



Power of data. Simplicity of design. Speed of innovation.

**Hands on Introduction to Apache Spark for
Data Engineers, Data Scientists, and Developers at Metlife**



Agenda

9 am – 10 am	Kickoff, Apache Spark Overview
10:00am – 11:00am	Lab 1, Hello Spark - Hand on exercise
11:00 am – 12:00pm	Apache Spark SQL Overview
12:00 pm – 1pm	Lunch
1pm – 2pm	Lab 2, Spark SQL - Hands on exercises
2pm – 3pm	Overview of Data Science & Machine Learning w/ Apache Spark
3pm – 4pm	Lab 3, Machine Learning w/ Spark – Hands on exercises
4pm – 4:30pm	Wrap up – Feedback from attendees

Introductions: Instructor

Joseph Kambourakis

Open Source Analytics Technical Evangelist

joseph.kambourakis@ibm.com

@mouthorjoe



Ground Rules

- **Interrupt me!**
- **The slides will be available**

Objectives

- **Understand how Spark works**
- **Write Spark programs**
- **List different ways IBM uses Spark**
- **Discuss how to apply to Spark to business problems**
- **Build relationship between IBM and MetLife**

Introductions: Attendees

- **What is your role?**
- **How familiar are you with Spark?**
 - Python?
 - Scala?
- **How familiar are you with Hadoop?**

Grace Hopper



If one ox could not do the job they did not try to grow a bigger ox, but used two oxen. When we need greater computer power, the answer is not to get a bigger computer, but...to build systems of computers and operate them in parallel.

— Grace Hopper —

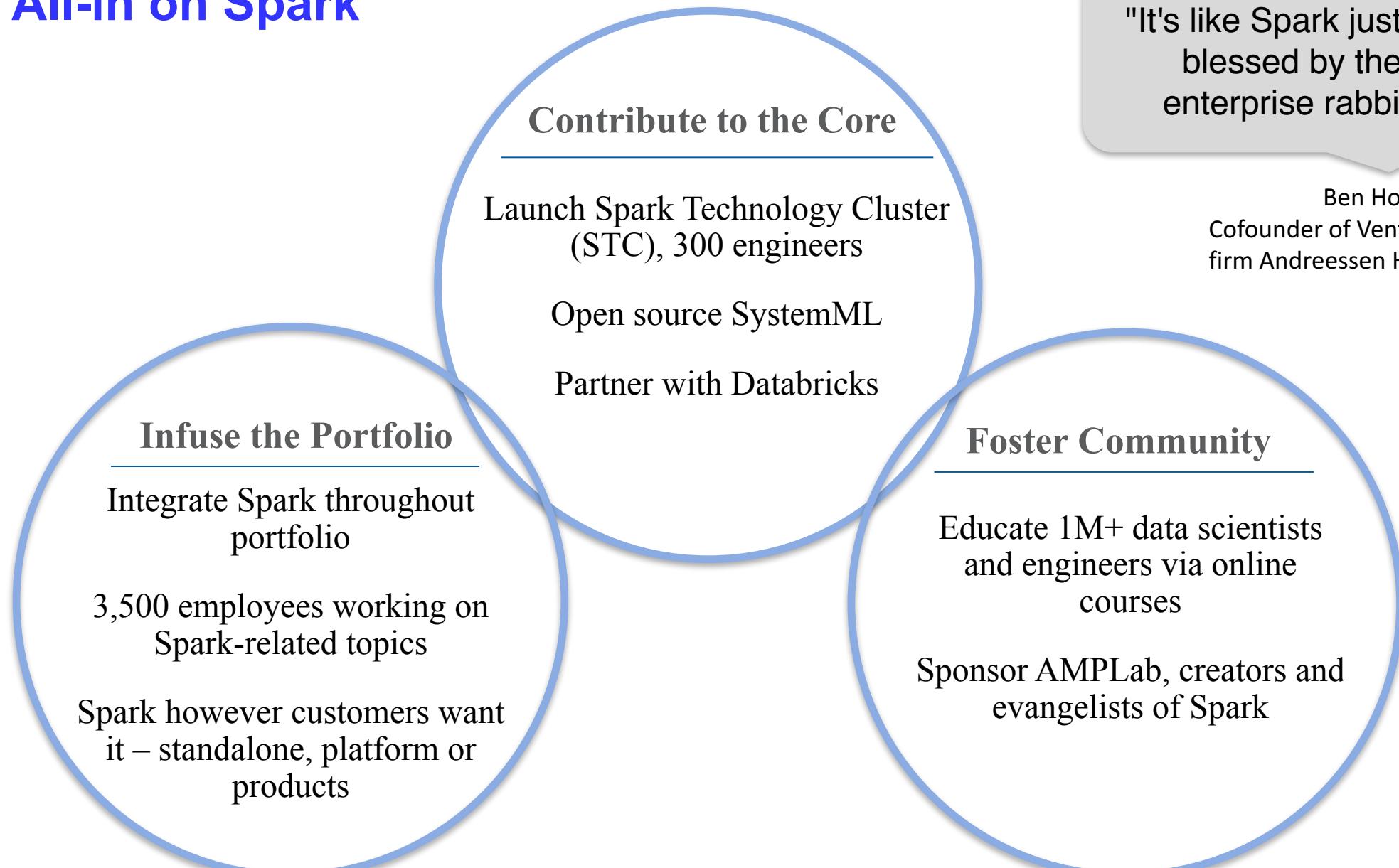


What is Spark?

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

“Analytic Operating System”

