Graph Analytics in Spark

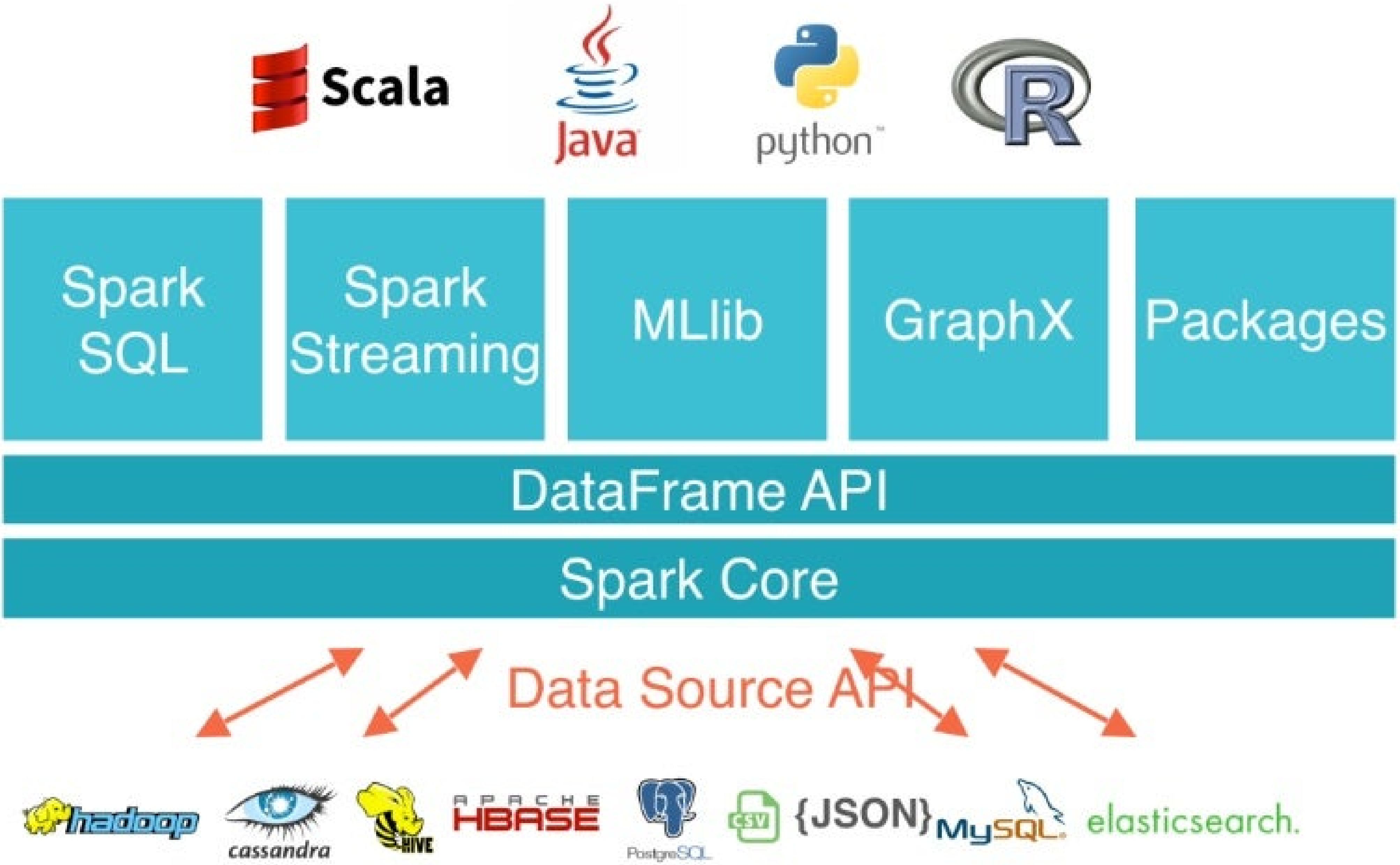
2015-06-08 • Scala Days • Amsterdam

Paco Nathan, @pacoid

Spark Overview

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Spark Overview: *Components*



Spark Overview: *Key Distinctions vs. MapReduce*

- generalized patterns
  ⇒ unified engine for many use cases
- lazy evaluation of the lineage graph
  ⇒ reduces wait states, better pipelining
- generational differences in hardware
  ⇒ off-heap use of large memory spaces
- functional programming / ease of use
  ⇒ reduction in cost to maintain large apps
- lower overhead for starting jobs
- less expensive shuffles

TL;DR: *Smashing The Previous Petabyte Sort Record*

databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html

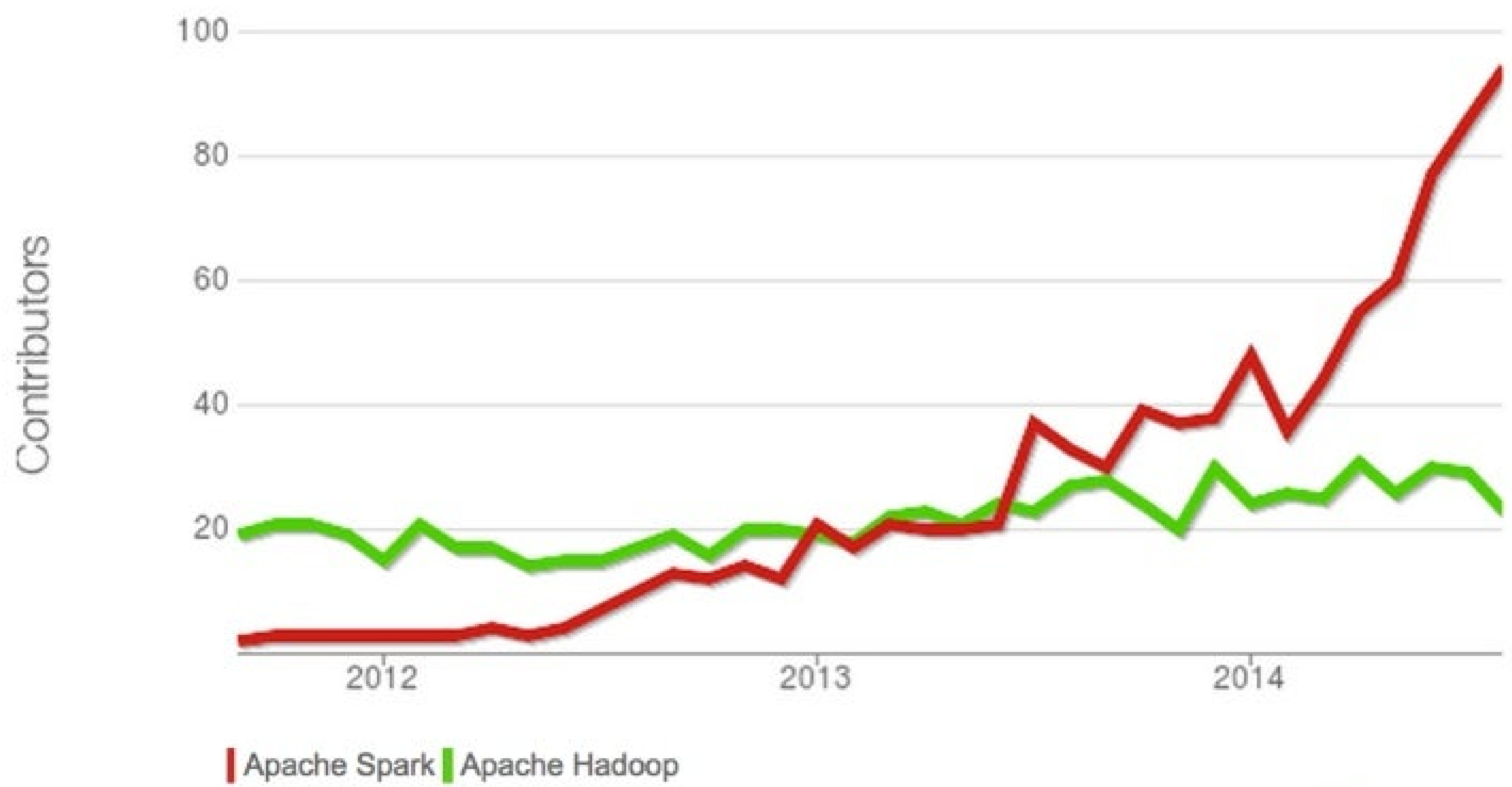
	Hadoop MR Record	**Spark Record**	**Spark 1 PB**
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

TL;DR: *Sustained Exponential Growth*

Spark is one of the most active Apache projects

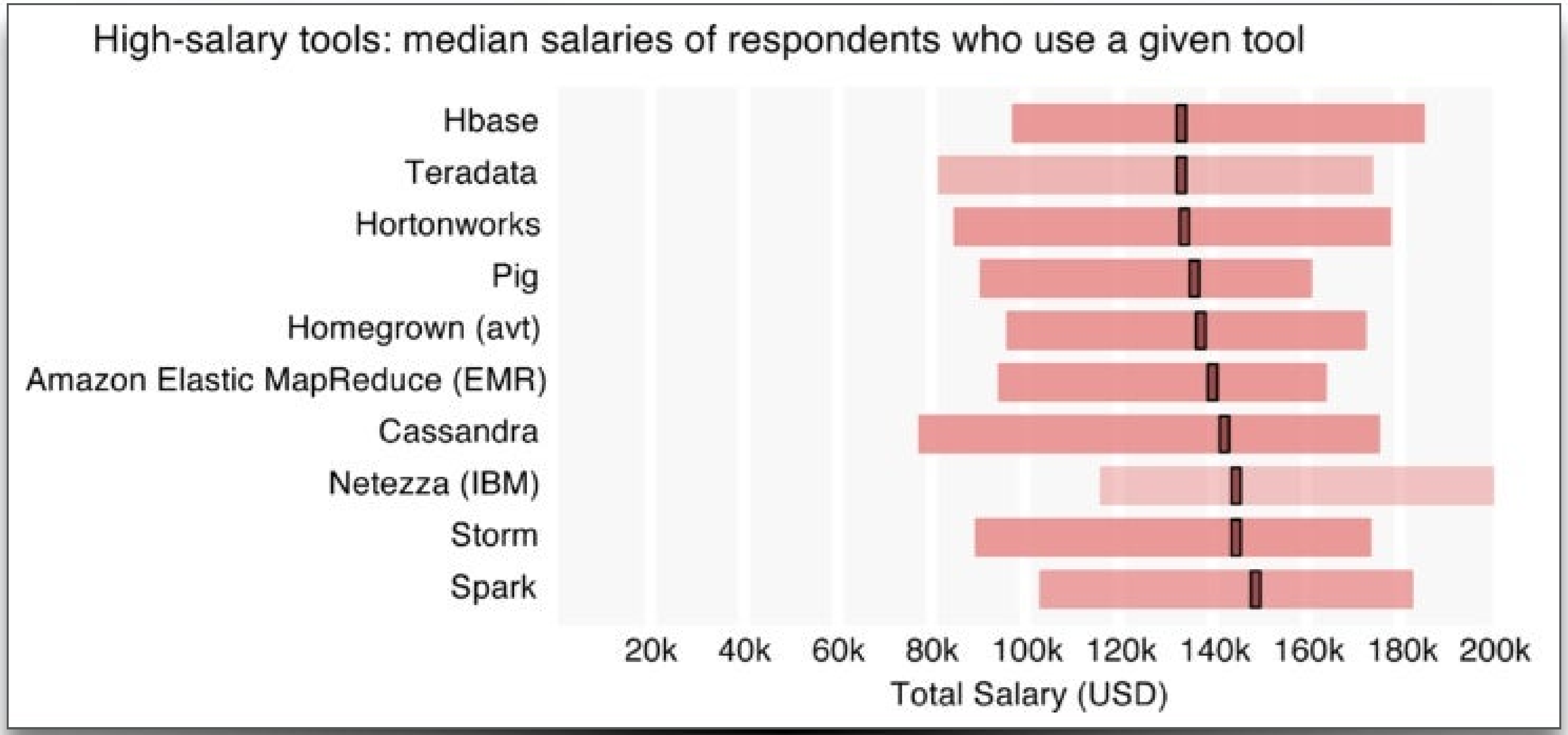
ohloh.net/orgs/apache

Number of contributors who made changes to the project source code each month.



