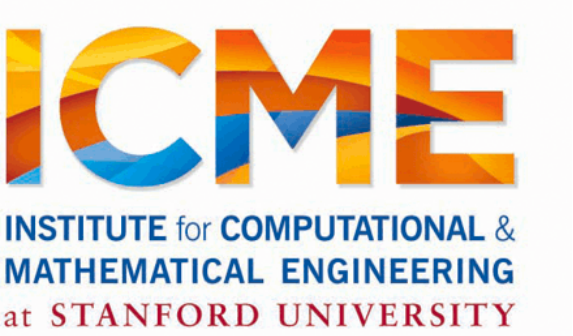
Distributed Machine Learning on Spark

Reza Zadeh



@Reza_Zadeh | http://reza-zadeh.com

Outline

Data flow vs. traditional network programming

Spark computing engine

Optimization Example

Matrix Computations

MLlib + {Streaming, GraphX, SQL}

Future of MLlib

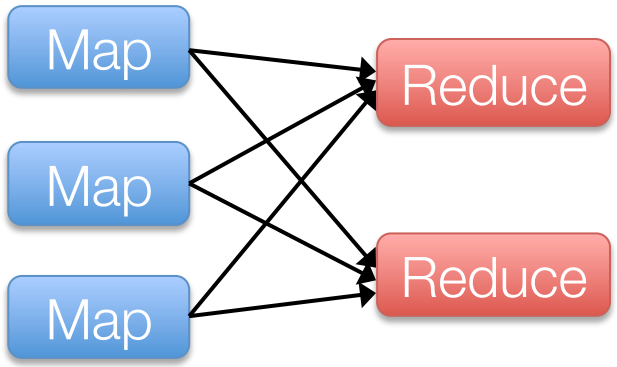
Data Flow Models

Restrict the programming interface so that the system can do more automatically

Express jobs as graphs of high-level operators

- » System picks how to split each operator into tasks and where to run each task
- » Run parts twice fault recovery

Biggest example: MapReduce



Spark Computing Engine

Extends a programming language with a distributed collection data-structure

- » “Resilient distributed datasets” (RDD)

Open source at Apache

- » Most active community in big data, with 50+ companies contributing

Clean APIs in Java, Scala, Python

Community: SparkR

Key Idea

Resilient Distributed Datasets (RDDs)

- » Collections of objects across a cluster with user controlled partitioning & storage (memory, disk, ...)
- » Built via parallel transformations (map, filter, …)
- » The world only lets you make make RDDs such that they can be:

Automatically rebuilt on failure

MLlib History

MLlib is a Spark subproject providing machine learning primitives

Initial contribution from AMPLab, UC Berkeley

Shipped with Spark since Sept 2013

MLlib: Available algorithms

classification: logistic regression, linear SVM, naïve Bayes, least squares, classification tree

regression: generalized linear models (GLMs), regression tree

collaborative filtering: alternating least squares (ALS), non-negative matrix factorization (NMF)

