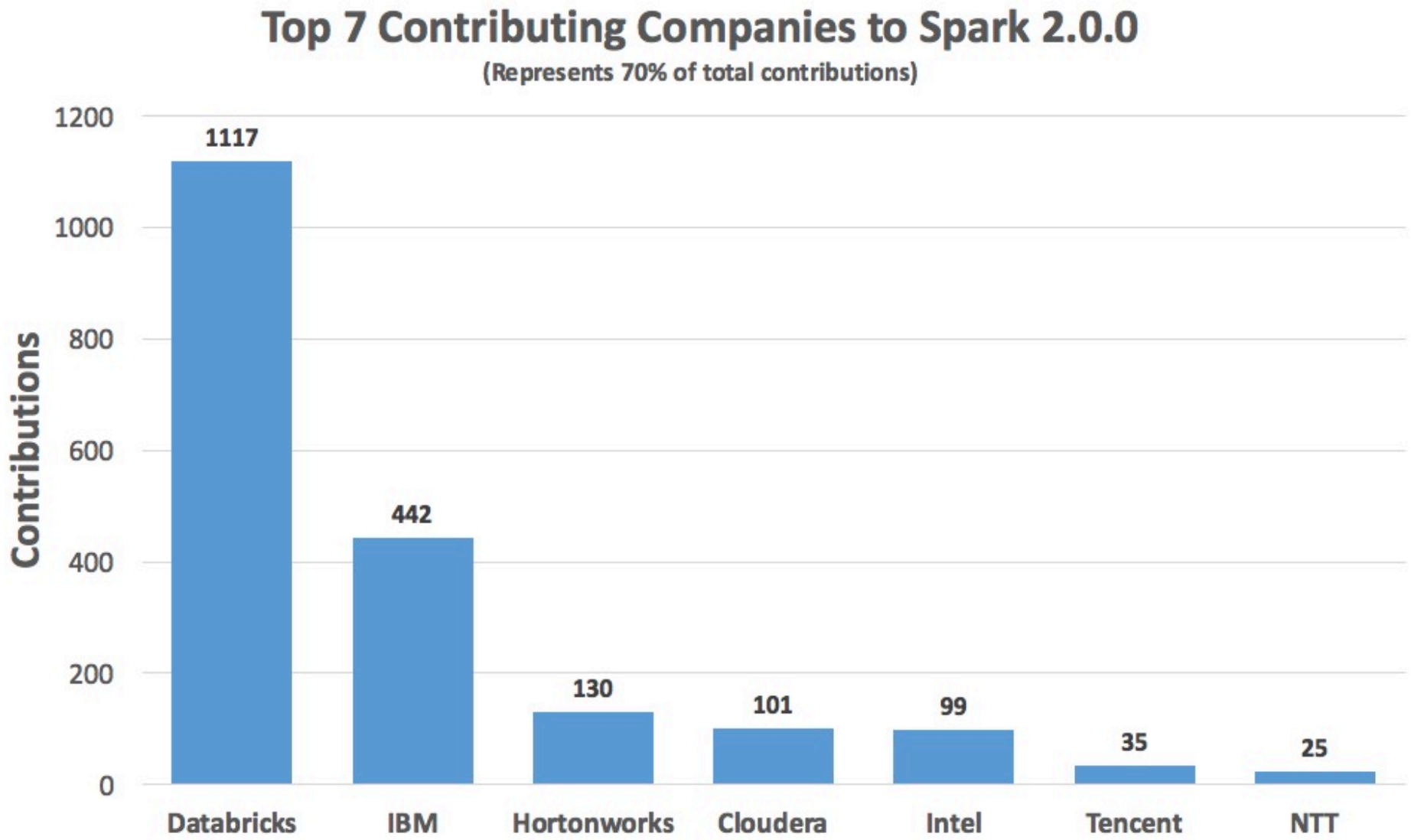
Spark however customers want it – standalone, platform or products

Foster Community

Educate 1M+ data scientists and engineers via online courses

Sponsor AMPLab, creators and evangelists of Spark

Contribute to the Core: Actual Code!



Apache Hadoop Ecosystem Committers

Committers	Projects
Juanjo Marron	Ambari
Eric Yang	Ambari, Hadoop
Di Li	Ambari
Tim Thorpe	Ambari
Tanping Wang	Hadoop
Songqing (Richardd) Ding	Pig
Shai Erera	Lucene
Tuong Truong	Sentry
Jing He	HBase
Doug Miel	HBase
Prashant Sharma	Spark
Trevor Grant	Mahout
Holden Karau	Spark