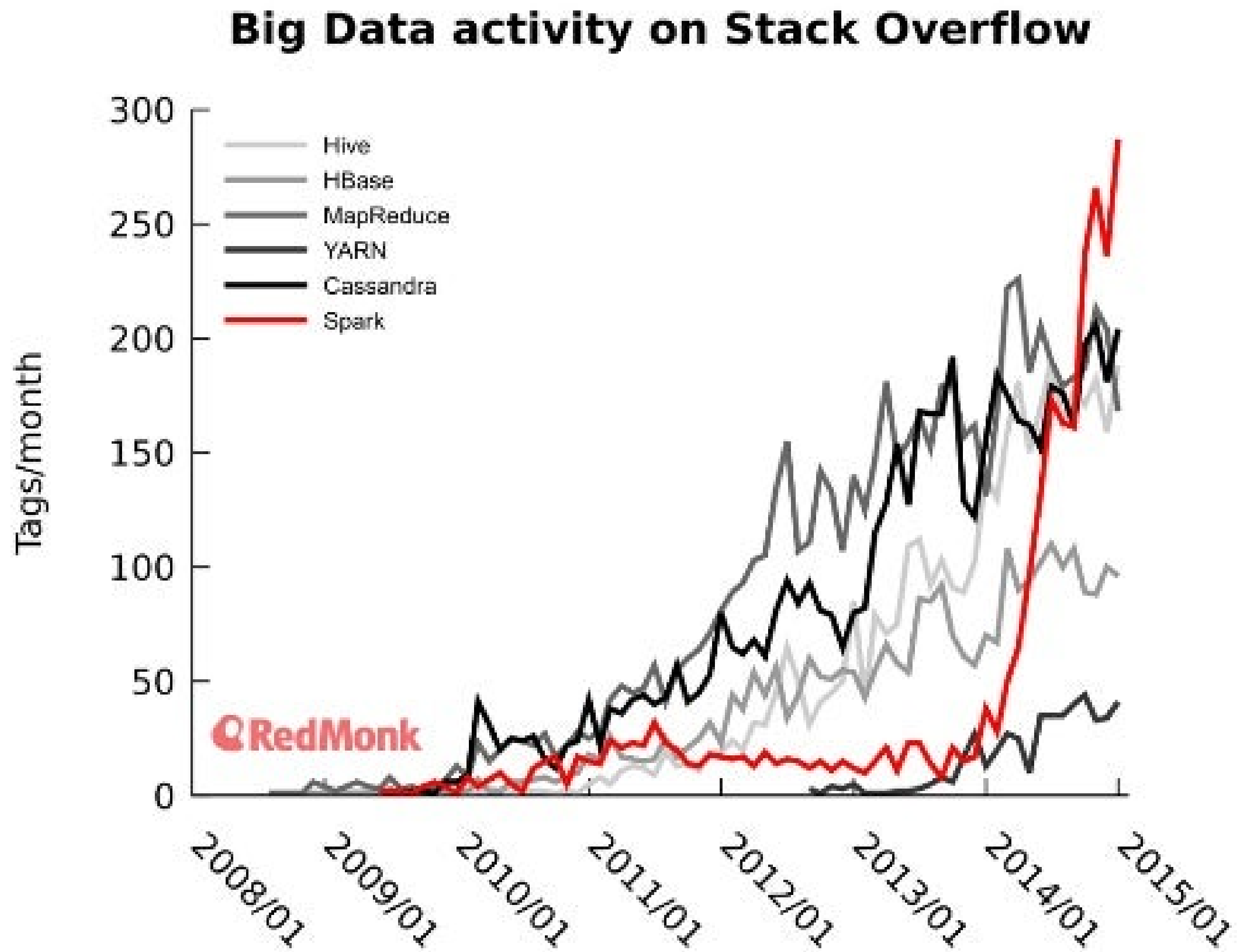
TL;DR: *Spark Expertise Tops Median Salaries within Big Data*

oreilly.com/data/free/2014-data-science-salary-survey.csp



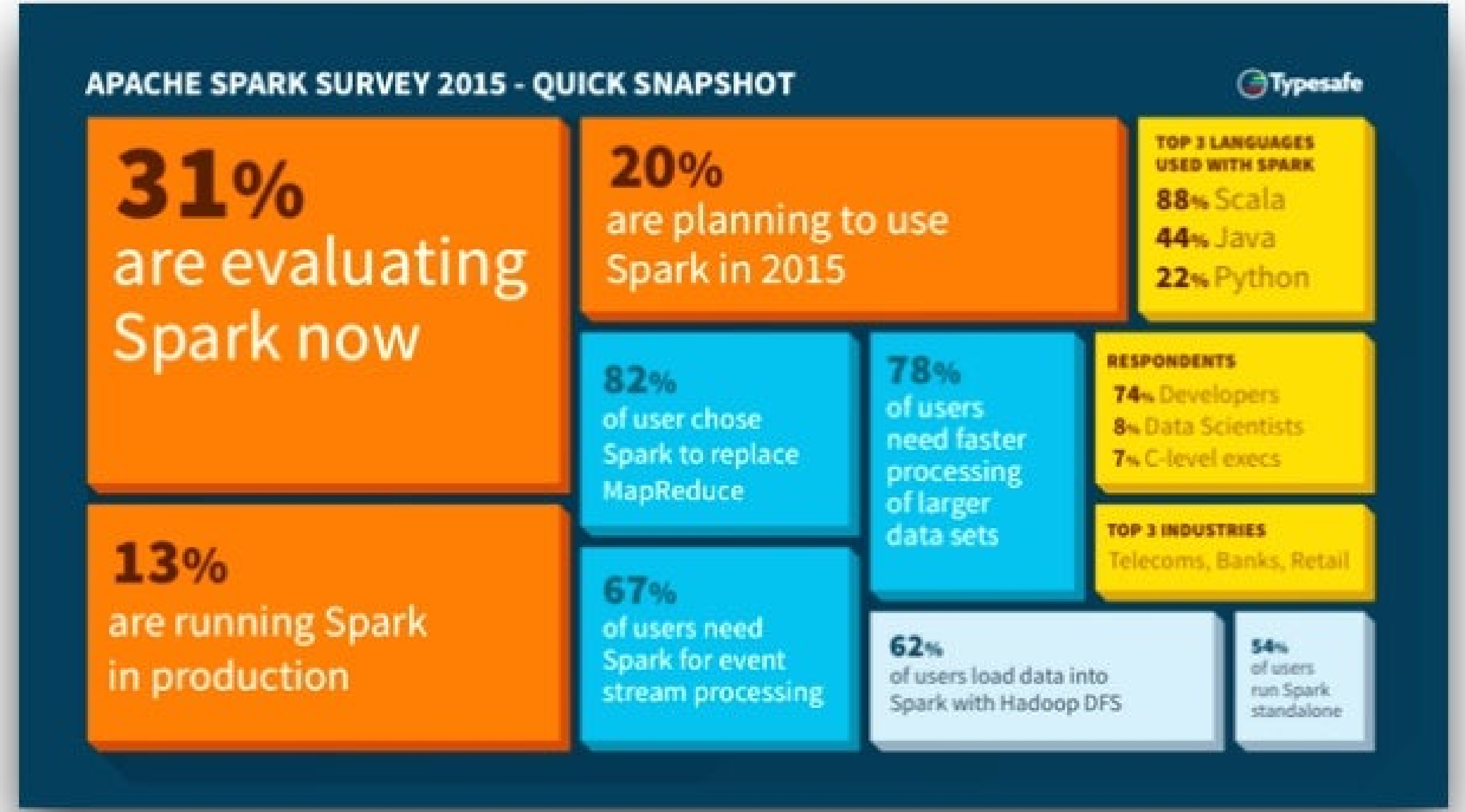
TL;DR: *Spark on StackOverflow*

twitter.com/dberkholz/status/568561792751771648

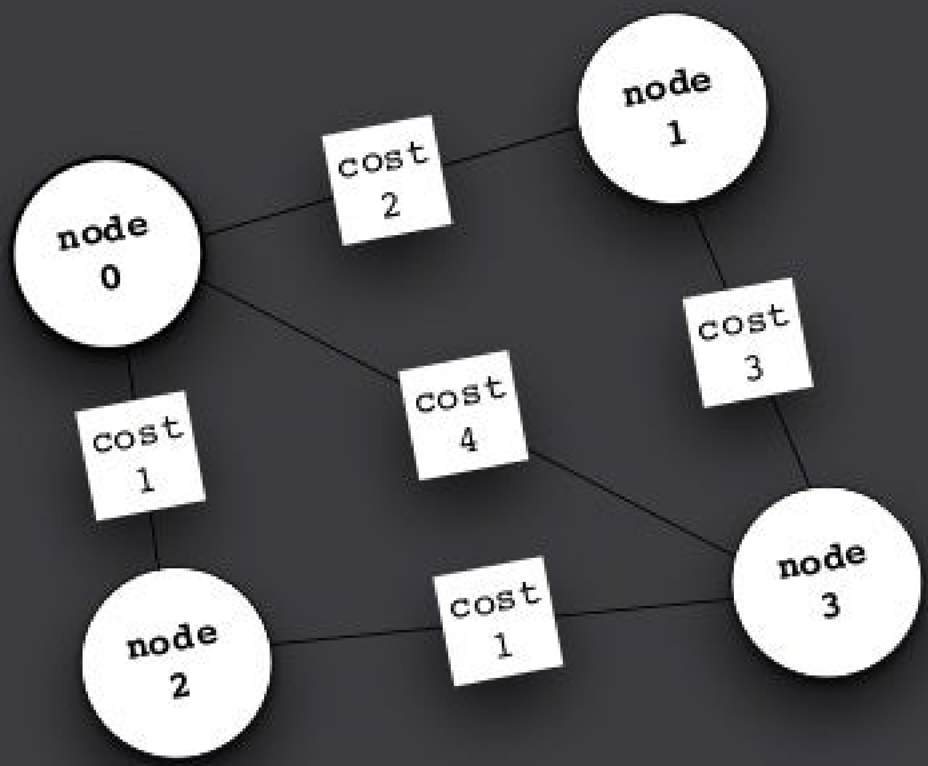


TL;DR: *Spark Survey 2015 by Databricks + Typesafe*

databricks.com/blog/2015/01/27/big-data-projects-are-hungry-for-simpler-and-more-powerful-tools-survey-validates-apache-spark-is-gaining-developer-traction.html



GraphX examples



GraphX:

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

Key Points:

- graph-parallel systems
- emphasis on integrated workflows
- optimizations

GraphX: *Further Reading…*

PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs
J. Gonzalez, Y. Low, H. Gu, D. Bickson, C. Guestrin
graphlab.org/files/osdi2012-gonzalez-low-gu-bickson-guestrin.pdf

Pregel: Large-scale graph computing at Google
Grzegorz Czajkowski, et al.
googleresearch.blogspot.com/2009/06/large-scale-graph-computing-at-google.html

GraphX: Graph Analytics in Spark
Ankur Dave, Databricks
spark-summit.org/east-2015/talk/graphx-graph-analytics-in-spark

Topic modeling with LDA: MLlib meets GraphX
Joseph Bradley, Databricks
databricks.com/blog/2015/03/25/topic-modeling-with-lda-mllib-meets-graphx.html

GraphX: *Compose Node + Edge RDDs into a Graph*

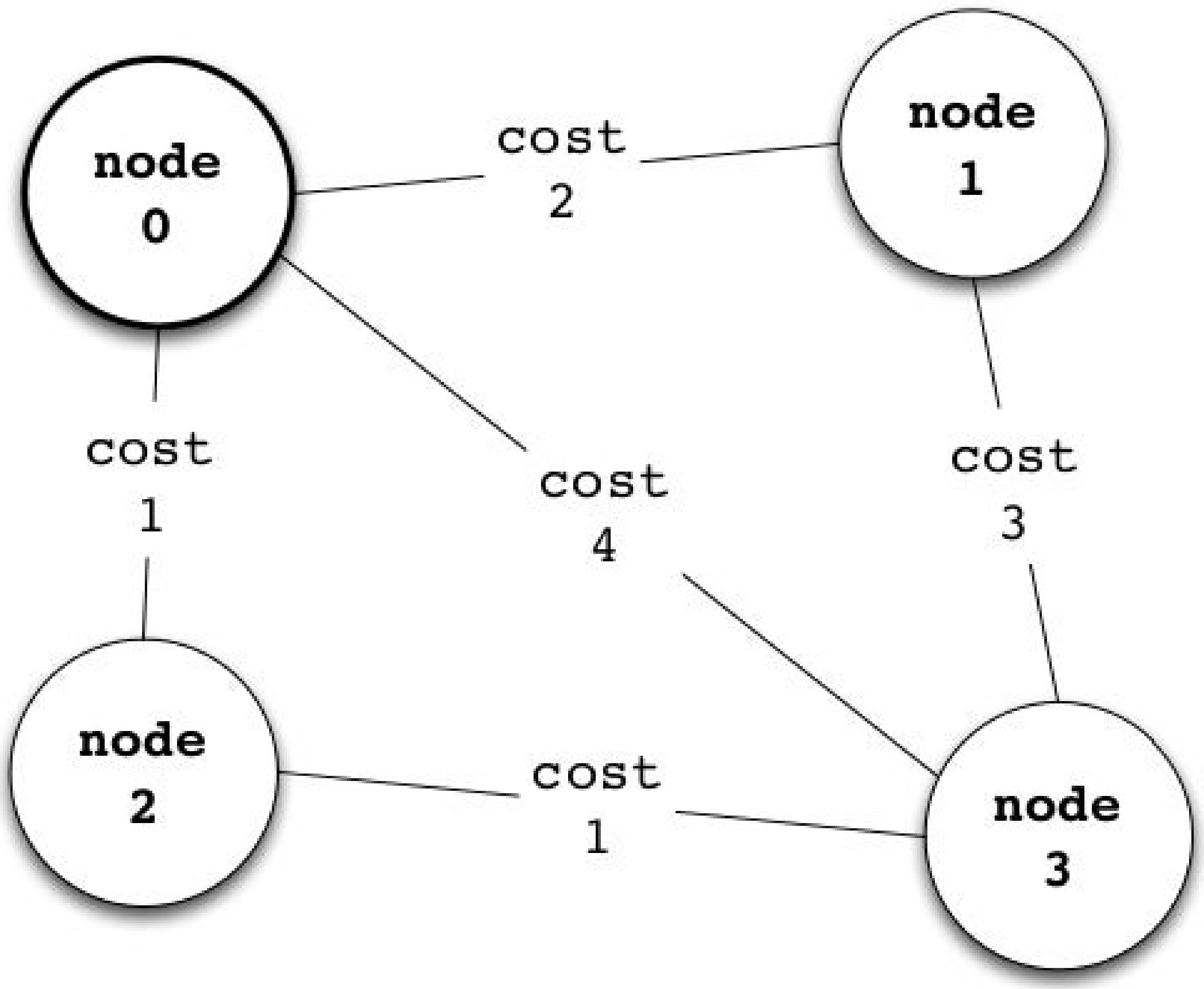
```
val nodeRDD: RDD[(Long, ND)] = sc.parallelize(…)
val edgeRDD: RDD[Edge[ED]] = sc.parallelize(…)

val g: Graph[ND, ED] = Graph(nodeRDD, edgeRDD)
```

GraphX: *Example – simple traversals*

```scala
// http://spark.apache.org/docs/latest/graphx-programming-guide.html

import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD

case class Peep(name: String, age: Int)

val nodeArray = Array(
  (1L, Peep("Kim", 23)), (2L, Peep("Pat", 31)),
  (3L, Peep("Chris", 52)), (4L, Peep("Kelly", 39)),
  (5L, Peep("Leslie", 45))
  )
val edgeArray = Array(
  Edge(2L, 1L, 7), Edge(2L, 4L, 2),
  Edge(3L, 2L, 4), Edge(3L, 5L, 3),
  Edge(4L, 1L, 1), Edge(5L, 3L, 9)
  )

val nodeRDD: RDD[(Long, Peep)] = sc.parallelize(nodeArray)
val edgeRDD: RDD[Edge[Int]] = sc.parallelize(edgeArray)
val g: Graph[Peep, Int] = Graph(nodeRDD, edgeRDD)

val results = g.triplets.filter(t => t.attr > 7)

for (triplet <- results.collect) {
  println(s"${triplet.srcAttr.name} loves ${triplet.dstAttr.name}")
}
```

