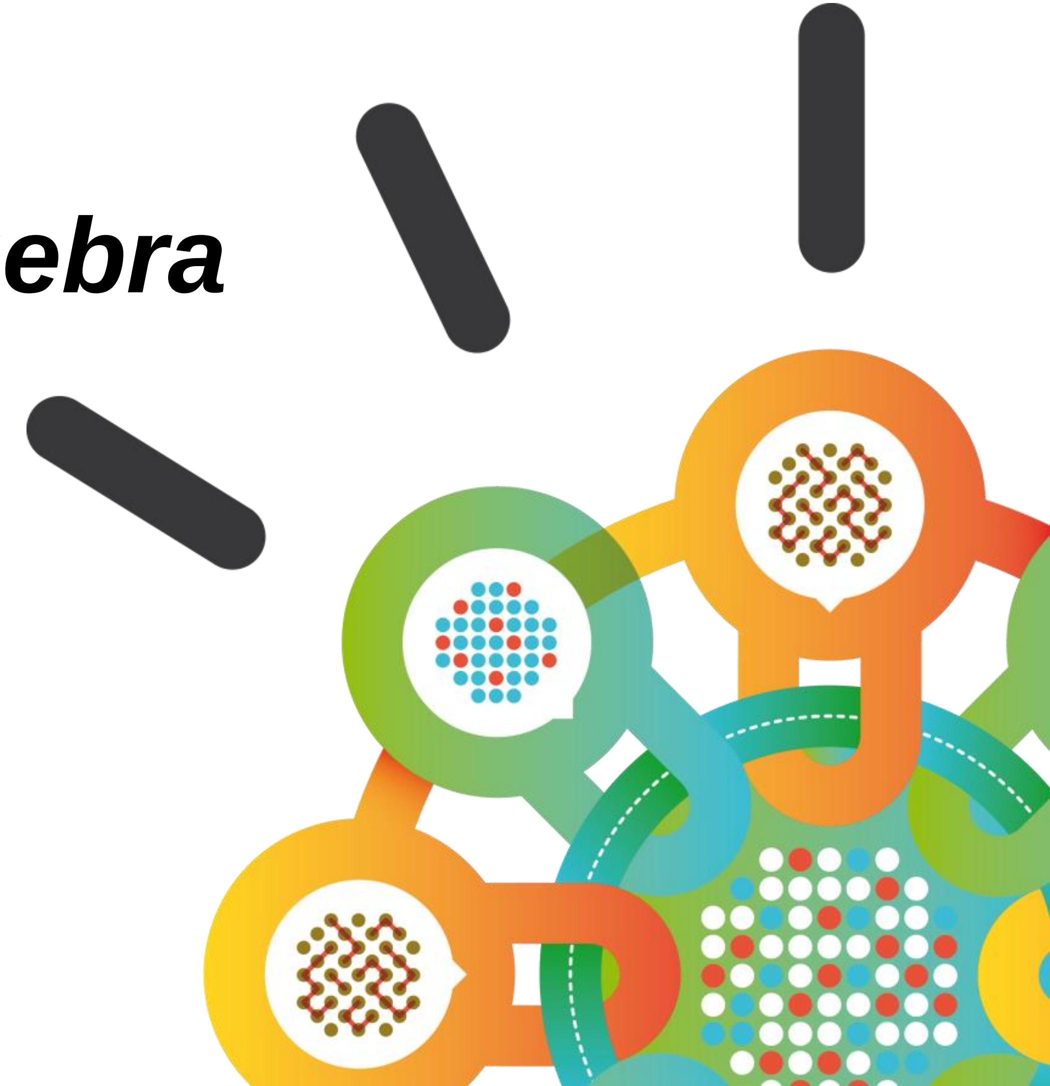
Cloud, Big Data & Linear Algebra

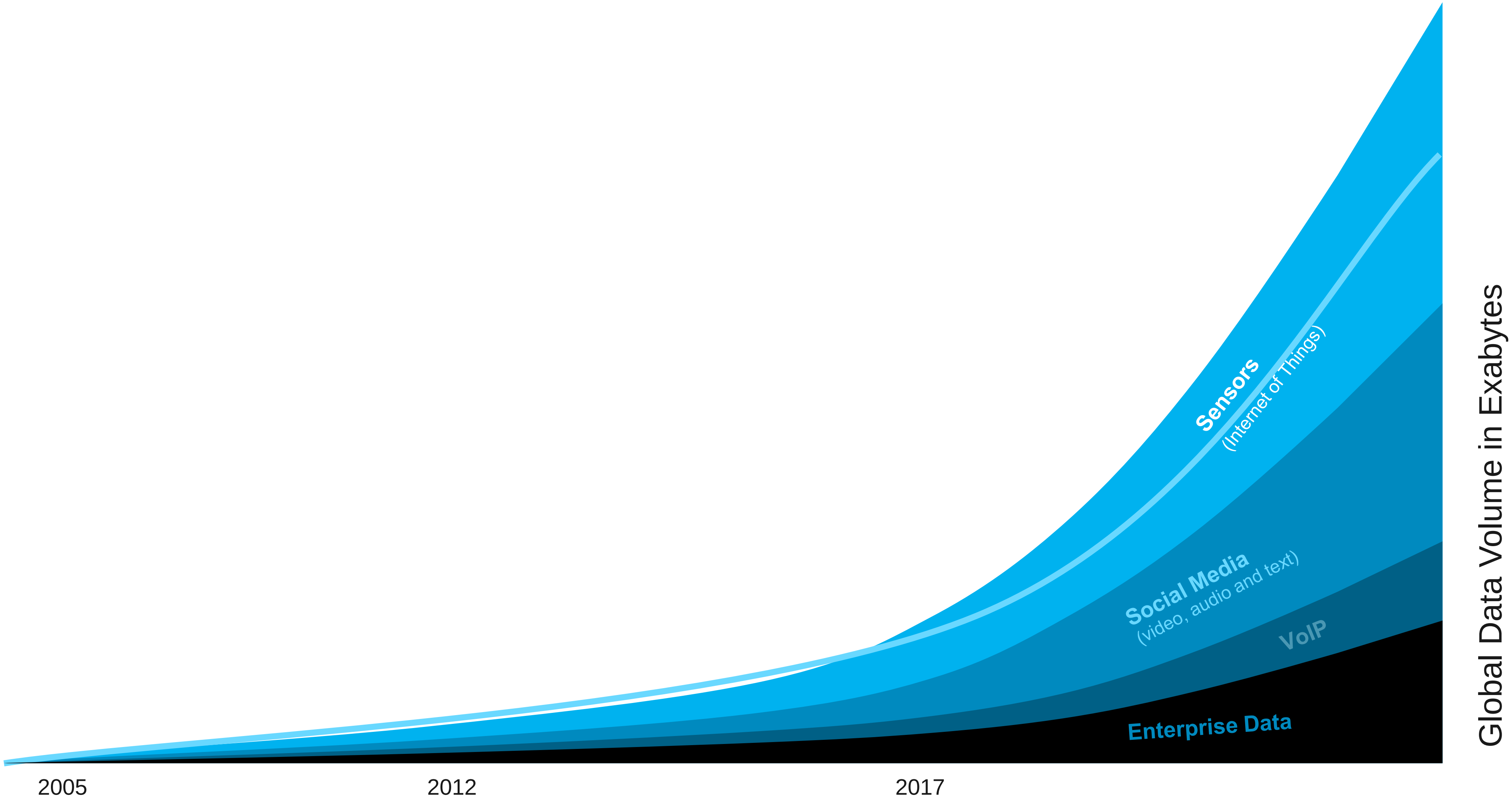
Shelly Garion

IBM Research -- Haifa

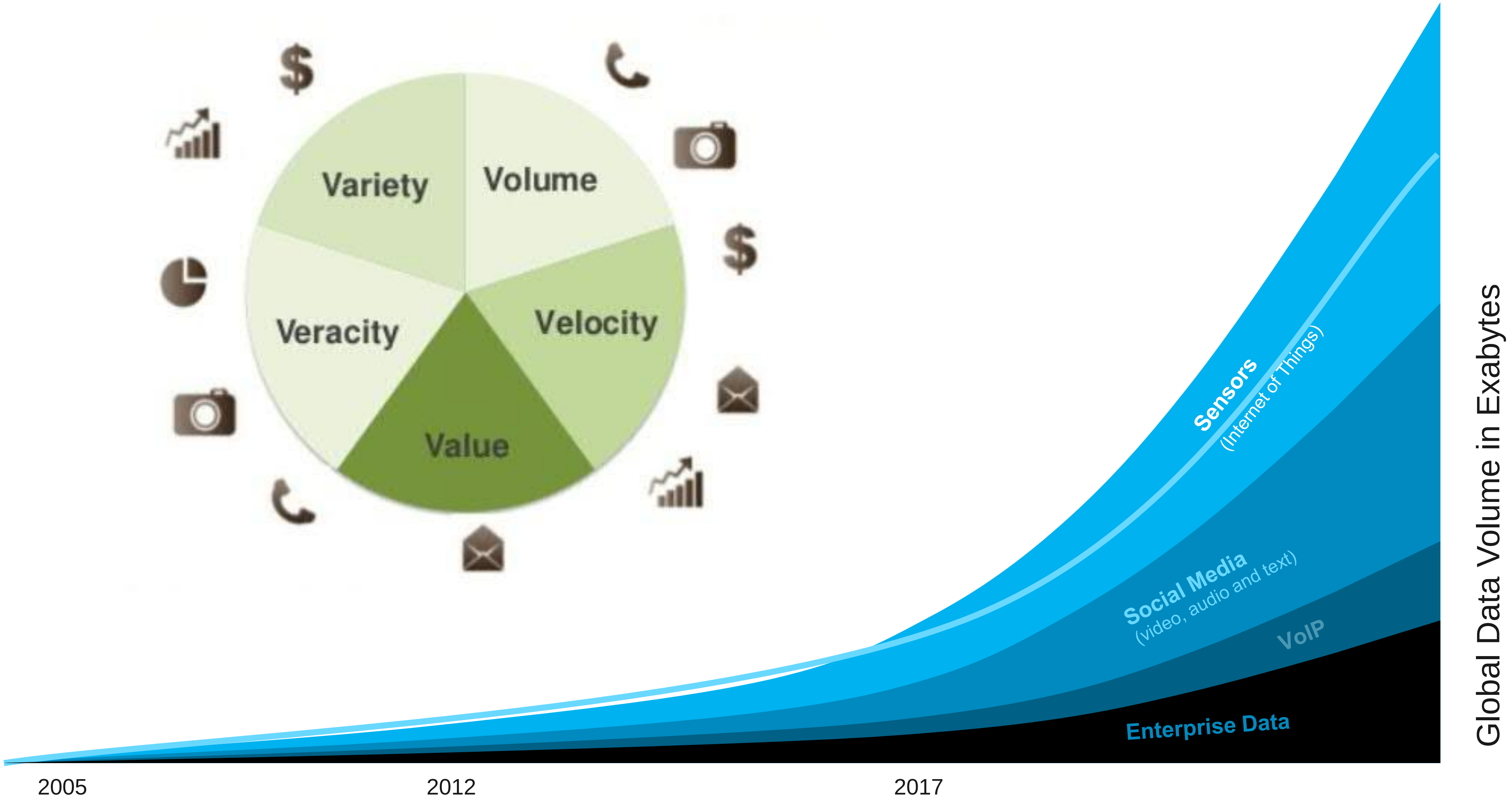


What is Big Data?

What is Big Data?



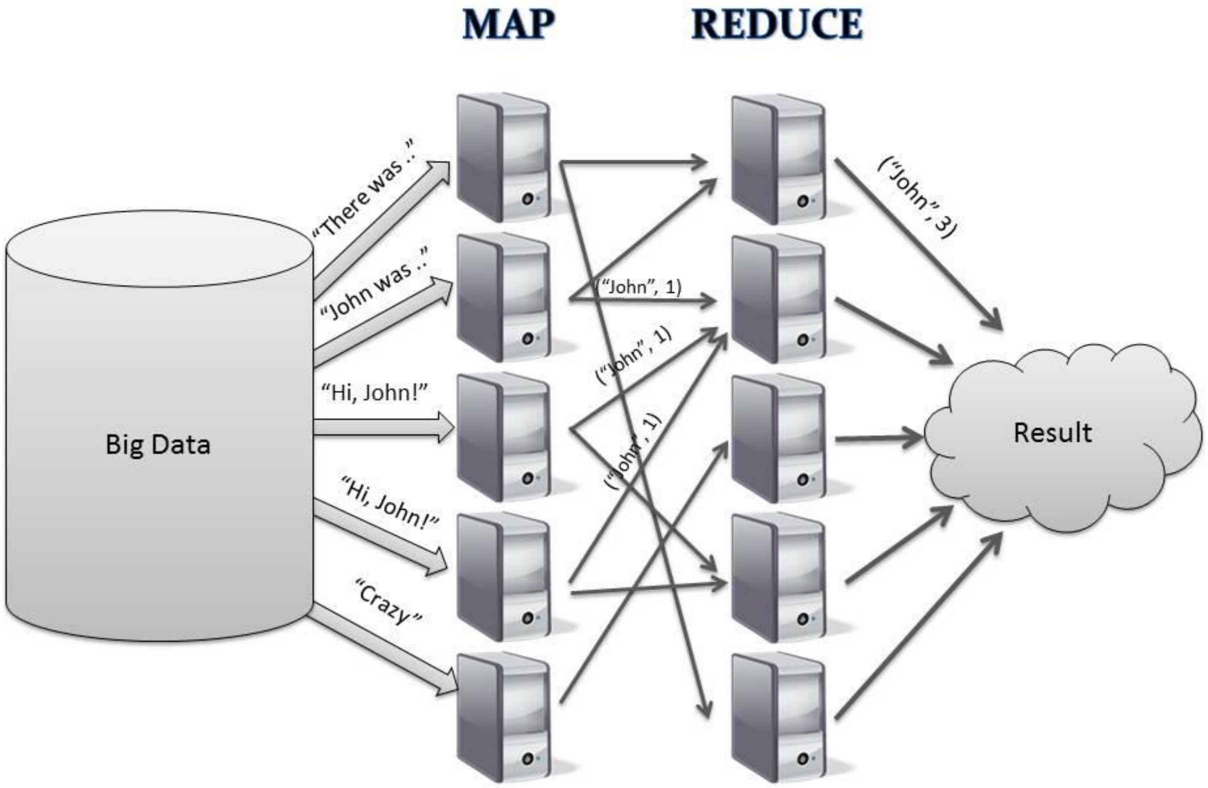
What is Big Data?



Big Data in the Cloud

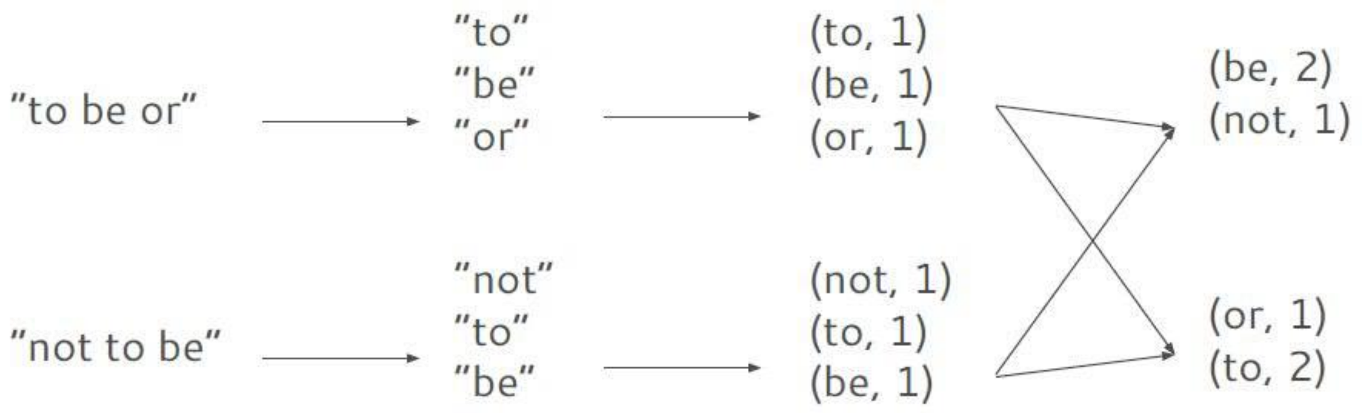
