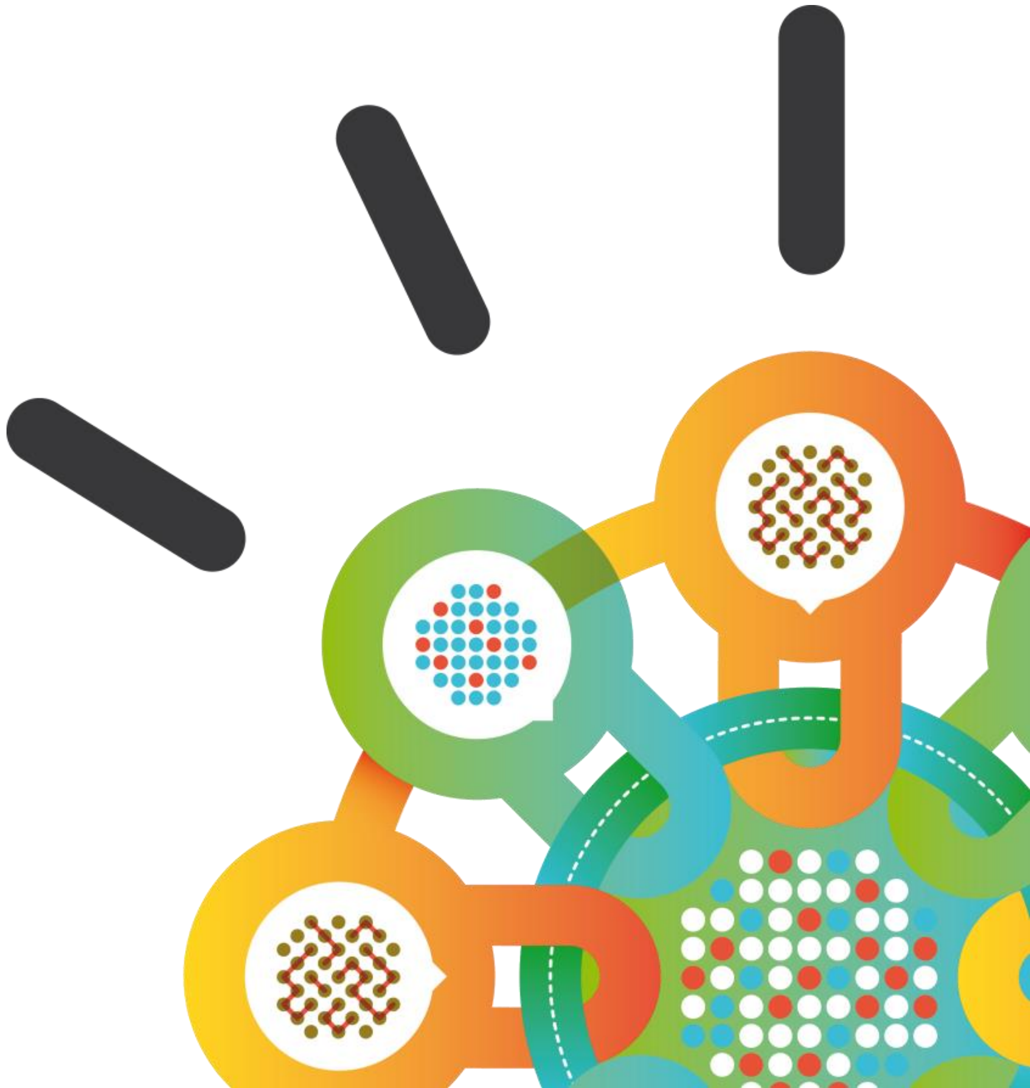
Data Science with Spark

Shelly Garion

IBM Research -- Haifa

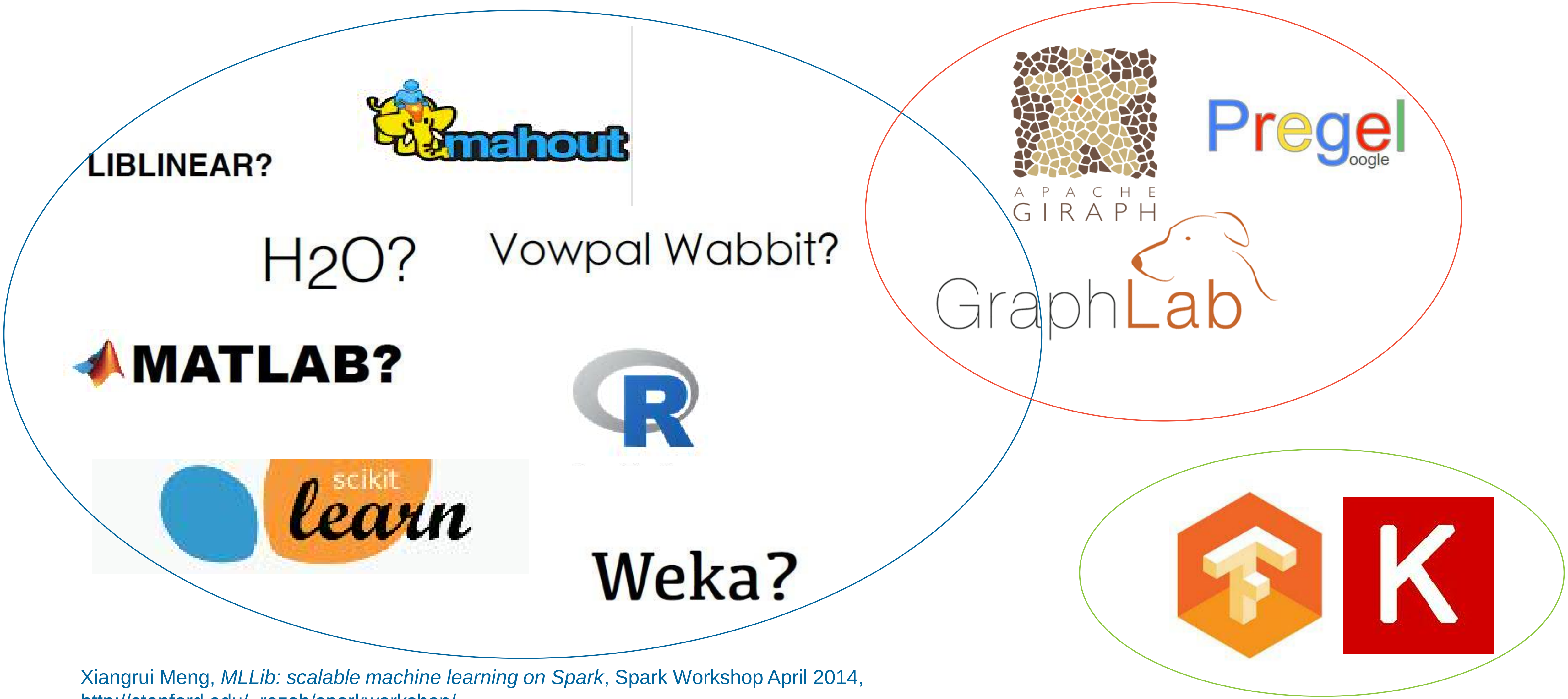


Overview – Advanced Data Analysis Tools

- Spark MLLib – large scale machine learning
  - –RDD based API
  - –DataFrame based API
- Spark GraphX – graph-parallel processing

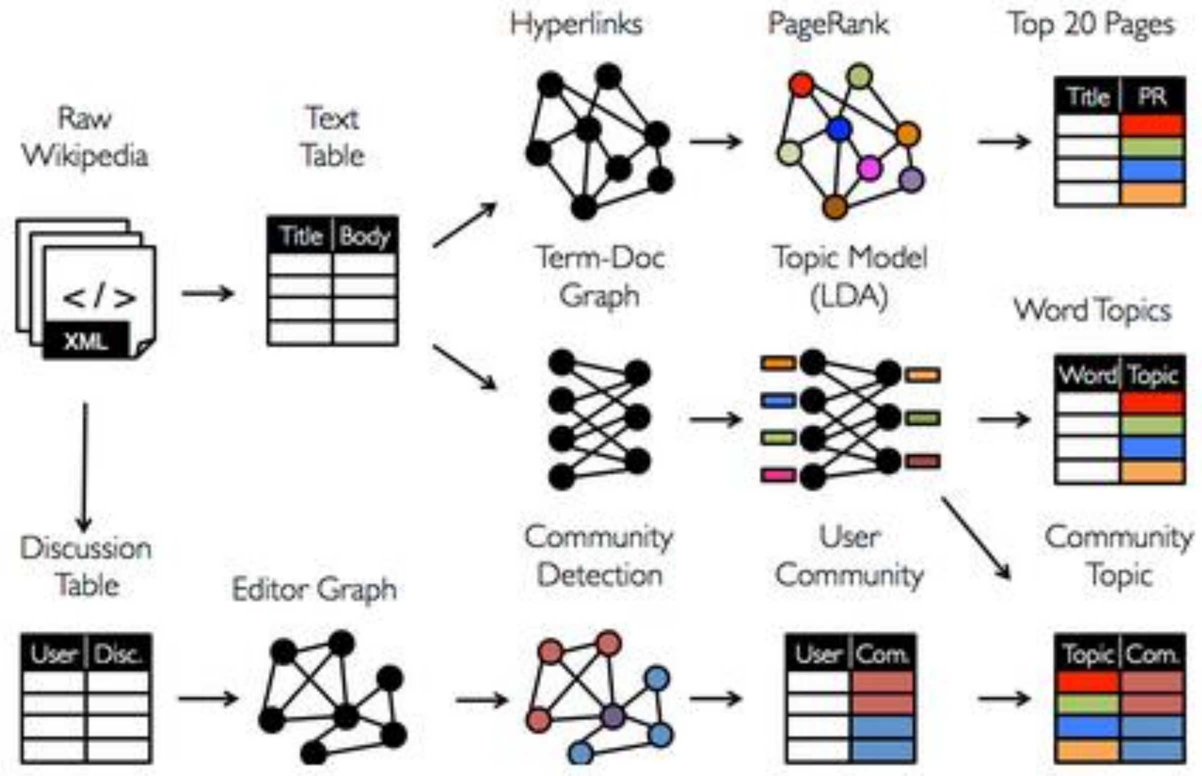
➢How to clean your data?
➢How to combine it all?
➢How to visualize it?

Why Spark MLLib & GraphX?



Xiangrui Meng, *MLLib: scalable machine learning on Spark*, Spark Workshop April 2014, http://stanford.edu/~rezab/sparkworkshop/

