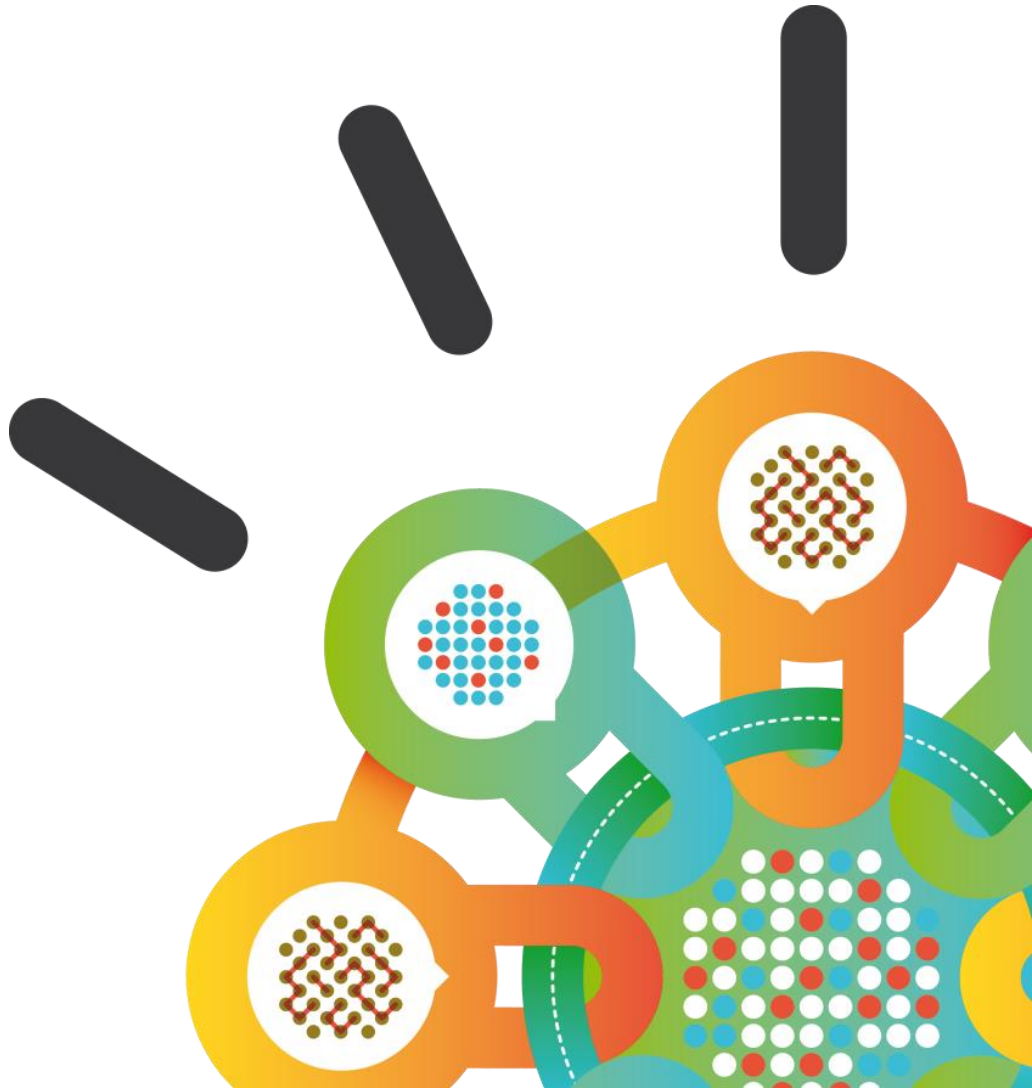
Analytic Cloud with Spark

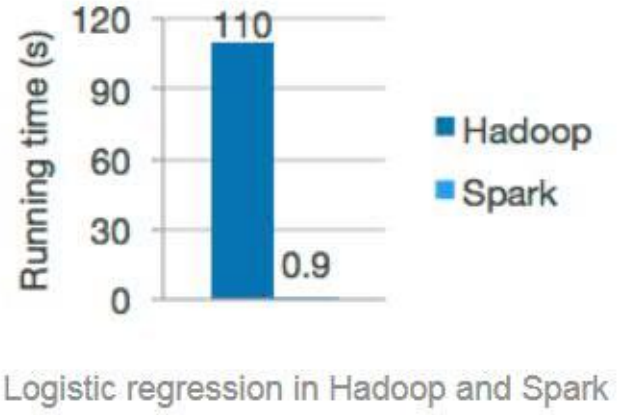


Shelly Garion

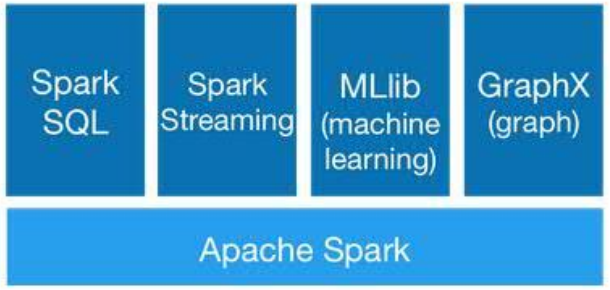
IBM Research -- Haifa

Why Spark?

- **Apache Spark™** is a fast and general open-source cluster computing engine for big data processing
- **Speed:** Spark is capable to run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk
- **Ease of use:** Write applications quickly in Java, Scala, Python and R, also with notebooks
- **Generality:** Combine SQL, streaming, and complex analytics – machine learning, graph processing
- **Runs everywhere:** runs on Apache Mesos, Hadoop YARN cluster manager, standalone, or in the cloud, and can read any existing Hadoop data, and data from HDFS, object store, databases etc.