How to Analyze Big Data?

How to Analyze Big Data?



Basic Example: Word Count (Spark & Python)

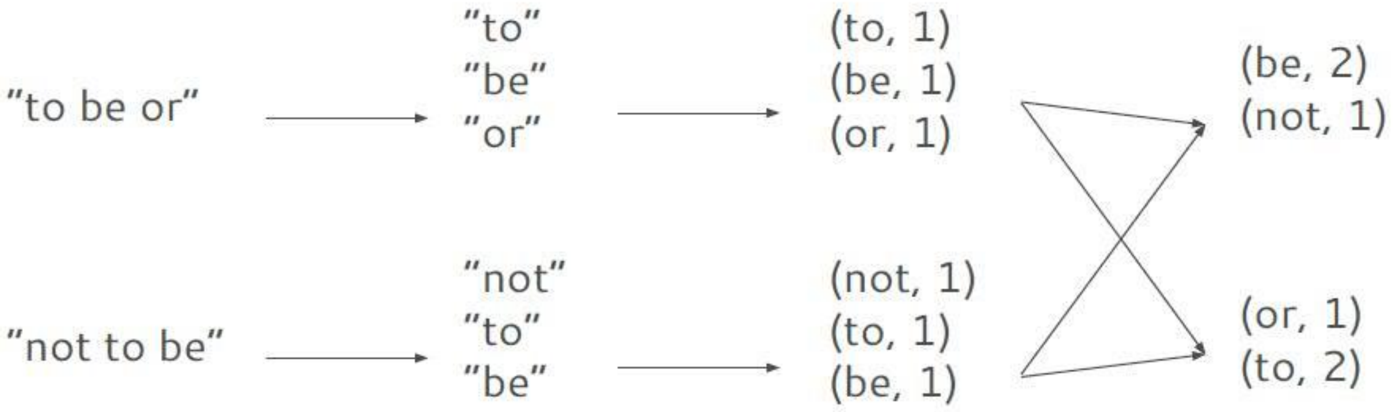
```
>lines = sc.textFile("hamlet.txt")
>counts = lines.flatMap(lambda line: line.split(" "))
                .map(lambda word => (word, 1))
                .reduceByKey(lambda x, y: x + y)
```