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

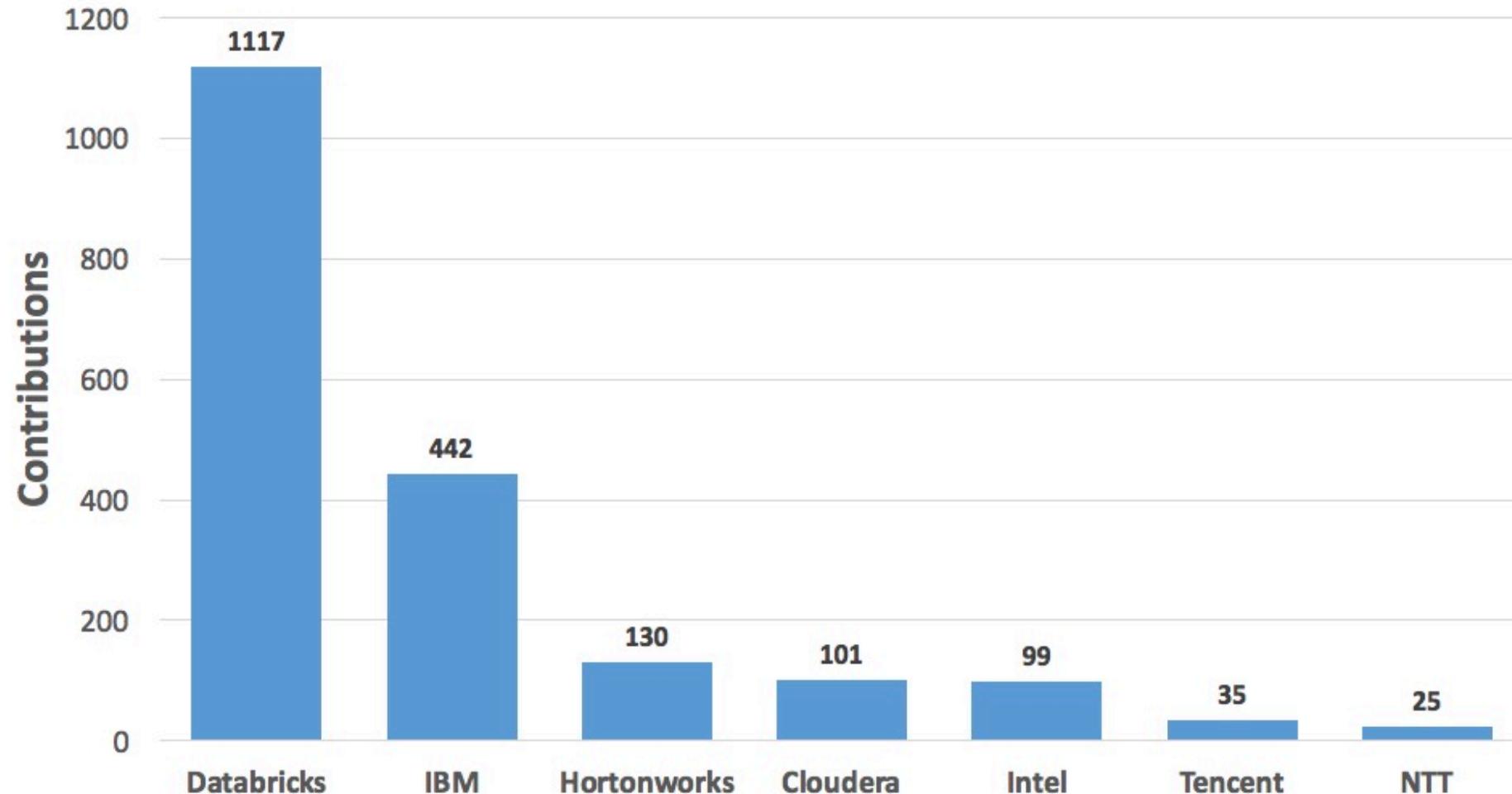
Contribute to the Core: Spark Technology Center



Contribute to the Core: Actual Code!

Top 7 Contributing Companies to Spark 2.0.0

(Represents 70% of total contributions)



Infuse the Portfolio

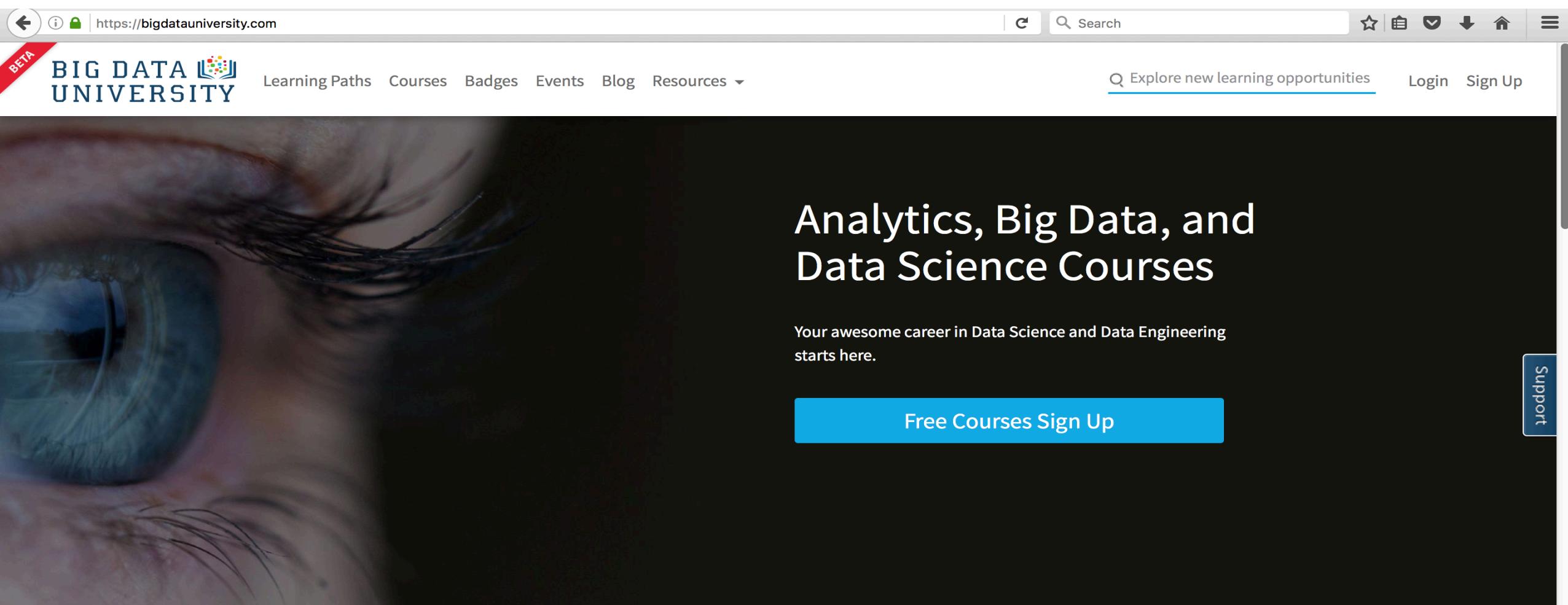


IBM Bluemix



IBM Watson Health

Foster the Community: Big Data University



The website features a large, high-resolution close-up of a person's eye on the left side of the main content area. The eye is dark brown with long, dark eyelashes. The rest of the page has a dark background.

BETA

BIG DATA UNIVERSITY

Learning Paths Courses Badges Events Blog Resources ▾

Search

Explore new learning opportunities

Login Sign Up

Analytics, Big Data, and Data Science Courses

Your awesome career in Data Science and Data Engineering starts here.

Free Courses Sign Up

Support

What are the benefits?

Foster the Community: Data Science Experience



IBM Data Science Experience

Log In

Sign Up

Master the art of data science.

Enjoy the best of open source and collaborate in a social environment, built for data scientists by data scientists.

Get Started

Watch the video

This is a preview of Data Science Experience, stay tuned for additional features.