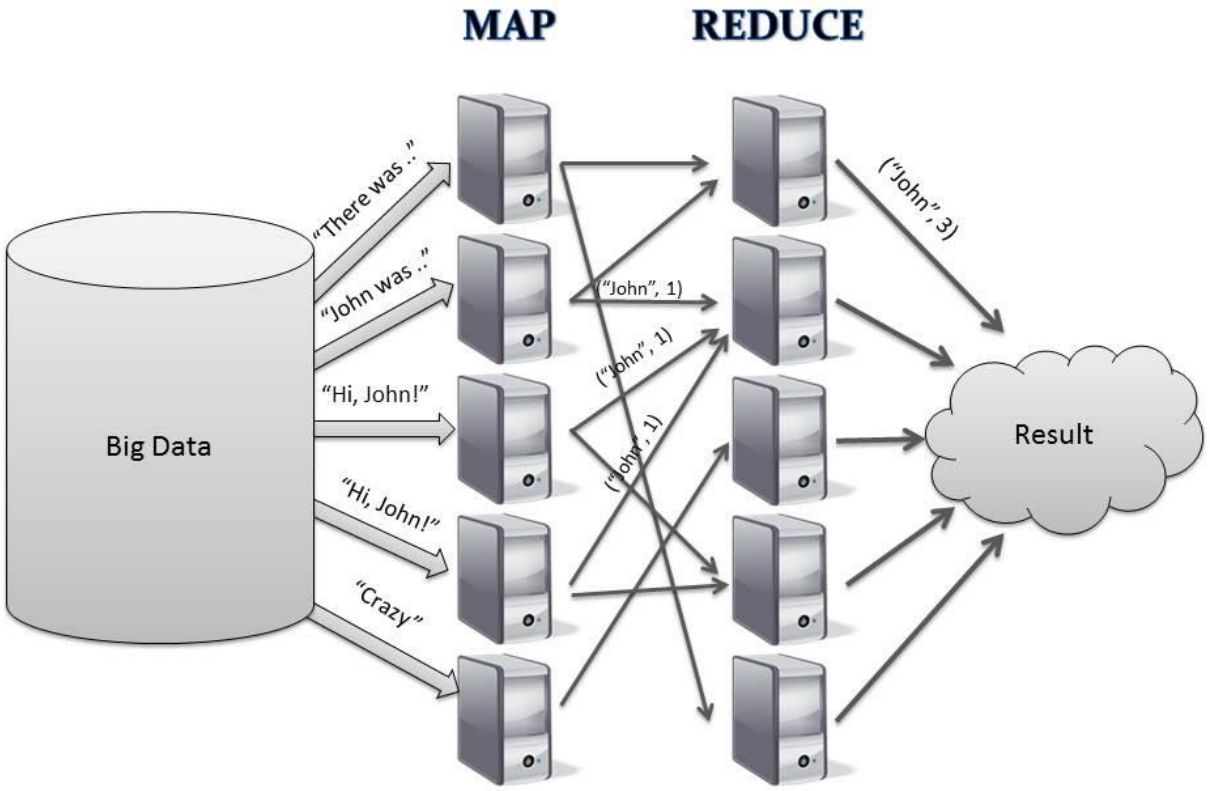
Logistic regression in Hadoop and Spark



History of Spark

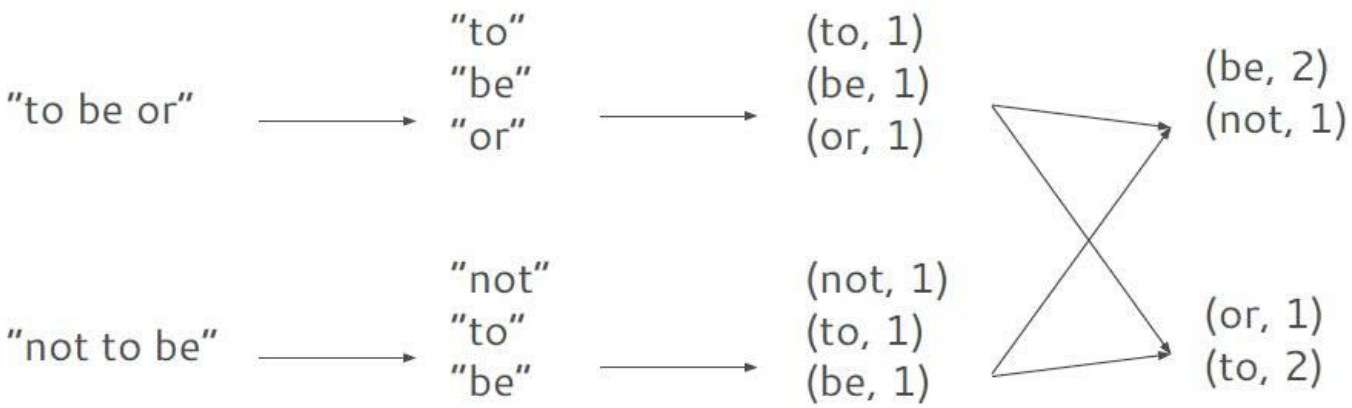
- Started in 2009 as a research project of UC Berkley
- Now it is an open source Apache project
  - Built by a wide set of developers from over 200 companies
  - more than 1000 developers have contributed to Spark
- IBM has decided to “bet big on Spark” at June 2015
  - Created Spark Technology Center (STC) - http://www.spark.tc/
  - “Spark as a Service” on Bluemix

How to Analyze BigData?



Basic Example: Word Count (Spark & Python)

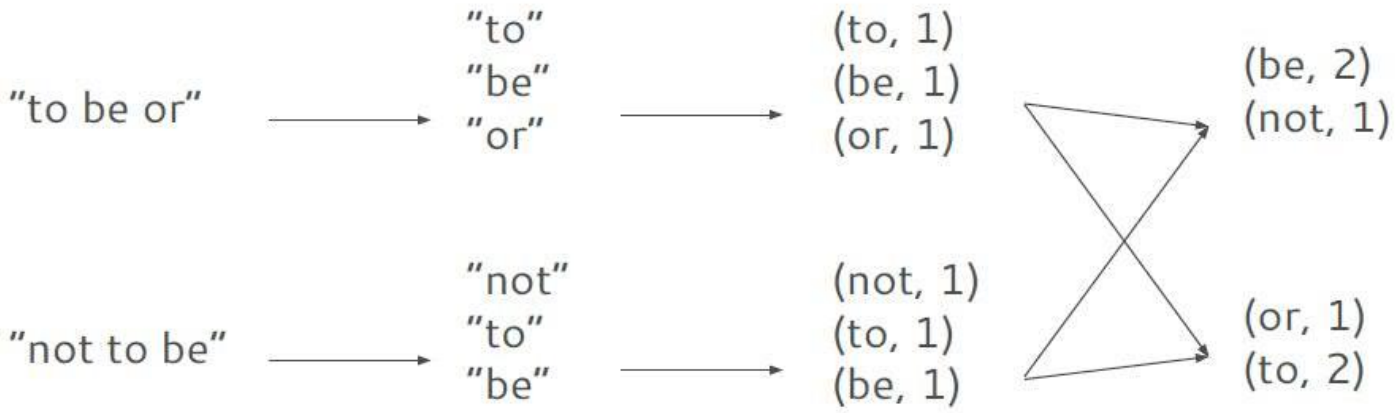
```
>lines = sc.textFile("hamlet.txt")
>counts = lines.flatMap(lambda line: line.split(" "))
               .map(lambda word => (word, 1))
               .reduceByKey(lambda x, y: x + y)
```



Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Basic Example: Word Count (Spark & Scala)

```
>val lines = sc.textFile("hamlet.txt")
>val counts = lines.flatMap(_.split(" "))
                  .map((_, 1))
                  .reduceByKey(_ + _)
```

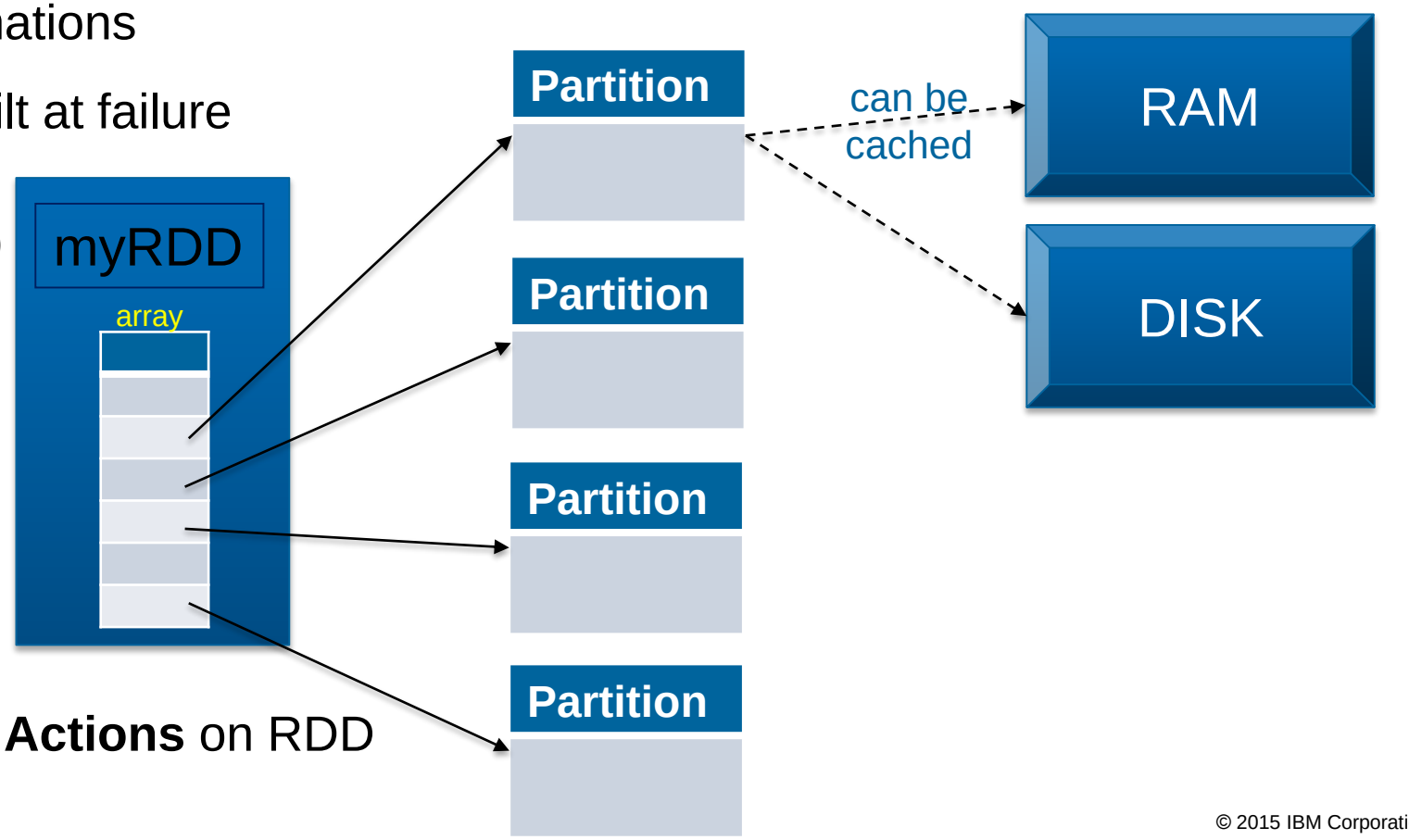


Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark RDD (Resilient Distributed Dataset)