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(_ + _)
```



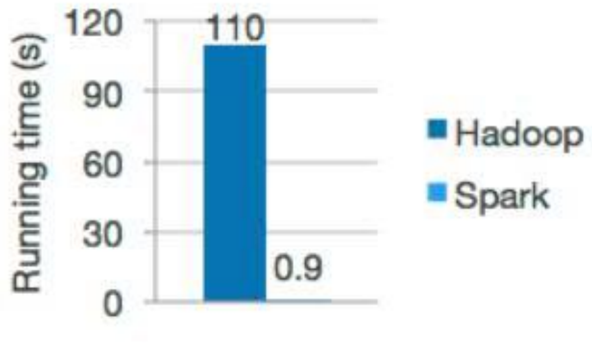
Holden Karau, *Making interactive BigData applications fast and easy*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

Some History…

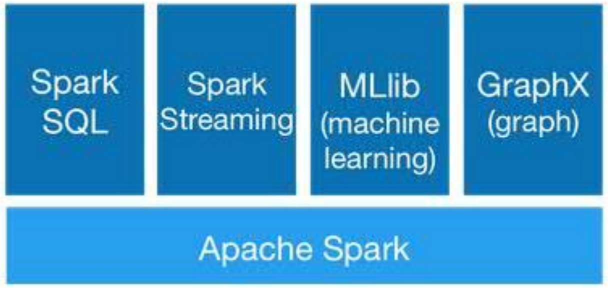
- **Map/Reduce** was invented by **Google:**
  - Inspired by functional programming languages map and reduce functions
  - Seminal paper: Jeffrey Dean and Sanjay Ghemawat (OSDI 2004), *"MapReduce: Simplified Data Processing on Large Clusters"*
  - Used at Google to completely regenerate Google's index of the World Wide Web
- **Hadoop** – open source implementation matches Google's specifications
- **Amazon EMR** (Elastic MapReduce) running on Amazon EC2
- **Spark** started in 2009 as a research project of UC Berkley
- **Spark** is now 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 created Spark Technology Center (STC) - http://www.spark.tc/

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 Map/Reduce in memory, or 10x faster on disk
- **Ease of use:** Write applications quickly in Java, Scala, Python and R, also with notebooks
- **Generality:** Combine streaming, SQL and complex analytics – machine learning, graph processing
- **Runs everywhere:** 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.

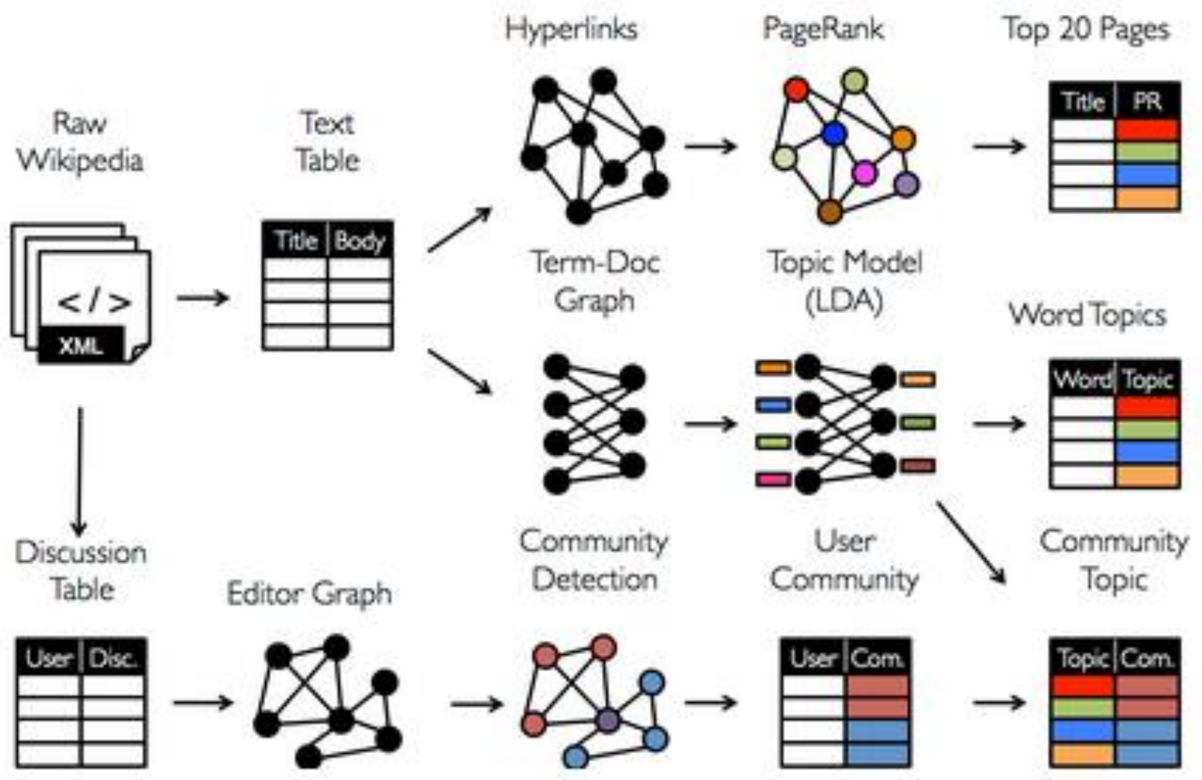