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

Machine Learning Algorithms

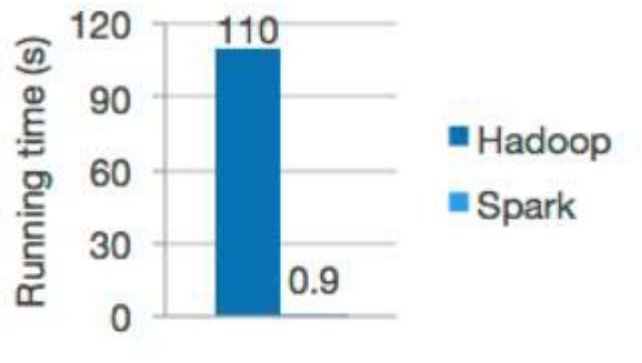
- Classification
  - Logistic regression
  - Linear support vector machine (SVM)
  - Naïve Bayes
  - Decision trees and forests
- Regression
  - Generalized linear regression (GLM)
- Recommendation
  - Alternating least squares (ALS)
- Clustering
  - K-means and Streaming K-means
  - Gaussian mixture
  - Latent Dirichlet allocation (LDA)
- Dimensionality reduction
  - Singular value decomposition (SVD)
  - Principal component analysis (PCA)
- Feature extraction & selection
  - Word2Vec
- …