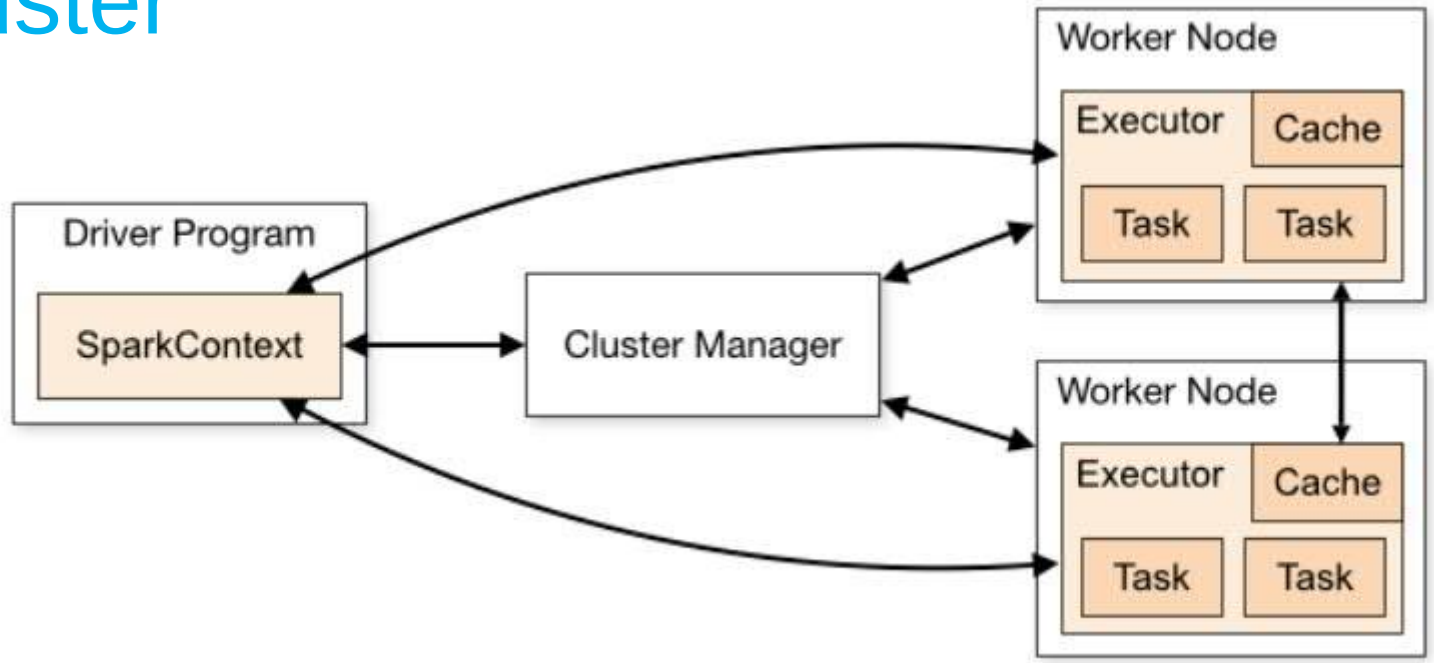
- Immutable, partitioned collections of objects spread across a cluster, stored in RAM or on Disk
- Built through lazy parallel transformations
- **Fault tolerance** – automatically built at failure

var *myRDD* **=** *sc.sequenceFile("hdfs:///…")*



- We can apply **Transformations** or **Actions** on RDD

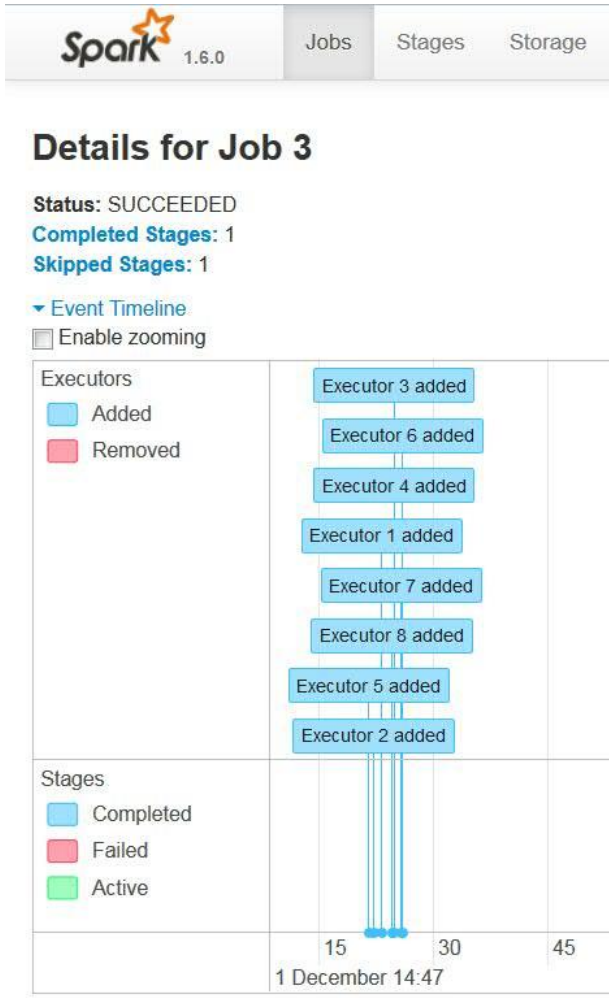
Spark Cluster

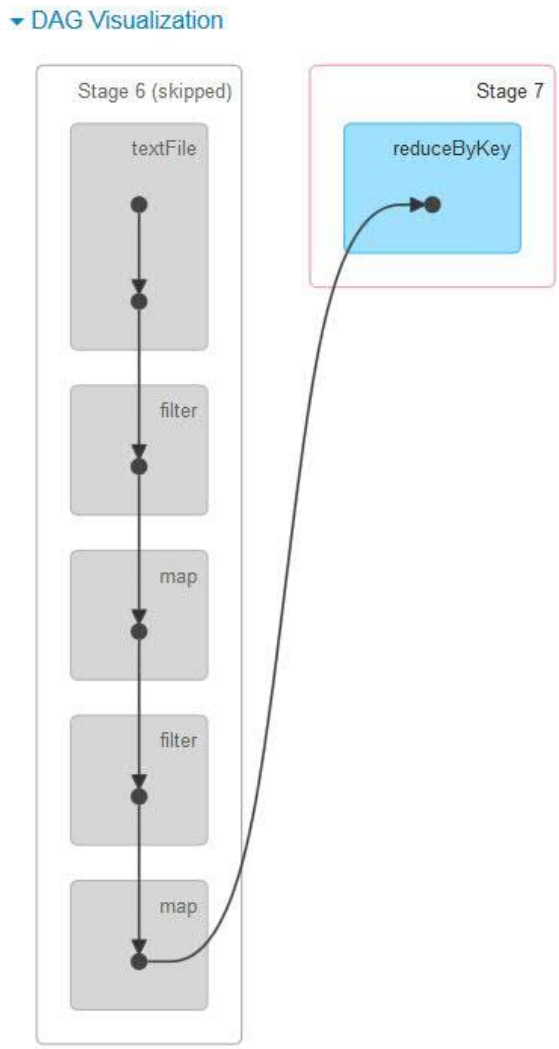


- **Driver program** – The process running the main() function of the application and creating the SparkContext
- **Cluster manager** – External service for acquiring resources on the cluster (e.g. standalone, Mesos, YARN)
- **Worker node** - Any node that can run application code in the cluster
- **Executor** – A process launched for an application on a worker node

Spark Scheduler

- **Task** - A unit of work that will be sent to one executor
- **Job** - A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action
- **Stage** - Each job gets divided into smaller sets of tasks called *stages* that depend on each other





Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
7	take at <console>:44 +details	2016/12/01 14:51:40	0.3 s	53/53			130.9 KB	

Scala

- **Spark** was originally written in Scala
  - Java and Python API were added later
- **Scala:** high-level language for the JVM
  - Object oriented
  - Functional programming
  - Immutable
  - Inspired by criticism of the shortcomings of Java
- Static types
  - Comparable in speed to Java
  - Type inference saves us from having to write explicit types most of the time
- Interoperates with Java
  - Can use any Java class
  - Can be called from Java code