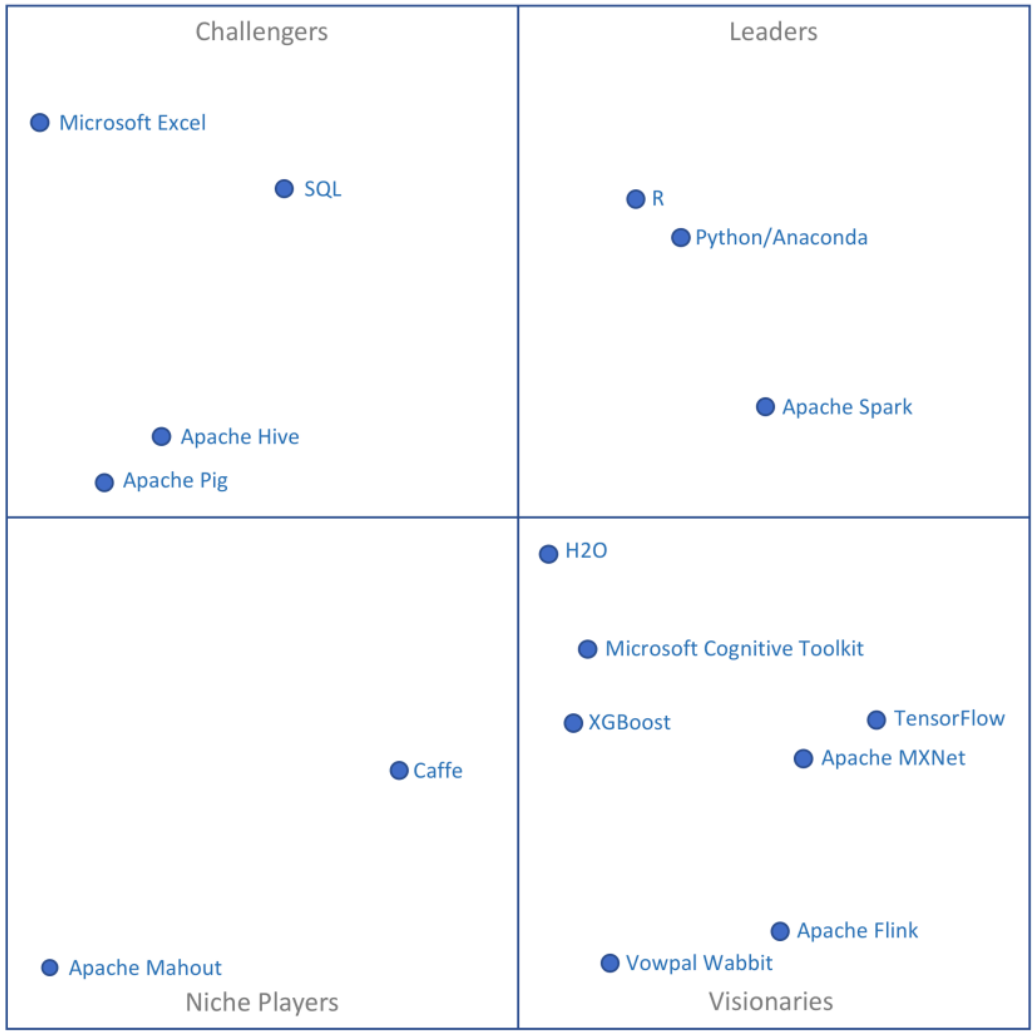
New Hires

Spark Technology Center

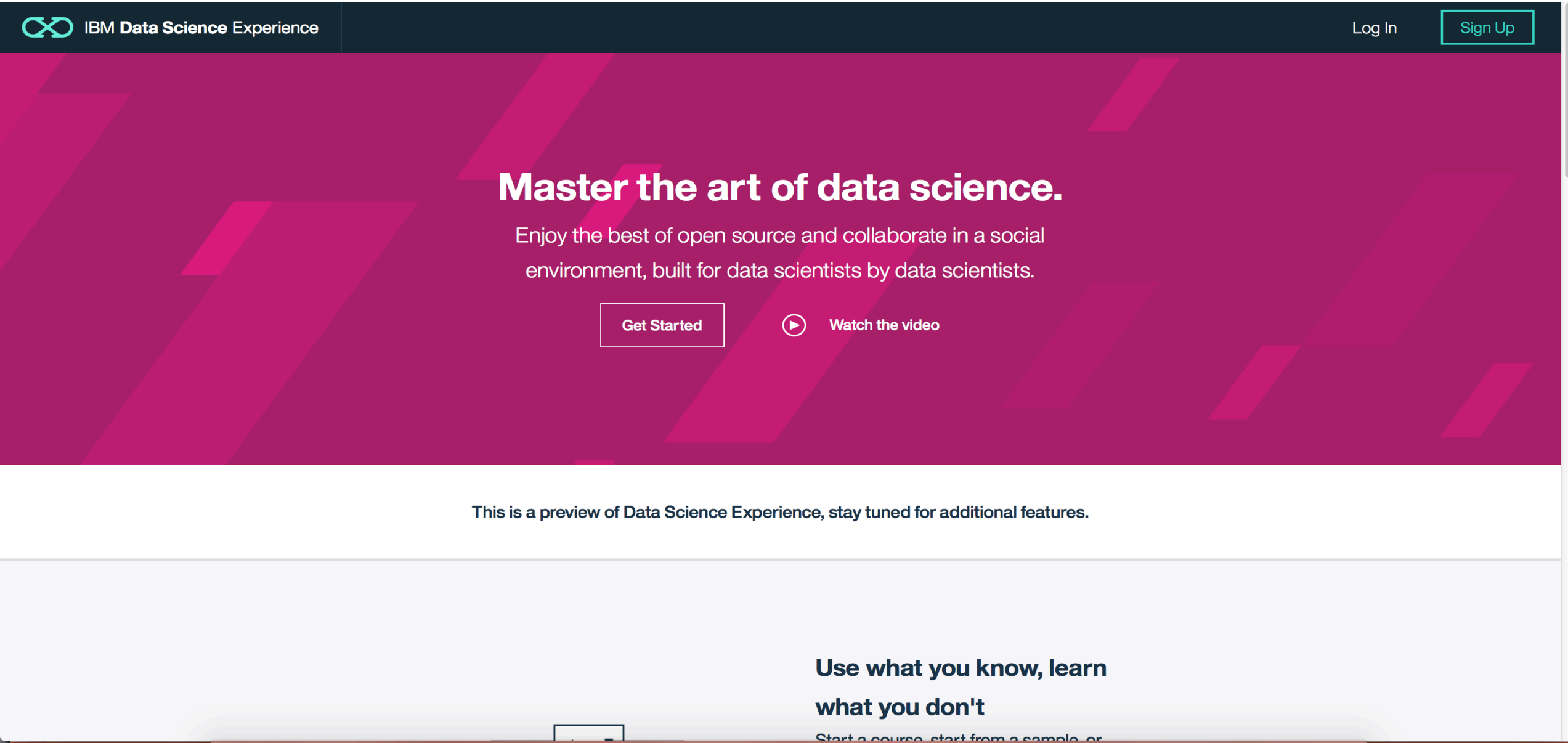
Spark Summit: Platinum Sponsor

Local Level: Meetups & Universities

Data Science Tools: What do you use?



Foster the Community: Data Science Experience



Data Science Tools: We built it out of leaders

Data Science Tools: We're actually a leader

What is Apache Spark?

What is Spark?

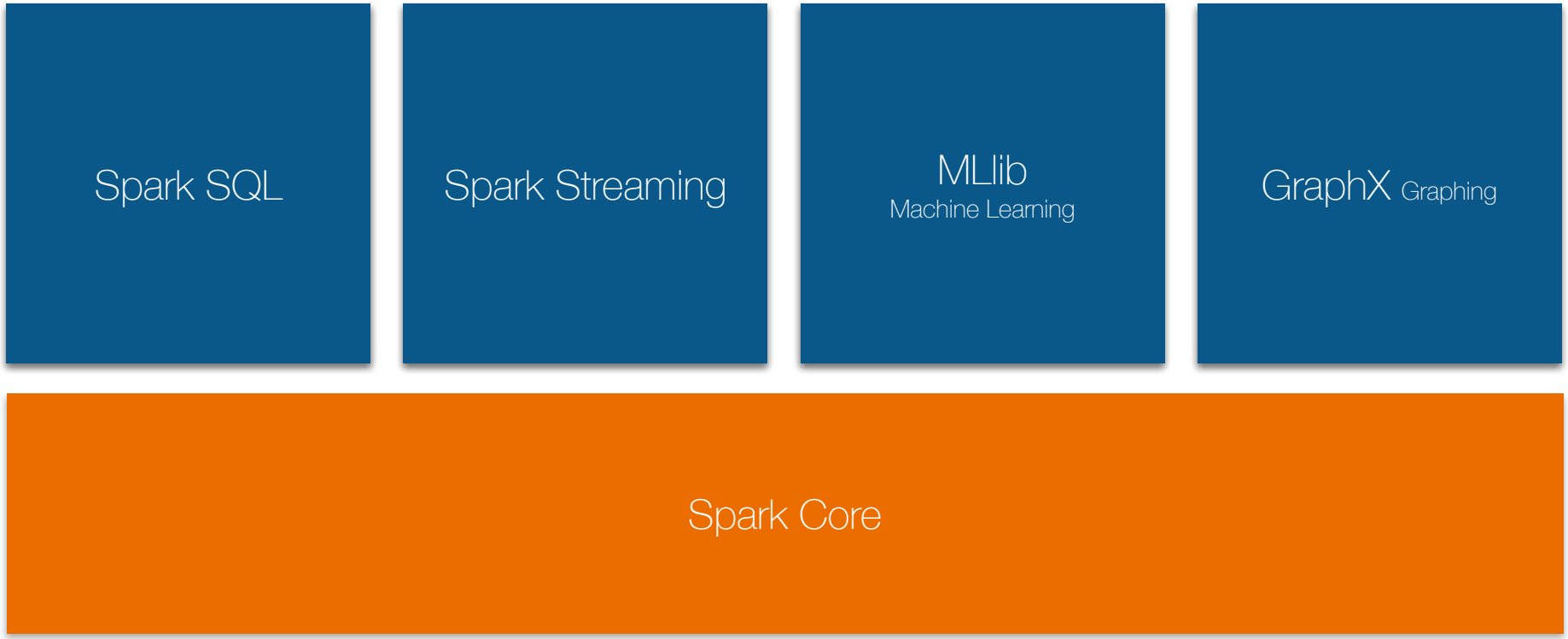
Spark is an **open** source
in-memory
application framework for
distributed data processing and
iterative analysis
on **massive** data volumes