See: https://spark.apache.org/docs/latest/mllib-guide.html

Performance of MLLib

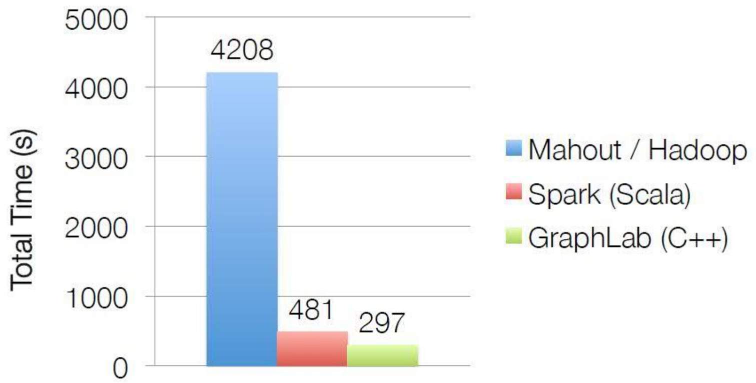
- It is built on Apache Spark, a fast and general engine for large-scale data processing.
- Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

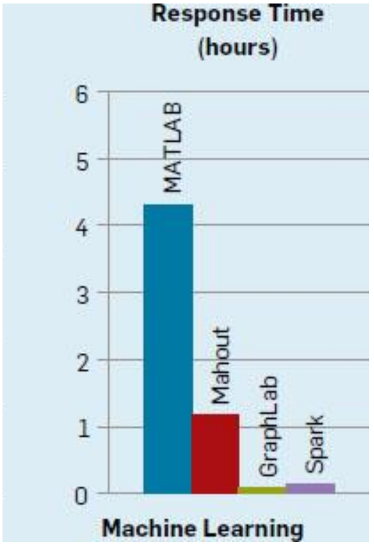
Logistic Regression



Logistic regression in Hadoop and Spark

ALS Results





- https://spark.apache.org/
- Reza Zadeh, CME 323: Distributed Algorithms and Optimization, Stanford University, http://stanford.edu/~rezab/dao/
- https://cacm.acm.org/magazines/2016/11/209116-apache-spark/fulltext

Performance of MLLib

- Speed-up between MLLib versions

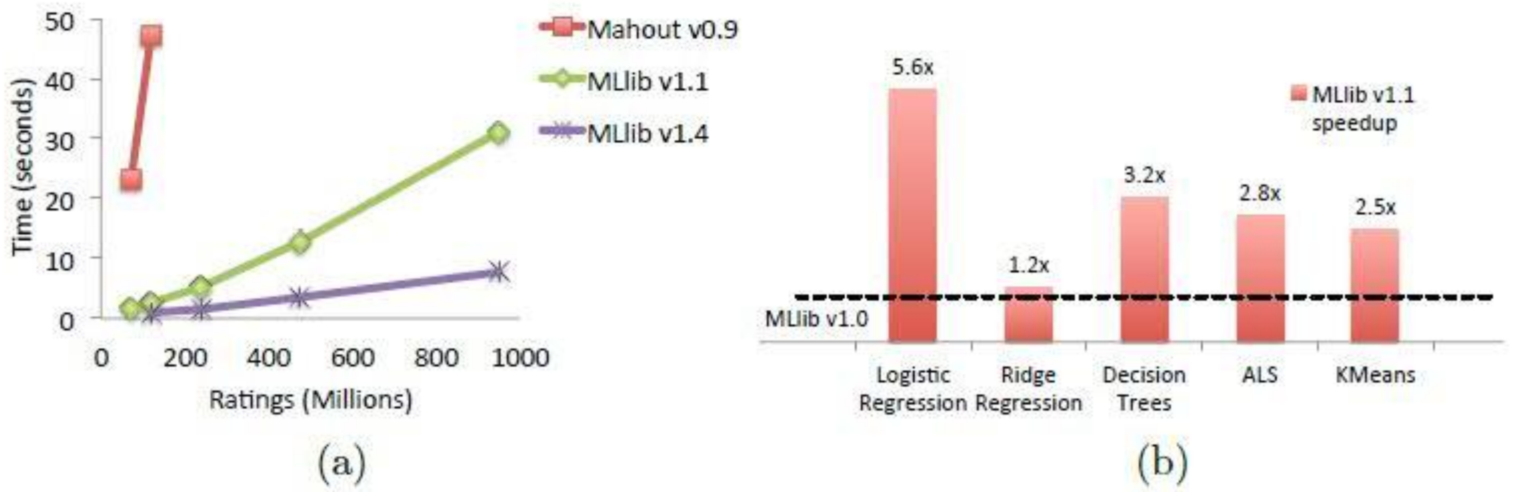
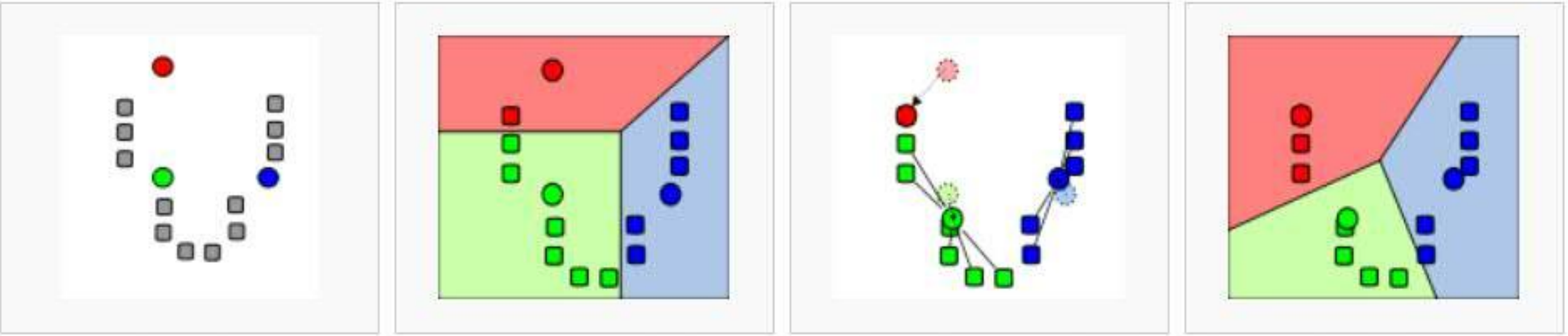


Figure 2: (a) Benchmarking results for ALS. (b) MLlib speedup between versions.