Scala vs. Java

Declaring variables:

```
var x: Int = 7
var x = 7 // type inferred
val y = "hi" // read-only
```

Java equivalent:

```
int x = 7;

final String y = "hi";
```

Functions:

```
def square(x: Int): Int = x*x
def square(x: Int): Int = {
  x*x
}
def announce(text: String) =
{
  println(text)
}
```

Java equivalent:

```
int square(int x) {
  return x*x;
}

void announce(String text) {
  System.out.println(text);
}
```

Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark & Scala: Creating RDD

```
# Turn a Python collection into an RDD
>sc.parallelize([1, 2, 3])

# Turn a Scala collection into an RDD
>sc.parallelize(List(1, 2, 3))

# Load text file from local FS, HDFS, or S3
>sc.textFile("file.txt")
>sc.textFile("directory/*.txt")
>sc.textFile("hdfs://namenode:9000/path/file")
>sc.textFile("swift://ContainerName.spark/ObjectName")
```

or SoftLayer object store

Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark & Scala: Basic Transformations

```
>val nums = sc.parallelize(List(1, 2, 3))

// Pass each element through a function
>val squares = nums.map(x: x*x)   // {1, 4, 9}

// Keep elements passing a predicate
>val even = squares.filter(x => x % 2 == 0) // {4}

// Map each element to zero or more others
>nums.flatMap(x => 0.to(x))
//=> {0, 1, 0, 1, 2, 0, 1, 2, 3}
```

Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark & Scala: Basic Actions

```
>val nums = sc.parallelize(List(1, 2, 3))

// Retrieve RDD contents as a local collection
>nums.collect() //=> List(1, 2, 3)

// Return first K elements
>nums.take(2)   //=> List(1, 2)

// Count number of elements
>nums.count()   //=> 3

// Merge elements with an associative function
>nums.reduce{case (x, y) => x + y}  //=> 6

// Write elements to a text file
>nums.saveAsTextFile("hdfs://file.txt")
```

Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark & Scala: Key-Value Operations

```
>pets = sc.parallelize(
  List(("cat", 1), ("dog", 1), ("cat", 2)))
>pets.reduceByKey(_ + _)
                  //=> ((cat, 3), (dog, 1))
>pets.groupByKey() //=> {(cat, [1, 2]), (dog, [1])}
>pets.sortByKey()  //=> {(cat, 1), (cat, 2), (dog, 1)}
```

Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Example: Spark Core API

Goal:

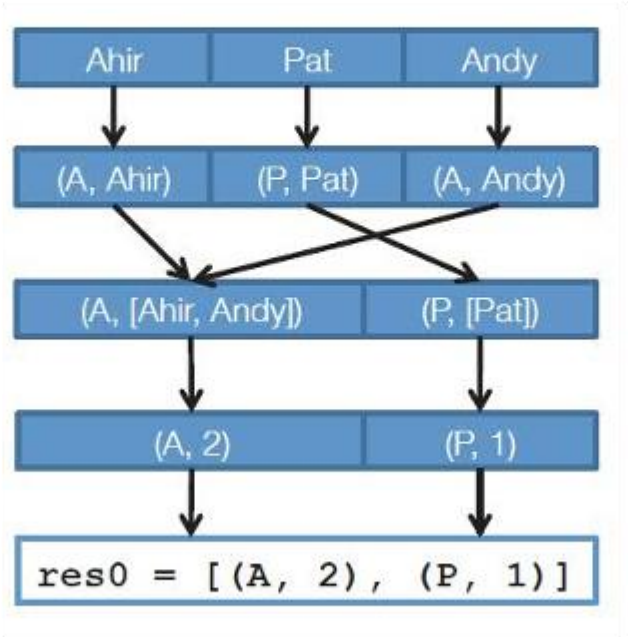
Find number of distinct names per "first letter".

Aaron Davidson, *A deeper understanding of Spark internals*, Spark Summit July 2014, https://spark-summit.org/2014/

Example: Spark Core API

Goal:

Find number of distinct names per "first letter".

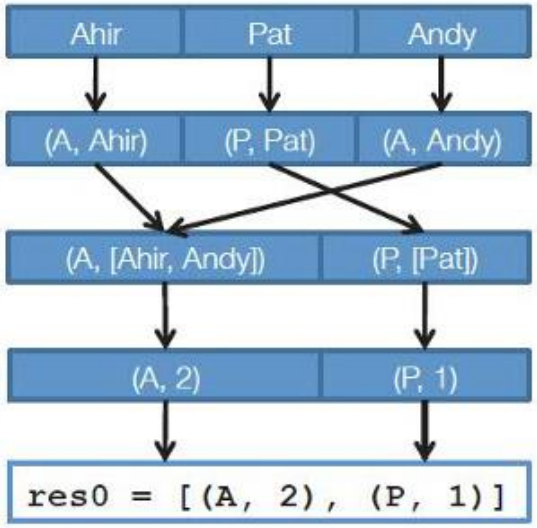


Aaron Davidson, *A deeper understanding of Spark internals*, Spark Summit July 2014, https://spark-summit.org/2014/

Example: Spark Core API

Goal: Find number of distinct names per "first letter"

```
sc.textFile("hdfs:/names")
  .map(name => (name.charAt(0), name))
  .groupByKey()
  .mapValues(names => names.toSet.size)
  .collect()
```



Aaron Davidson, *A deeper understanding of Spark internals*, Spark Summit July 2014, https://spark-summit.org/2014/

Example: Spark Core API

Better implementation:

```
sc.textFile("hdfs:/names")
   .distinct(numPartitions = 6)
   .map(name => (name.charAt(0), 1))
   .reduceByKey(_ + _)
   .collect()
```

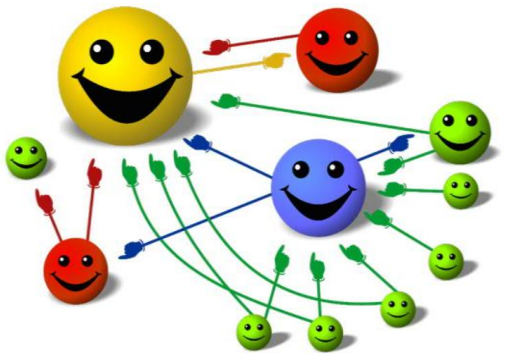
Original:

```
sc.textFile("hdfs:/names")
   .map(name => (name.charAt(0), name))
   .groupByKey()
   .mapValues { names => names.toSet.size }
   .collect()
```

Aaron Davidson, *A deeper understanding of Spark internals*, Spark Summit July 2014, https://spark-summit.org/2014/

Example: PageRank

Popular algorithm originally introduced by Google



How to implement PageRank algorithm using Map/Reduce?

```
val links = // load RDD of (url, neighbors) pairs
var ranks = // load RDD of (url, rank) pairs

for (i <- 1 to ITERATIONS) {
  val contribs = links.join(ranks).flatMap {
    case (url, (links, rank)) =>
      links.map(dest => (dest, rank/links.size))
  }
  ranks = contribs.reduceByKey(_ + _)
    .mapValues(0.15 + 0.85 * _)
}
ranks.saveAsTextFile(...)
```