GraphX: *Example – routing problems*

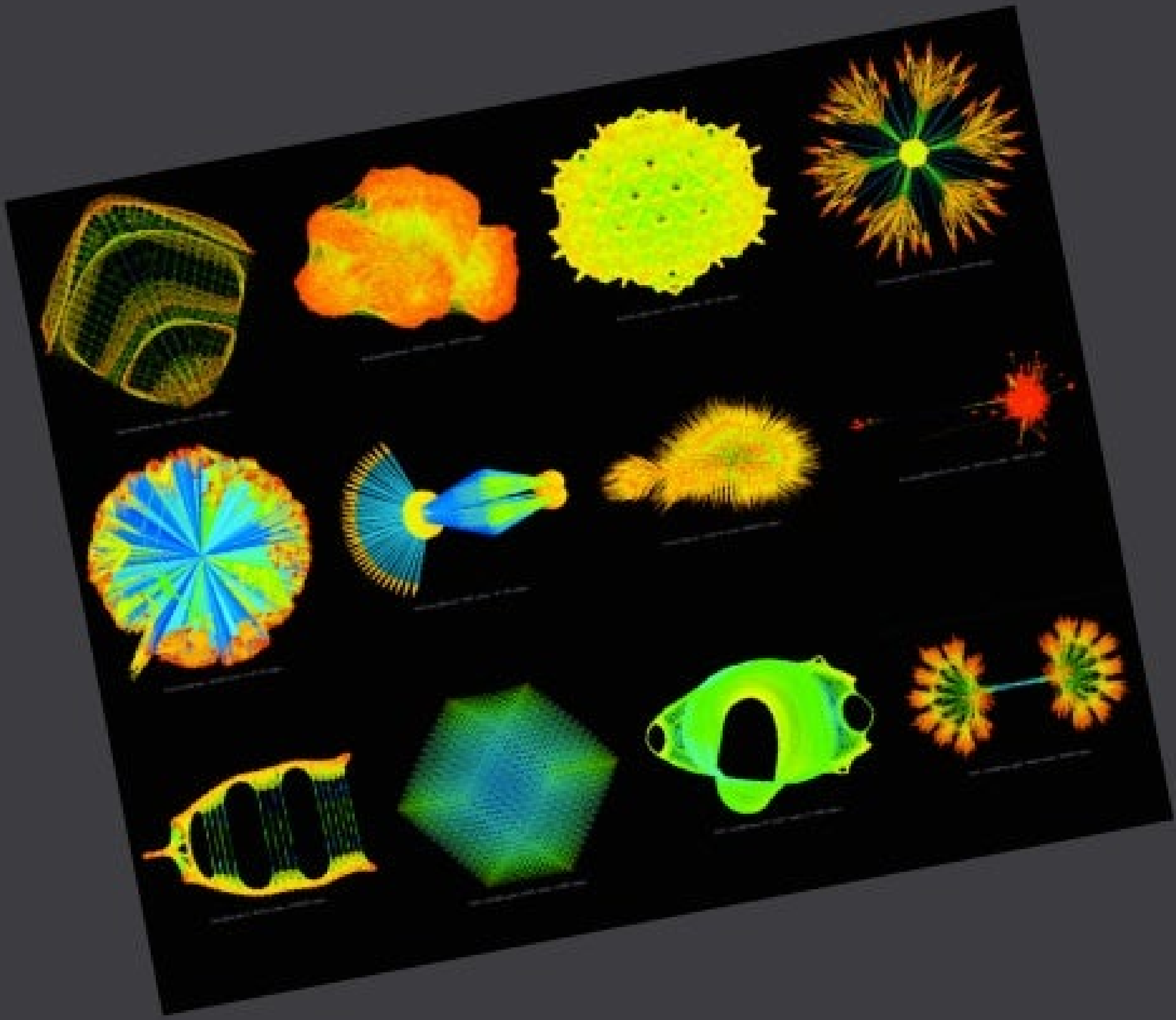
What is the cost to reach **node 0** from any other node in the graph? This is a common use case for graph algorithms, e.g., **Dijkstra**



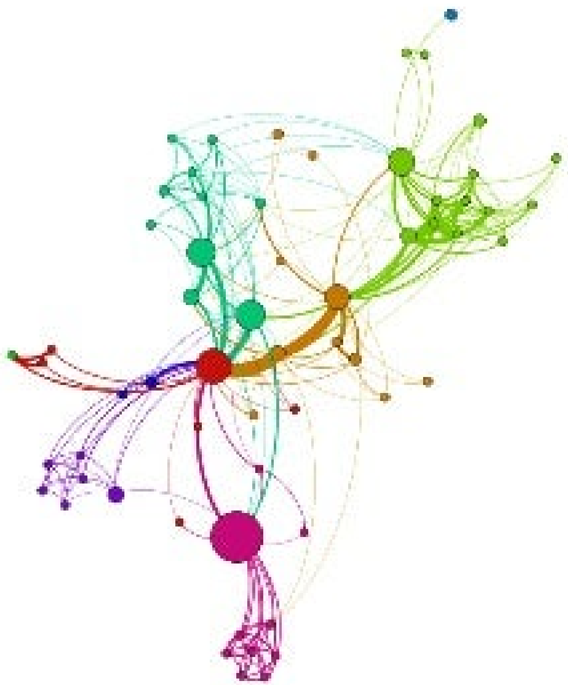
GraphX: *code examples…*

Let’s check some code!

Graph Analytics

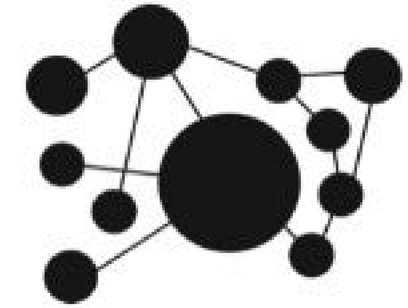


Graph Analytics: *terminology*

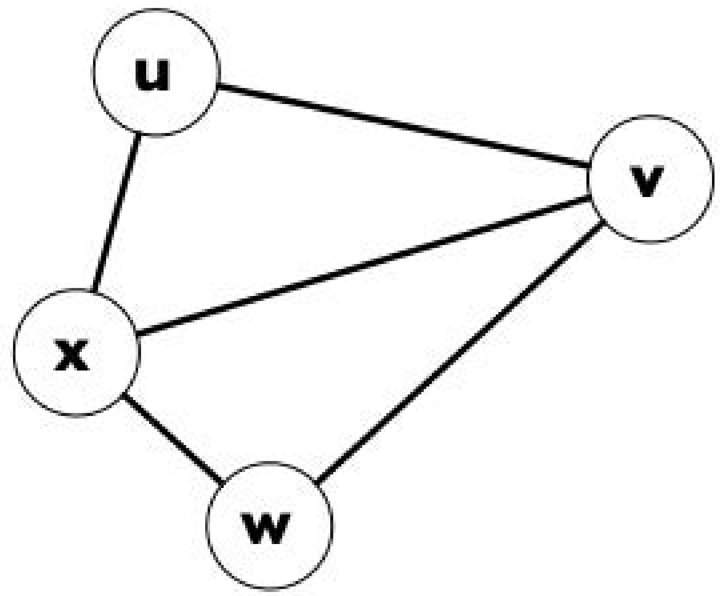


- many real-world problems are often represented as *graphs*
- graphs can generally be converted into *sparse matrices* (bridge to linear algebra)
- *eigenvectors* find the stable points in a system defined by matrices – which may be more efficient to compute
- beyond simpler graphs, complex data may require work with *tensors*

Graph Analytics: *example*



Suppose we have a graph as shown below:



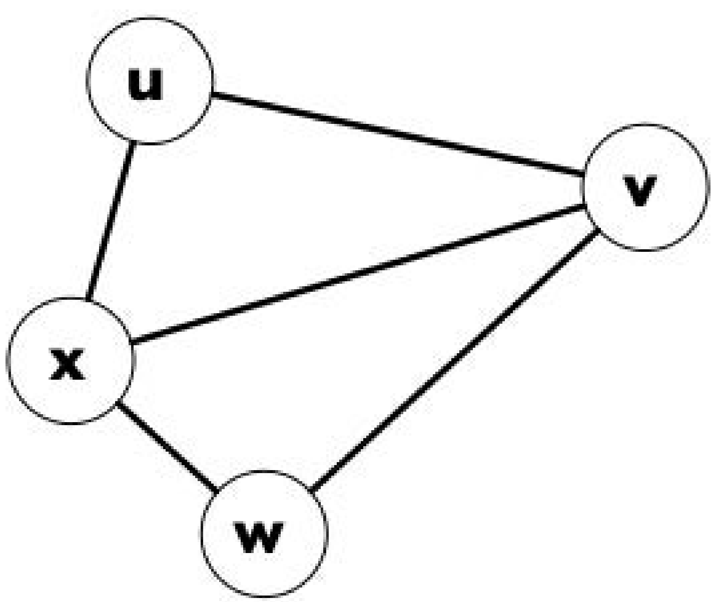
We call **x** a *vertex* (sometimes called a *node*)

An *edge* (sometimes called an *arc*) is any line connecting two vertices

Graph Analytics: *representation*

We can represent this kind of graph as an *adjacency matrix*:

- label the rows and columns based on the vertices
- entries get a 1 if an edge connects the corresponding vertices, or 0 otherwise



	u	v	w	x
u	0	1	0	1
v	1	0	1	1
w	0	1	0	1
x	1	1	1	0

Graph Analytics: *algebraic graph theory*

An *adjacency matrix* always has certain properties:

- it is *symmetric*, i.e., $\mathbf{A} = \mathbf{A}^{\mathrm{T}}$
- it has real eigenvalues

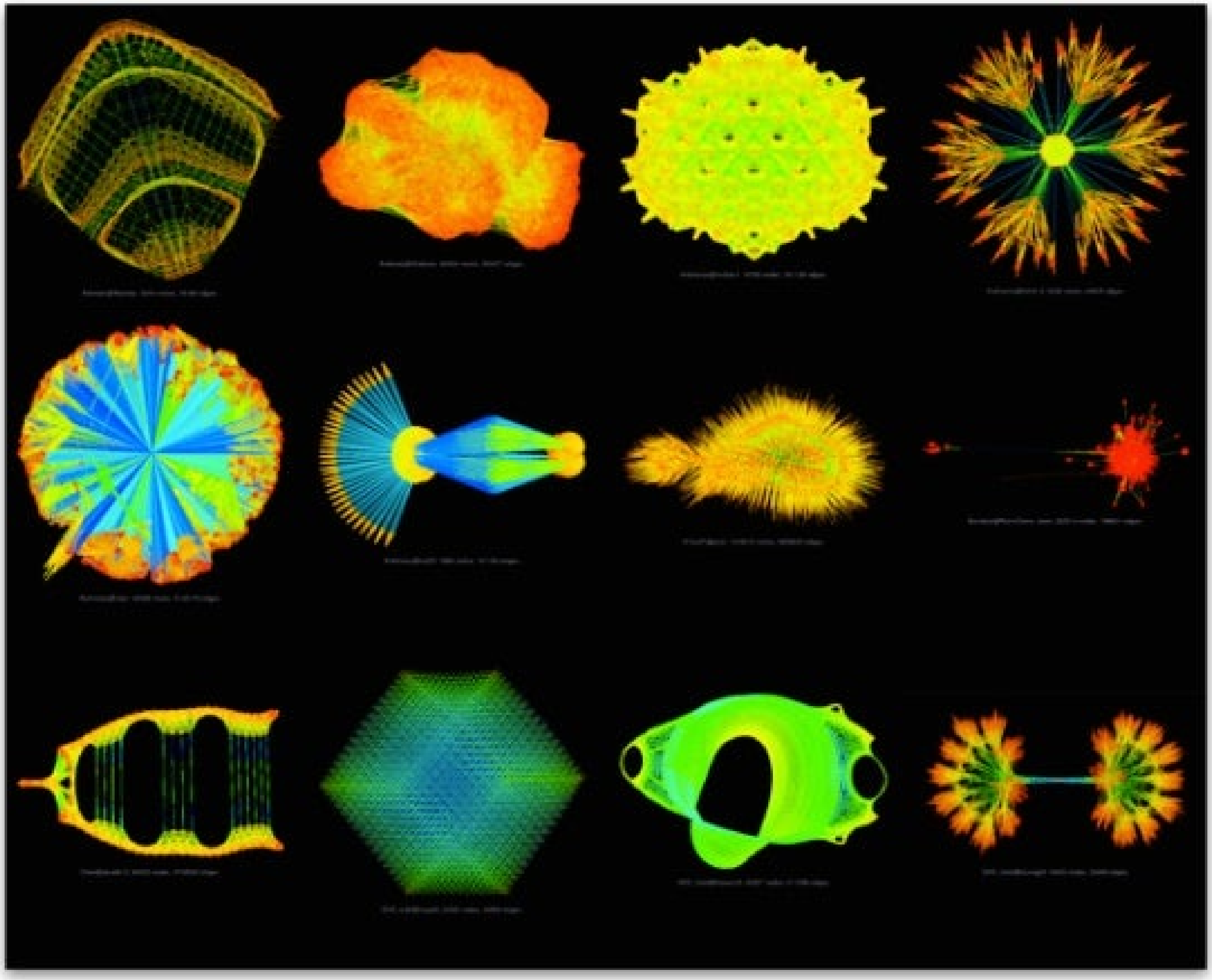
$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Therefore *algebraic graph theory* bridges between *linear algebra* and *graph theory*

Graph Analytics: *beauty in sparsity*

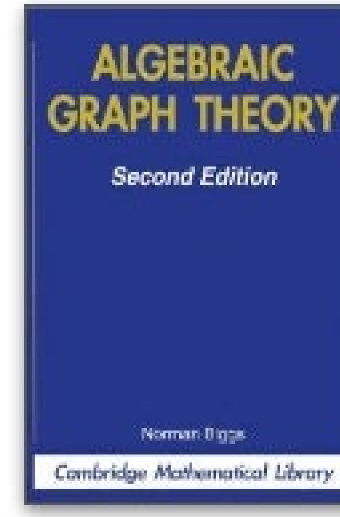
Sparse Matrix Collection… for when you **really** need a wide variety of sparse matrix examples, e.g., to evaluate new ML algorithms