“Analytic Operating System”

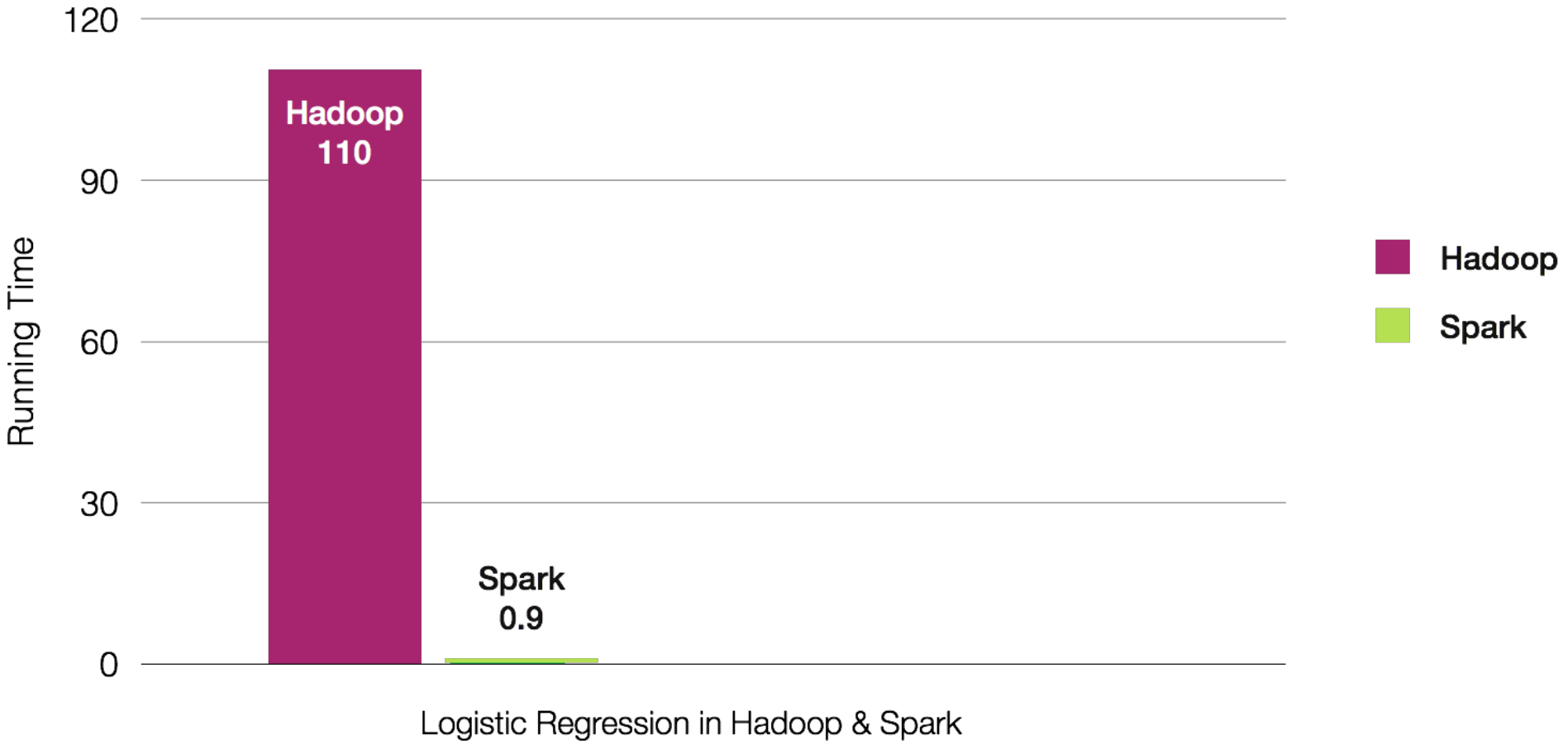
Introducing Apache Spark

- **General purpose distributed data processing engine**
- **AMPLab at UC Berkeley**
  - Started by Matei Zaharia in 2009
  - Creators founded Databricks
    - IBM partners with DB
- **Apache open source project**
  - Initial release May 2014
  - Current version 2.1 released December 2016

Spark and the Key Libraries



Why Spark? Performance



Why Spark? Productive

- Production ready code
- Concise
- Focus on the business problems
- and not coding

Word Count

In this example, we use a few transformations to build a dataset of (String, Int) pairs called `counts` and then save it to a file.

Python | Scala | Java

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

Example: WordCount v1.0

Before we jump into the details, lets walk through an example MapReduce application to get a flavour for how they work.

`WordCount` is a simple application that counts the number of occurences of each word in a given input set.

This works with a local-standalone, pseudo-distributed or fully-distributed Hadoop installation (Single Node Setup).