PageRank Algorithm

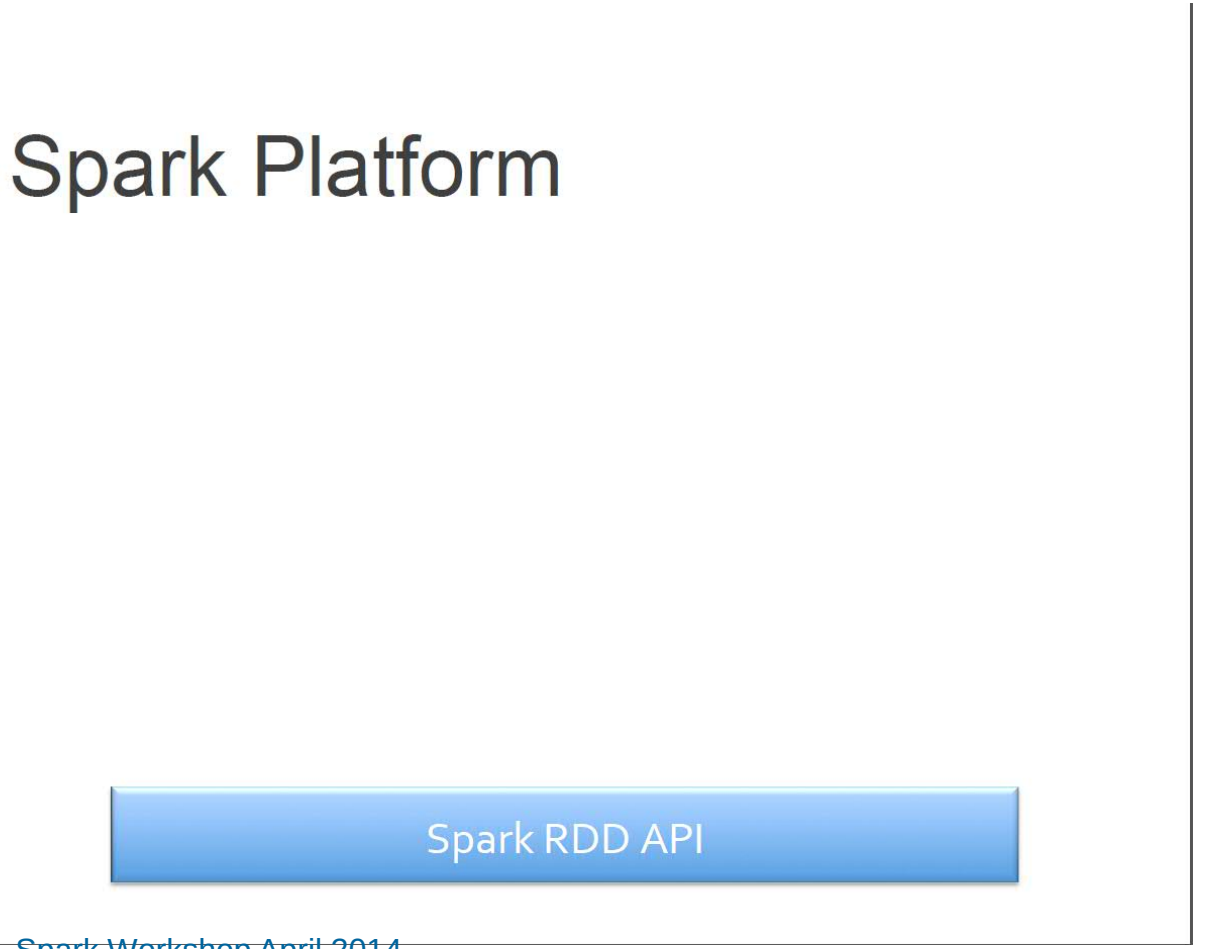
- Start each page with a rank of 1
- On each iteration:

A. $contrib = \frac{curRank}{|neighbors|}$

B. $curRank = 0.15 + 0.85 \sum_{neighbors} contrib_i$

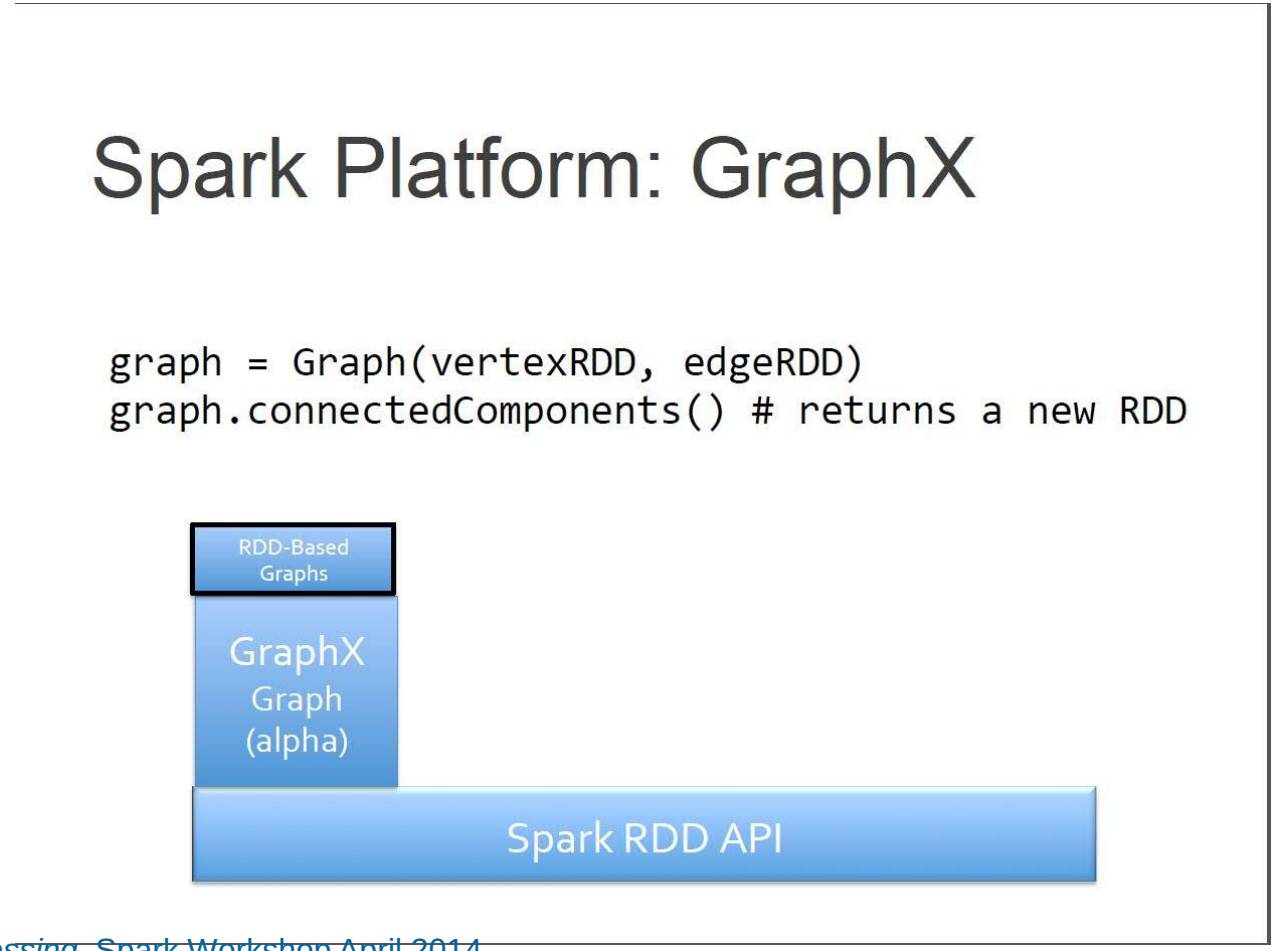
Hossein Falaki, *Numerical Computing with Spark*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark Platform



Patrick Wendell, *Big Data Processing*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

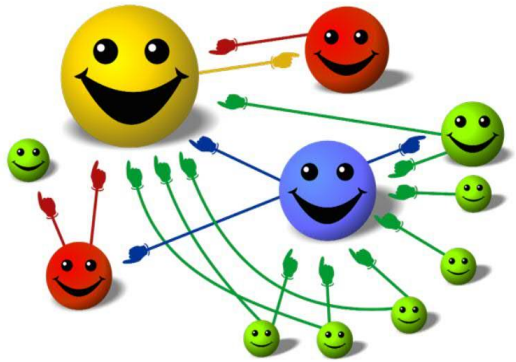
Spark Platform: GraphX



Patrick Wendell, *Big Data Processing*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark Platform: GraphX Example: PageRank

Popular algorithm originally introduced by Google



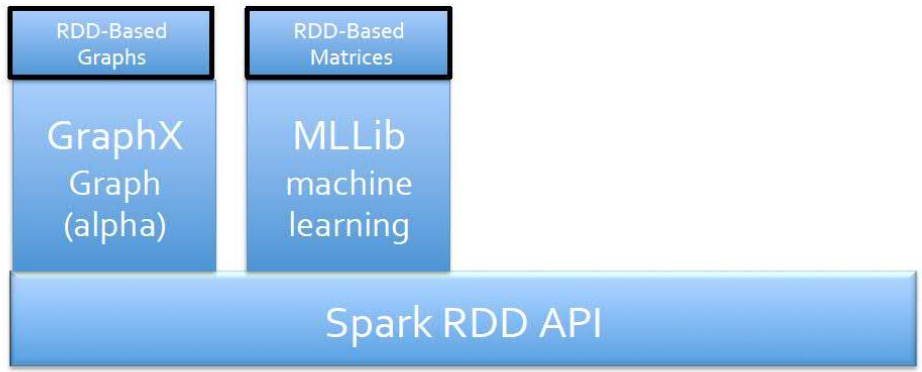
PageRank is implemented using **Pregel** graph processing