University of Florida
Sparse Matrix Collection
cise.ufl.edu/
research/sparse/
matrices/

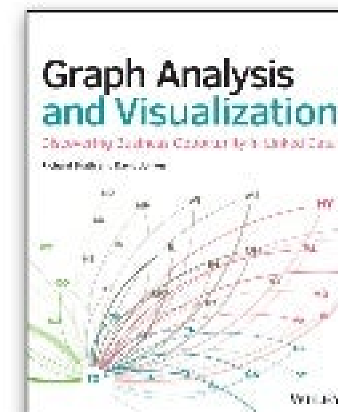


Graph Analytics: *resources*

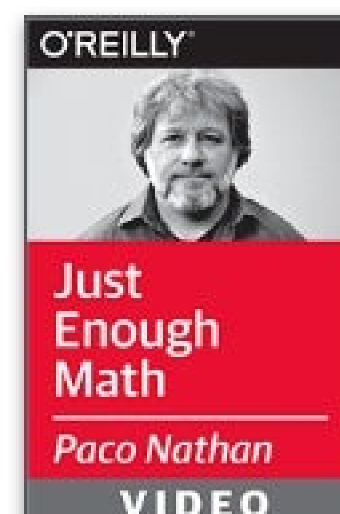
Algebraic Graph Theory
Norman Biggs
Cambridge (1974)
amazon.com/dp/0521458978



Graph Analysis and Visualization
Richard Brath, David Jonker
Wiley (2015)
shop.oreilly.com/product/9781118845844.do



See also examples in: **Just Enough Math**



Graph Analytics: *tensor solutions emerging*

Although tensor factorization is considered problematic, it may provide more general case solutions, and some work leverages Spark:

The Tensor Renaissance in Data Science

Anima Anandkumar @UC Irvine

radar.oreilly.com/2015/05/the-tensor-renaissance-in-data-science.html

Spacey Random Walks and Higher Order Markov Chains

David Gleich @Purdue

slideshare.net/dgleich/spacey-random-walks-and-higher-order-markov-chains

Data Preparation

Data Prep: *Exsto Project Overview*

- insights about dev communities, via data mining their email forums
- works with any Apache project email archive
- applies NLP and ML techniques to analyze message threads
- graph analytics surface themes and interactions
- results provide feedback for communities, e.g., leaderboards

Data Prep: *Exsto Project Overview – four links*

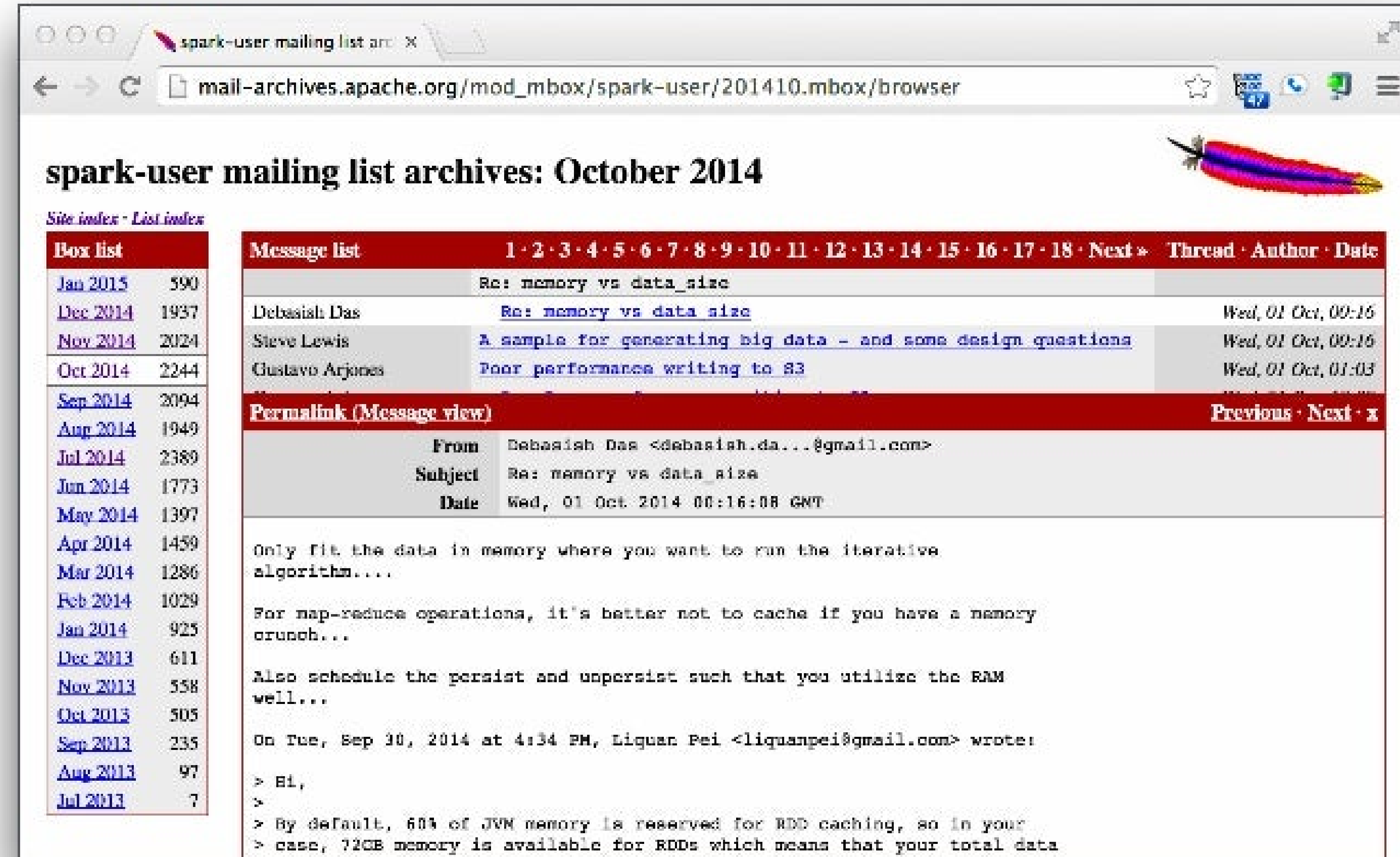
https://github.com/ceteri/spark-exercises/tree/master/exsto/dbc

http://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

http://mail-archives.apache.org/mod_mbox/spark-user/

https://class01.cloud.databricks.com/#notebook/67011

Data Prep: *Scraper pipeline*



+

github.com/ceteri/spark-exercises/tree/master/exsto/dbc

Data Prep: *Scraper pipeline*

Typical data rates, e.g., for dev@spark.apache.org:

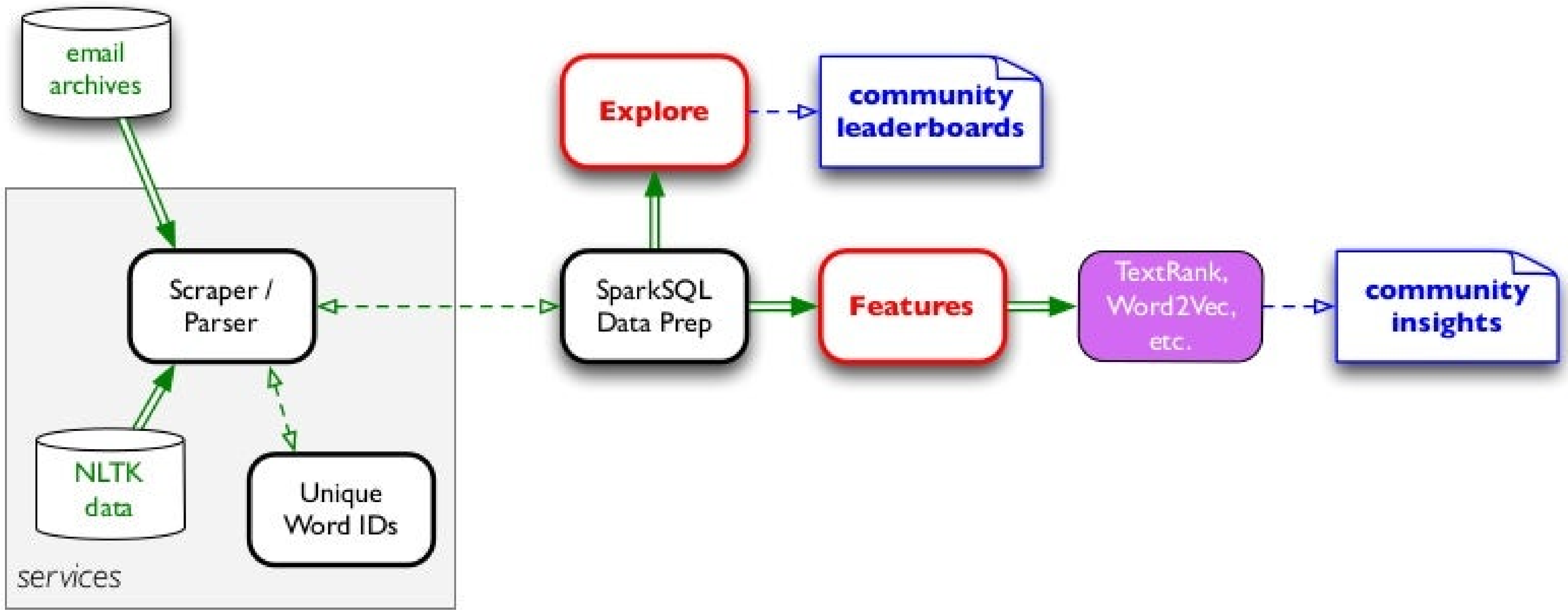
- ~2K msgs/month
- ~18 MB/month parsed in JSON

Six months' list activity represents a graph of:

- 1882 senders
- 1,762,113 nodes
- 3,232,174 edges

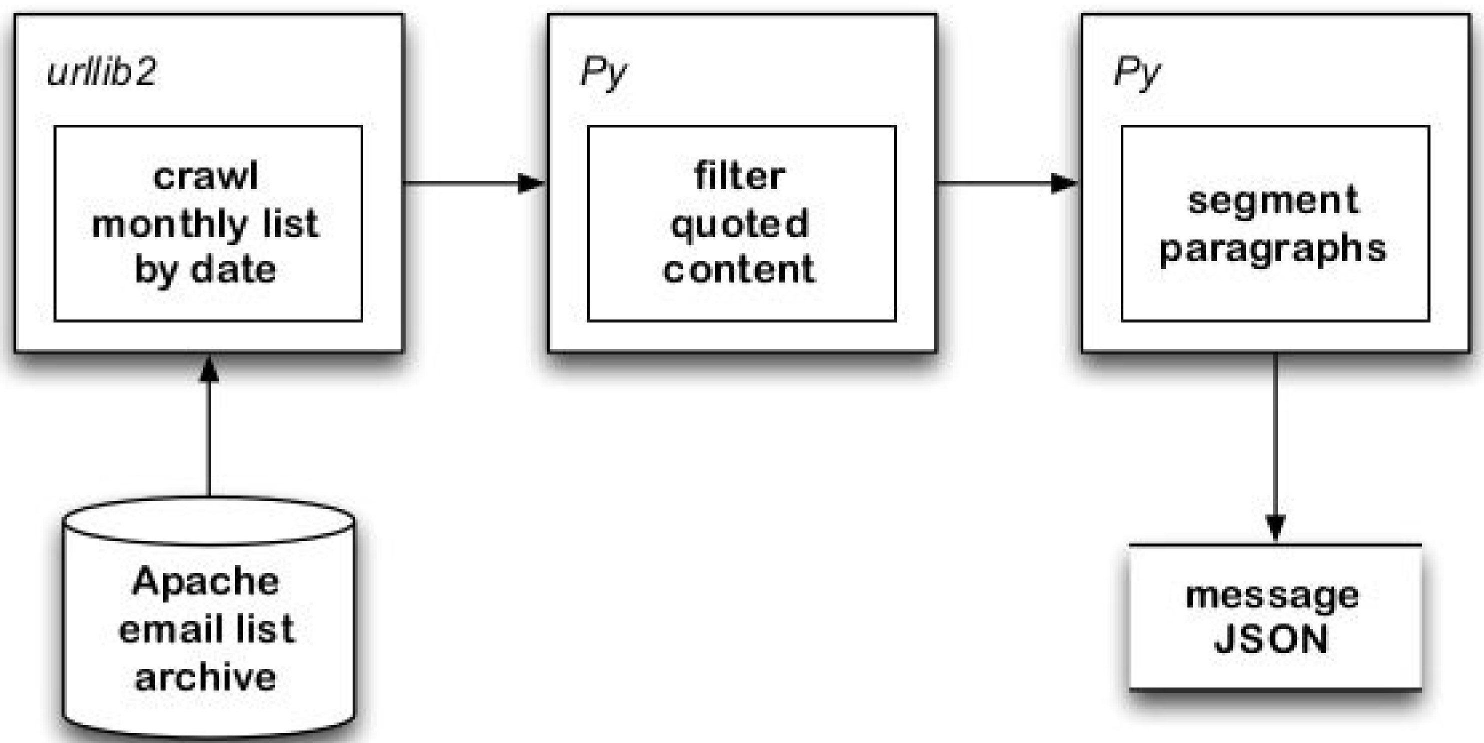
A large graph?! In any case, it satisfies definition of a *graph-parallel system* – lots of data locality to leverage