Logistic regression in Hadoop and Spark



https://spark.apache.org/

Combined Analytics of Data with Spark



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

Spark Example

Goal:

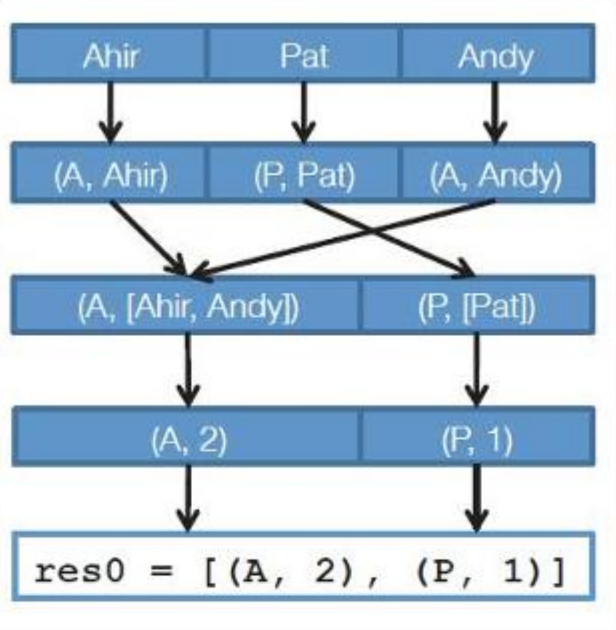
Find number of distinct names per "first letter".

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

Spark Example

Goal:

Find number of distinct names per "first letter".

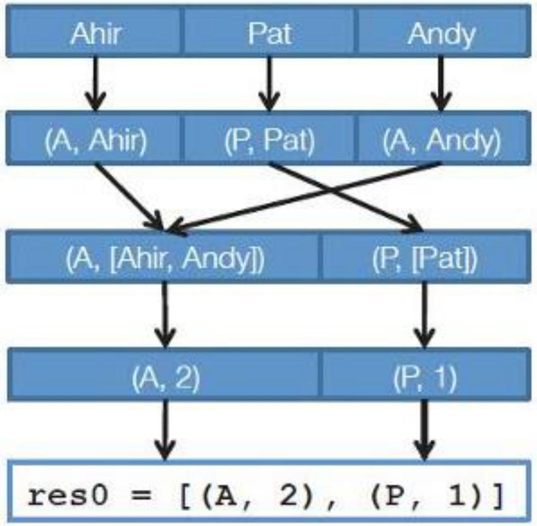


Aaron Davidson, *A deeper understanding of Spark internals*, Spark Summit July 2014, https://spark-summit.org/2014/

Spark Example

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

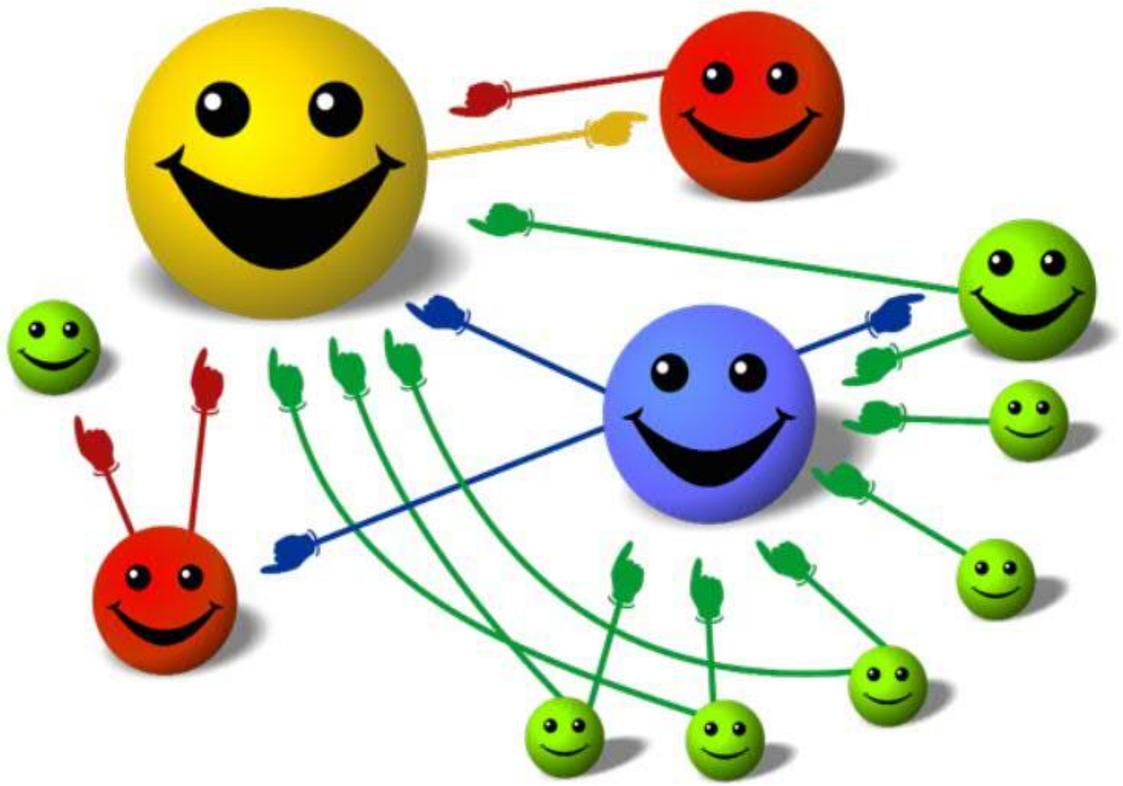
```
sc.textFile("hdfs:/names")
   .map(name => (name.charAt(0), name))
   .groupByKey()
   .mapValues(names => names.toSet.size)
   .collect()
```



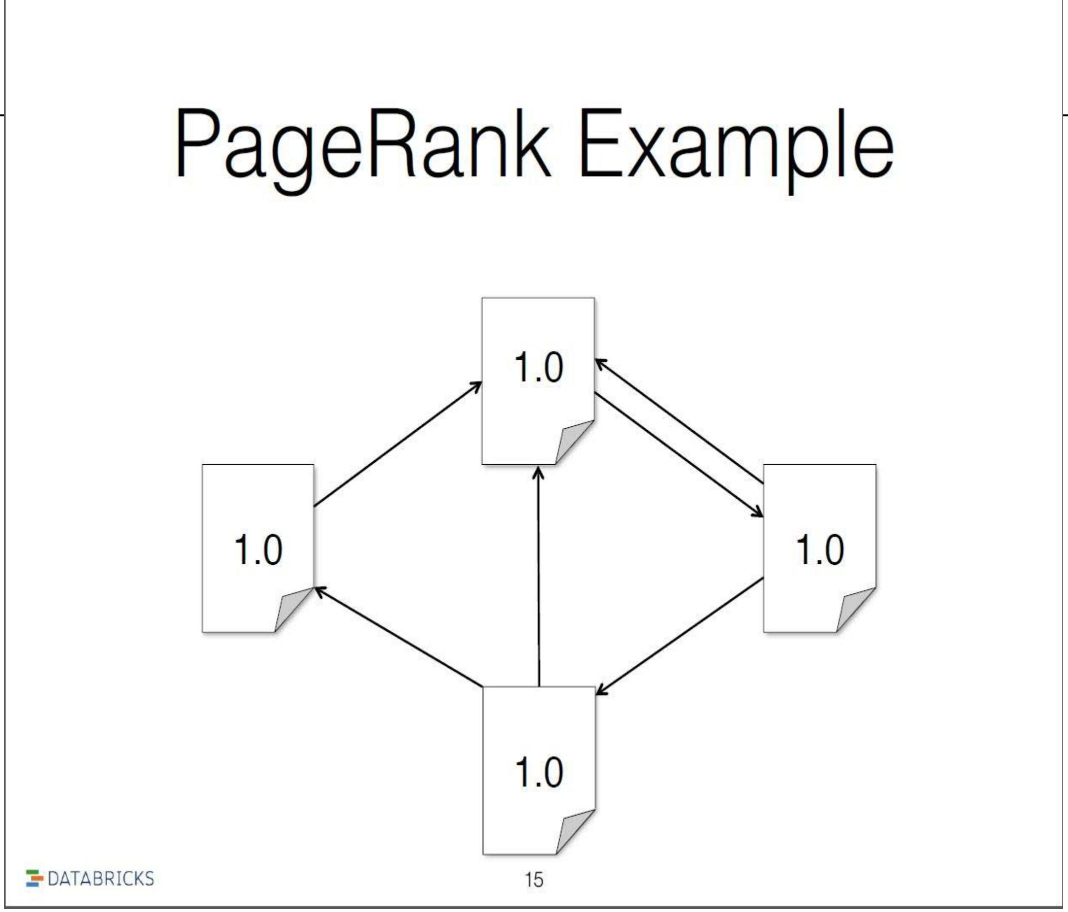
Aaron Davidson, *A deeper understanding of Spark internals*, Spark Summit July 2014, https://spark-summit.org/2014/

PageRank

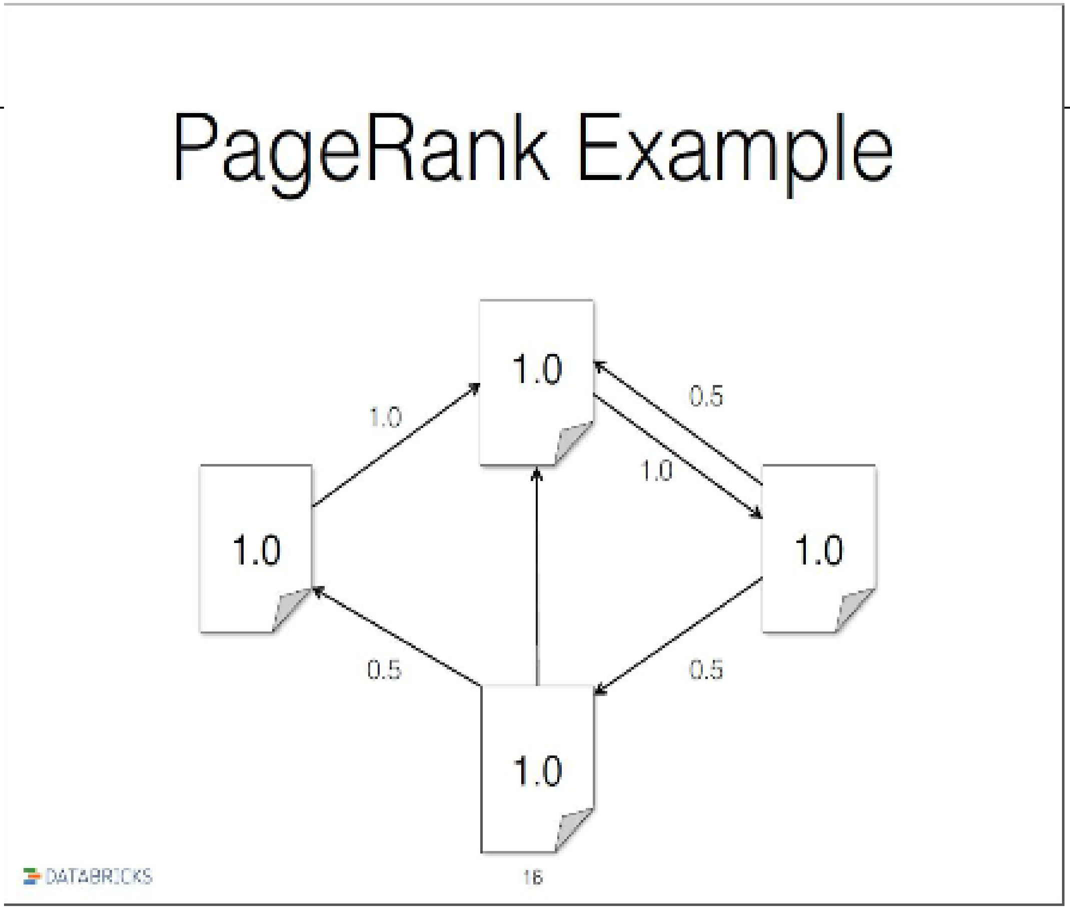
Popular algorithm originally introduced by Google



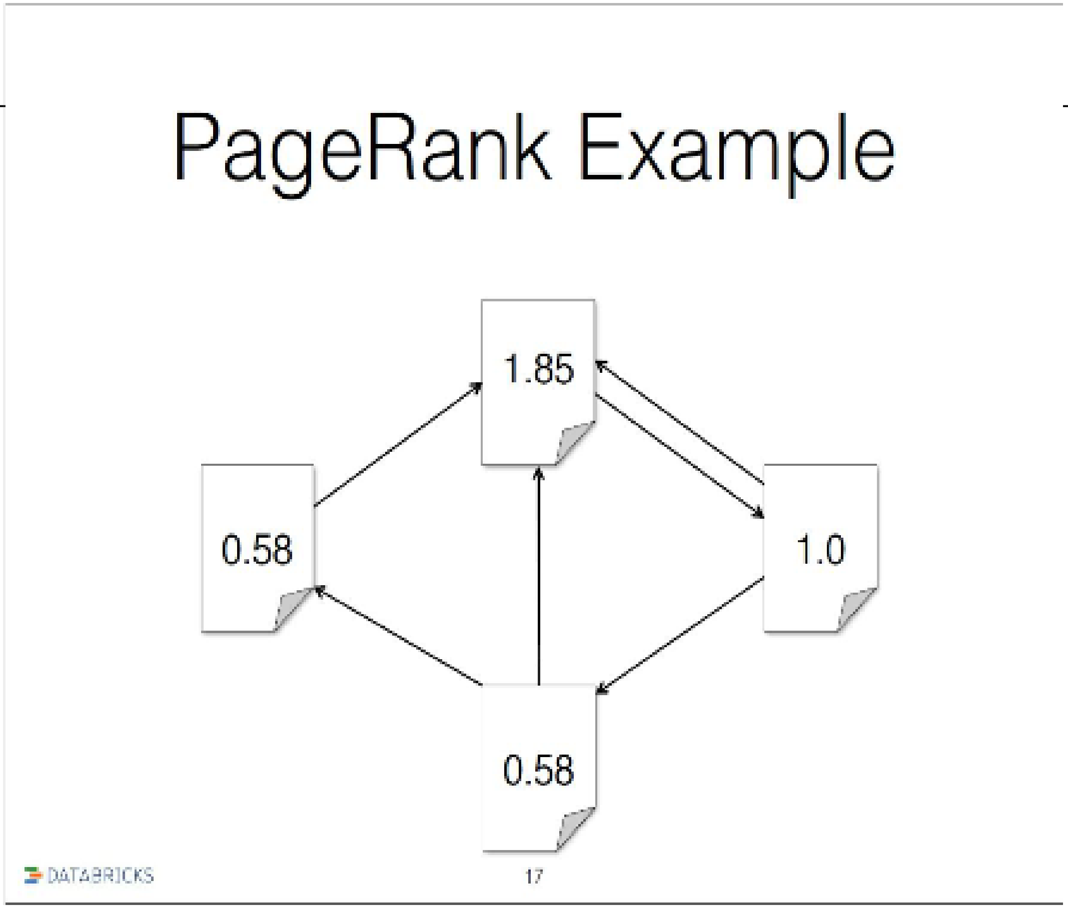
Sergei Brin and Lawrence Page, *"The anatomy of a large-scale hypertextual Web search engine“*, Computer Networks and ISDN Systems. (1998) 30: 107–117.


PageRank Example
1.0
1.0
1.0
1.0
DATABRICKS
15