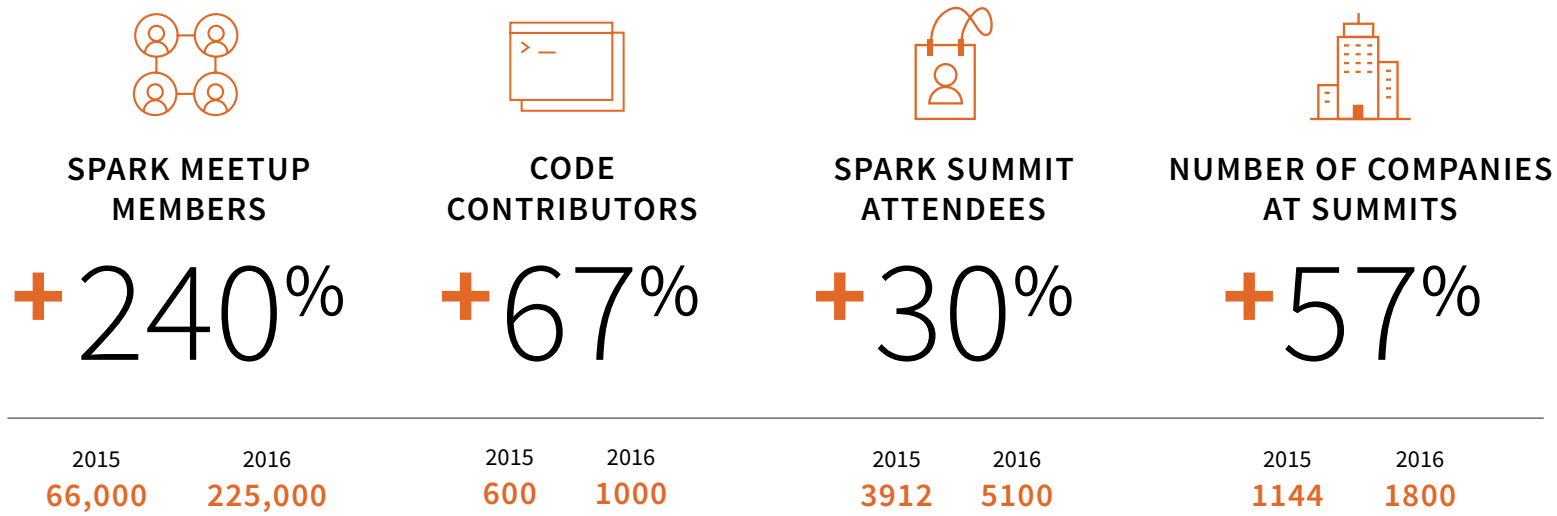
Source Code

WordCount.java

```
1.  package org.myorg;
2.
3.  import java.io.IOException;
4.  import java.util.*;
5.
6.  import org.apache.hadoop.fs.Path;
7.  import org.apache.hadoop.conf.*;
8.  import org.apache.hadoop.io.*;
9.  import org.apache.hadoop.mapred.*;
10. import org.apache.hadoop.util.*;
11.
12. public class WordCount {
13.
14.   public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
15.     private final static IntWritable one = new IntWritable(1);
16.     private Text word = new Text();
17.
18.     public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
19.       String line = value.toString();
20.       StringTokenizer tokenizer = new StringTokenizer(line);
21.       while (tokenizer.hasMoreTokens()) {
                  kenizer.nextToken());
                  ect(word, one);
                class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
                educe(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
                s.hasNext()) {
                es.next().get();
                ct(key, new IntWritable(sum));
                void main(String[] args) throws Exception {
                = new JobConf(WordCount.class);
                me("wordcount");
                tKeyClass(Text.class);
                tValueClass(IntWritable.class);
44.
45.     conf.setMapperClass(Map.class);
46.     conf.setCombinerClass(Reduce.class);
47.     conf.setReducerClass(Reduce.class);
48.
49.     conf.setInputFormat(TextInputFormat.class);
50.     conf.setOutputFormat(TextOutputFormat.class);
51.
52.     FileInputFormat.setInputPaths(conf, new Path(args[0]));
53.     FileOutputFormat.setOutputPath(conf, new Path(args[1]));
54.
55.     JobClient.runJob(conf);
57.   }
58. }
59.
```

Why Spark? Community

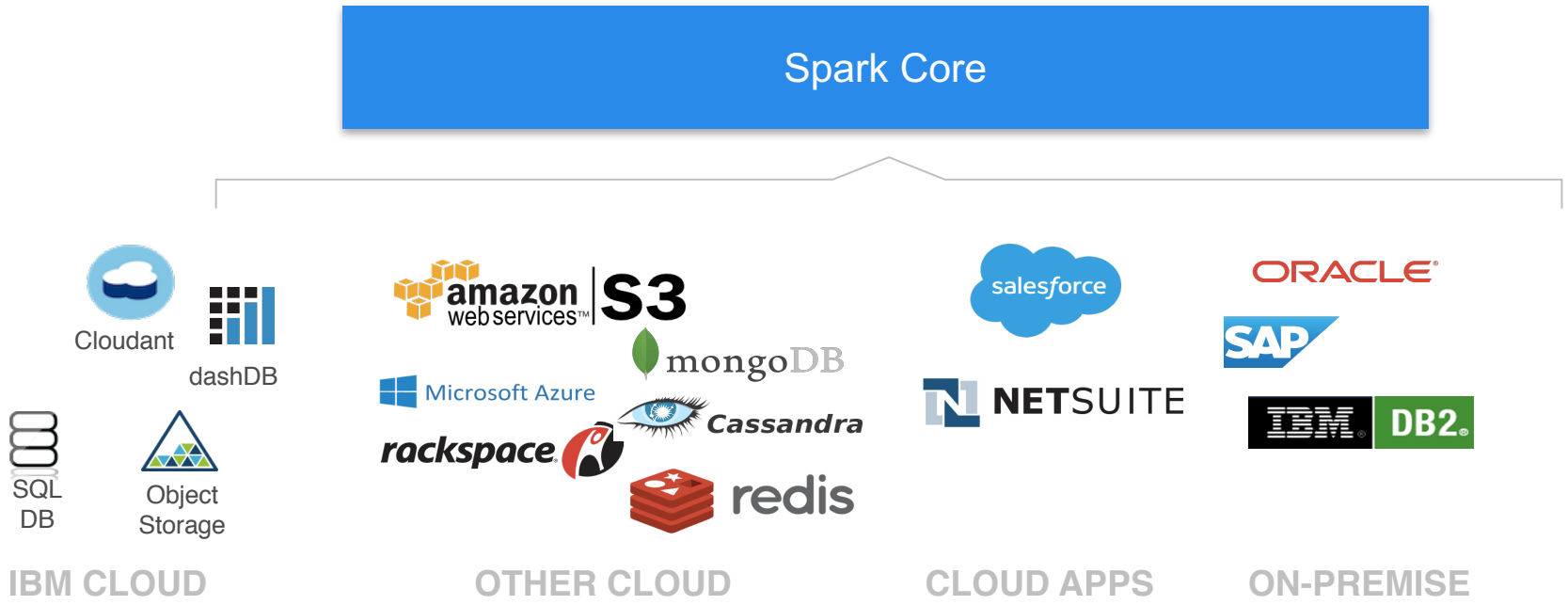
- **Open source community**
- **Fastest growing**
- **Wide adoption**