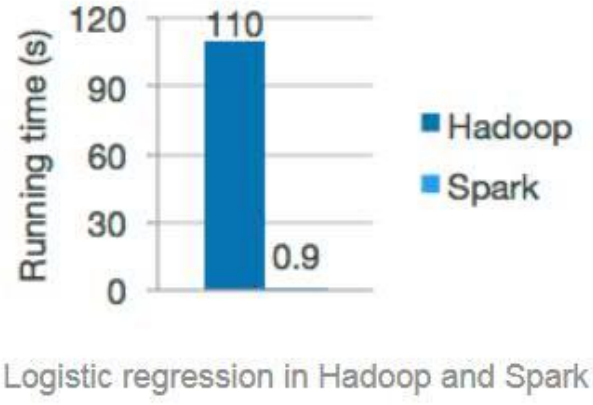
```
// get people with top-k pageranks
def findTopPageRank(allPeople: RDD[String], links: RDD[(String, String, Double)], k: Int) = {
  val versRDD = allPeople.map(p => (uid(p), p))
  val edgesRDD = links.map{ case (l, r, score) => Edge(uid(l), uid(r), score) }

  val g = Graph(versRDD, edgesRDD).cache
  val ranks = g.pageRank(0.001)

  ranks.vertices.top(k)(Ordering.by( . 2)).map(p => (fromUid(p._1), p._2))
}
```

Spark Platform: MLLib

Spark Platform: MLLib

```
model = LogisticRegressionWithSGD.train(trainRDD)
dataRDD.map(point => model.predict(point))
```





Logistic regression in Hadoop and Spark

Patrick Wendell, *Big Data Processing*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark Platform: MLLib
Example: K-Means Clustering

Goal:

Segment tweets into clusters by geolocation using Spark MLLib K-means clustering

```
<longitude>, <latitude>, <timestamp>, <userId>, <tweet message>

-56.544541,-29.089541,1403918487000,1706271294,Por que ni estamos jugando, son más pajeros e
-69.922686,18.462675,1403918487000,2266363318,Aprenda hablar amigo
-118.565107,34.280215,1403918487000,541836358,today a boy told me I'm pretty and he loved me
121.039399,14.72272,1403918487000,362868852,@Kringgelss labuyoo. Hahaha
-34.875339,-7.158832,1403918487000,285758331,@keithmeneses_ oi td bem? sdds 😊💚
103.766123,1.380696,1403918487000,121042839,Xian Lim on iShine 3 2
```

https://chimpler.wordpress.com/2014/07/11/segmenting-audience-with-kmeans-and-voronoi-diagram-using-spark-and-mllib/

Spark Platform: MLLib
Example: K-Means Clustering

To run the k-means algorithm in Spark, we need to first read the csv file

```
val sc = new SparkContext("local[4]", "kmeans")
// Load and parse the data, we only extract the latitude and longitude of each line
val data = sc.textFile(arg)
val parsedData = data.map {
  line =>
    Vectors.dense(line.split(',').slice(0, 2).map(_.toDouble))
}
```

Then we can run the spark kmeans algorithm:

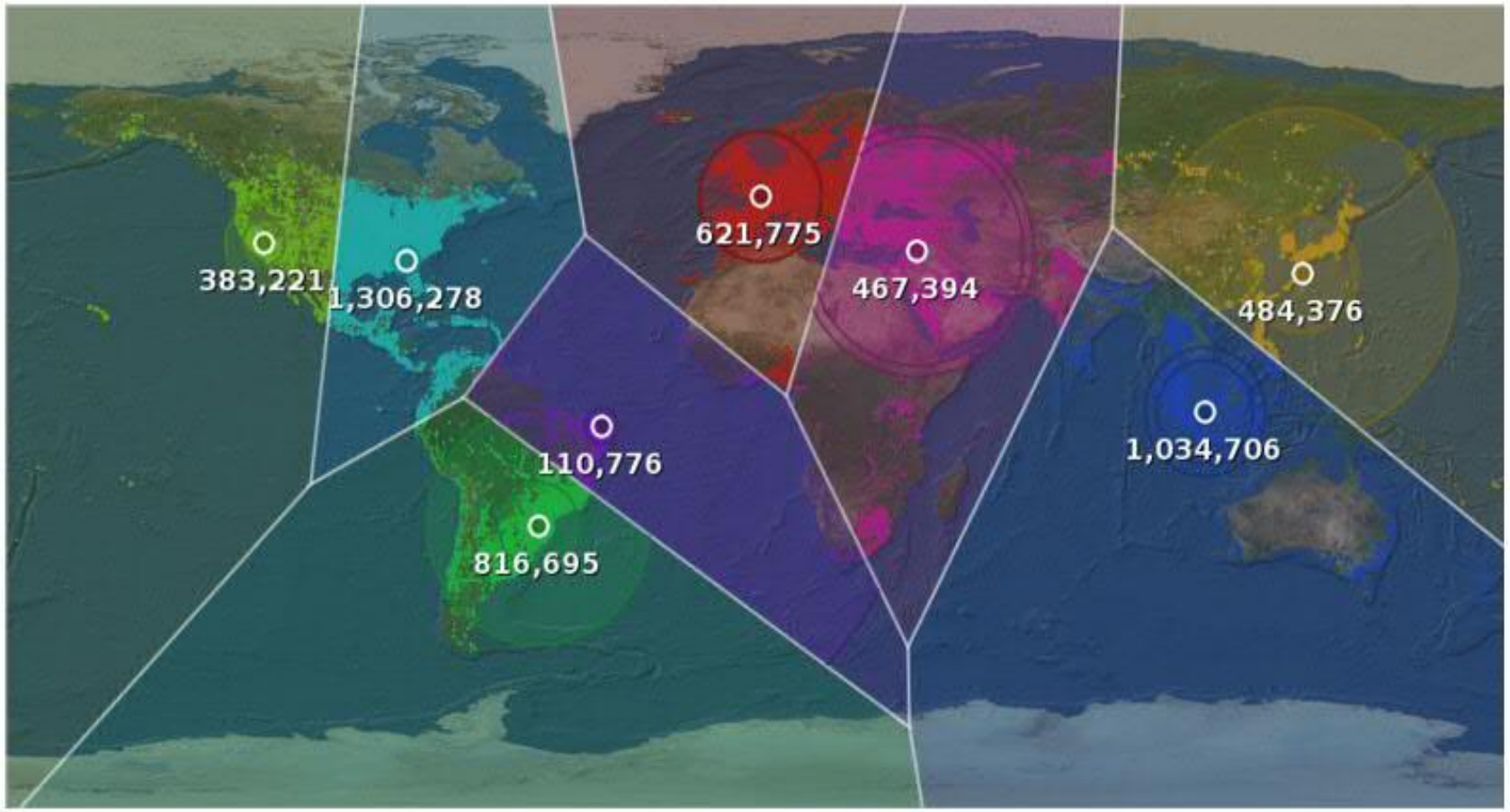
```
val iterationCount = 100
val clusterCount = 10
val model = KMeans.train(parsedData, clusterCount, iterationCount)
```

From the model we can get the cluster centers and group the tweets by cluster:

```
val clusterCenters = model.clusterCenters map (_.toArray)

val cost = model.computeCost(parsedData)
println("Cost: " + cost)

val tweetsByGoup = data
  .map {_.split(',').slice(0, 2).map(_.toDouble)}
  .groupBy{rdd => model.predict(Vectors.dense(rdd))}
  .collect()
sc.stop()
```