```
    [0 0.5 1 0.5]
A = [1 0   0   0 ]
    [0 0   0 0.5]
    [0 0.5 0   0 ]

    [1]
V = [1]
    [1]
    [1]

B = 0.85*A
U = 0.15*V


B*V+U = ?
```

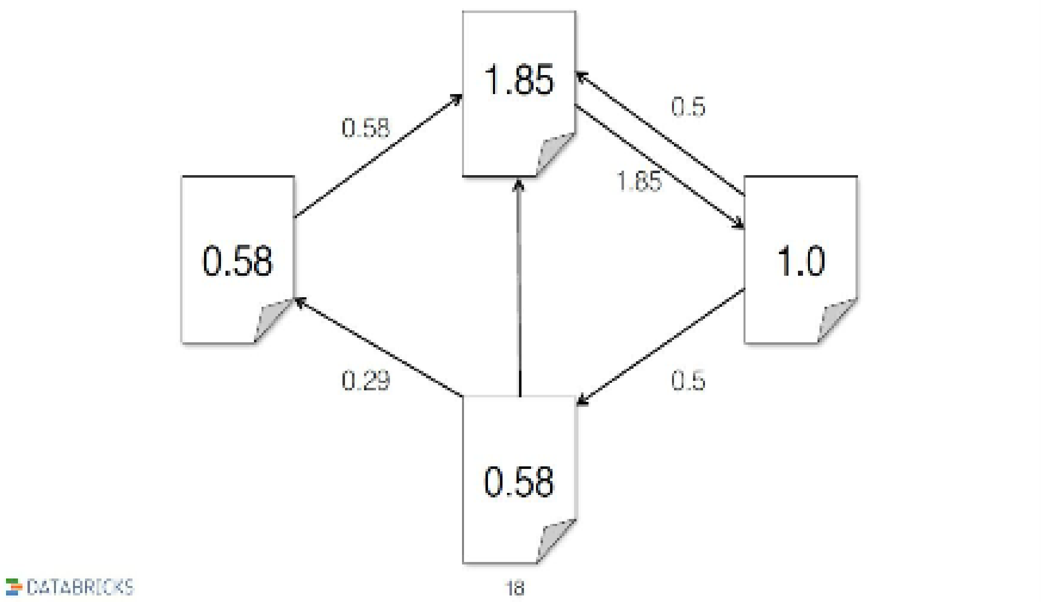


```
    [0 0.5 1 0.5]
A = [1 0   0  0 ]
    [0 0   0 0.5]
    [0 0.5 0  0 ]

    [1]
V = [1]
    [1]
    [1]

B = 0.85*A
U = 0.15*V

          [1.85 ]
B*V+U =   [1.0  ]
          [0.575]
          [0.575]
```

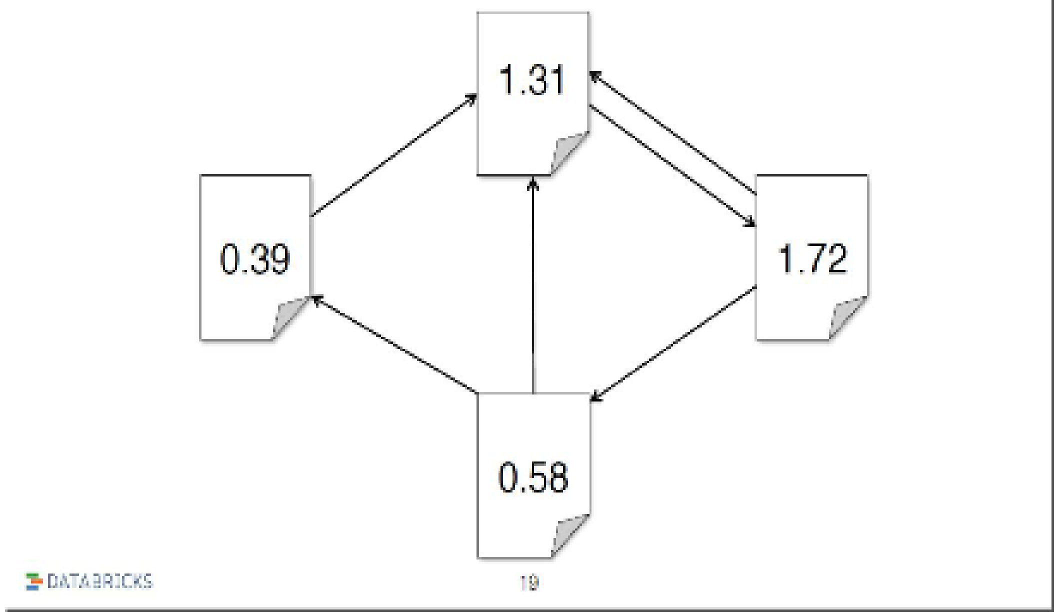


PageRank Example

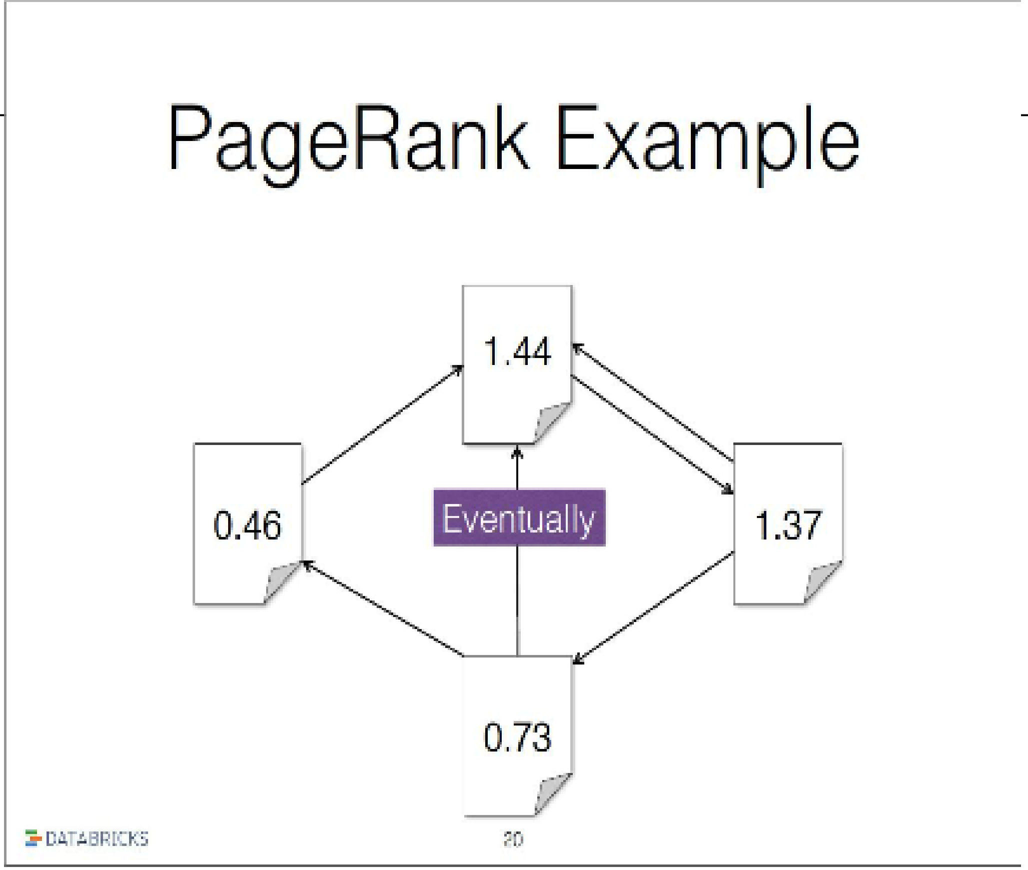
```
        [0  0.5 1  0.5]
A  =    [1  0    0   0 ]
        [0  0    0  0.5]
        [0  0.5  0   0 ]

        [1]
V  =    [1]
        [1]
        [1]

B  =  0.85*A
U  =  0.15*V


B*(B*V+U)+U  =  ?
```



PageRank Example




```
     [0  0.5 1  0.5]
A =  [1  0    0   0 ]
     [0  0    0  0.5]
     [0  0.5  0   0 ]

     [1]
V =  [1]
     [1]
     [1]

B = 0.85*A
U = 0.15*V

                  [1.31]
B*(B*V+U)+U =     [1.72]
                  [0.39]
                  [0.58]