Meng et.al. "MLLib: Machine Learning in Apache Spark", Journal of Machine Learning Research 17 (2016)

Example: K-Means Clustering (RDD based API)

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)

Example: K-Means Clustering (RDD based API)

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/

Example: K-Means Clustering (RDD based API)

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)
```

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

Example: K-Means Clustering (RDD based API)

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

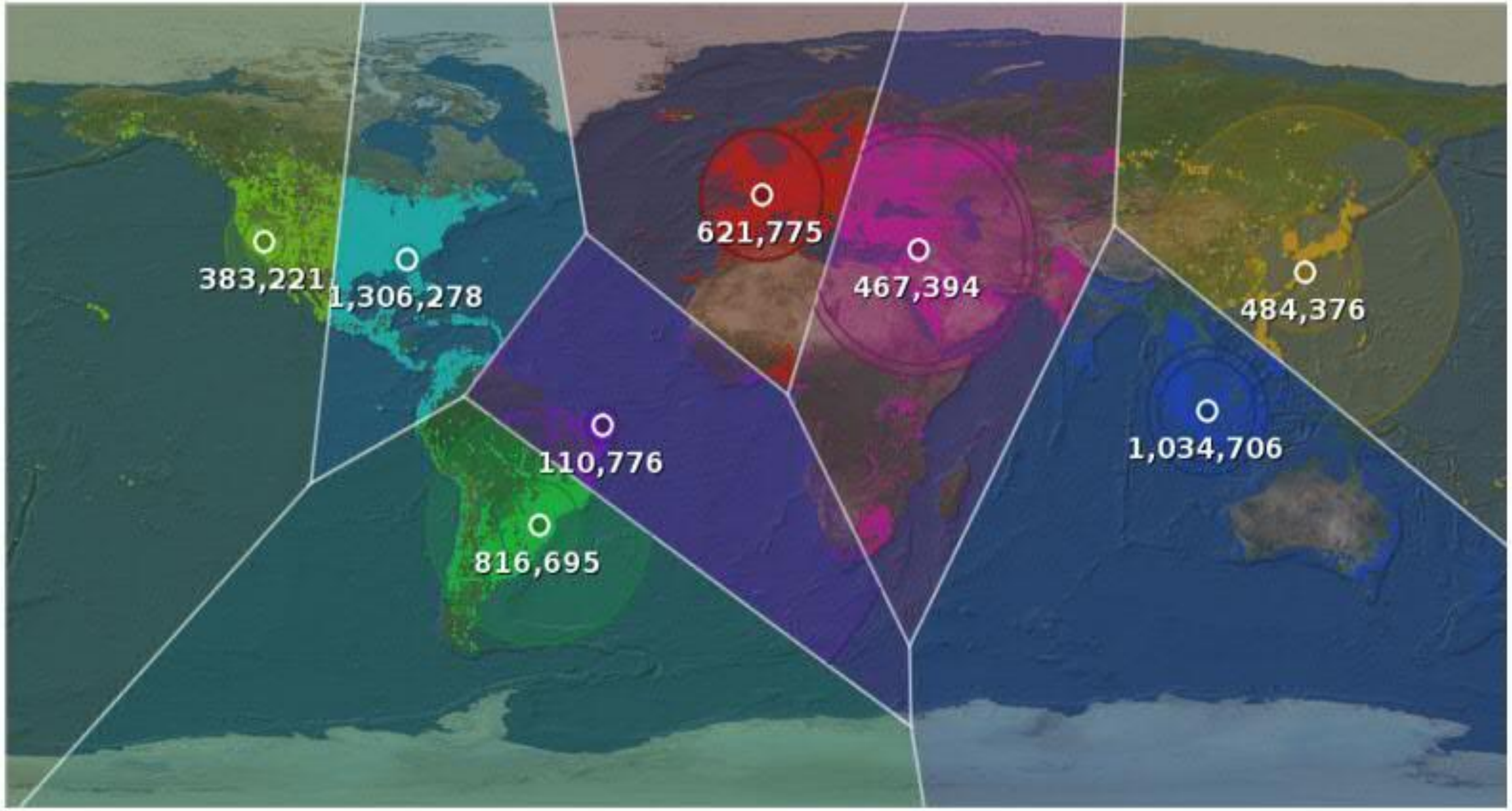
```
val clusterCenters = model.clusterCenters map (_.toArray)

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/

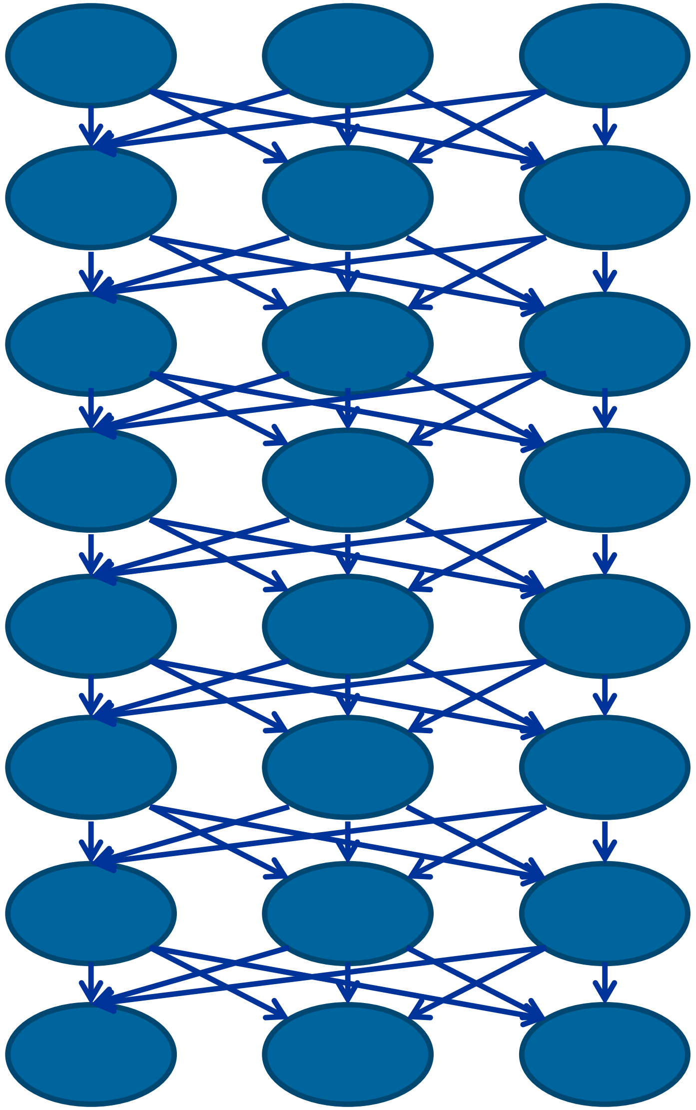
Example: K-Means Clustering (RDD based API)