source: DataBricks 2016 Survey

Why Spark? Ease of Use

- Four APIs
- Documentation
- Data Platform Flexibility



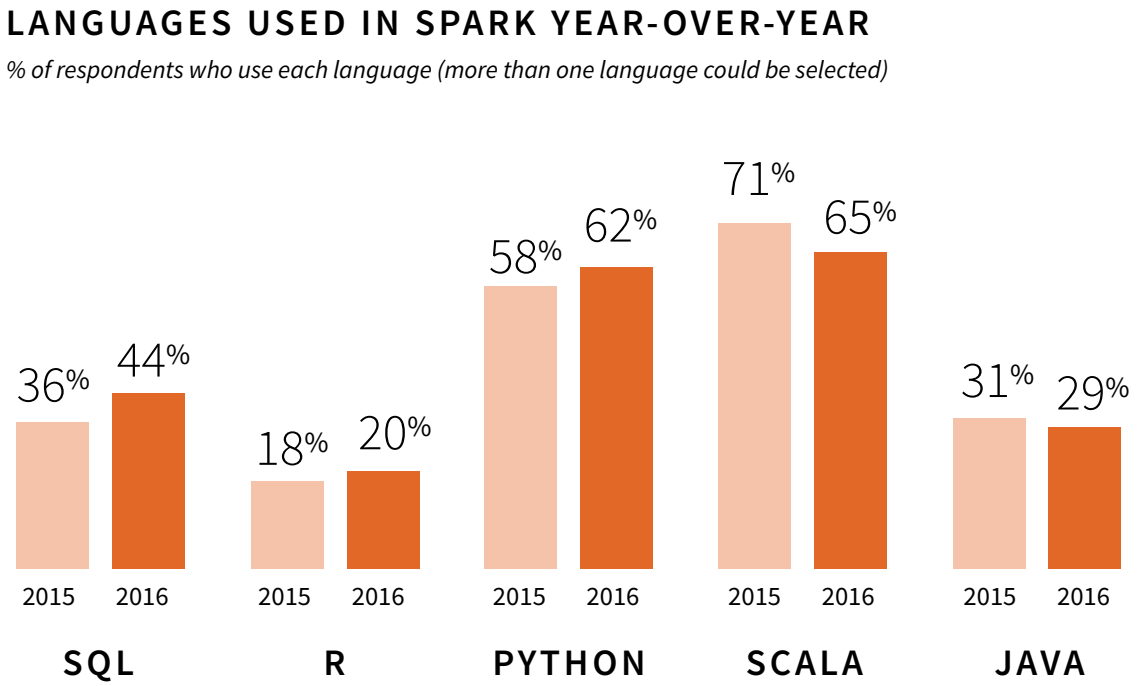
What Spark Is Not

- **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 real time** – Spark Streaming is micro-batching, not true streaming, and cannot handle the real-time complex event processing

Apache Spark Programming

Spark Programming Languages

- **Scala**
  - Functional programming
  - Spark written in Scala
  - Scala compiles into Java byte code
- **Python**
  - Easily ports to Spark
- **Java**
  - New features in Java 8 makes for more compact coding
- **R**
  - Statistical language used to create and manipulate data



source: DataBricks 2016 Survey

Functional Programming

- **Spark depends on functional programming**
- **Closures or in line anonymous functions**
  - Functions have inputs and outputs only
  - One can pass functions as inputs to other functions as parameters

- **Python**
  - function(lambda x: x)

- **Scala**
  - function(x => x)

Functions!

- Two types of operations
  - Transformations
    - map()
    - filter()
    - flatMap()
  - Actions
    - first()
    - collect()
    - take()
- Lazy evaluation

A Few Good Functions

map - Return a new RDD by applying a function to each element of this RDD.

filter - Return a new RDD containing only the elements that satisfy a predicate.

reduce - Reduces the elements of this RDD using the specified commutative and associative binary operator. Currently reduces partitions locally.

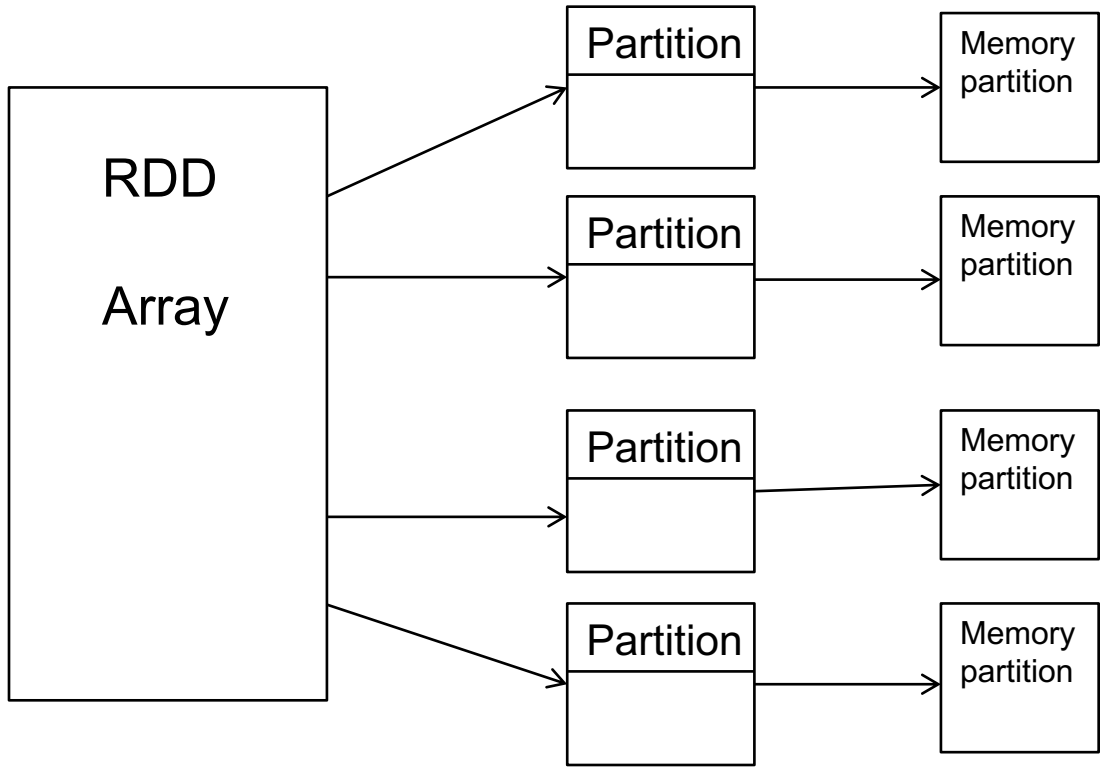
flatMap - Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results.

distinct - Return a new dataset that contains the distinct elements of the source dataset.

sortByKey - 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, as specified in the boolean **ascending** argument

Resilient Distributed Dataset (RDD)

- **Immutable**
- **Holds references to partition objects**
- **Each partition is a subset of the overall data**
- **Partitions are assigned to nodes on the cluster**
- **Partitions are in memory by default**
- **Fault tolerance**
  - If data in memory is lost it will be recreated from lineage