B*(B*(B*V+U)+U)+U = ...
```




```
    [0  0.5 1  0.5]
A = [1  0    0   0 ]
    [0  0    0  0.5]
    [0  0.5  0   0 ]

    [1]
V = [1]
    [1]
    [1]

B = 0.85*A
U = 0.15*V
```

At the k-th step: $B^k v+(B^{k-1}+B^{k-2}+\ldots+B^2+B+I)U=$
$B^k v+(I-B^k)(I-B)^{-1}U$

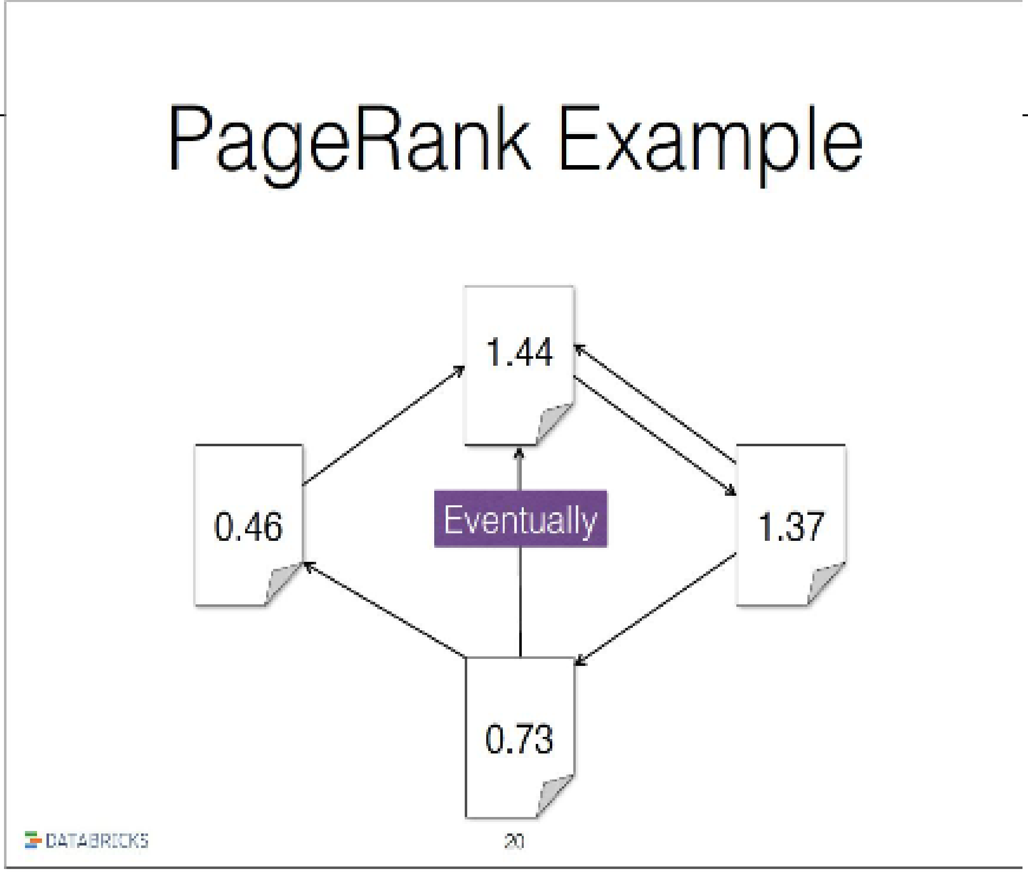
For k=10:

```
[1.43]
[1.37]
[0.46]
[0.73]
```

```
        [0  0.5  1  0.5]
A  =    [1  0    0    0 ]
        [0  0    0  0.5]
        [0  0.5  0    0 ]

        [1]
V  =    [1]
        [1]
        [1]

B  =  0.85*A
U  =  0.15*V
```

Where k goes to infinity:

```
                    [1.44]
(I-B)^(-1)*U  =     [1.37]
                    [0.46]
                    [0.73]
```



$$B^k v \longrightarrow 0$$

$$B^k v + (I-B^k)(I-B)^{-1}U \longrightarrow (I-B)^{-1}U$$

$$A = \begin{bmatrix} 0 & 0.5 & 1 & 0.5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0.5 & 0 & 0 \end{bmatrix} \qquad B=0.85*A$$

- Characteristic polynomial of A:

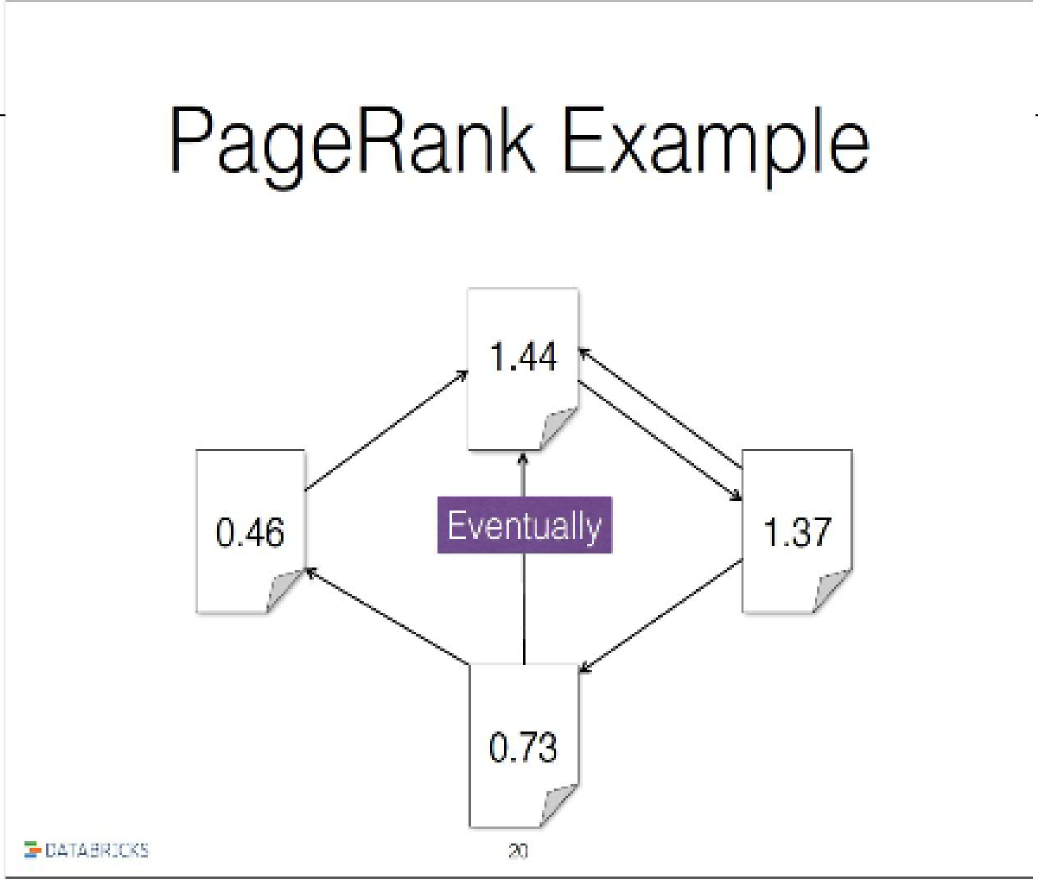
  $$x^4 - 0.5x^2 - 0.25x - 0.25$$

- A is a stochastic matrix,
- 1 is the largest eigen value of A (in its absolute value),
- 1 corresponds to the eigen vector:

$$E = \begin{bmatrix} 1.0 \\ 1.0 \\ 0.25 \\ 0.5 \end{bmatrix}$$

Where k goes to infinity: $A^k v \longrightarrow cE$

$B^k v \longrightarrow 0$



PageRank

PageRank Algorithm

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

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

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

Sergei Brin and Lawrence Page, "The anatomy of a large-scale hypertextual Web search engine", Computer Networks and ISDN Systems. (1998) 30: 107–117.

PageRank

- Rank of each page is the probability of landing on that page for a random surfer on the web
- Probability of visiting all pages after k steps is

$$V_k = A^k \times V^t$$

V: the initial rank vector

A: the link structure (sparse matrix)

- Each page is identified by its unique URL rather than an index
- Ranks vectors (**V**): RDD[(URL, Double)]
- Links matrix (**A**): RDD[(URL, List(URL))]

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

PageRank in Spark