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

Machine Learning Pipeline with Spark MLLib

- Data pre-processing
- Feature extraction
- Model fitting
- Model training
- Validation
- Model prediction

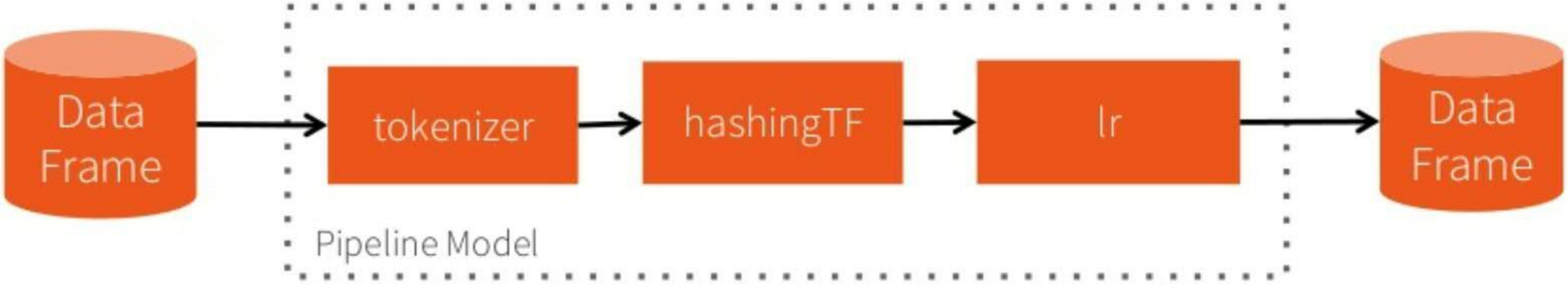


Spark MLLib Pipeline (DataFrame based API)

```
// 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")
```



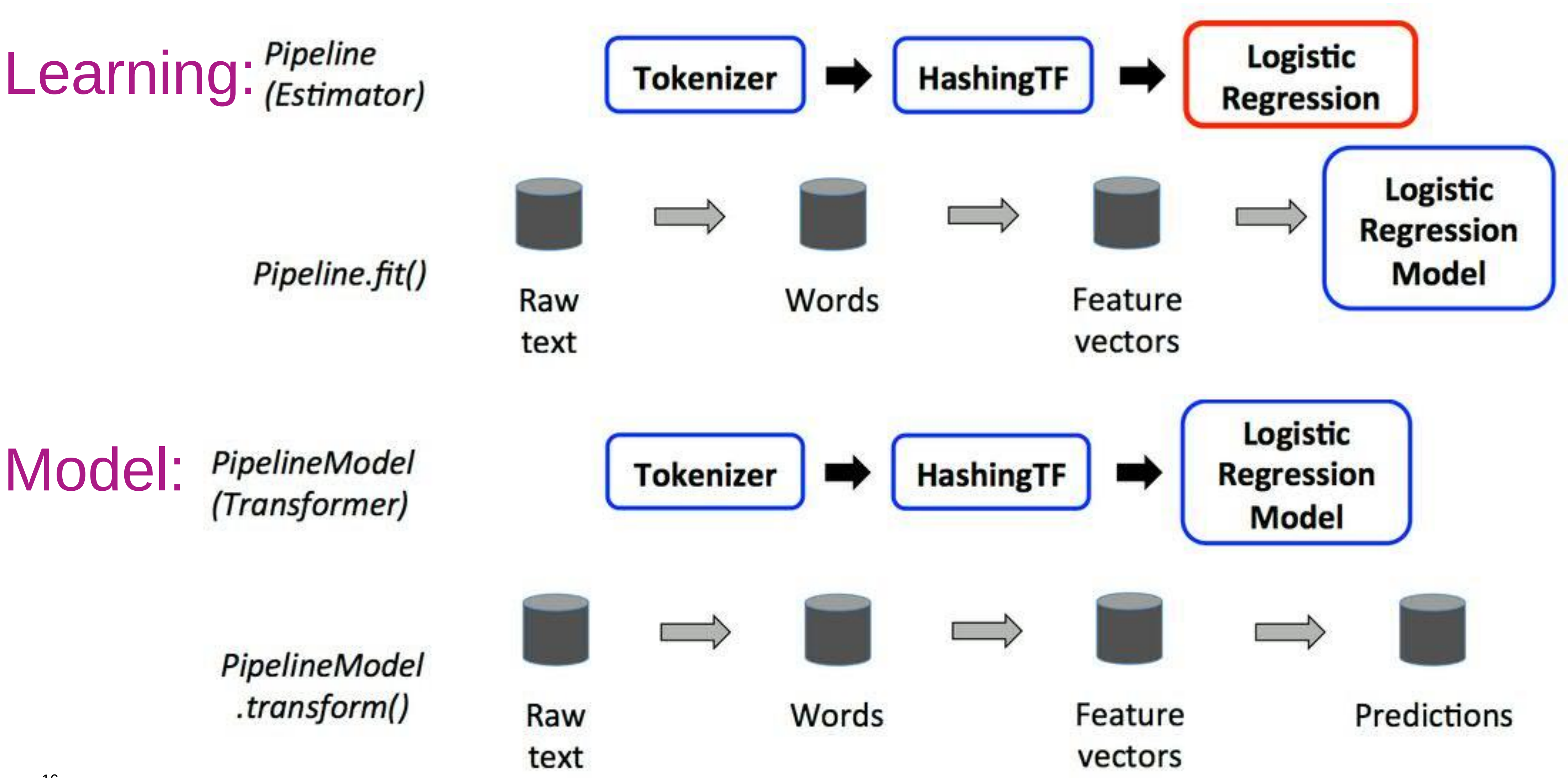
Patrick Wendell, Matei Zaharia, “Spark community update”, https://spark-summit.org/2015/events/keynote-1/