Use what you know, learn
what you don't

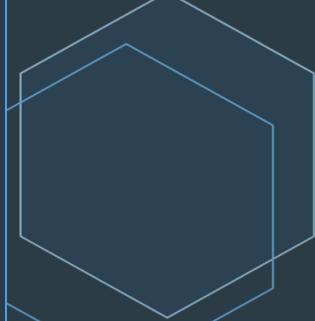
Start a course, start from a sample, or

Foster the Community: Bluemix

https://new-console.ng.bluemix.net

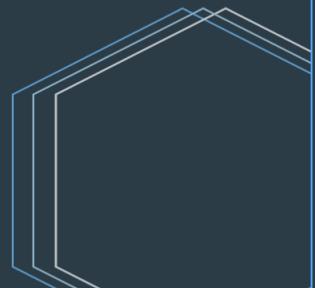
IBM Bluemix Console ▾ Go to Classic Experience Docs Log In Sign Up

Welcome to Bluemix



Start Using the Bluemix Platform
Bluemix is the home of 130+ unique services, including offerings like IBM Watson and Weather.com, and millions of running applications, containers, servers, and more.

[Go to Catalog](#)



Ready to start?

[Create a Free Account](#) [Log In](#)

Learn More about Bluemix:
[Pricing](#) [Products](#) [Blog](#) [Status](#)

Explore by Category:

 Compute	 Network	 Storage	 Data & Analytics	 Watson	 Internet of Things
 https://new-console.ng.bluemix.net/dashboard/storage					

Apache Spark Use Cases



Use Case: SolutionInc



Business challenge

Wi-Fi provider SolutionInc wanted to give its clients insight into user engagement with public Wi-Fi spots. To extract value from its Wi-Fi network data, SolutionInc needed a new analytics capability.



Glen Lavigne
President and Chief Executive Officer
SolutionInc

Transformation

SolutionInc uses IBM's managed Spark service, IBM Analytics for Apache Spark, to mine network activity data that reveals customer behavior as customers visit and move between different locations.

Business benefits:

241 million
data sets analyzed and visualized rapidly in the cloud

Enables

the development of new location-based analytics services

Supports

better design decisions to adapt locations to customer needs

SolutionInc

Rapid big data analytics opens up new business opportunities

SolutionInc is a leader in managing public Internet access in hotels, convention centers, airports, coffee shops and other public venues around the world. The company offers on-premise and cloud-based solutions to provide high-demand public Wi-Fi solutions, enhancing the overall customer experience.

"Data analytics help customers translate investments in Wi-Fi solutions into a valuable business tool."

Glen Lavigne
President and Chief Executive Officer
SolutionInc

Share this



Use Case: Sensitel



Business challenge

Many retailers suffer from the same headache. When a shopper enters a store seeking advice, if they cannot find a sales assistant quickly, they leave empty-handed – and the store loses out on a sale.

Transformation

Sensitel SENS analyzes sensor, Wi-Fi, and video data to recognize shoppers' faces and track their in-store locations, helping retailers send staff to serve people who need help, and make personalized offers.



Ray Sikka
CEO
Sensitel

Business benefits:

Recognizes
faces and anonymizes visit patterns, enabling 1-on-1 marketing and service

Accelerates
insight, enabling real-time decision-making while customers are in-store

Increases
sales by an estimated 5% for every 10% increase in staff engaging with customers

Sensitel

Harnessing analytics to deliver a smarter shopping experience

Sensitel specializes in leveraging Internet of Things (IoT) and big data technologies to help companies make smarter decisions and deliver excellent customer experiences. The firm is based in Santa Clara, California.

"IBM is helping us develop innovative IoT solutions that will change the face of entire industries."

Ray Sikka
CEO
Sensitel

Share this



What is Apache Spark?

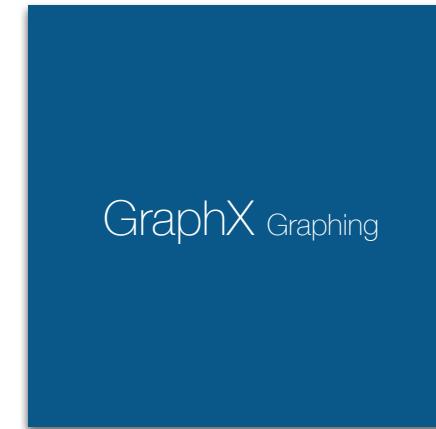
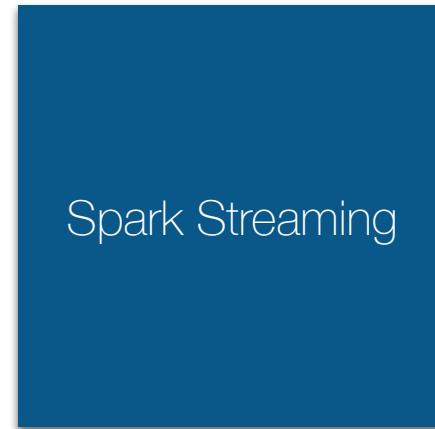


Introducing Apache Spark

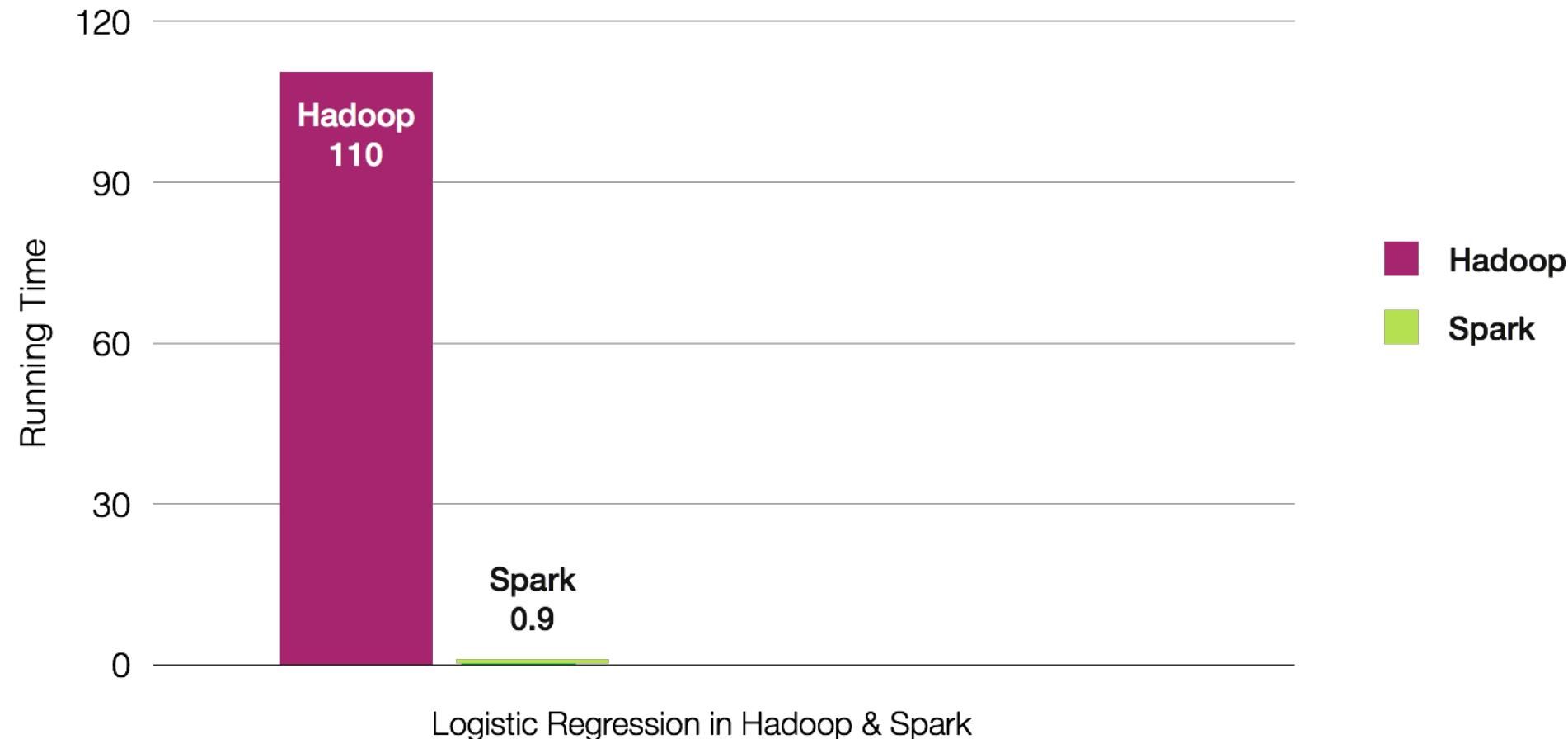
- **General purpose distributed data processing engine**
- **AMPLab at UC Berkeley**
 - Started by Matei Zaharia in 2009
 - Creators founded Databricks
 - IBM partners with DB
- **Apache open source project**
 - Initial release May 2014
 - Current version 2.0.2 released November 2016