Data Prep: *Microservices meet Parallel Processing*

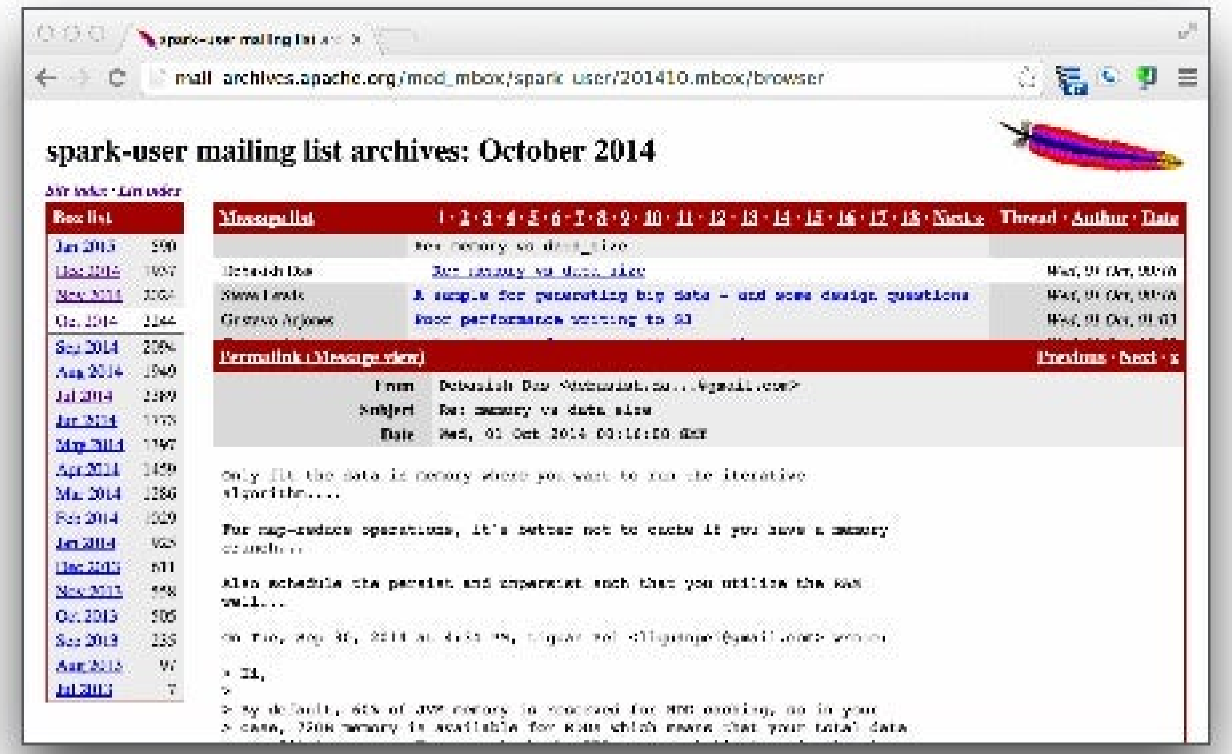


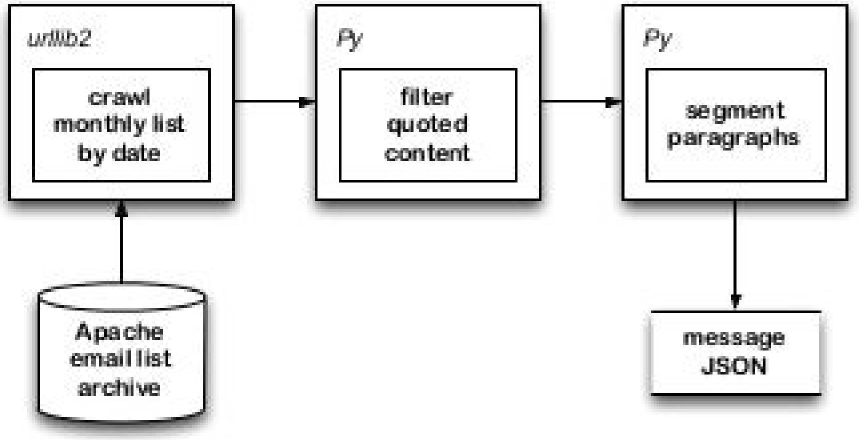
not so big data… *relatively big compute…*

Data Prep: *Scraper pipeline*



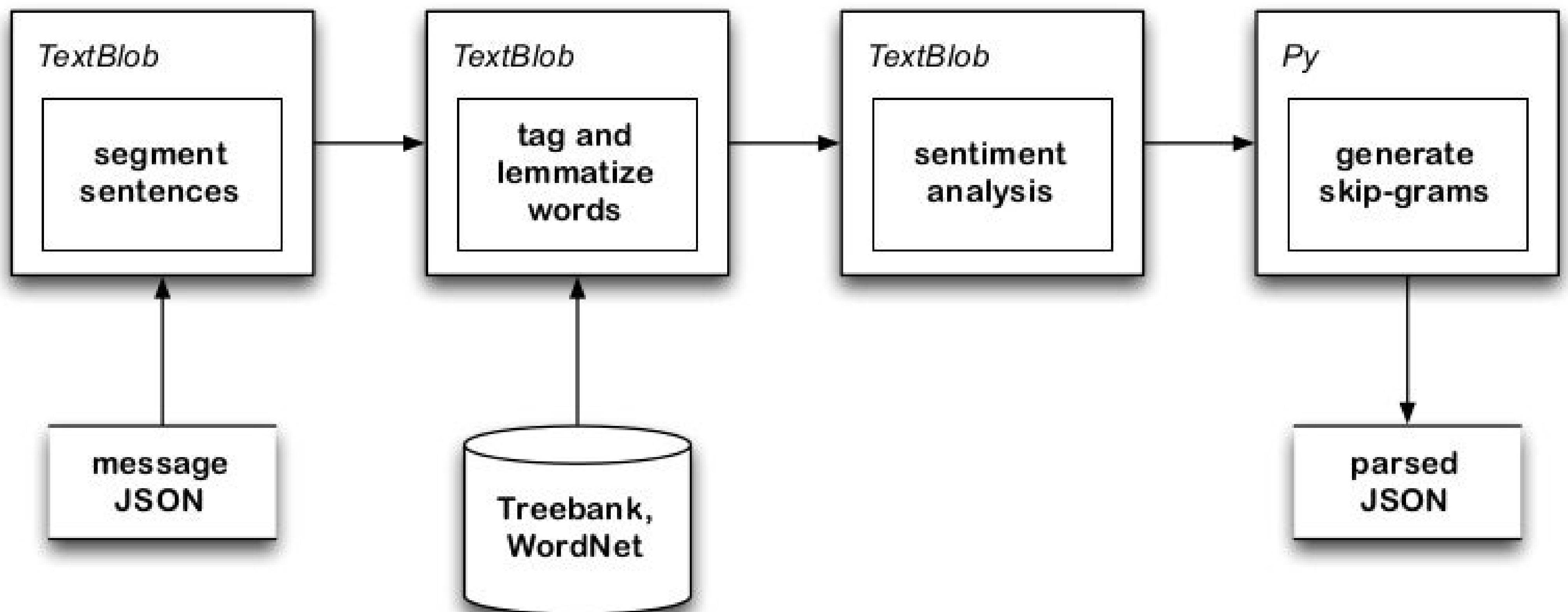
Data Prep: *Scraper pipeline*





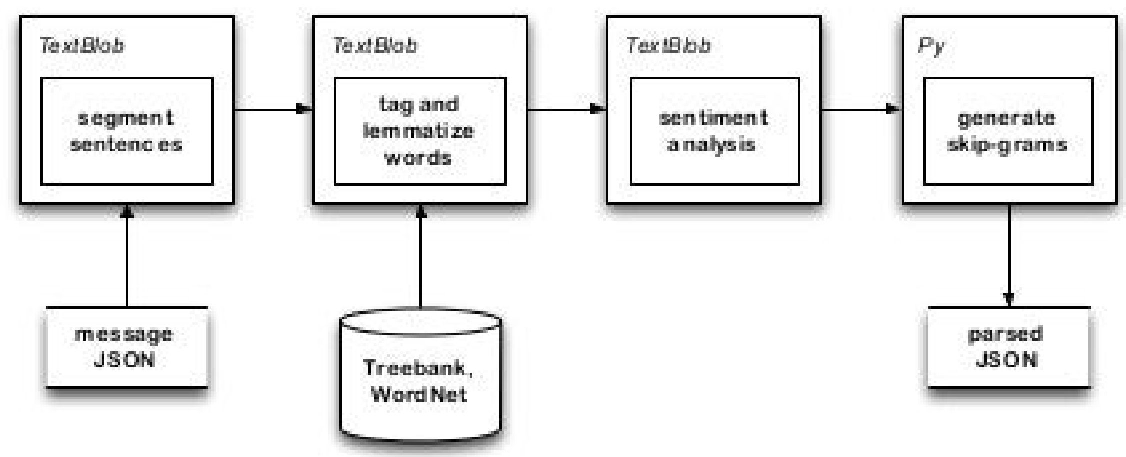

```
{
    "date": "2014-10-01T00:16:08+00:00",
    "id": "CA+B-+fyrBU1yGZAYJM_u=gnBVtzB=sXoBHkhmS-6L1n8K5Hhbw",
    "next_thread": "CALEj8eP5hpQDM=p2xryL-JT-x_VhkRcD59Q+9Qr9LJ9sYLeLVg",
    "next_url": "http://mail-archives.apache.org/mod_mbox/spark-user/201410.mbox/%3cCALEj8eP5hpQ
    "prev_thread": "",
    "sender": "Debasish Das <debasish.da...@gmail.com>",
    "subject": "Re: memory vs data_size",
    "text": "\nOnly fit the data in memory where you want to run the iterative\nalgorithm....\n\
}
```

Data Prep: *Parser pipeline*



Data Prep: *Parser pipeline*

```
{
    "date": "2014-10-01T00:16:08+00:00",
    "id": "CA+B-+fyrBU1yGZAYJM_u=gnBVtzB=sXoBHkhmS-6L1n8K5Hhbw",
    "next_thread": "CALEj8eP5hpQDM=p2xryL-JT-x_VhkRcD59Q+9Qr9LJ9sYLeLVg",
    "next_url": "http://mail-archives.apache.org/mod_mbox/spark-user/201410.mbox/%3cCALEj8eP5hpQDM=p
    "prev_thread": "",
    "sender": "Debasish Das <debasish.da...@gmail.com>",
    "subject": "Re: memory vs data_size",
    "text": "\nOnly fit the data in memory where you want to run the iterative\nalgorithm....\n\nFor
}
```

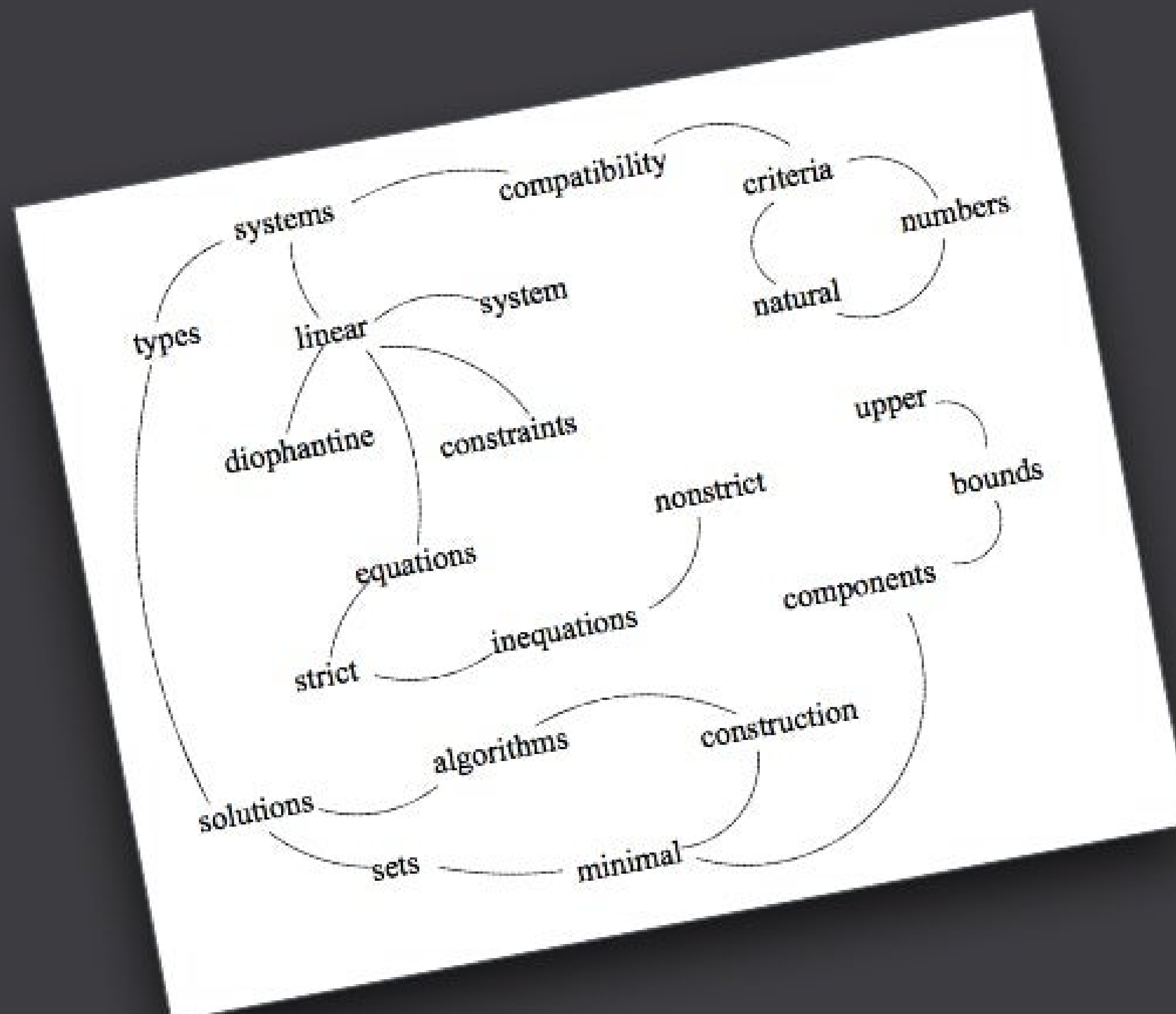



```
{
    "graf": [ [1, "Only", "only", "RB", 1, 0], [2, "fit", "fit", "VBP", 1, 1 ] ... ],
    "id": "CA+B-+fyrBU1yGZAYJM_u=gnBVtzB=sXoBHkhmS-6L1n8K5Hhbw",
    "polr": 0.2,
    "sha1": "178b7a57ec6168f20a8a4f705fb8b0b04e59eeb7",
    "size": 14,
    "subj": 0.7,
    "tile": [ [1, 2], [2, 3], [3, 4] ... ]
    ]
}
```

Data Prep: *code examples…*

Let’s check some code!