Spark MLLib Pipeline (DataFrame based API)

- **DataFrame:**
  - Use DataFrame from Spark SQL as ML dataset
  - Can have different columns storing text, feature vectors, true labels, and predictions
- **Transformer:**
  - A Transformer implements a method `transform()`
  - Algorithm that transforms one DataFrame to another DataFrame
    - Feature transformers (e.g., OneHotEncoder)
    - Trained ML models (e.g., LogisticRegressionModel)
- **Estimator:**
  - An Estimator implements a method `fit()`
  - Algorithm which can be fit on a DataFrame to produce a transformer
    - ML algorithms which trains on a DataFrame and produces a model (e.g., LogisticRegression)
- **Pipeline:**
  - Chains multiple Transformers and Estimators together to specify an ML workflow

https://spark.apache.org/docs/latest/ml-pipeline.html

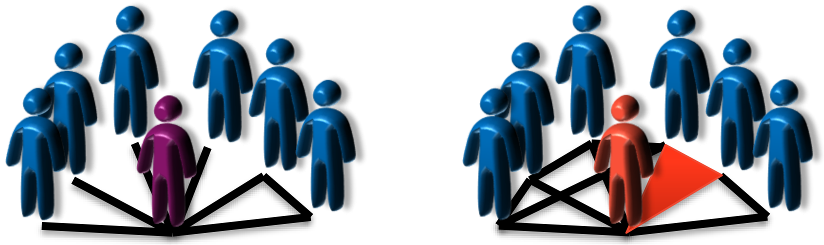
Machine Learning Pipeline with Spark MLLib



https://spark.apache.org/docs/latest/ml-pipeline.html

Spark GraphX
Key idea

- Graphs are essential to analytics (e.g. social networks)
- Tables & Graphs are composable views of the same physical data
- Each view has its own operators that exploit the semantics of the view to achieve efficient execution
- Graph algorithms are based on Pregel API



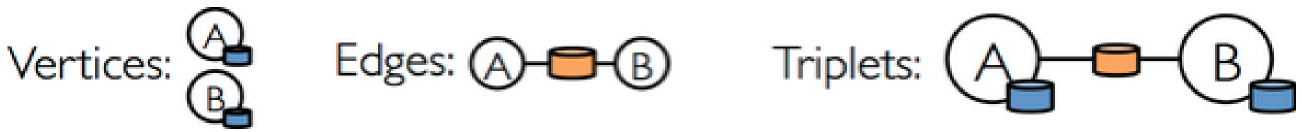
Fewer Triangles
Weaker Community

More Triangles
Stronger Community

Joseph Gonzalez, Reynold Xin, Daniel Crankshaw, Ankur Dave, Michael Franklin, and Ion Stoica, GraphX: *Unifying Data-Parallel and Graph-Parallel Analytics,* *https://amplab.cs.berkeley.edu/wp-content/uploads/2014/02/graphx@strata2014_final.pdf*

Spark GraphX
Main components

- **VertexRDD** maps IDs to vertex content
- **EdgeRDD** are of the form (ID1, ID2, ET)
- **Triplets** are a combination of Vertex & Edge RDDs

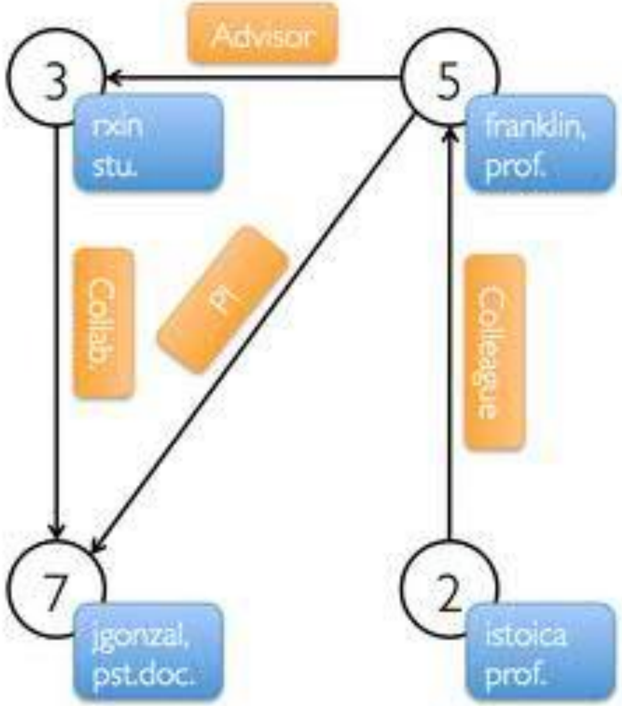



```
def Graph(vertices: Table[ (Id, V) ],
          edges: Table[ (Id, Id, E) ])
// Table Views -----------------
def vertices: Table[ (Id, V) ]
def edges: Table[ (Id, Id, E) ]
def triplets: Table [ ((Id, V), (Id, V), E)]
```

Spark GraphX Example

```scala
val users: RDD[(VertexId, (String, String))] =
  sc.parallelize(Array((3L, ("rxin", "student")),
                       (7L, ("jgonzal", "postdoc")),
                       (5L, ("franklin", "prof")),
                       (2L, ("istoica", "prof"))))
// Create an RDD for edges
val relationships: RDD[Edge[String]] =
  sc.parallelize(Array(Edge(3L, 7L, "collab"),
                       Edge(5L, 3L, "advisor"),
                       Edge(2L, 5L, "colleague"),
                       Edge(5L, 7L, "pi")))
// Define a default user in case there are
relationship with missing user
val defaultUser = ("John Doe", "Missing")
// Build the initial Graph
val graph = Graph(users, relationships, defaultUser)
```