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, let's walk through an example MapReduce application to get a flavour for how they work.

WordCount is a simple application that counts the number of occurrences 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

```
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()) {
22.                 word.set(tokenizer.nextToken());
23.                 output.collect(word, one);
24.             }
25.         }
26.     }
27.
28.     public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
29.         public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
30.             int sum = 0;
31.             for (IntWritable val : values) {
32.                 sum += val.get();
33.             }
34.             output.collect(key, new IntWritable(sum));
35.         }
36.     }
37.
38.     public static void main(String[] args) throws Exception {
39.         JobConf conf = new JobConf(WordCount.class);
40.         conf.set("mapred.mapper.class", WordCount.Map.class);
41.         conf.set("mapred.reducer.class", WordCount.Reduce.class);
42.         conf.set("mapred.inputformat", TextInputFormat.class.getName());
43.         conf.set("mapred.outputformat", TextOutputFormat.class.getName());
44.         conf.setMapperClass(Map.class);
45.         conf.setCombinerClass(Reduce.class);
46.         conf.setReducerClass(Reduce.class);
47.         conf.setInputFormat(TextInputFormat.class);
48.         conf.setOutputFormat(TextOutputFormat.class);
49.         FileInputFormat.setInputPaths(conf, new Path(args[0]));
50.         FileOutputFormat.setOutputPath(conf, new Path(args[1]));
51.         JobClient.runJob(conf);
52.     }
53. }
```

```
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()) {
22.                 word.set(tokenizer.nextToken());
23.                 output.collect(word, one);
24.             }
25.         }
26.     }
27.
28.     public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
29.         public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
30.             int sum = 0;
31.             for (IntWritable val : values) {
32.                 sum += val.get();
33.             }
34.             output.collect(key, new IntWritable(sum));
35.         }
36.     }
37.
38.     public static void main(String[] args) throws Exception {
39.         JobConf conf = new JobConf(WordCount.class);
40.         conf.set("mapred.mapper.class", WordCount.Map.class);
41.         conf.set("mapred.reducer.class", WordCount.Reduce.class);
42.         conf.set("mapred.inputformat", TextInputFormat.class.getName());
43.         conf.set("mapred.outputformat", TextOutputFormat.class.getName());
44.         conf.setMapperClass(Map.class);
45.         conf.setCombinerClass(Reduce.class);
46.         conf.setReducerClass(Reduce.class);
47.         conf.setInputFormat(TextInputFormat.class);
48.         conf.setOutputFormat(TextOutputFormat.class);
49.         FileInputFormat.setInputPaths(conf, new Path(args[0]));
50.         FileOutputFormat.setOutputPath(conf, new Path(args[1]));
51.         JobClient.runJob(conf);
52.     }
53. }
```

Why Spark? Community

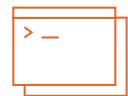
- Open source community
- Fastest growing
- Wide adoption