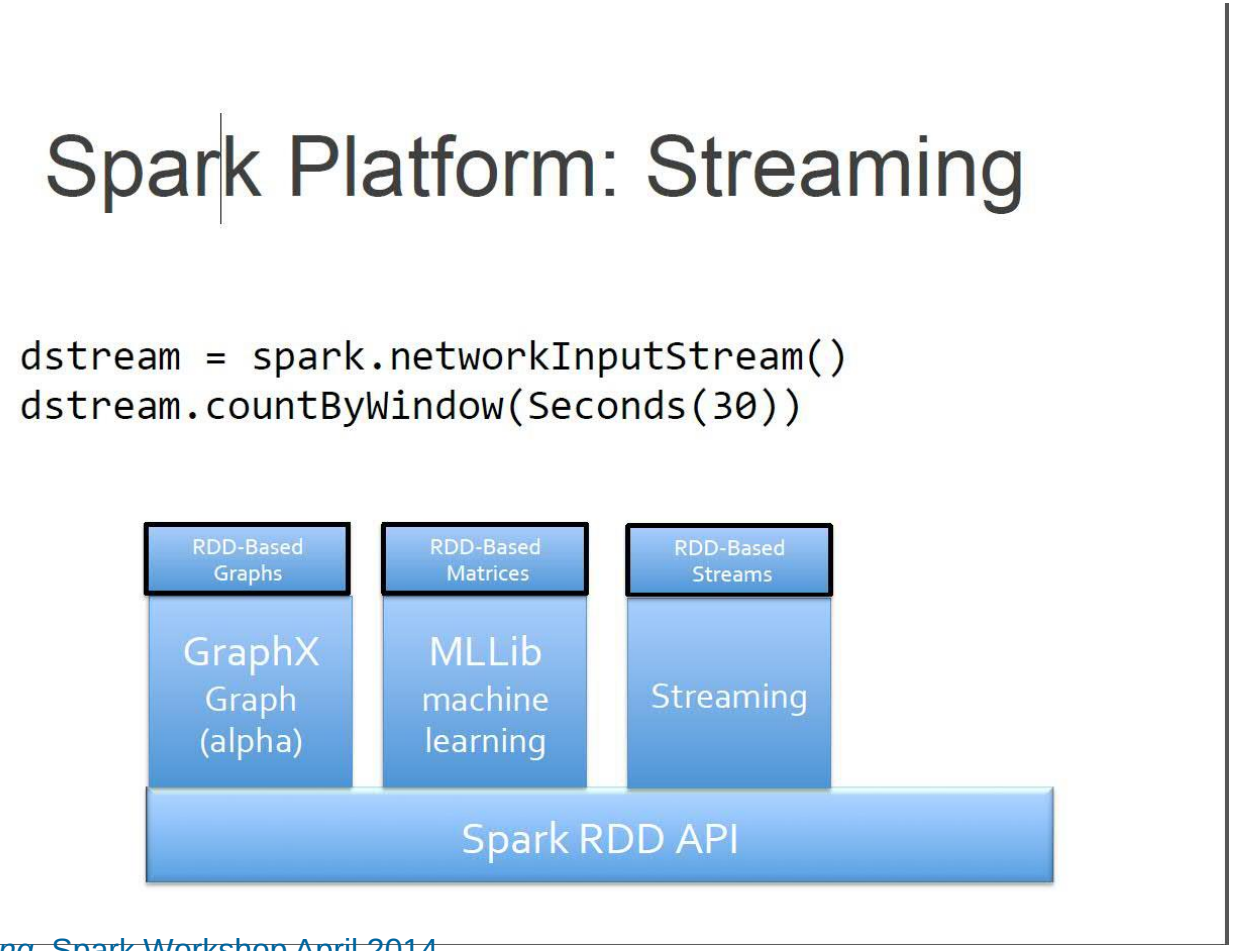
https://chimpler.wordpress.com/2014/07/11/segmenting-audience-with-kmeans-and-voronoi-diagram-using-spark-and-mllib/

Spark Platform: MLLib
Example: K-Means Clustering



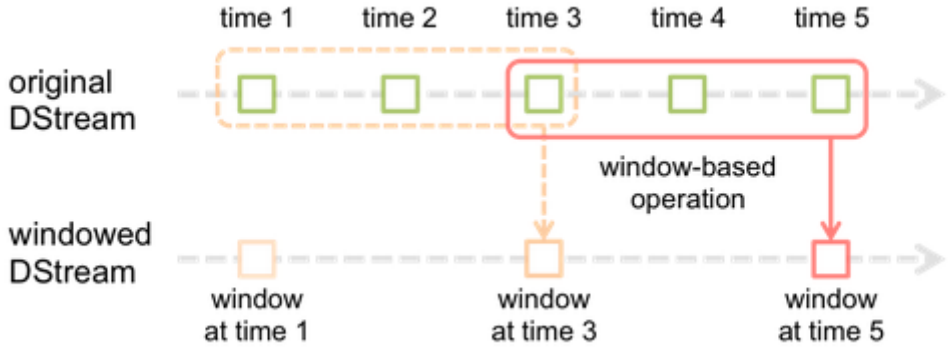
https://chimpler.wordpress.com/2014/07/11/segmenting-audience-with-kmeans-and-voronoi-diagram-using-spark-and-mllib/

Spark Platform: Streaming



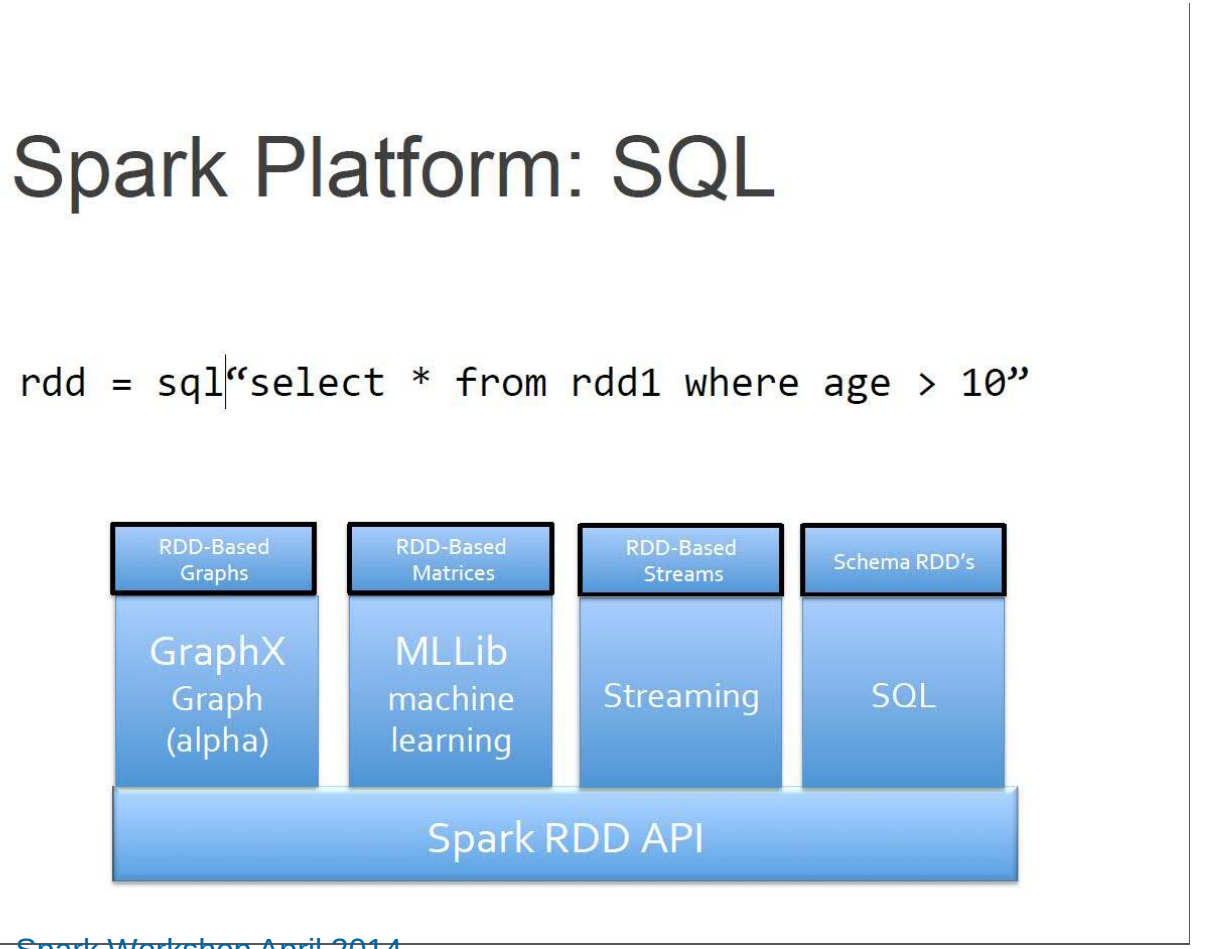
Patrick Wendell, *Big Data Processing*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Spark Platform: Streaming Example




```
// popular topics of last 3 minutes
def latestTopics(docDStream: DStream[String], k: Int) = {
  val topicDStream = docDStream.flatMap(d => getTopics(d).map(t => (t, 1)))
  val topicCountDStream = topicDStream.reduceByKeyAndWindow(
    { (x, y) => x + y },
    { (x, y) => x - y },
    Minutes(3),   // window size
    Minutes(1)    // batch size
  )

  val sortedTopics = topicCountDStream.transform(rdd => rdd.sortBy(_._2, false))

  sortedTopics
}
```