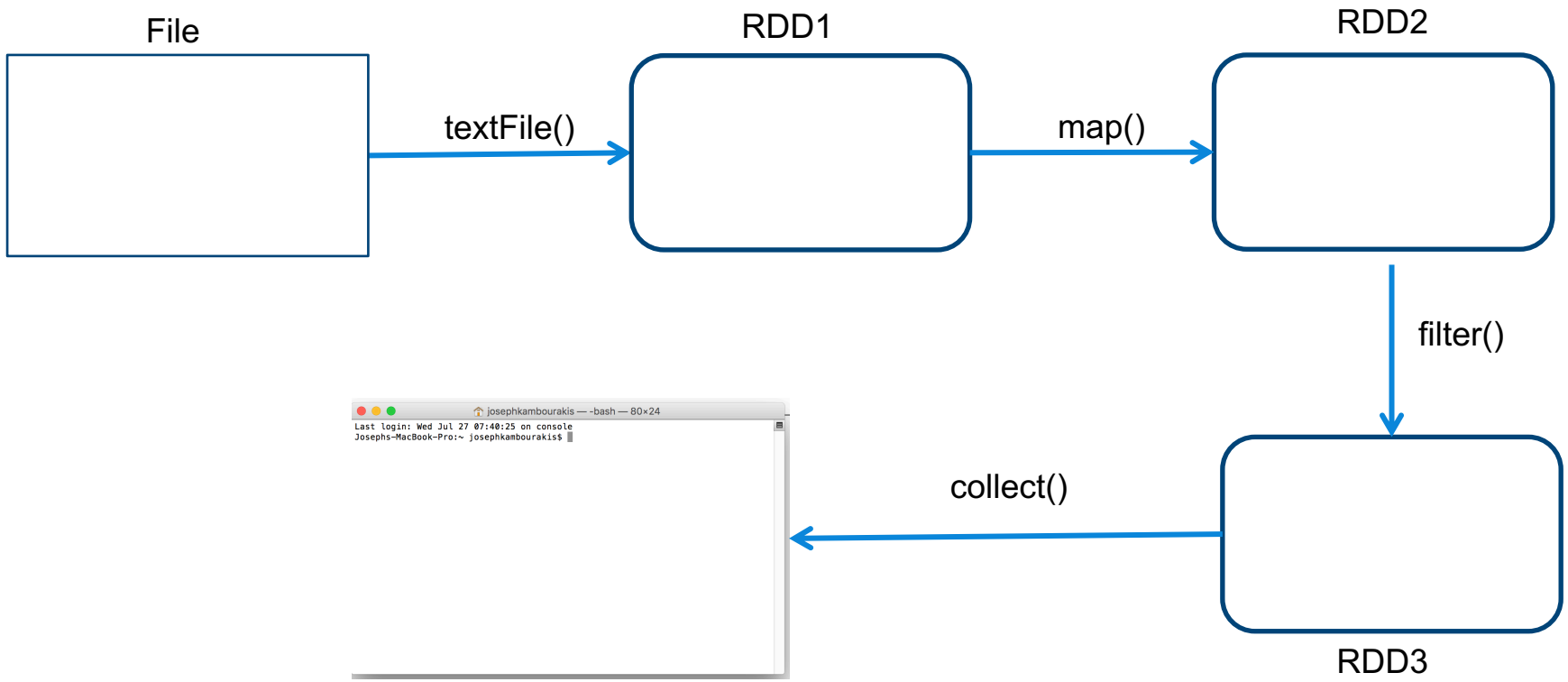


Dataframes

- **Subset of RDDs**
- **Main data object abstraction in SparkSQL**
  - Organized into NAMED columns
- **Can be created from:**
  - Existed structured data source
    - json, database table, Apache Parquet
  - Another RDD
  - Transformation of another dataframe
  - Programmatically

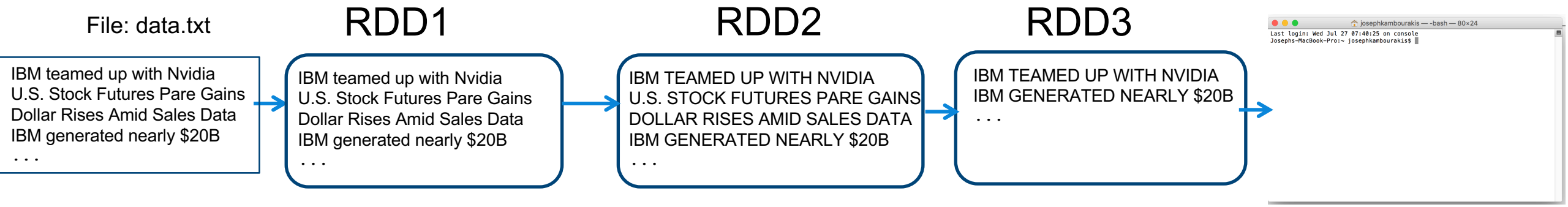
Apache Spark Code Execution

Directed Acyclic Graph - DAG



Sample Code

- **RDD1 = sc.textFile(data.txt)**
  RDD2 = RDD1.map(lambda line: line.upper())
  RDD3 = RDD2.filter(lambda line: 'IBM' in line)
  RDD3.collect()

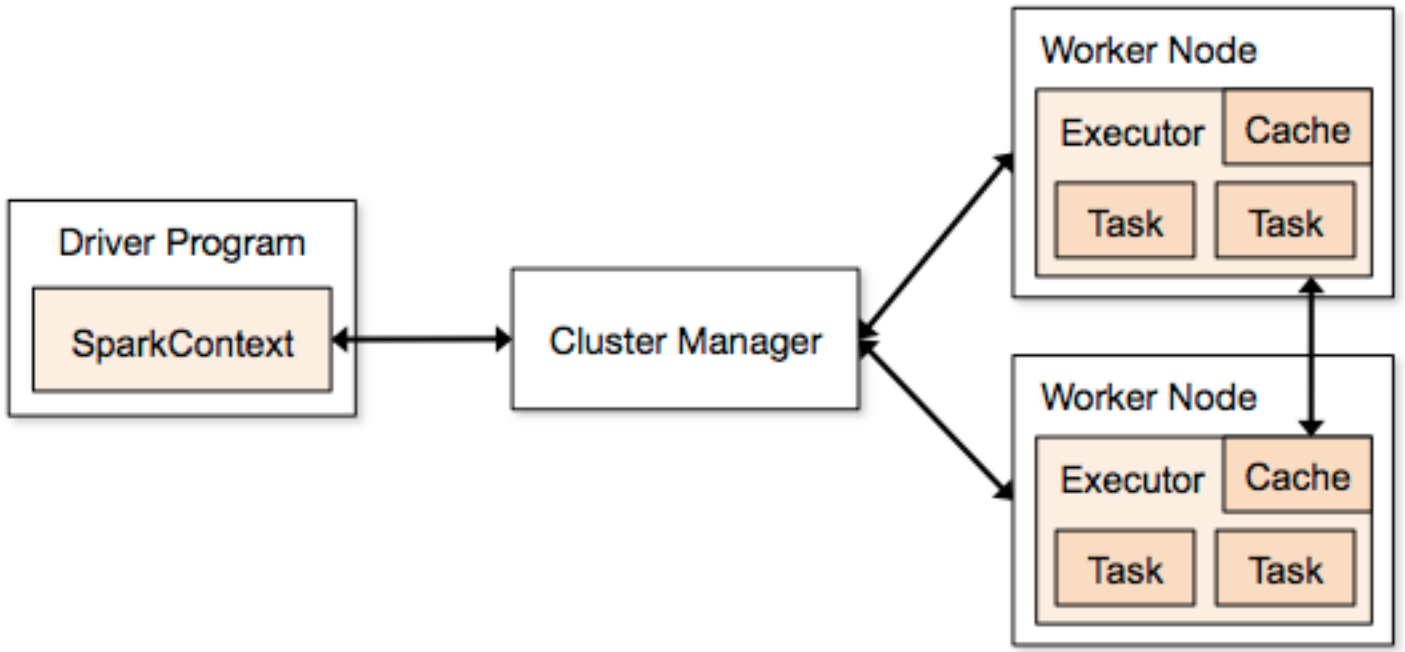


Apache Spark Under the hood!