SPARK MEETUP
MEMBERS

+240%

2015 2016
66,000 225,000



CODE
CONTRIBUTORS

+67%

2015 2016
600 1000



SPARK SUMMIT
ATTENDEES

+30%

2015 2016
3912 5100



NUMBER OF COMPANIES
AT SUMMITS

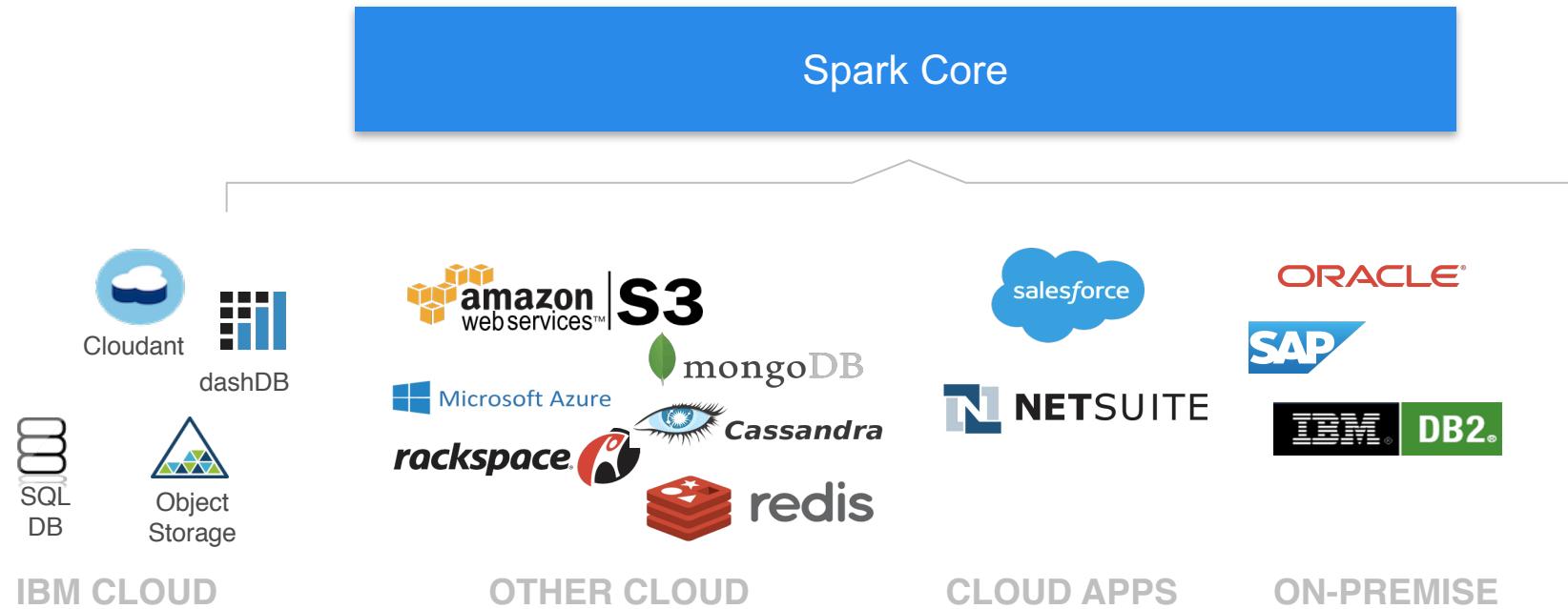
+57%

2015 2016
1144 1800

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**

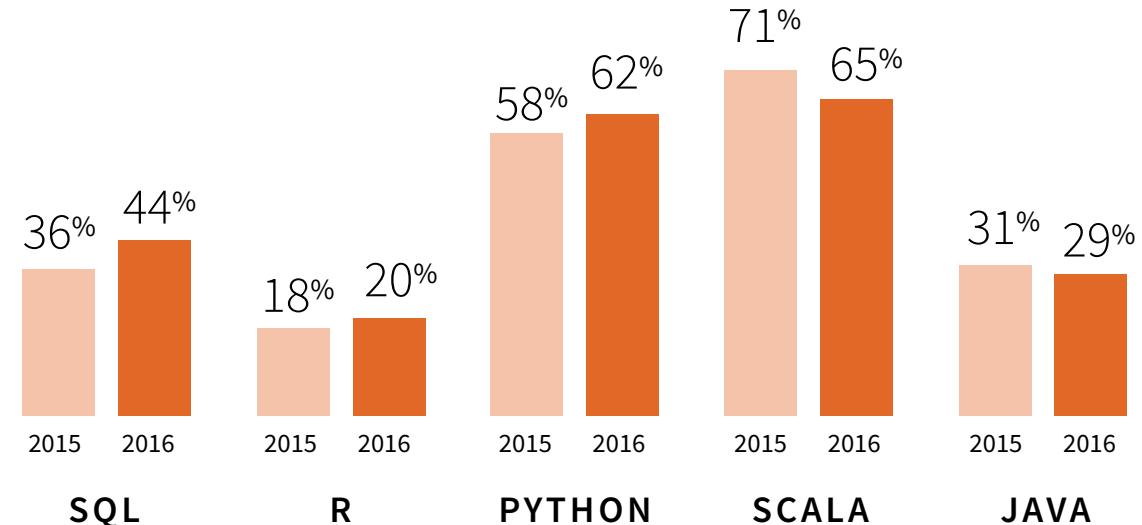
- New features in Java 8 makes for more compact coding

- **R**

- Statistical language used to create and manipulate data

LANGUAGES USED IN SPARK YEAR-OVER-YEAR

% of respondents who use each language (more than one language could be selected)