Spark Platform: SQL and DataFrames



Patrick Wendell, *Big Data Processing*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

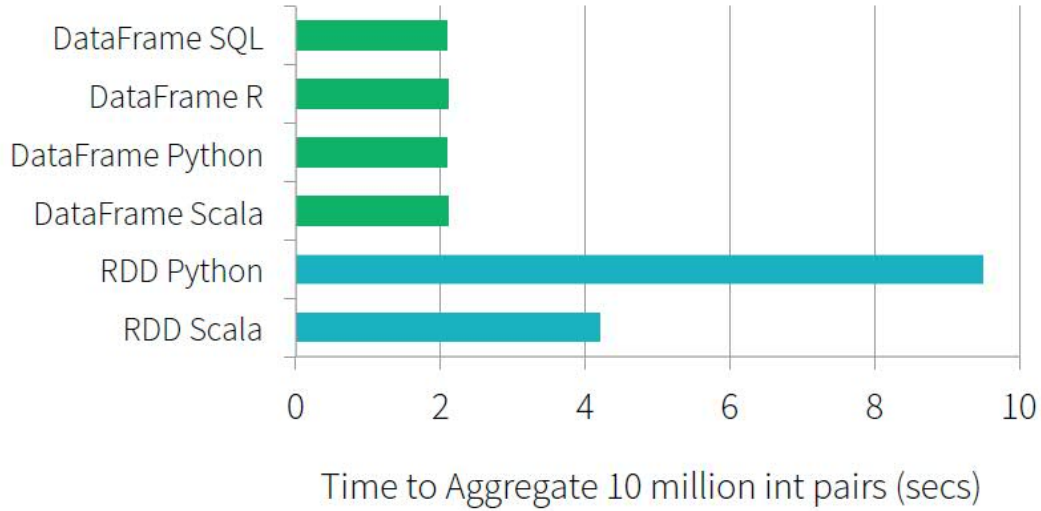
Spark Platform: SQL and DataFrames Example

Using RDDs

```
data = sc.textFile(...).split("\t")
data.map(lambda x: (x[0], [int(x[1]), 1])) \
    .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
    .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
    .collect()
```

Using DataFrames

```
sqlCtx.table("people") \
    .groupBy("name") \
    .agg("name", avg("age")) \
    .collect()
```

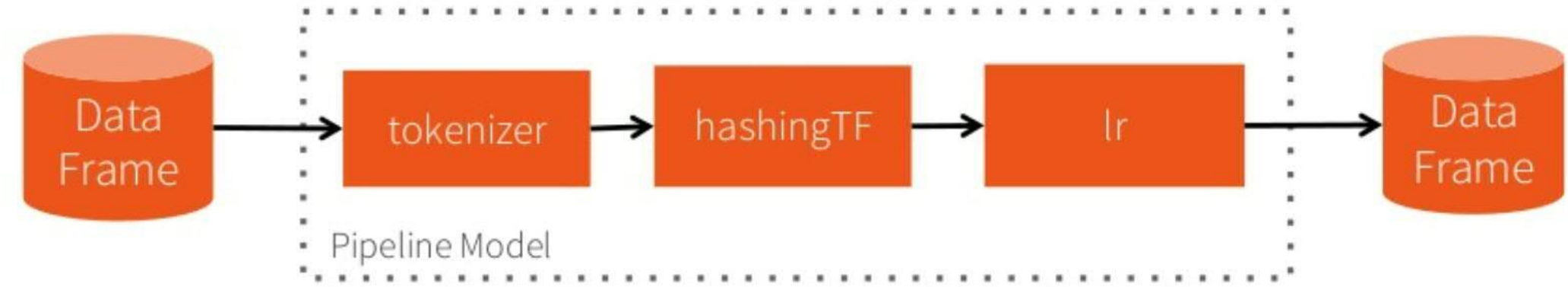


Michael Armbrust, “Spark DataFrames: Simple and Fast Analytics on Structured Data”, Spark Summit 2015

Machine Learning Pipeline with Spark ML

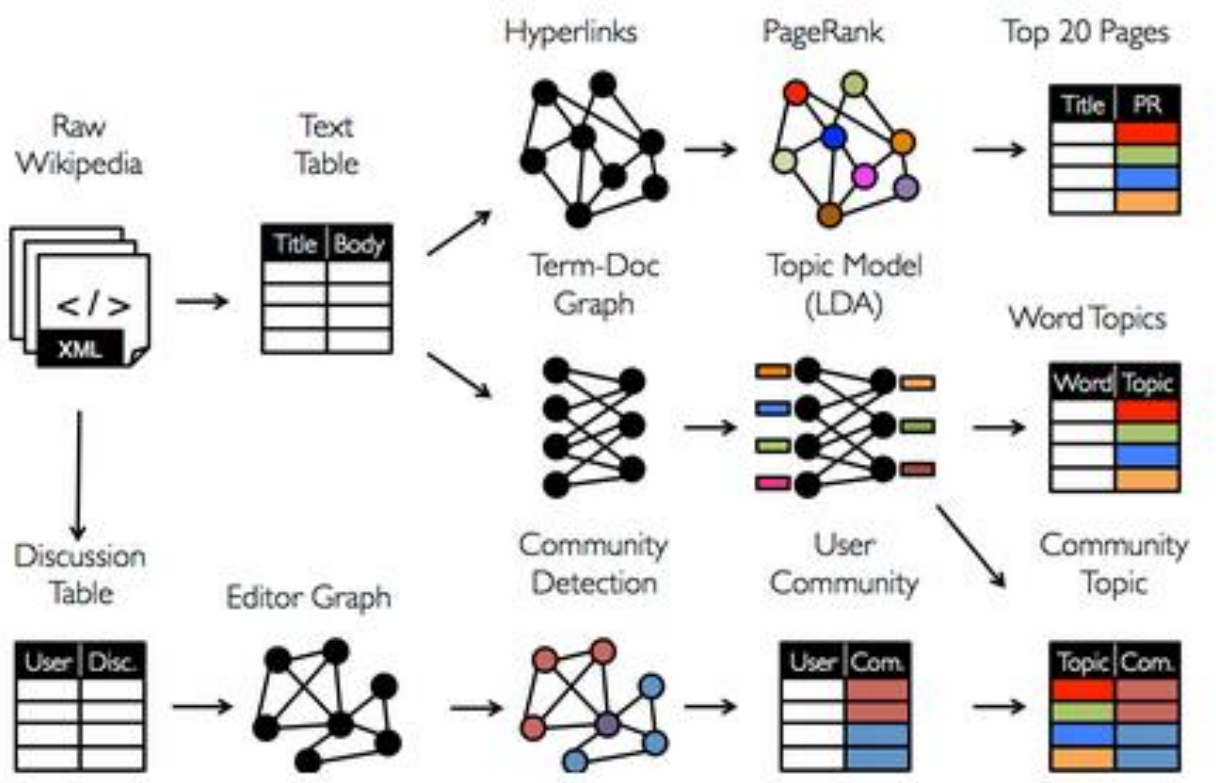
```
// create pipeline
tok = Tokenizer(in="text", out="words”)
tf = HashingTF(in=“words”, out="features”)
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tok, tf, lr])
```

```
// train pipeline
df = sqlCtx.table(”training”)
model = pipeline.fit(df)

// make predictions
df = sqlCtx.read.json("/path/to/test”)
model.transform(df)
  .select(“id”, “text”, “prediction”)
```



Patrick Wendell, Matei Zaharia, “Spark community update”, https://spark-summit.org/2015/events/keynote-1/

Combined Analytics of Data



Analyze tabular data with SQL

Analyze graph data using GraphX graph analytics engine

Use same machine learning Infrastructure

Use same solution for streaming data

Joseph Gonzalez, Reynold Xin, Ankur Dave, Daniel Crankshaw, Michael Franklin, and Ion Stoica, "GRAPHX: UNIFIED GRAPH ANALYTICS ON SPARK", spark summit July 2014