clustering: k-means||

decomposition: SVD, PCA

optimization: stochastic gradient descent, L-BFGS

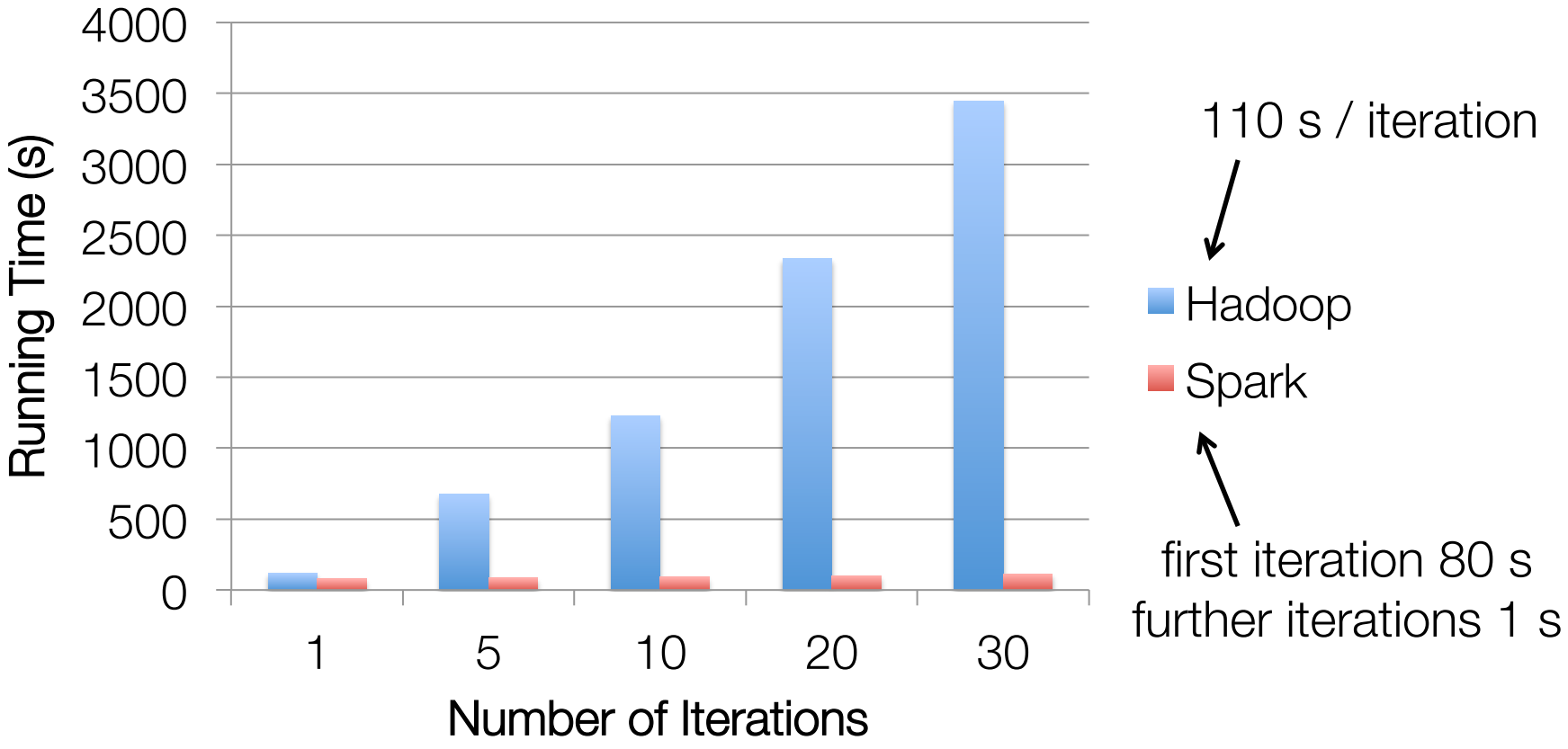
Optimization Example

Logistic Regression

$$w \leftarrow w - \alpha \cdot \sum_{i=1}^{n} g(w; x_i, y_i)$$

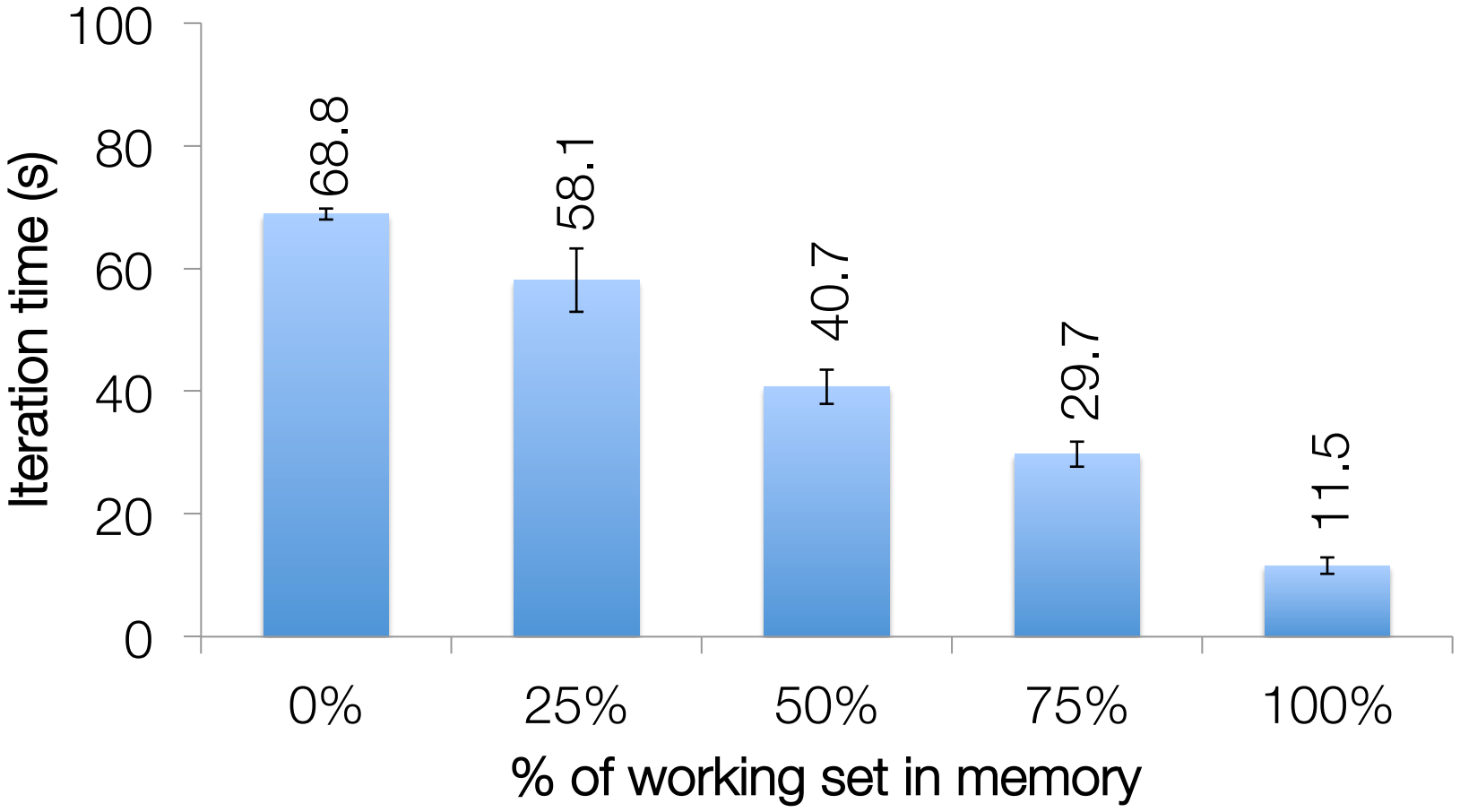
```
val points = spark.textFile(...).map(parsePoint).cache()
var w = Vector.zeros(d)
for (i <- 1 to numIterations) {
  val gradient = points.map { p =>
    (1 / (1 + exp(-p.y * w.dot(p.x)) - 1) * p.y * p.x
  ).reduce(_ + _)
  w -= alpha * gradient
}
```

Logistic Regression Results



100 GB of data on 50 m1.xlarge EC2 machines

Behavior with Less RAM



Optimization

At least two large classes of optimization problems humans can solve:

» Convex Programs

» Spectral Problems

Distributing Matrix Computations

Distributing Matrices

How to distribute a matrix across machines?

» By Entries (CoordinateMatrix)

» By Rows (RowMatrix)

» By Blocks (BlockMatrix) As of version 1.3

All of Linear Algebra to be rebuilt using these partitioning schemes

Distributing Matrices

Even the simplest operations require thinking about communication e.g. multiplication