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

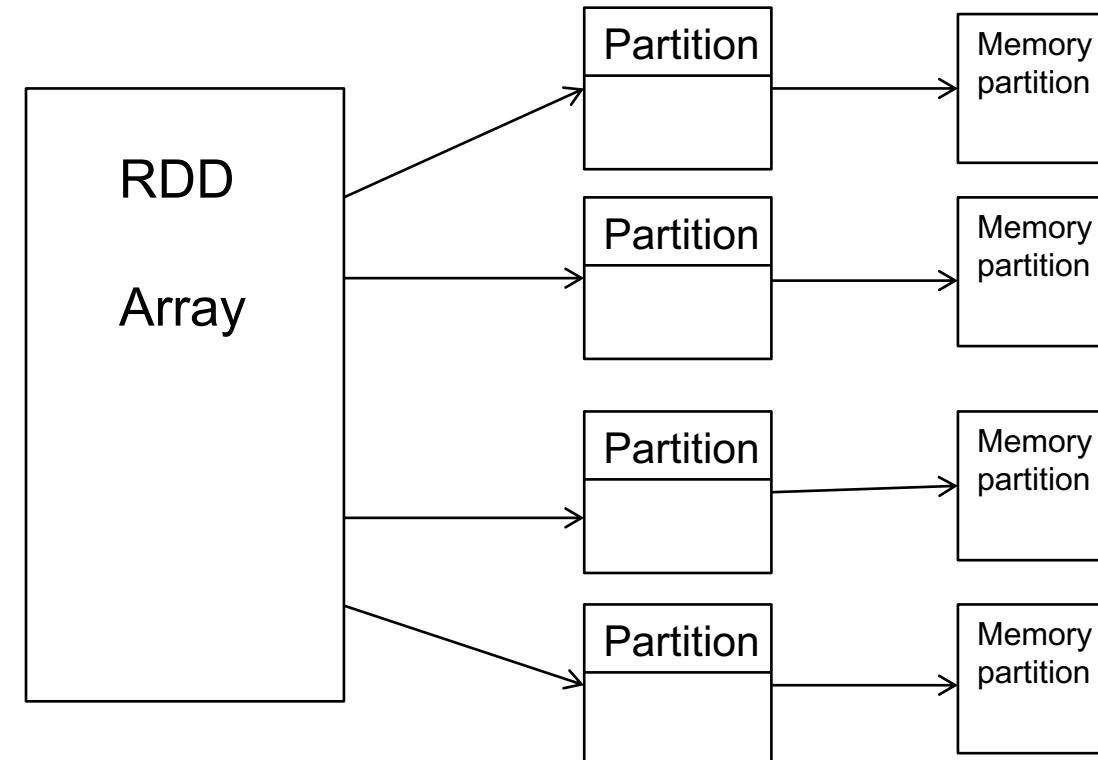
–`function(lambda x: x)`

▪ Scala

–`function(x => x)`

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

More about RDDs and DFs

- **Two types of operations**

- **Transformations**

- `map()`
 - `filter()`
 - `flatMap()`

- **Actions**

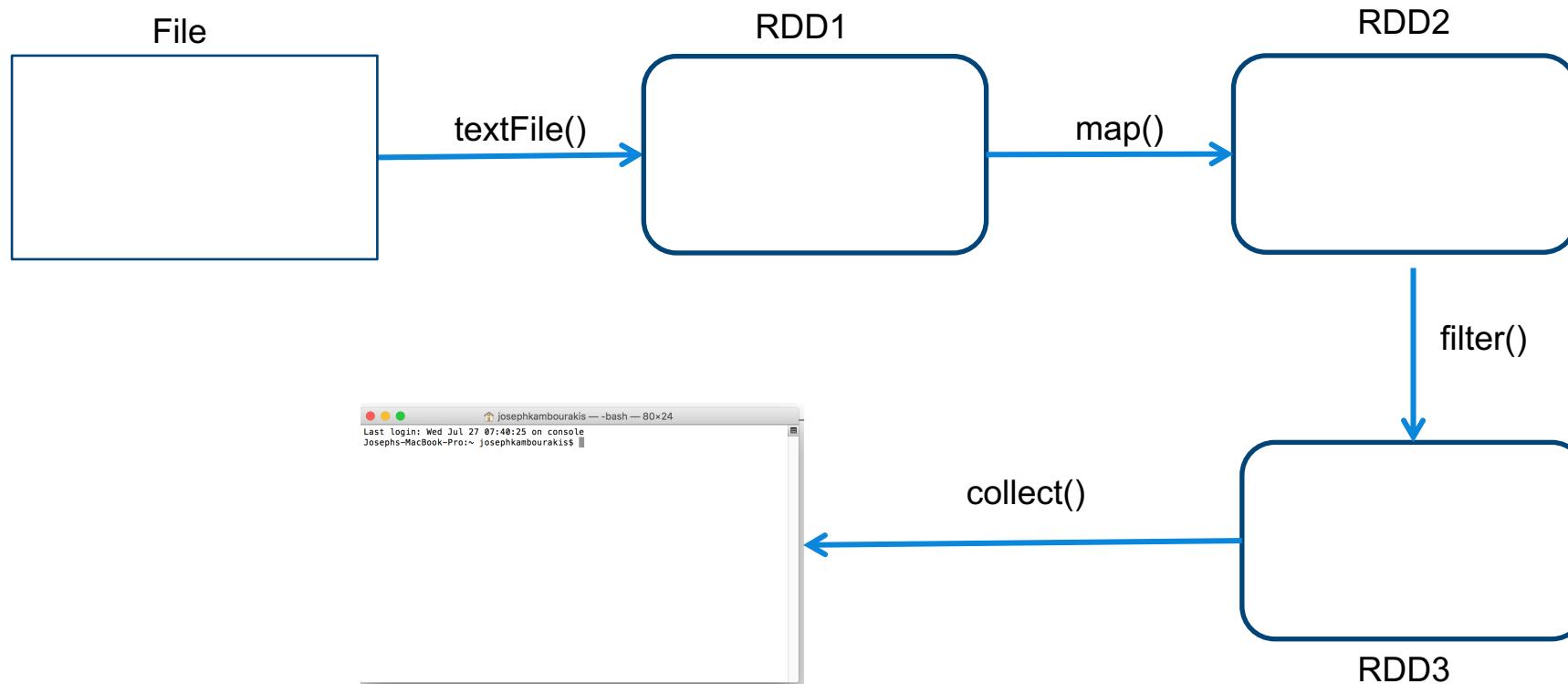
- `first()`
 - `collect()`
 - `take()`

- **Lazy evaluation**

Apache Spark Code Execution

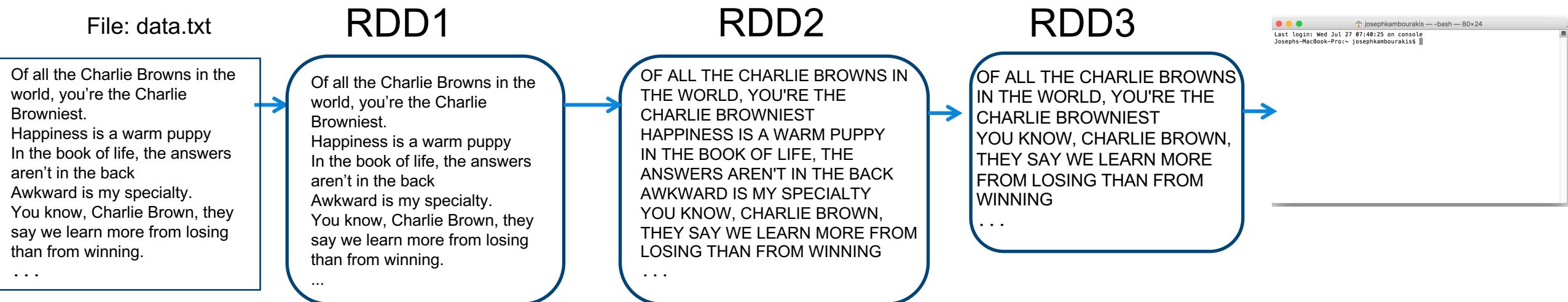


Directed Acyclic Graph - DAG



Sample Code

```
▪ RDD1 = sc.textFile(data.txt)
RDD2 = RDD1.map(lambda line: line.upper())
RDD3 = RDD2.filter(lambda line: 'CHARLIE' in line)
RDD3.collect()
```



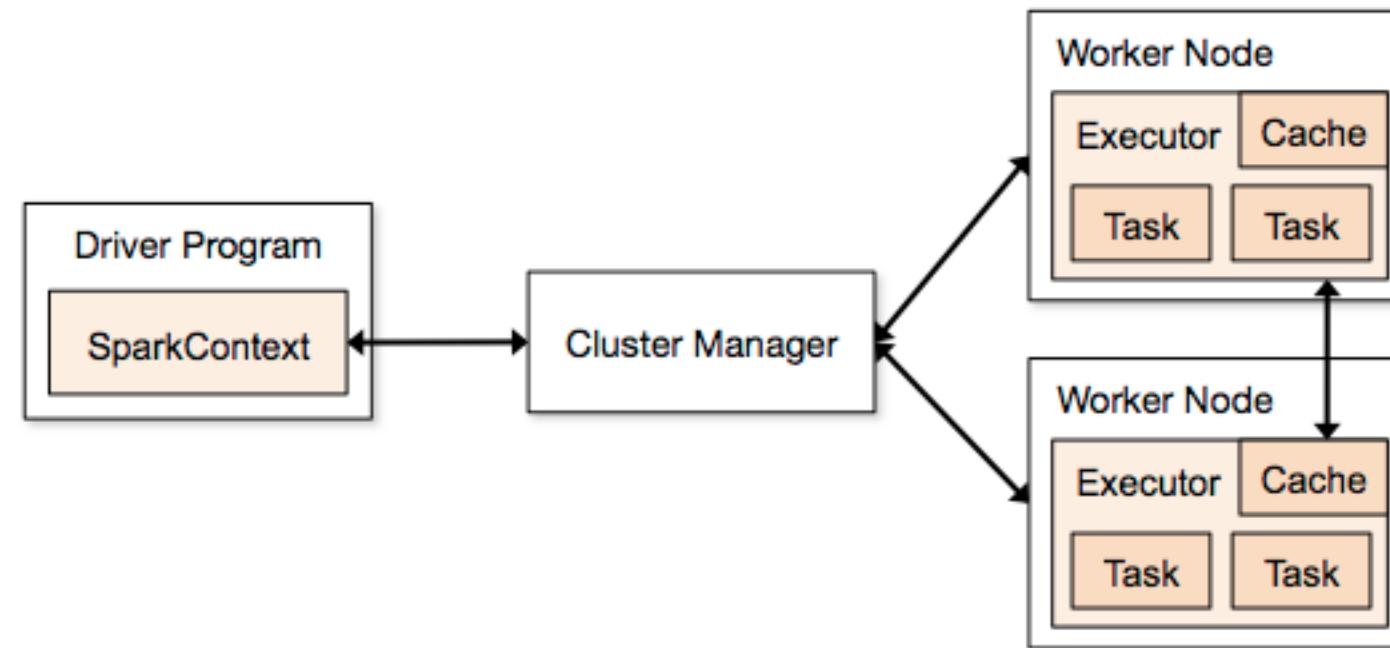
Apache Spark Under the hood!