Spark Application Architecture

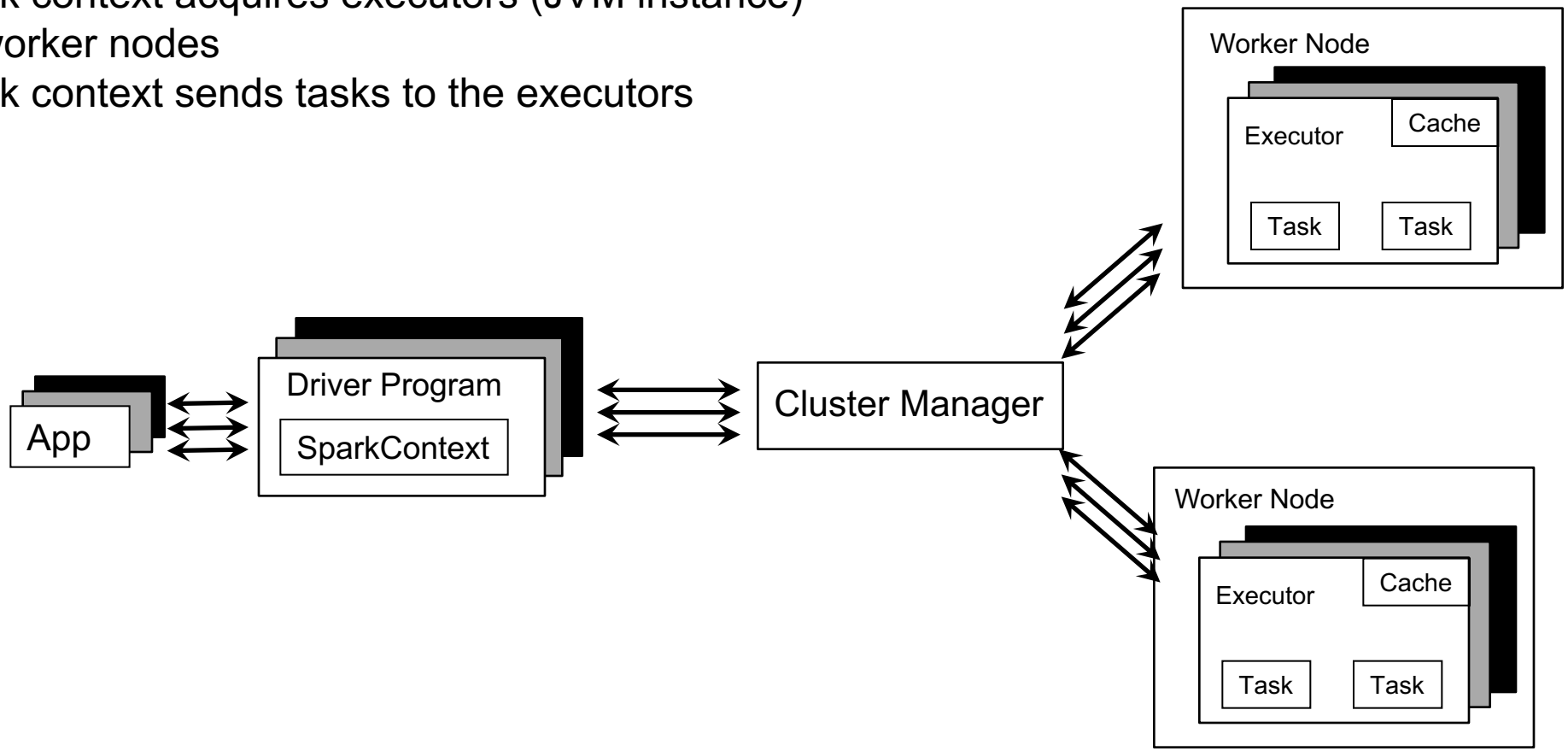
- **A Spark application is initiated from a driver program**

- **Spark execution modes:**
  - Standalone with the built-in cluster manager
  - Use Mesos as the cluster manager
  - Use YARN as the cluster manager
  - Standalone cluster on any cloud (BlueMix, IBM Softlayer, Amazon, Azure, …)



Showing multiple applications

- Each Spark application runs as a set of processes coordinated by the Spark context object (driver program)
  - Spark context connects to Cluster Manager (standalone, Mesos/Yarn)
  - Spark context acquires executors (JVM instance) on worker nodes
  - Spark context sends tasks to the executors



Apache Spark Libraries

SparkSQL

- **Component of Spark**
  - Project started in 2012
  - First alpha release in Spring 2013
  - Out of alpha with Spark 0.9.0
  - New features in Spark 2.0
- **Allows you to interact with Apache Hive**

- **More in next section!**

Spark Streaming

- **Component of Spark**
  - Project started in 2012
  - First alpha release in Spring 2013
  - Out of alpha with Spark 0.9.0
  - New features in Spark 2.0

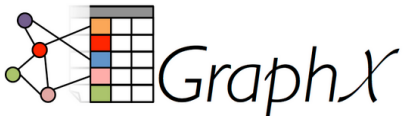
- **Discretized Stream (DStream) programming abstraction**
  - Represented as a sequence of RDDs (micro-batches)
  - RDD: set of records for a specific time interval
  - Supports Scala, Java, and Python (with limitations)

- **Fundamental architecture: batch processing of datasets**

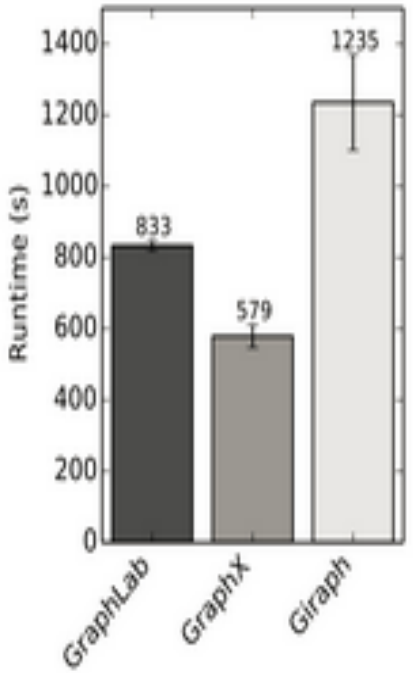
Spark MLlib

- **Spark MLlib for machine learning library**
- **Provides common algorithm and utilities**
  - Classification
  - Regression
  - Clustering
  - Collaborative filtering
  - Dimensionality reduction
- **Leverages iteration and yields better results than one-pass approximations sometimes used with MapReduce**
- **More on this later!**