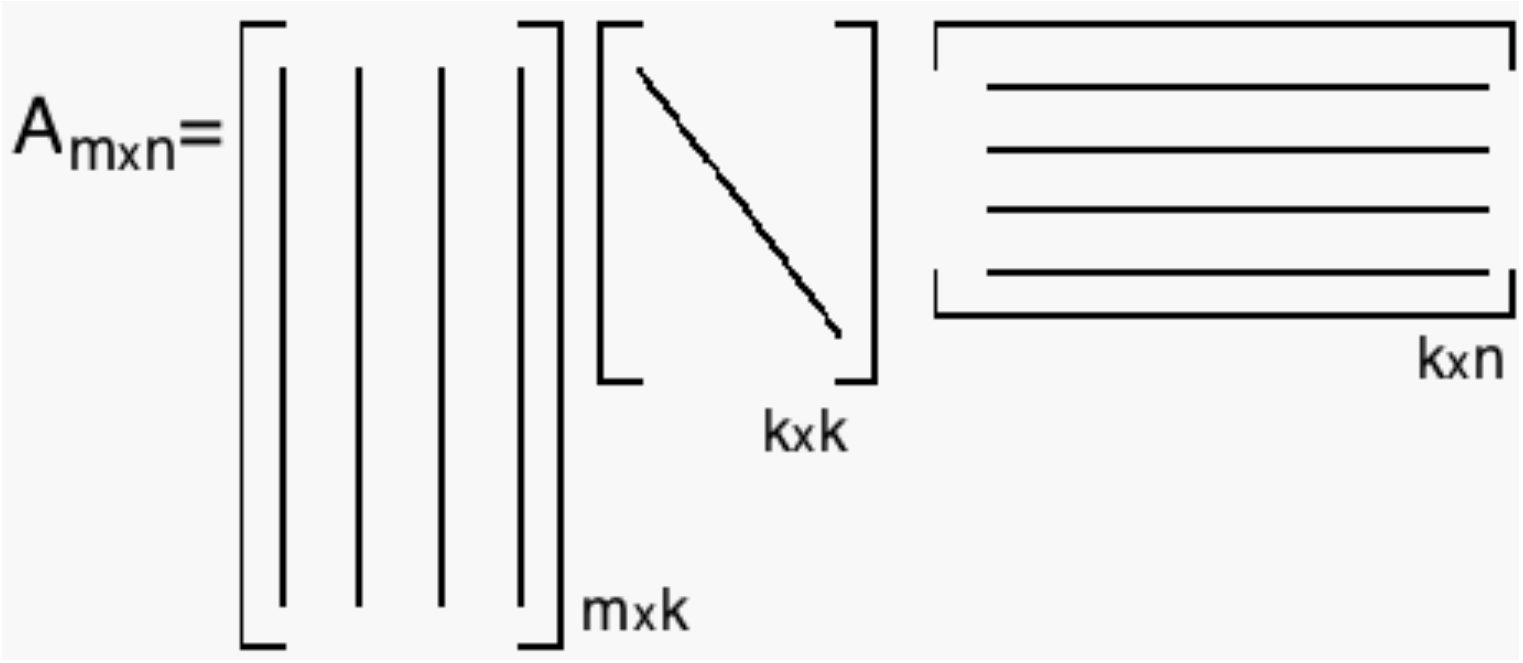
How many different matrix multiplies needed?

» At least one per pair of {Coordinate, Row, Block, LocalDense, LocalSparse} = 10

» More because multiplies not commutative

Singular Value Decomposition on Spark

Singular Value Decomposition



Singular Value Decomposition

Two cases

» Tall and Skinny

» Short and Fat (not really)

» Roughly Square

SVD method on RowMatrix takes care of which one to call.

Tall and Skinny SVD

- Given $m \times n$ matrix A, with $m \gg n$.
- We compute $A^T A$.
- $A^T A$ is $n \times n$, considerably smaller than A.
- $A^T A$ is dense.
- Holds dot products between all pairs of columns of A.

$$A = U\Sigma V^T \qquad\qquad A^T A = V\Sigma^2 V^T$$

Tall and Skinny SVD

$$A^T A = V \Sigma^2 V^T$$

Gets us V and the singular values

$$A = U \Sigma V^T$$

Gets us U by one matrix multiplication

Square SVD

ARPACK: Very mature Fortran77 package for computing eigenvalue decompositions

JNI interface available via netlib-java

Distributed using Spark – how?

Square SVD via ARPACK

Only interfaces with distributed matrix via matrix-vector multiplies

$$K_n = \begin{bmatrix} b & Ab & A^2b & \cdots & A^{n-1}b \end{bmatrix}$$

The result of matrix-vector multiply is small.

The multiplication can be distributed.

Square SVD

Matrix size	Number of nonzeros	Time per iteration (s)	Total time (s)
23,000,000 x 38,000	51,000,000	0.2	10
63,000,000 x 49,000	440,000,000	1	50
94,000,000 x 4,000	1,600,000,000	0.5	50

With 68 executors and 8GB memory in each, looking for the top 5 singular vectors

Communication-Efficient $A^T A$

All pairs similarity on Spark (DIMSUM)

All pairs Similarity

All pairs of cosine scores between n vectors

» Don’t want to brute force (n choose 2) m

» Essentially computes $A^T A$

Compute via DIMSUM

» Dimension Independent Similarity Computation using MapReduce

Intuition

Sample columns that have many non-zeros with lower probability.

On the flip side, columns that have fewer non-zeros are sampled with higher probability.