Property Graph



Vertex Table

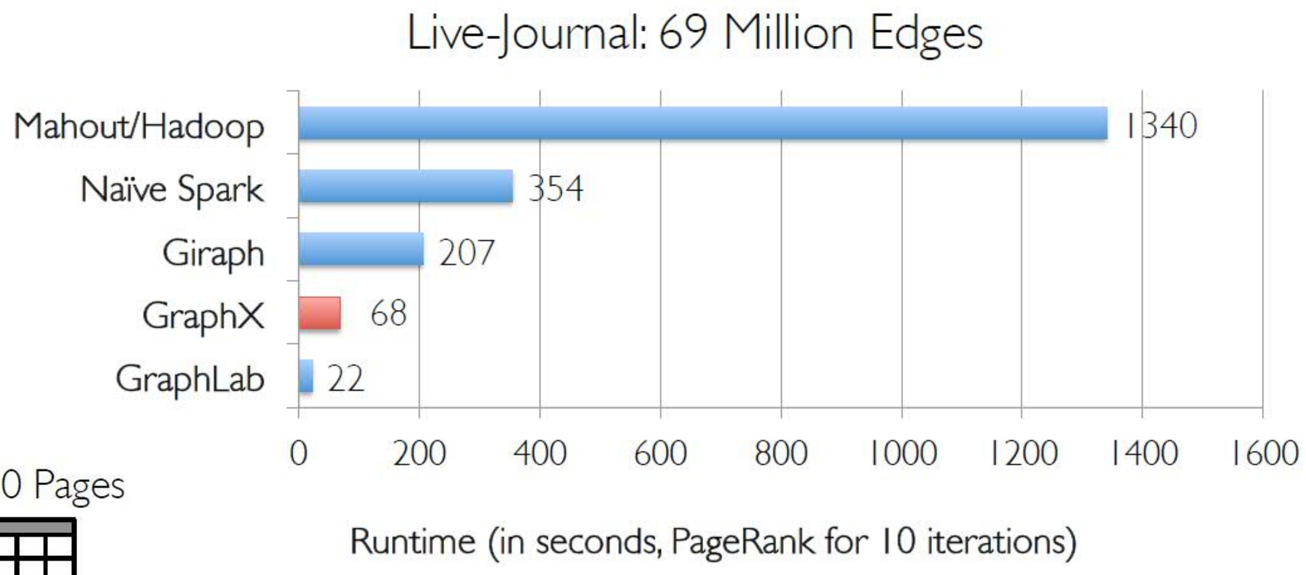
Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

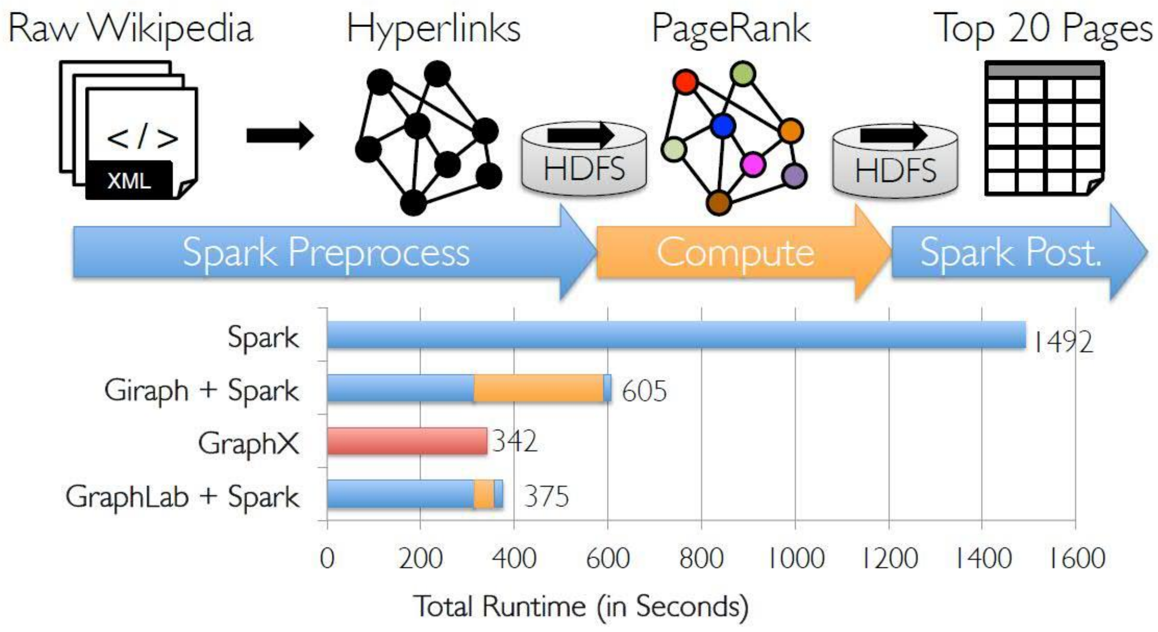
Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