Spark GraphX

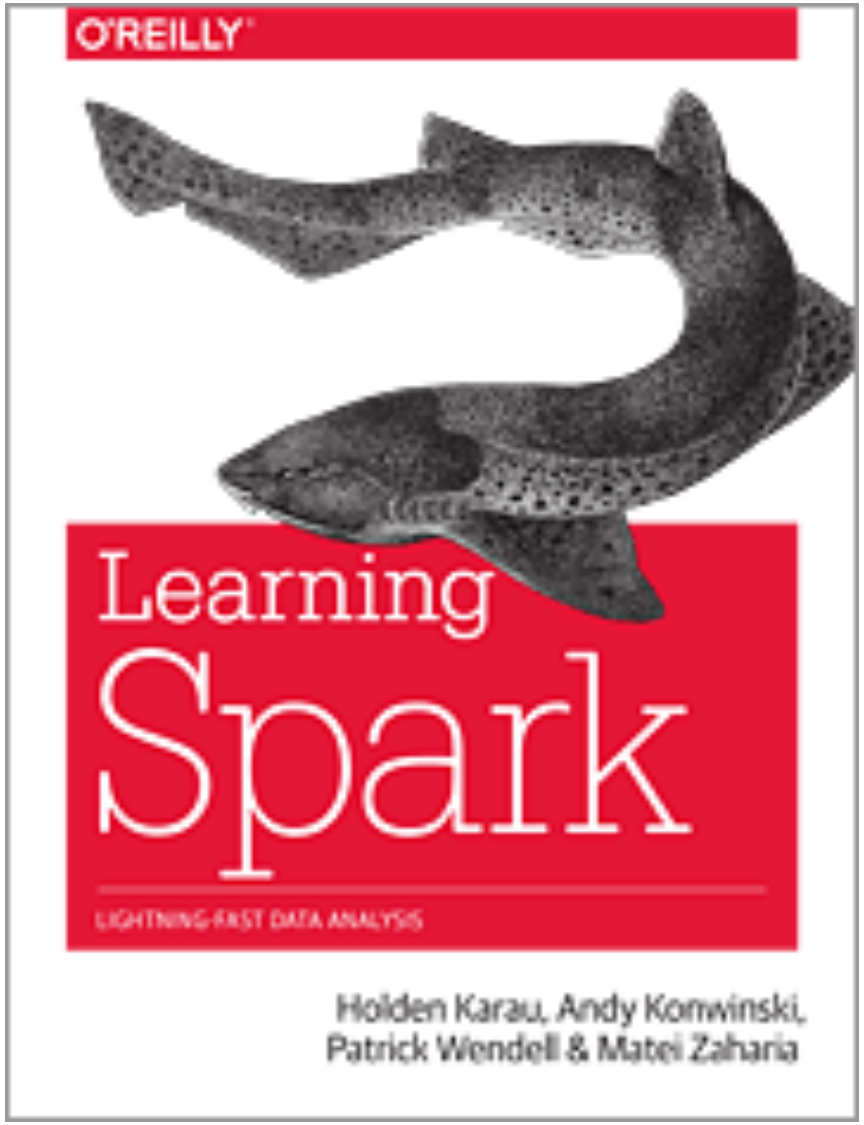


- **Scala only!**
- **Flexible Graphing**
  - GraphX unifies ETL, exploratory analysis, and iterative graph computation
  - You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms with the API
  - GraphFrames - graph library based on Data Frames
- **Speed**
  - Comparable performance to the fastest specialized graph processing systems.
- **Algorithms**
  - Choose from a growing library of graph algorithms
  - In addition to a highly flexible API, GraphX comes with a variety of graph algorithms

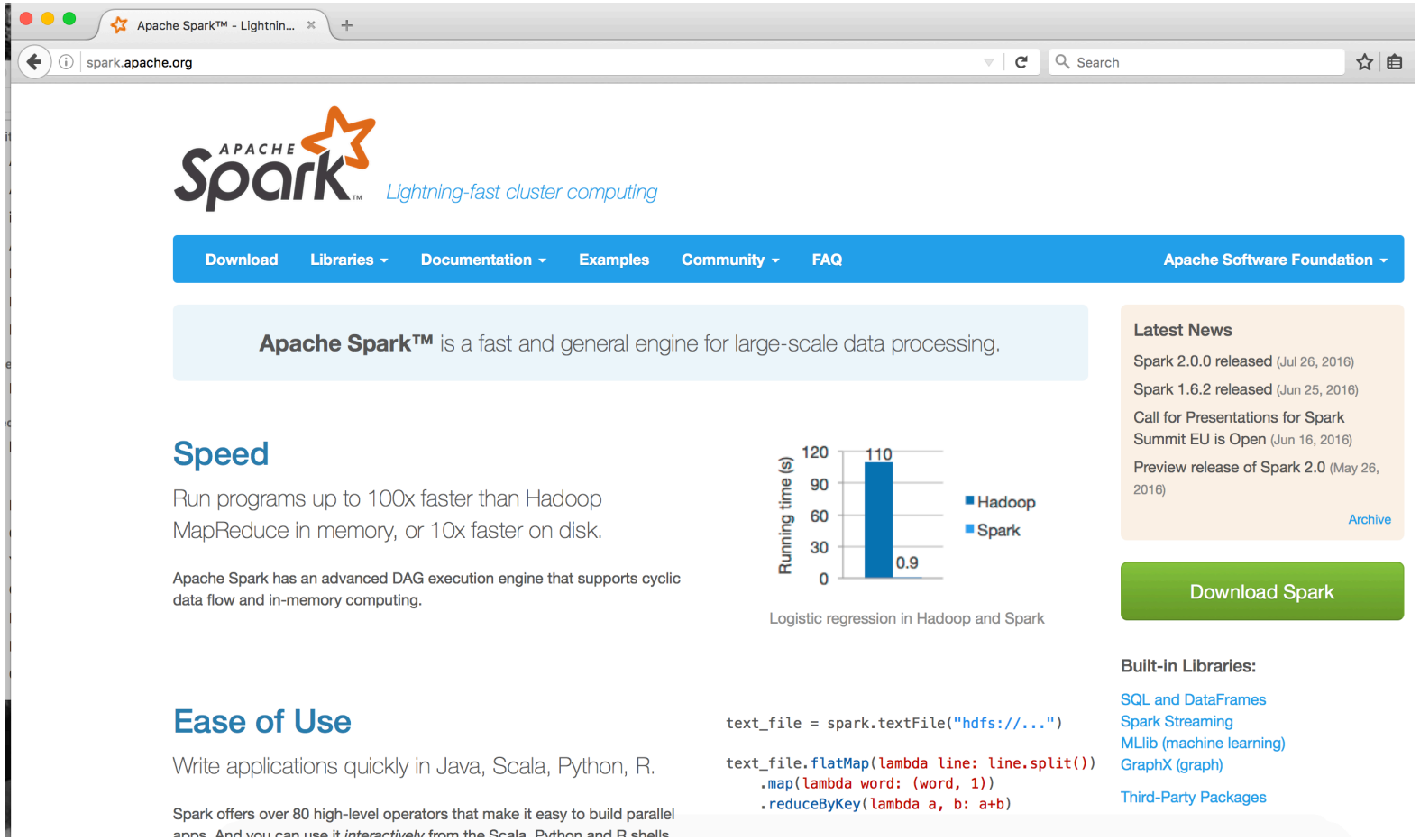


Apache Further Resources

Additional Resources



Additional Resources



Notebook Intro

- **Data Science Experience**
  - Jupyter Notebook
- **Student Code**
  - Notebook and solutions will be provided after the lab and we'll walk through them.

Small Demo

- datascience.ibm.com