```
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(...)
```

```
// Load the edges as a graph
val graph = GraphLoader.edgeListFile(sc, "graphx/data/followers.txt")
// Run PageRank
val ranks = graph.pageRank(0.0001).vertices
```

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

Machine Learning: K-Means Clustering

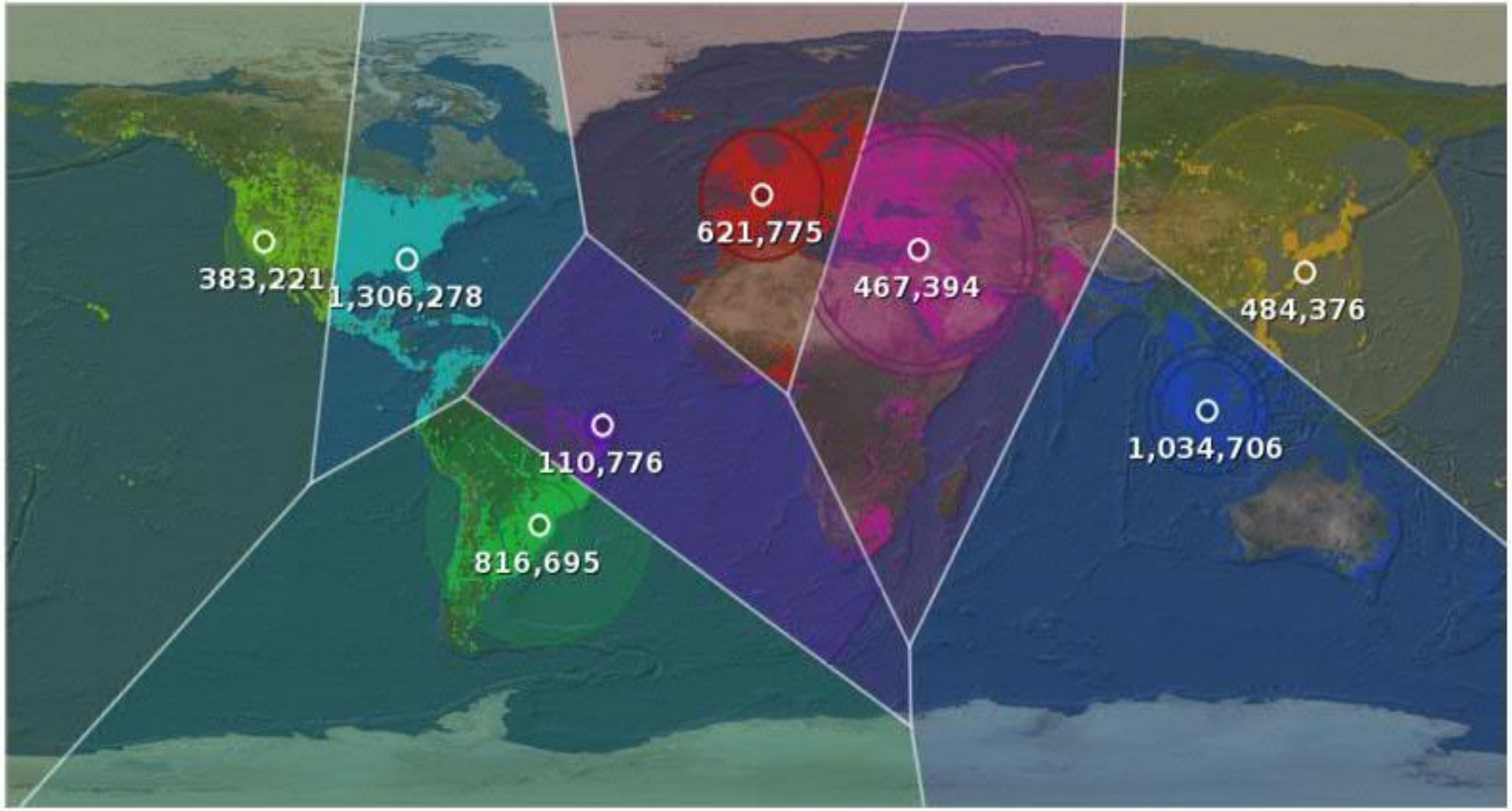
Goal:

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/

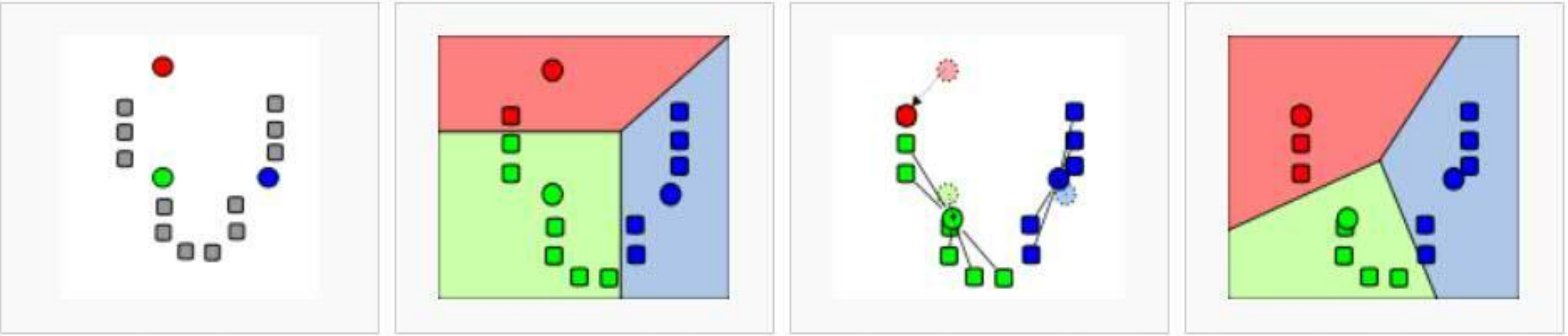
Machine Learning: K-Means Clustering



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

Machine Learning: K-Means Clustering

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

(from Wikipedia)

K-Means Clustering with Spark MLLib

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:

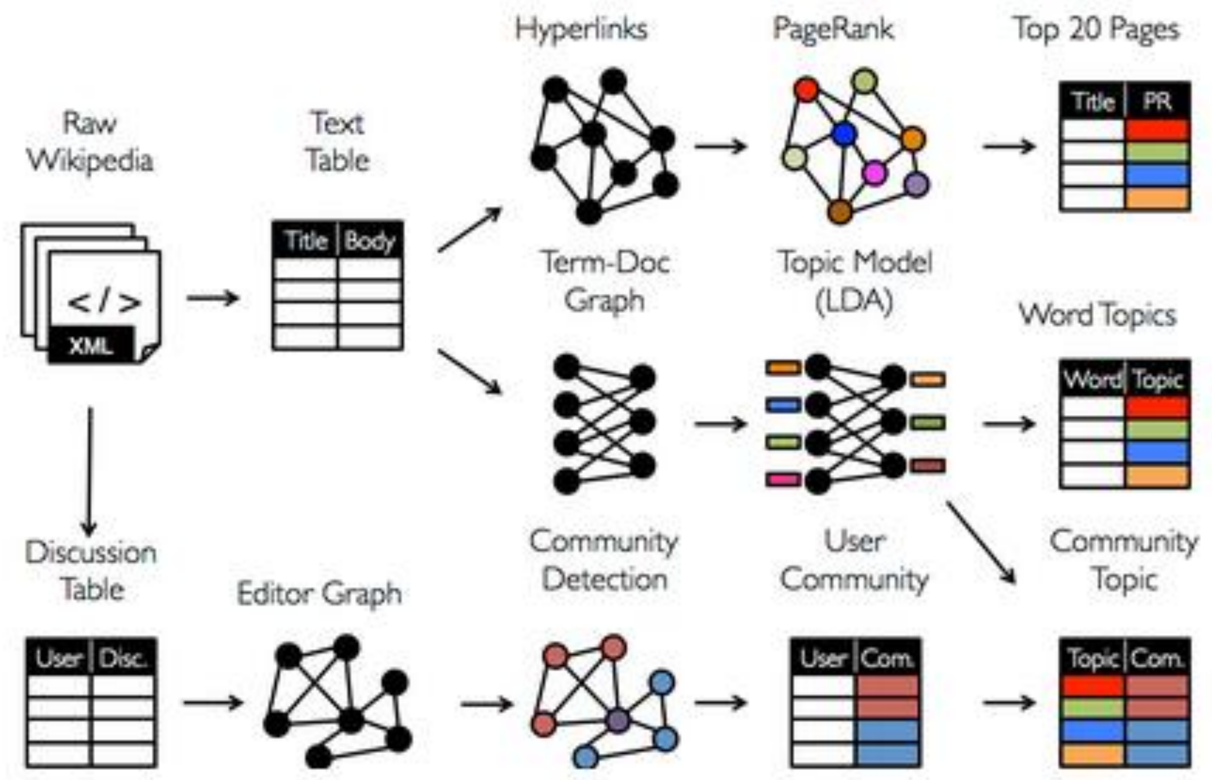
```
val iterationCount = 100
val clusterCount = 10
val model = KMeans.train(parsedData, clusterCount, iterationCount)
```

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/