https://spark.apache.org/docs/latest/graphx-programming-guide.html

Performance of GraphX

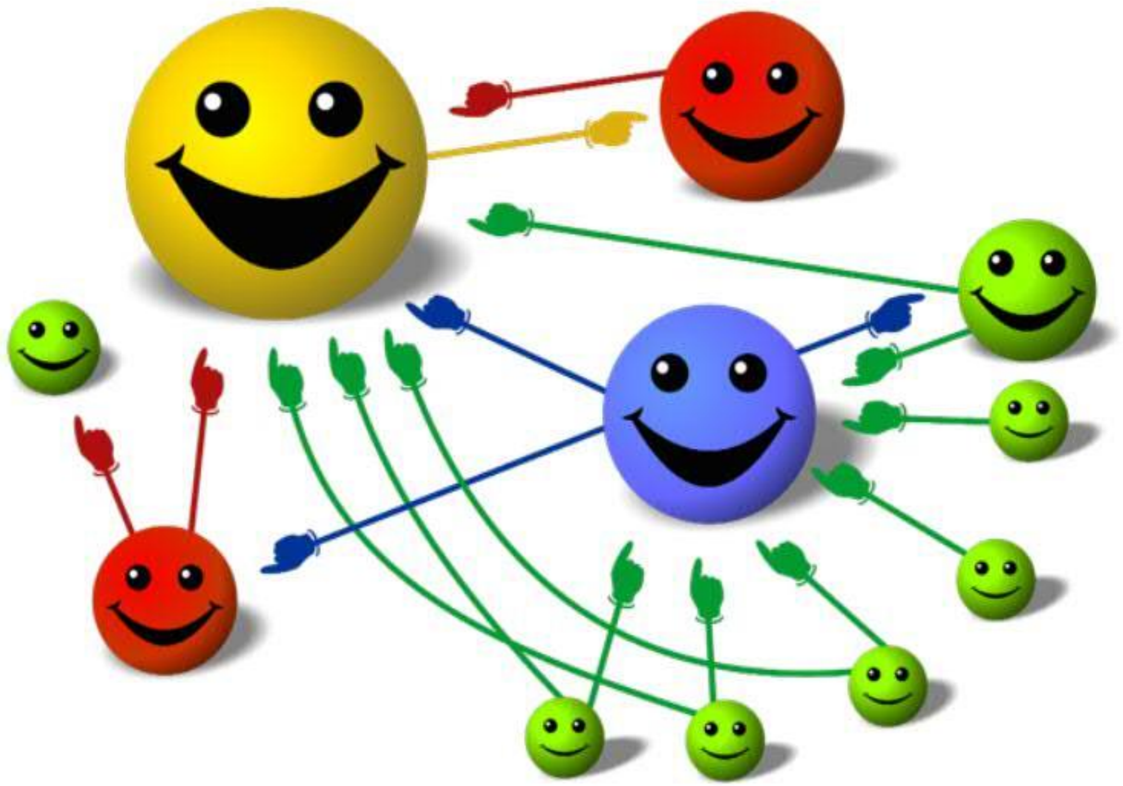




Joseph Gonzalez, Reynold Xin, Daniel Crankshaw, Ankur Dave, Michael Franklin, and Ion Stoica, GraphX: *Unifying Data-Parallel and Graph-Parallel Analytics,* https://amplab.cs.berkeley.edu/wp-content/uploads/2014/02/graphx@strata2014_final.pdf

Example - PageRank

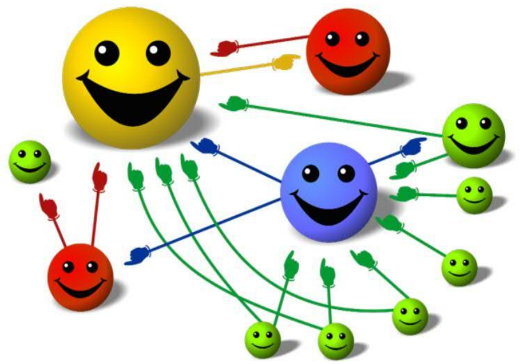
Popular algorithm originally introduced by Google



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

Example - PageRank

Popular algorithm originally introduced by Google



PageRank Algorithm

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

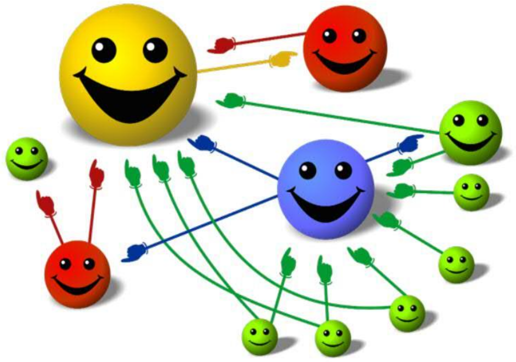
A. $contrib = \frac{curRank}{|neighbors|}$

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.

Example: PageRank
Spark GraphX

Popular algorithm originally introduced by Google



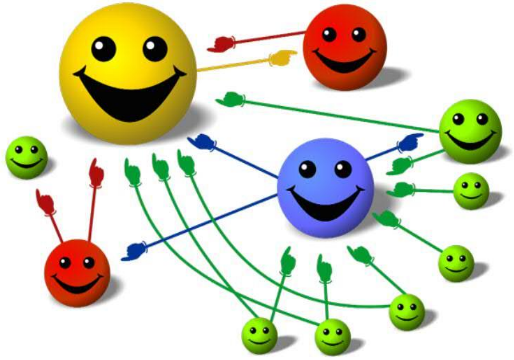
```
// get people with top-k pageranks
def findTopPageRank(allPeople: RDD[String], links: RDD[(String, String, Double)], k: Int) = {
  val versRDD = allPeople.map(p => (uid(p), p))
  val edgesRDD = links.map{ case (l, r, score) => Edge(uid(l), uid(r), score) }

  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))
}
```

Example: PageRank
How to implement it with Map/Reduce?

Popular algorithm originally introduced by Google



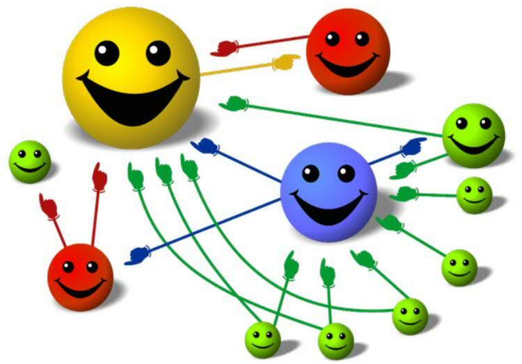
```
val links = // load RDD of (url, neighbors) pairs
var ranks = // load RDD of (url, rank) pairs

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(...)
```

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

Example: PageRank
How is it implemented in Pregel?

Popular algorithm originally introduced by Google



```
def PageRank(v: Id, msgs: List[Double]) {
  // Compute the message sum
  var msgSum = 0
  for (m <- msgs) { msgSum += m }
  // Update the PageRank
  PR(v) = 0.15 + 0.85 * msgSum
  // Broadcast messages with new PR
  for (j <- OutNbrs(v)) {
    msg = PR(v) / NumLinks(v)
    send_msg(to=j, msg)
  }
  // Check for termination
  if (converged(PR(v))) voteToHalt(v)
}
```

Reynold S. Xin, Daniel Crankshaw, Ankur Dave, Joseph E. Gonzalez, Michael J. Franklin, Ion Stoica.
GraphX: Unifying Data-Parallel and Graph-Parallel Analytics. *OSDI 2014*. October 2014.

Open your notebooks…