Spark Application Architecture

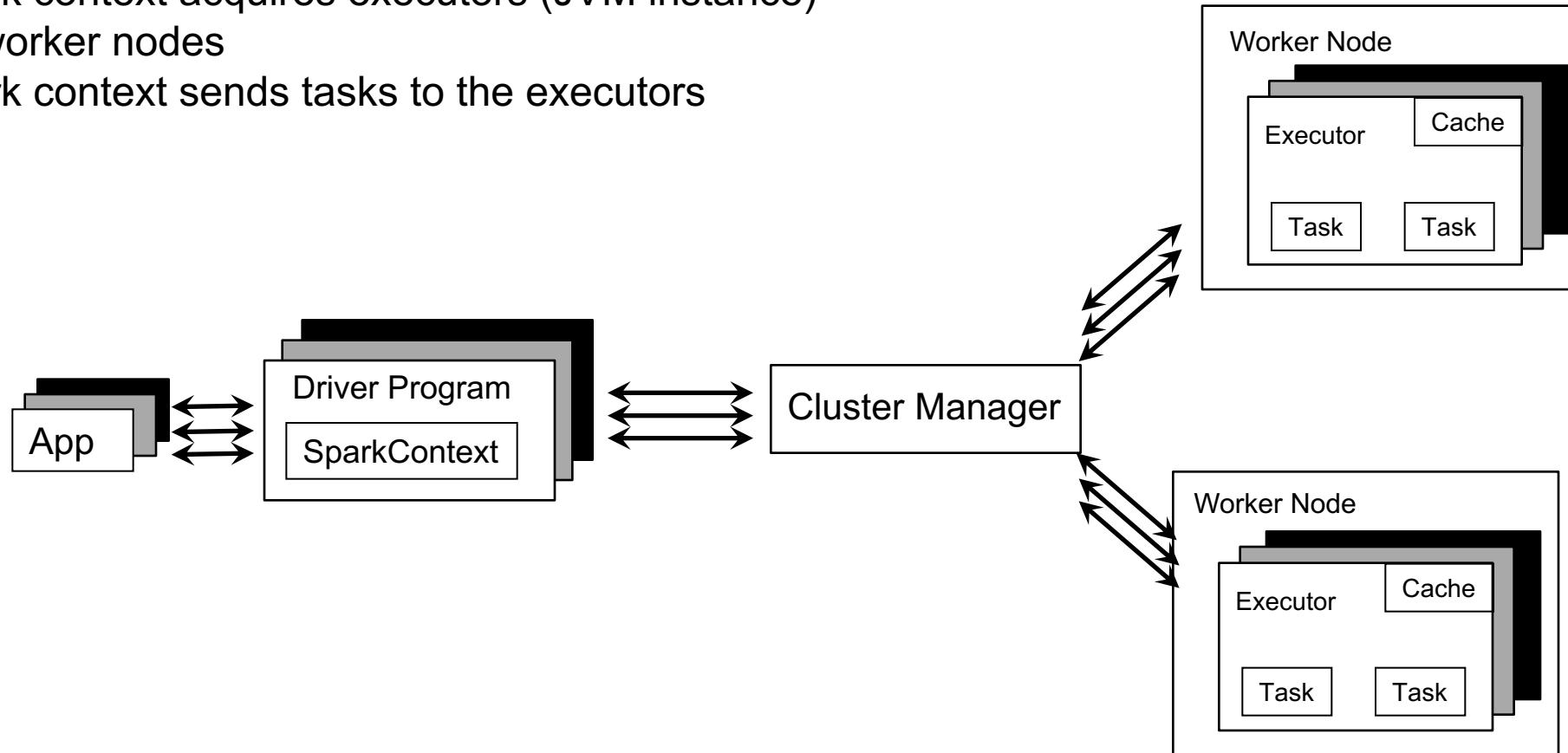


- A Spark application is initiated from a driver program
- Spark execution modes:
 - Standalone with the built-in cluster manager
 - Use Mesos as the cluster manager
 - Use YARN as the cluster manager
 - Standalone cluster on any cloud (BlueMix, IBM Softlayer, Amazon, Azure, ...)