Combined Analytics of Data with Spark



Analyze tabular data with SQL

Analyze graph data using GraphX graph analytics engine

Use same machine learning Infrastructure

Use same solution for streaming data

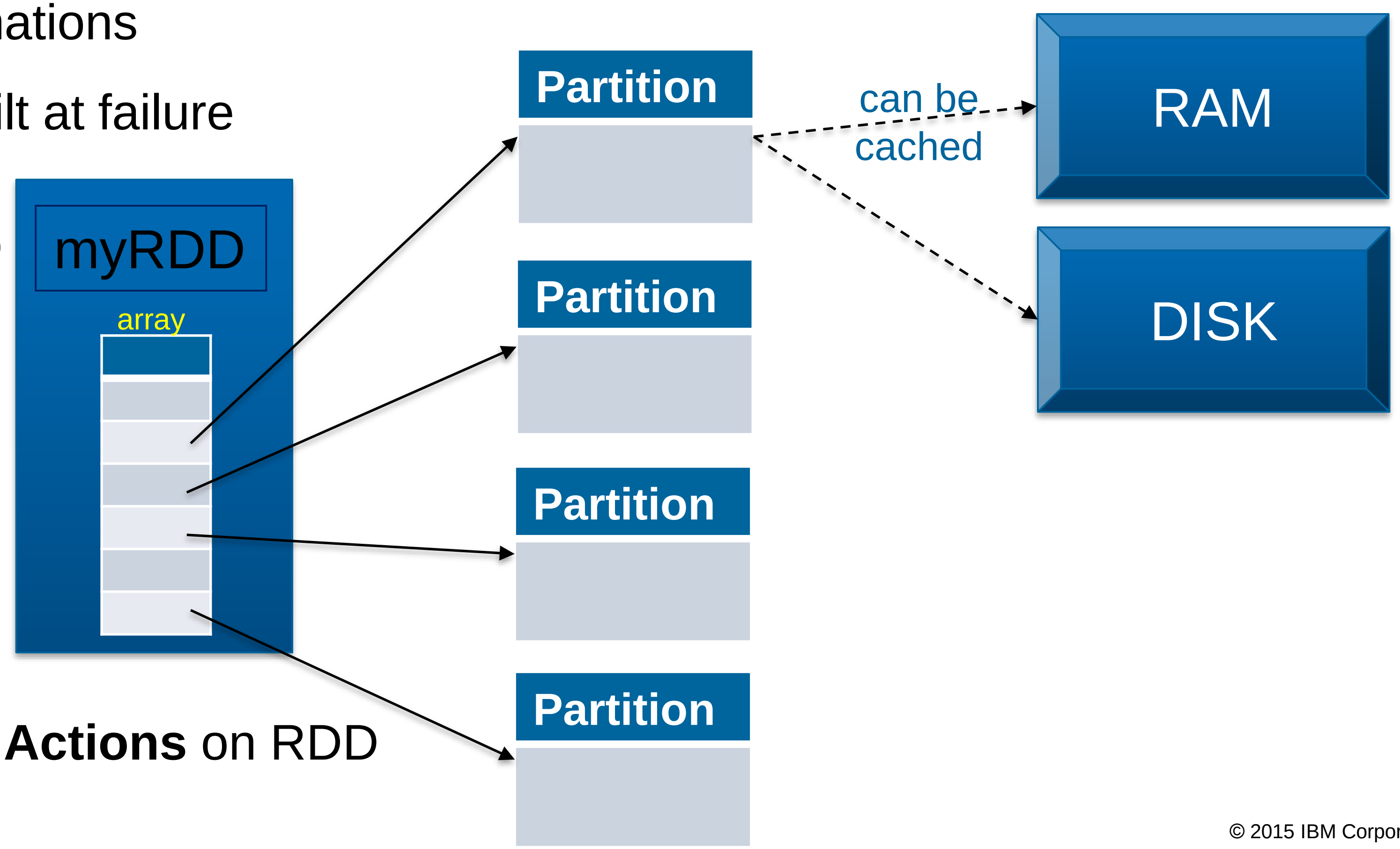
Joseph Gonzalez, Reynold Xin, Ankur Dave, Daniel Crankshaw, Michael Franklin, and Ion Stoica, *"GRAPHX: UNIFIED GRAPH ANALYTICS ON SPARK"*, spark summit July 2014

How Does Spark Work?

Spark RDD (Resilient Distributed Dataset)

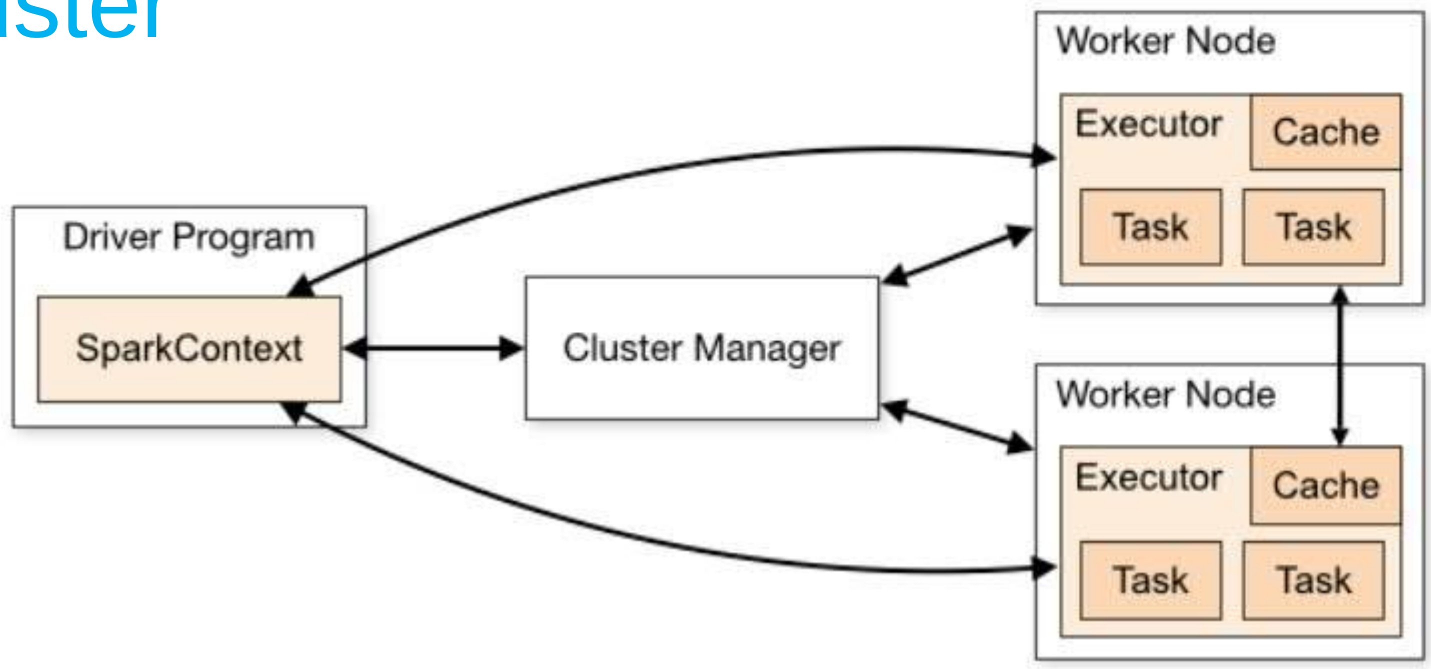
- Immutable, partitioned collections of objects spread across a cluster, stored in RAM or on Disk
- Built through lazy parallel transformations
- **Fault tolerance** – automatically built at failure

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



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

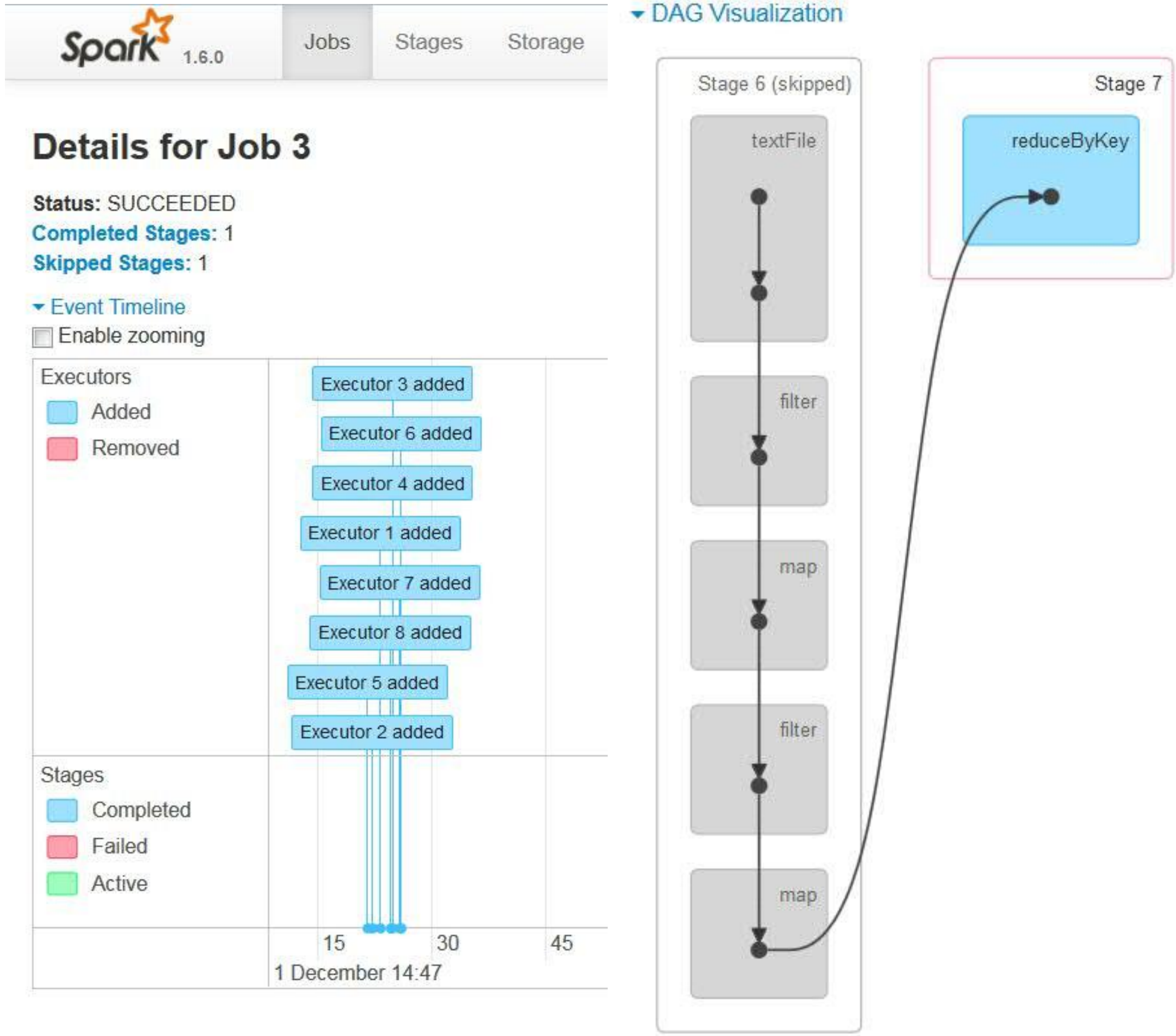
Spark Cluster



- **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	