TextRank in Spark



TextRank: Bringing Order into Texts

Rada Mihalcea, Paul Tarau

Conference on Empirical Methods in Natural Language Processing (July 2004)

https://goo.gl/AJnA76

http://web.eecs.umich.edu/~mihalcea/papers.html

http://www.cse.unt.edu/~tarau/

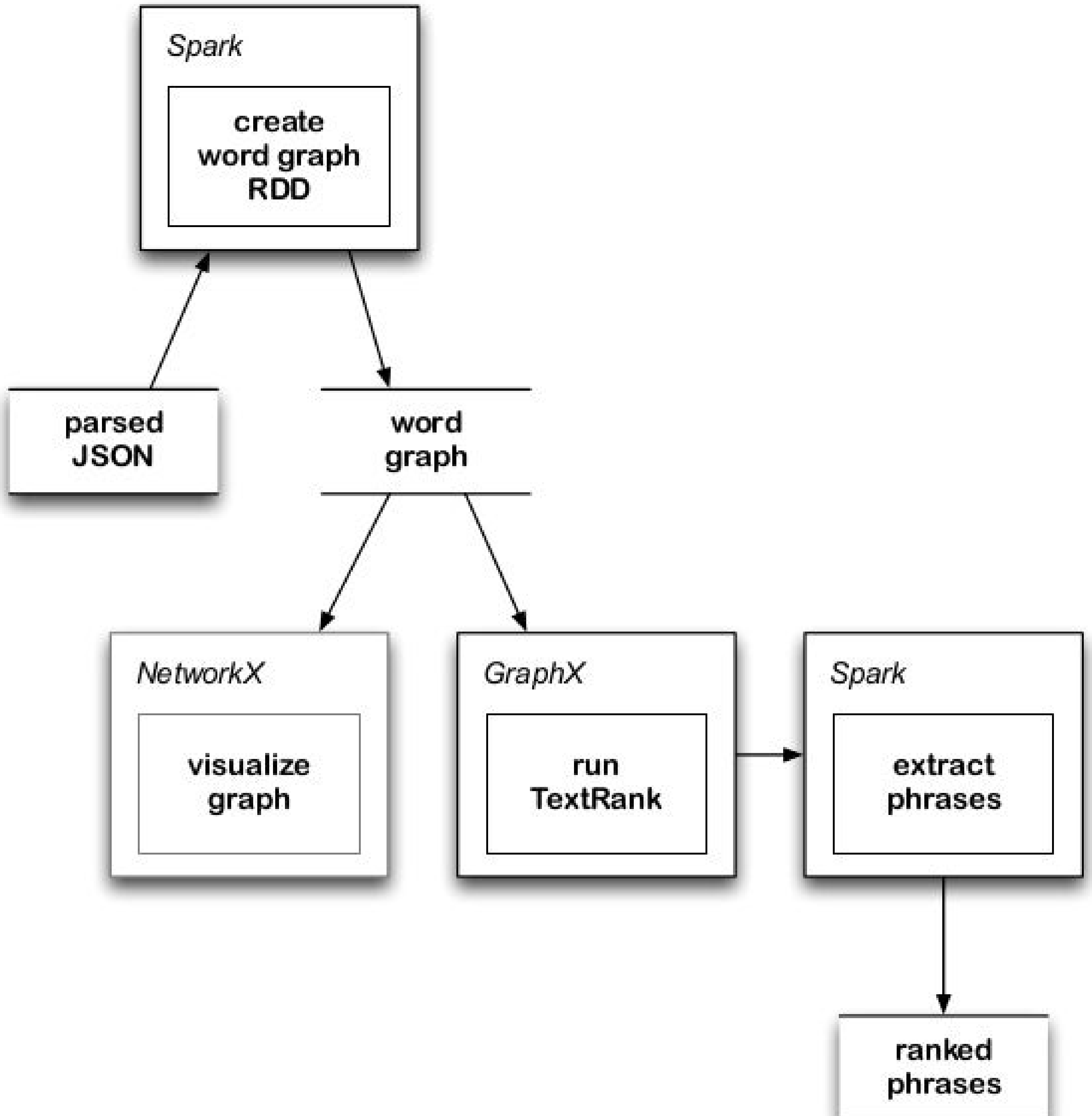
TextRank: *other implementations*

Jeff Kubina (Perl / English):
http://search.cpan.org/~kubina/Text-Categorize-Textrank-0.51/lib/Text/Categorize/Textrank/En.pm

Paco Nathan (Hadoop / English+Spanish):
https://github.com/ceteri/textrank/

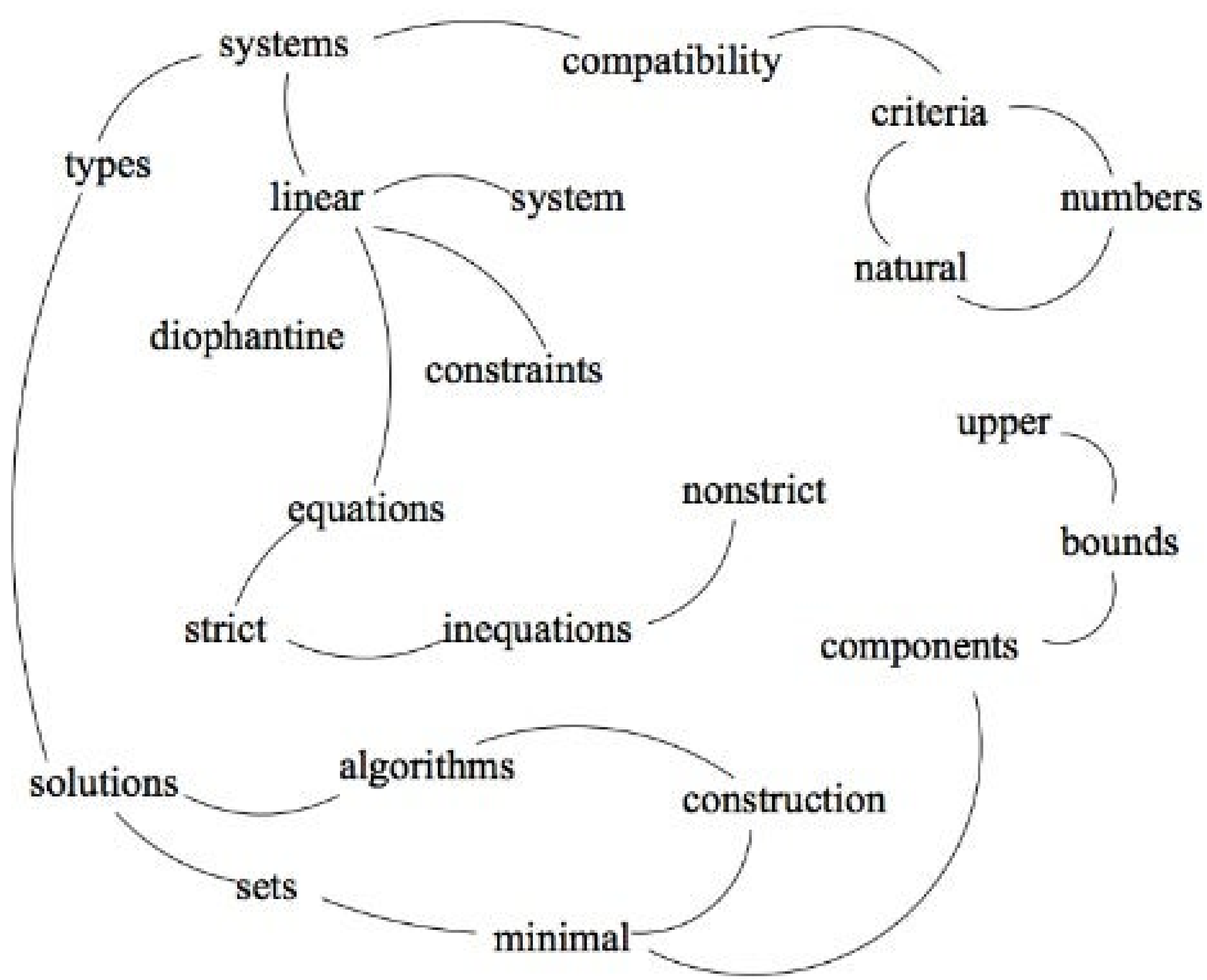
Karin Christiasen (Java / Icelandic):
https://github.com/karchr/icetextsum

TextRank: *Spark-based pipeline*



TextRank: *raw text input*

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.



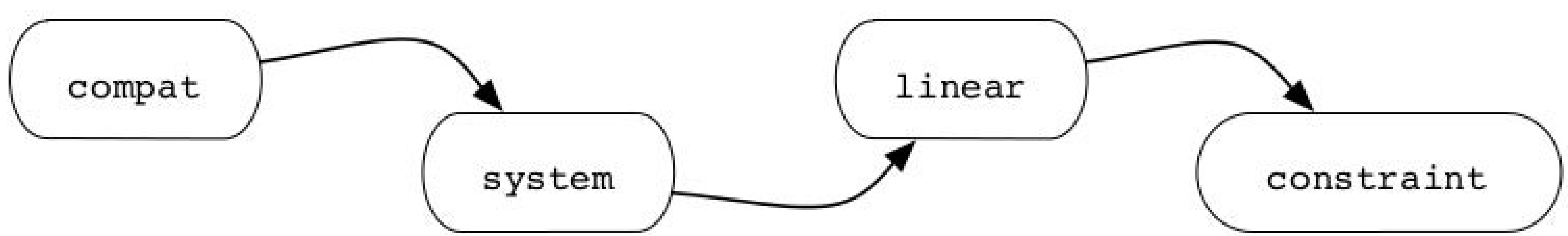
TextRank: *data results*

1: `"Compatibility of systems of linear constraints"`

2:

```
[{'index': 0, 'stem': 'compat', 'tag': 'NNP','word': 'compatibility'},
 {'index': 1, 'stem': 'of', 'tag': 'IN', 'word': 'of'},
 {'index': 2, 'stem': 'system', 'tag': 'NNS', 'word': 'systems'},
 {'index': 3, 'stem': 'of', 'tag': 'IN', 'word': 'of'},
 {'index': 4, 'stem': 'linear', 'tag': 'JJ', 'word': 'linear'},
 {'index': 5, 'stem': 'constraint', 'tag': 'NNS','word': 'constraints'}]
```

3:

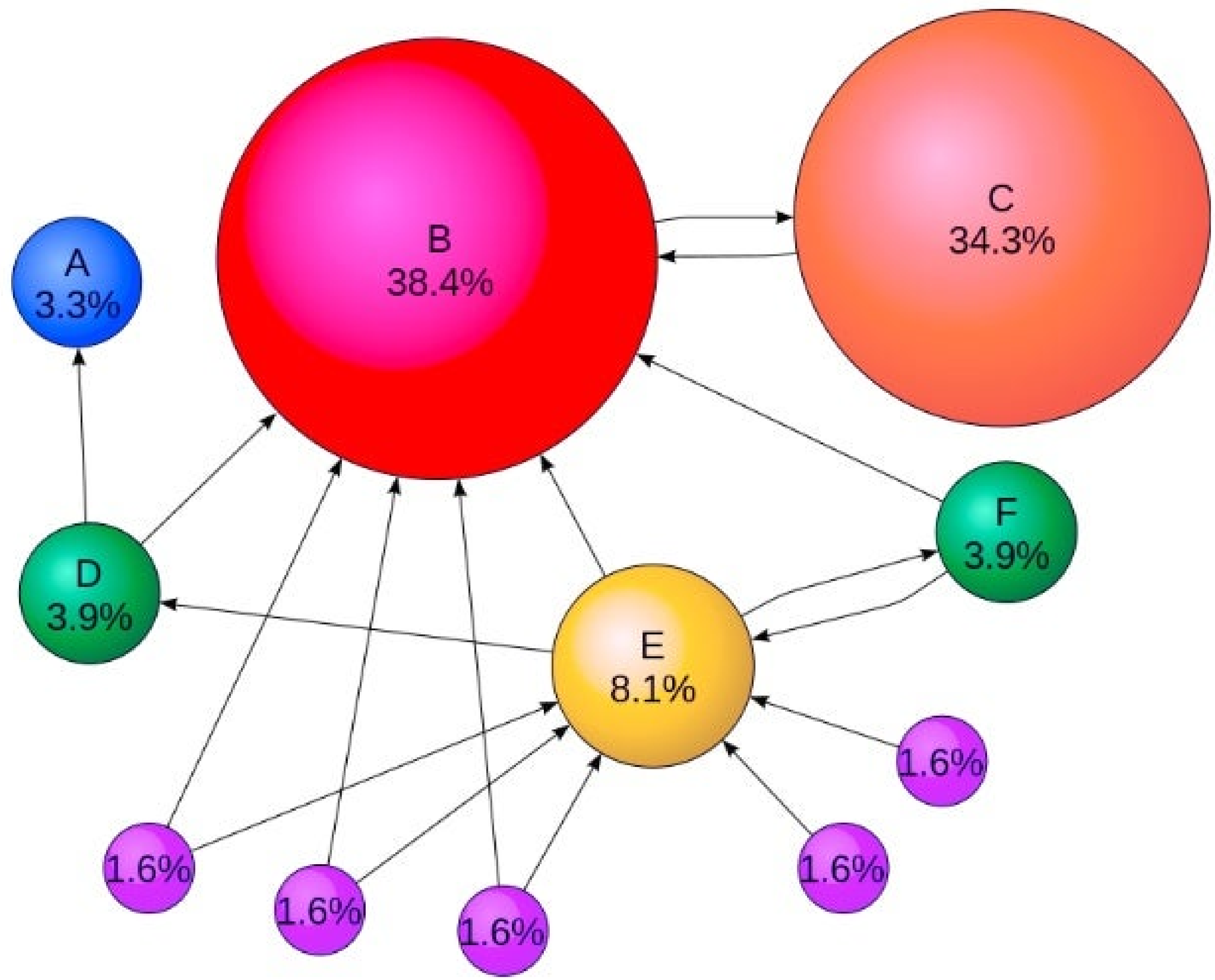


TextRank: *dependencies*

TextBlob

TextRank: *how it works*

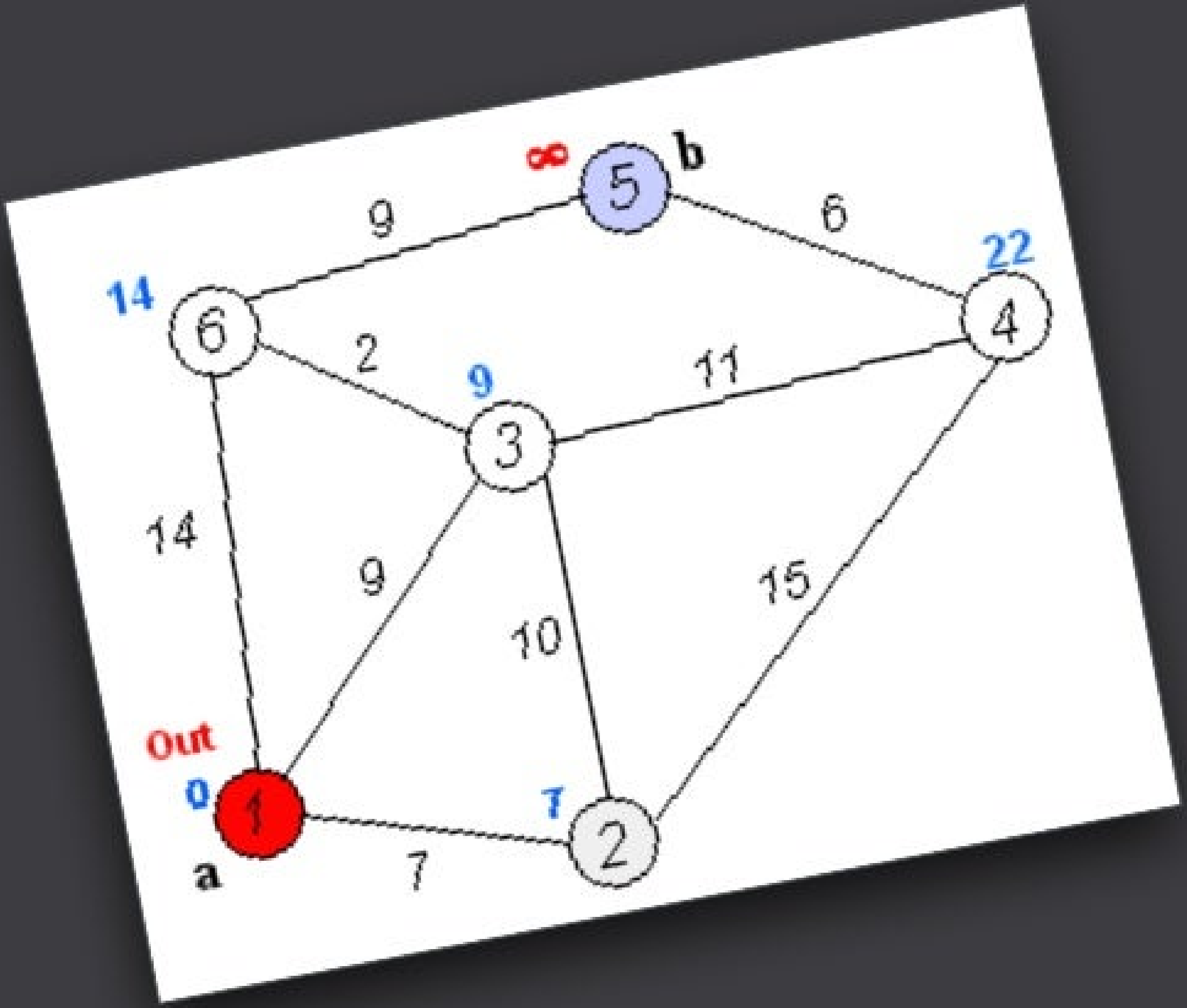
https://en.wikipedia.org/wiki/PageRank



TextRank: *code examples…*

Let's check some code!

Social Graph



Social Graph: *use GraphX to run graph analytics*

```
// run graph analytics
val g: Graph[String, Int] = Graph(nodes, edges)
val r = g.pageRank(0.0001).vertices
r.join(nodes).sortBy(_._2._1, ascending=false).foreach(println)

// define a reduce operation to compute the highest degree vertex
def max(a: (VertexId, Int), b: (VertexId, Int)): (VertexId, Int) = {
  if (a._2 > b._2) a else b
}

// compute the max degrees
val maxInDegree: (VertexId, Int)  = g.inDegrees.reduce(max)
val maxOutDegree: (VertexId, Int) = g.outDegrees.reduce(max)
val maxDegrees: (VertexId, Int)   = g.degrees.reduce(max)

// connected components
val scc = g.stronglyConnectedComponents(10).vertices
node.join(scc).foreach(println)
```

Social Graph: *PageRank of top dev@spark email, 4Q2014*

```
(389,(22.690229478710016,Sean Owen <so...@cloudera.com>))
(857,(20.832469059298248,Akhil Das <ak...@sigmoidanalytics.com>))
(652,(13.281821379806798,Michael Armbrust <mich...@databricks.com>))
(101,(9.963167550803664,Tobias Pfeiffer <...@preferred.jp>))
(471,(9.614436778460558,Steve Lewis <lordjoe2...@gmail.com>))
(931,(8.217073486575732,shahab <shahab.mok...@gmail.com>))
(48,(7.653814912512137,ll <duy.huynh....@gmail.com>))
(1011,(7.602002681952157,Ashic Mahtab <as...@live.com>))
(1055,(7.572376489758199,Cheng Lian <lian.cs....@gmail.com>))
(122,(6.87247388819558,Gerard Maas <gerard.m...@gmail.com>))
(904,(6.252657820614504,Xiangrui Meng <men...@gmail.com>))
(827,(6.0941062762076115,Jianshi Huang <jianshi.hu...@gmail.com>))
(887,(5.835053915864531,Davies Liu <dav...@databricks.com>))
(303,(5.724235650446037,Ted Yu <yuzhih...@gmail.com>))
(206,(5.430238461114108,Deep Pradhan <pradhandeep1...@gmail.com>))
(483,(5.332452537151523,Akshat Aranya <aara...@gmail.com>))
(185,(5.259438927615685,SK <skrishna...@gmail.com>))
(636,(5.235941228955769,Matei Zaharia <matei.zaha…@gmail.com>))

// seaaaaaaaaaan!
maxInDegree: (org.apache.spark.graphx.VertexId, Int) = (389,126)
maxOutDegree: (org.apache.spark.graphx.VertexId, Int) = (389,170)
maxDegrees: (org.apache.spark.graphx.VertexId, Int) = (389,296)
```

Social Graph: *code examples…*

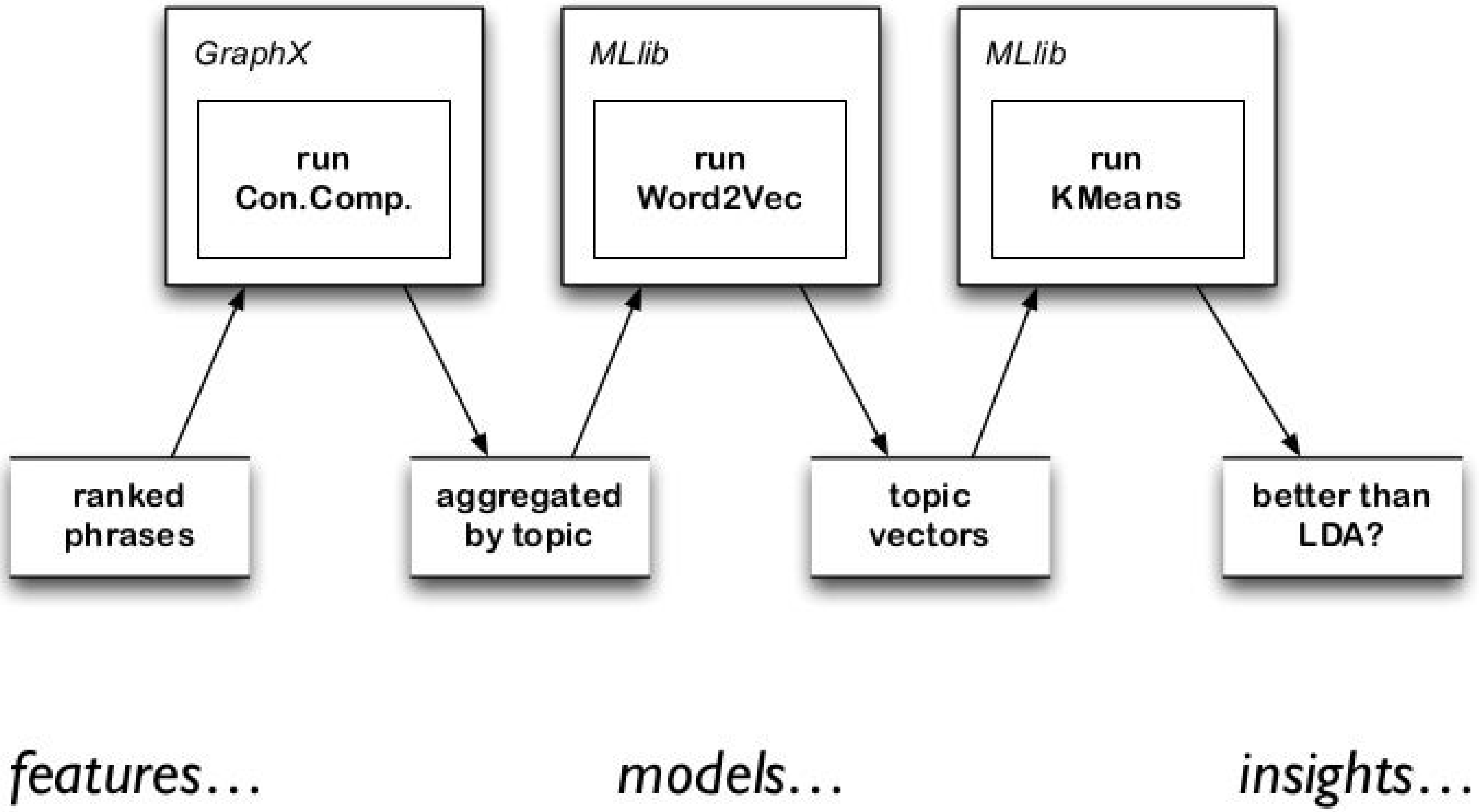
Let's check some code!

Misc., Etc., Maybe:

Feature learning with Word2Vec

Matt Krzus

www.yseam.com/blog/WV.html



Resources

Spark Developer Certification

- go.databricks.com/spark-certified-developer
- defined by Spark experts @Databricks
- assessed by O'Reilly Media
- establishes the bar for Spark expertise

Developer Certification: *Overview*

- 40 multiple-choice questions, 90 minutes
- mostly structured as choices among code blocks
- expect some Python, Java, Scala, SQL
- understand theory of operation
- identify best practices
- recognize code that is more parallel, less memory constrained

Overall, you need to write Spark apps in practice

community:

spark.apache.org/community.html

events worldwide: goo.gl/2YqJZK

YouTube channel: goo.gl/N5Hx3h

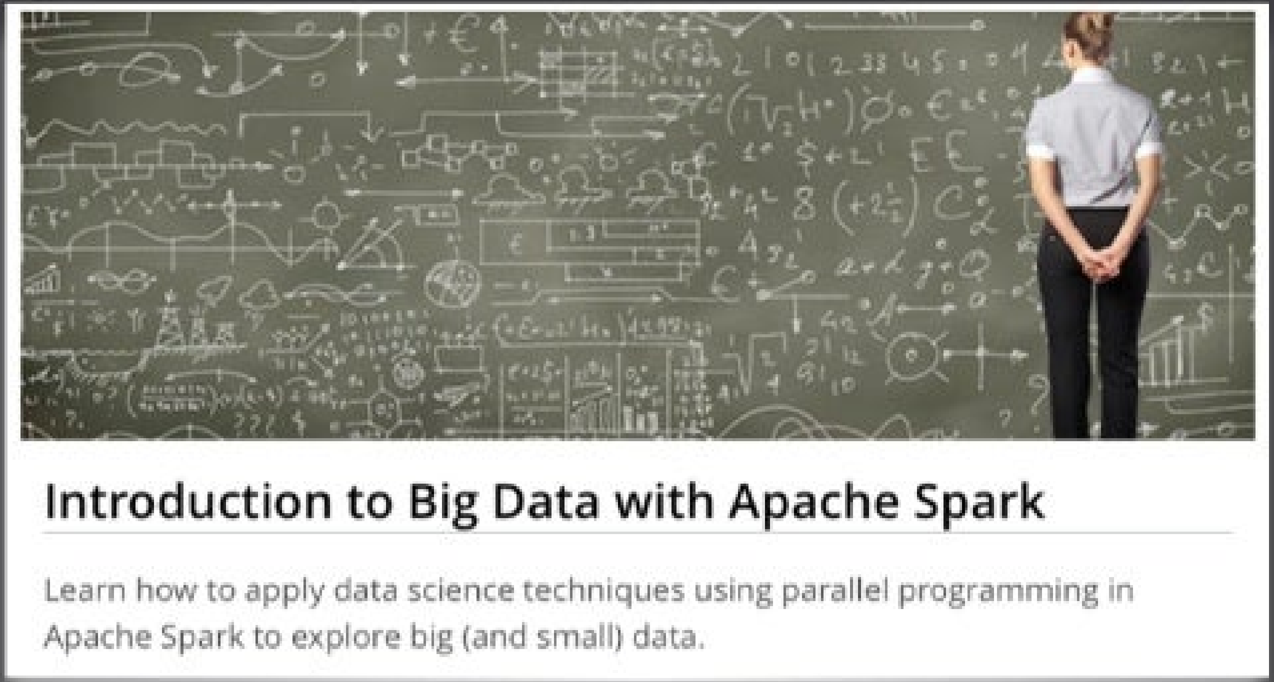
video+preso archives: spark-summit.org

resources: databricks.com/spark/developer-resources

workshops: databricks.com/spark/training

MOOCs:

Anthony Joseph
UC Berkeley
early June 2015
edx.org/course/uc-berkeleyx/uc-berkeleyx-cs100-1x-introduction-big-6181

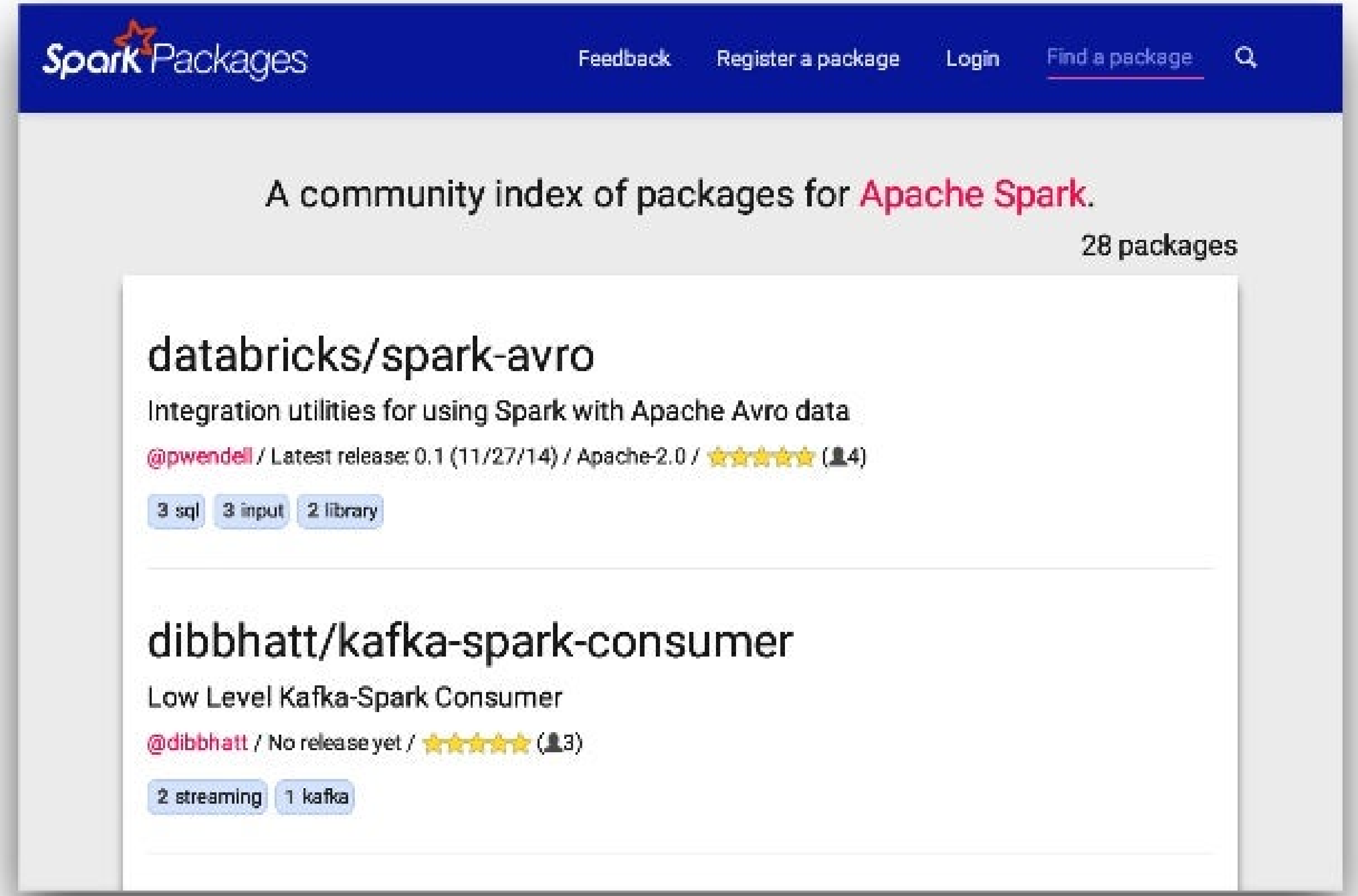


Ameet Talwalkar
UCLA
late June 2015
edx.org/course/uc-berkeleyx/uc-berkeleyx-cs190-1x-scalable-machine-6066

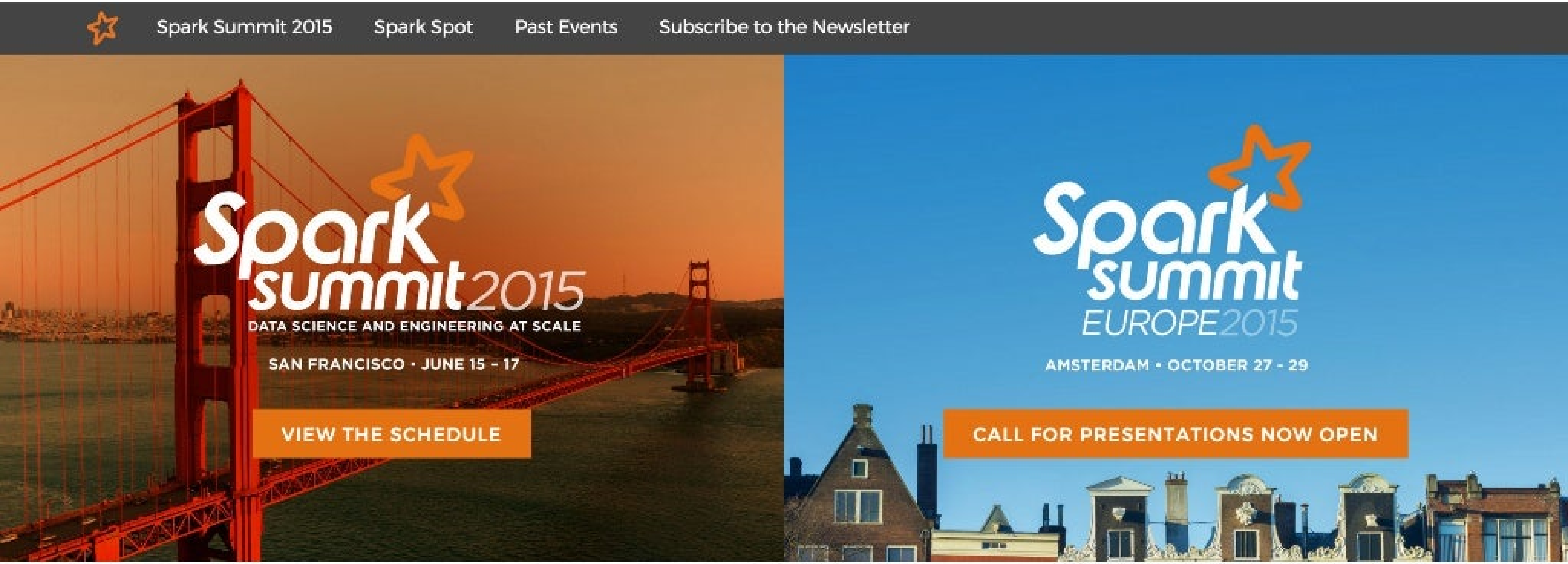
Resources: *Spark Packages*

Looking for other libraries and features? There are a variety of third-party packages available at:

http://spark-packages.org/



Resources: *Spark Summit conferences*



http://spark-summit.org/

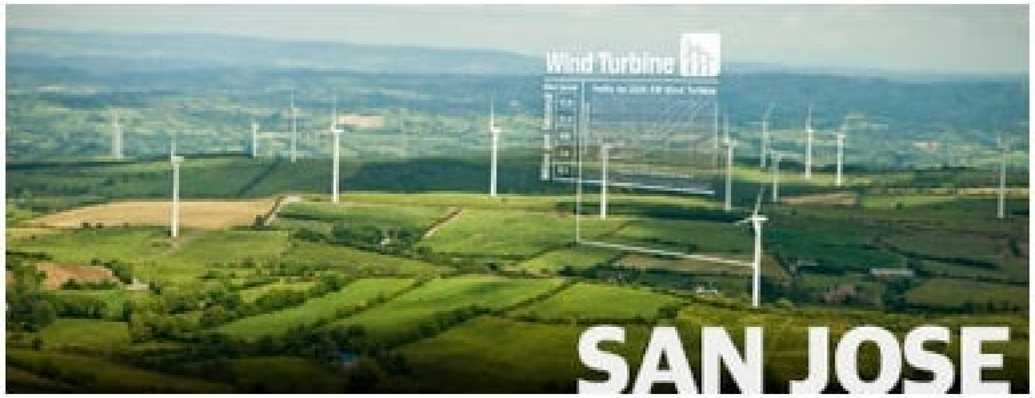
discount code ***datamining15*** *for 15% off registration*

Resources: *Strata + Hadoop World conferences*

Strata+ Hadoop WORLD

New York, NY | Sept 29–Oct 1, 2015

Singapore | December 1–3, 2015



San Jose, CA | March 29–31, 2016
Visit the Strata + Hadoop World San Jose 2015 website

SAVE THE DATE

London, UK | June 1–3, 2016
Visit the Strata + Hadoop World in London 2015 website

SAVE THE DATE

http://strataconf.com/

Resources: *O'Reilly Podcast*

O'REILLY®

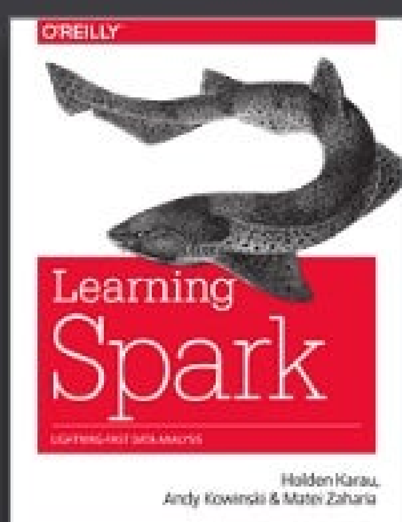
The Data Show

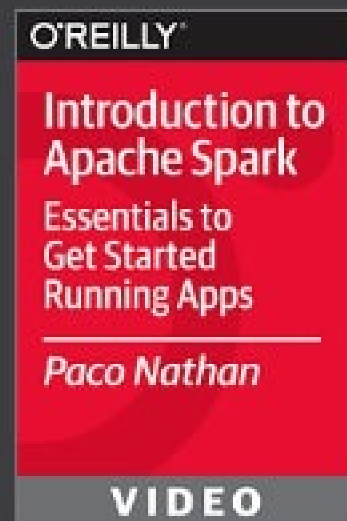
New Ideas. True Stories. Blistering Insights.

https://itunes.apple.com/podcast/id944929220

books+videos:

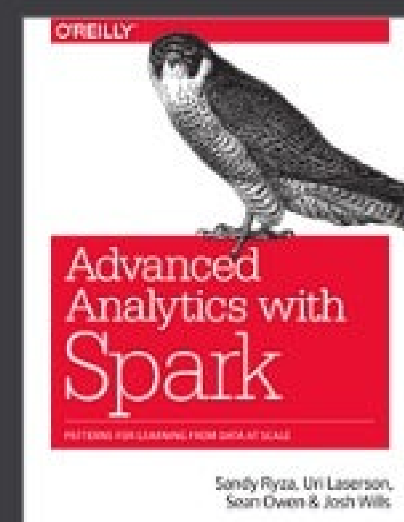
Learning Spark
Holden Karau, Andy Konwinski, Parick Wendell, Matei Zaharia
O'Reilly (2015)
shop.oreilly.com/product/0636920028512.do

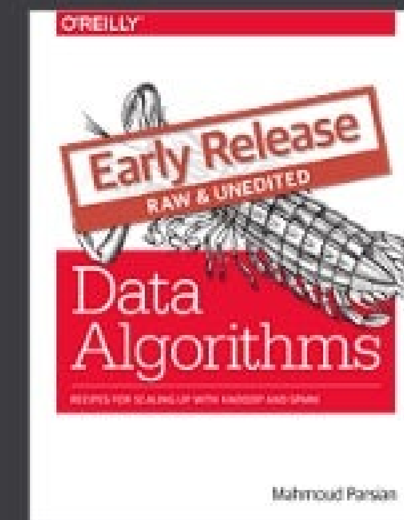




Intro to Apache Spark
Paco Nathan
O'Reilly (2015)
shop.oreilly.com/product/0636920036807.do

Advanced Analytics with Spark
Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills
O'Reilly (2015)
shop.oreilly.com/product/0636920035091.do



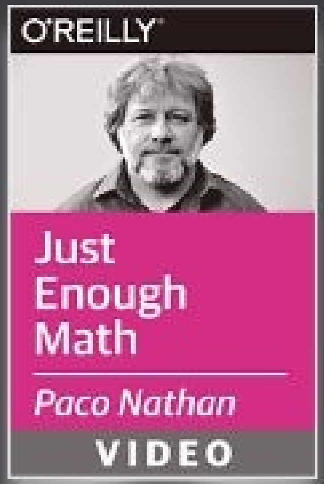


Data Algorithms
Mahmoud Parsian
O'Reilly (2014)
shop.oreilly.com/product/0636920033950.do

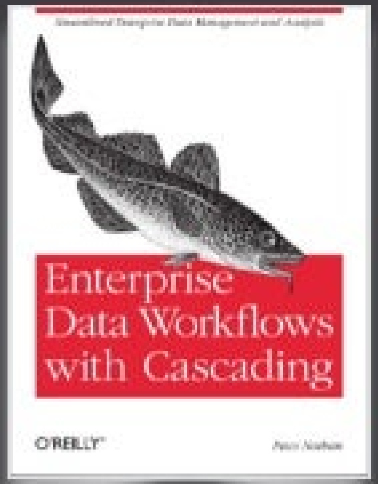
presenter:

monthly newsletter for updates, events, conf summaries, etc.:

liber118.com/pxn/



Just Enough Math
O'Reilly (2014)
justenoughmath.com
preview: youtu.be/TQ58cWgdCpA



Enterprise Data Workflows with Cascading
O'Reilly (2013)
shop.oreilly.com/product/0636920028536.do