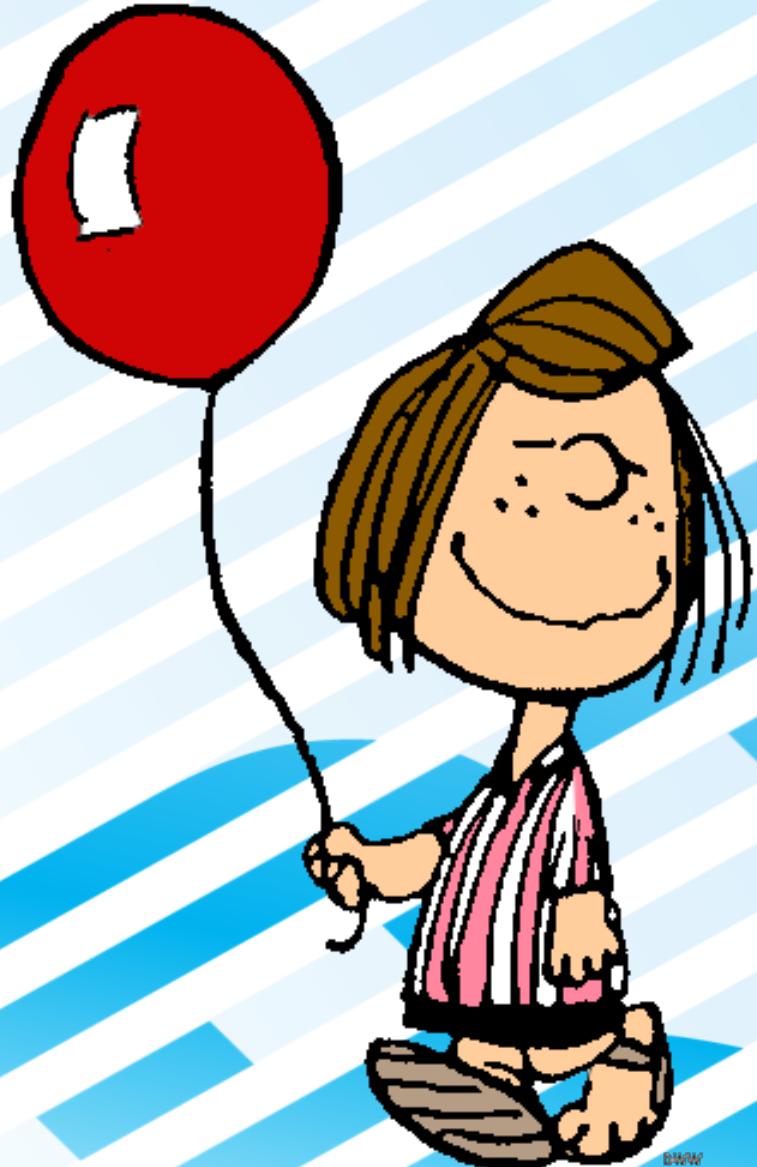


Showing multiple applications

- 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**

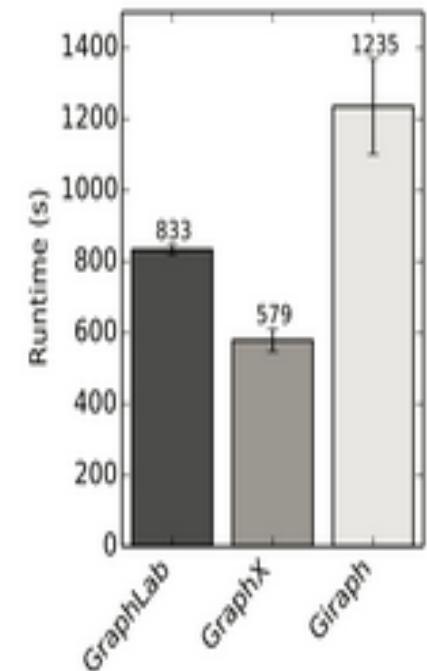


Spark MLlib

- **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

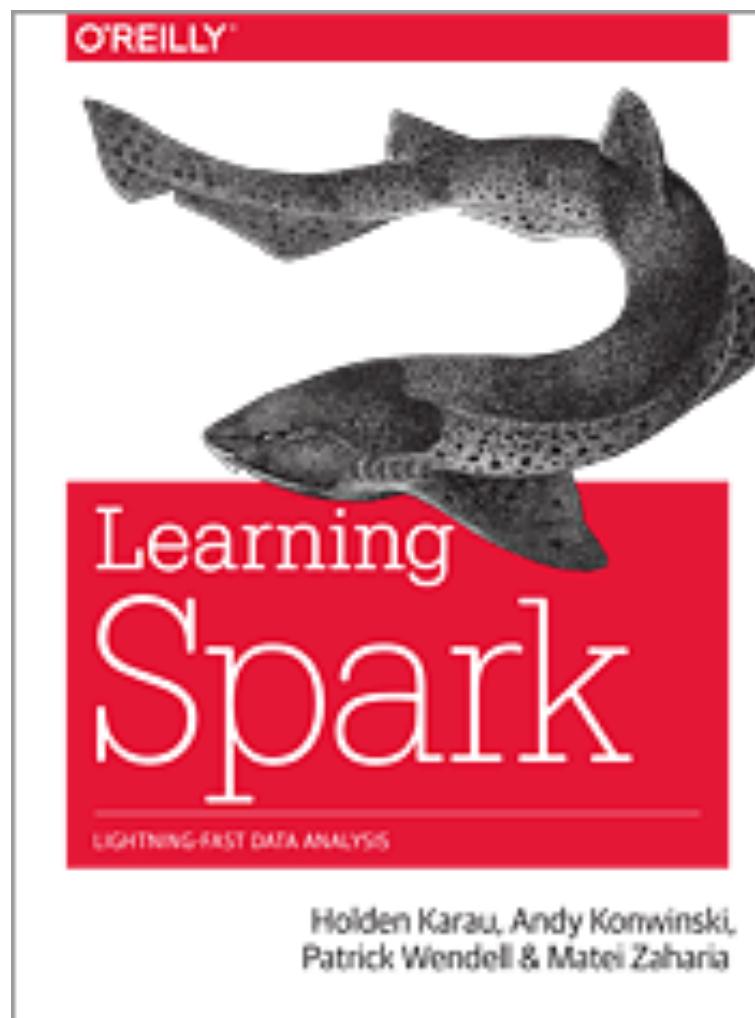
- **Scala only!**
- **Flexible Graphing**
 - GraphX unifies ETL, exploratory analysis, and iterative graph computation
 - You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms with the API
 - GraphFrames - 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

The screenshot shows the Apache Spark website on a web browser. The header includes the Apache logo and the text "Lightning-fast cluster computing". The main navigation bar has links for Download, Libraries, Documentation, Examples, Community, and FAQ. A sub-navigation bar for the Apache Software Foundation is also present. A banner at the top states: "Apache Spark™ is a fast and general engine for large-scale data processing." Below this, there are two sections: "Speed" and "Ease of Use". The "Speed" section compares Hadoop and Spark's running times for logistic regression, showing Spark is significantly faster. The "Ease of Use" section shows sample Scala code for processing a text file. To the right, there is a sidebar for "Latest News" and a "Download Spark" button.

Apache Spark™ is a fast and general engine for large-scale data processing.

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

Ease of Use

Write applications quickly in Java, Scala, Python, R.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it interactively from the Scala, Python and R shells.

Running time (s)

System	Running time (s)
Hadoop	110
Spark	0.9

Logistic regression in Hadoop and Spark

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

Latest News

- Spark 2.0.0 released (Jul 26, 2016)
- Spark 1.6.2 released (Jun 25, 2016)
- Call for Presentations for Spark Summit EU is Open (Jun 16, 2016)
- Preview release of Spark 2.0 (May 26, 2016)

[Archive](#)

Download Spark

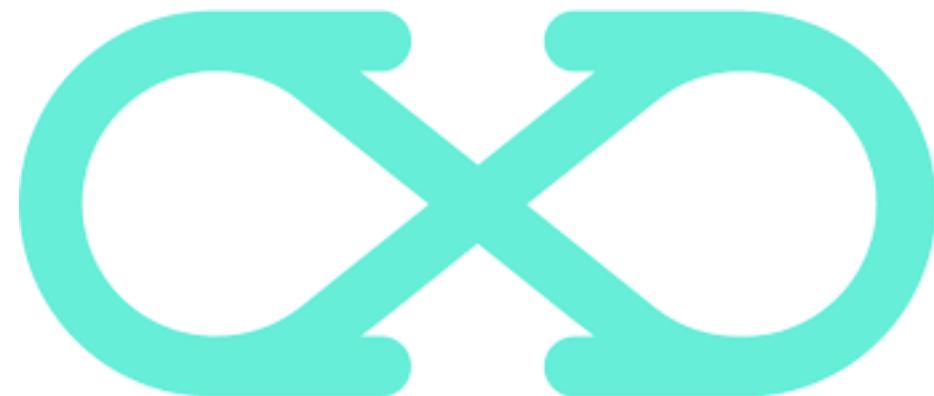
Built-in Libraries:

- SQL and DataFrames
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)

Third-Party Packages

Intro to lab

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



Intro to lab

Figure 1. Magic Quadrant for Data Science Platforms



Source: Gartner (February 2017)

Links to Material

- **Box**

- <https://ibm.box.com/v/MetLife>

- **Github**

- <https://github.com/JosephKambourakisIBM/Metlife>