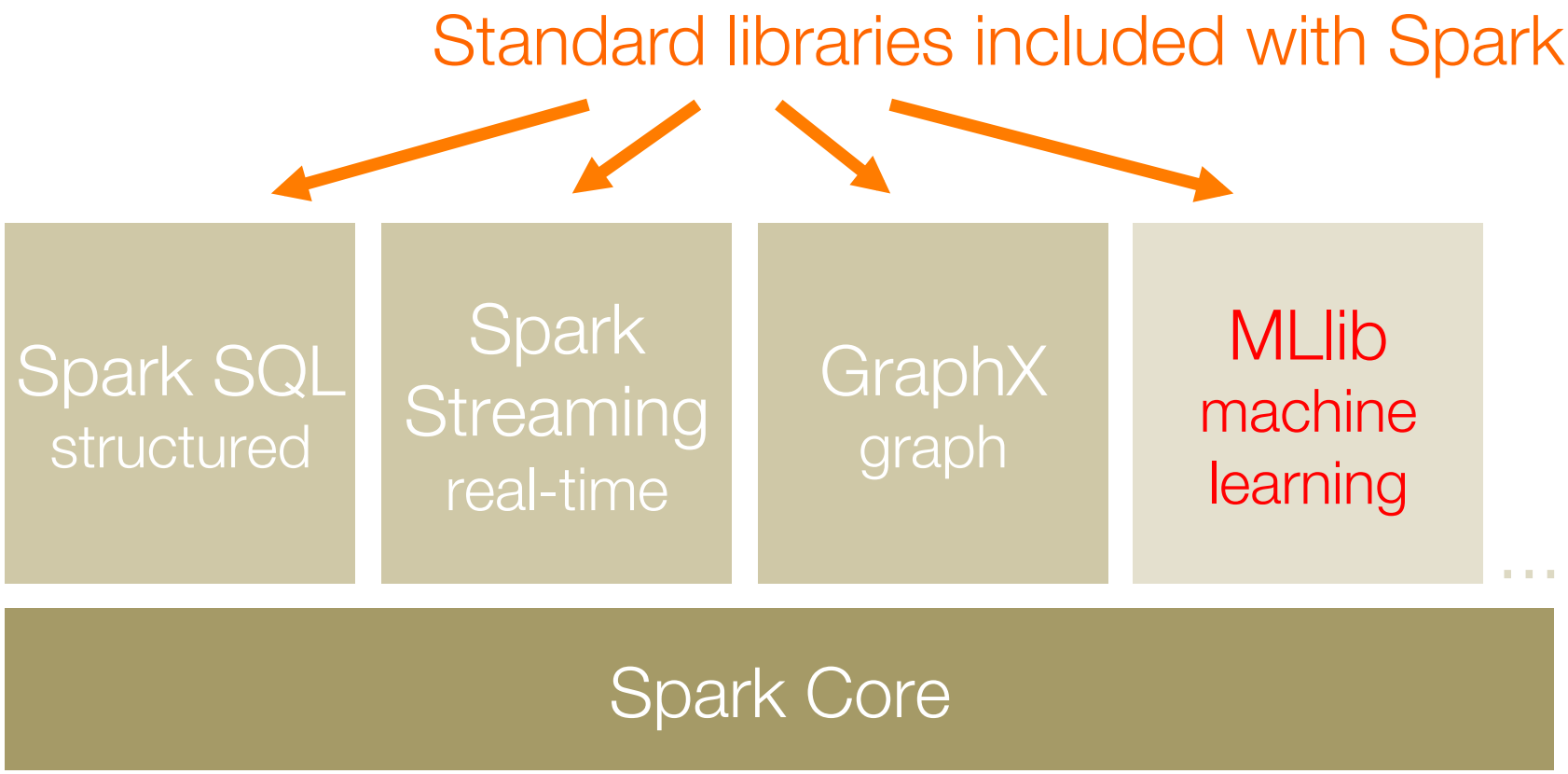
Results provably correct and independent of larger dimension, m.

Spark implementation

```scala
// Load and parse the data file.
val rows = sc.textFile(filename).map { line =>
  val values = line.split(' ').map(_.toDouble)
  Vectors.dense(values)
}
val mat = new RowMatrix(rows)

// Compute similar columns perfectly, with brute force.
val simsPerfect = mat.columnSimilarities()

// Compute similar columns with estimation using DIMSUM
val simsEstimate = mat.columnSimilarities(threshold)
```

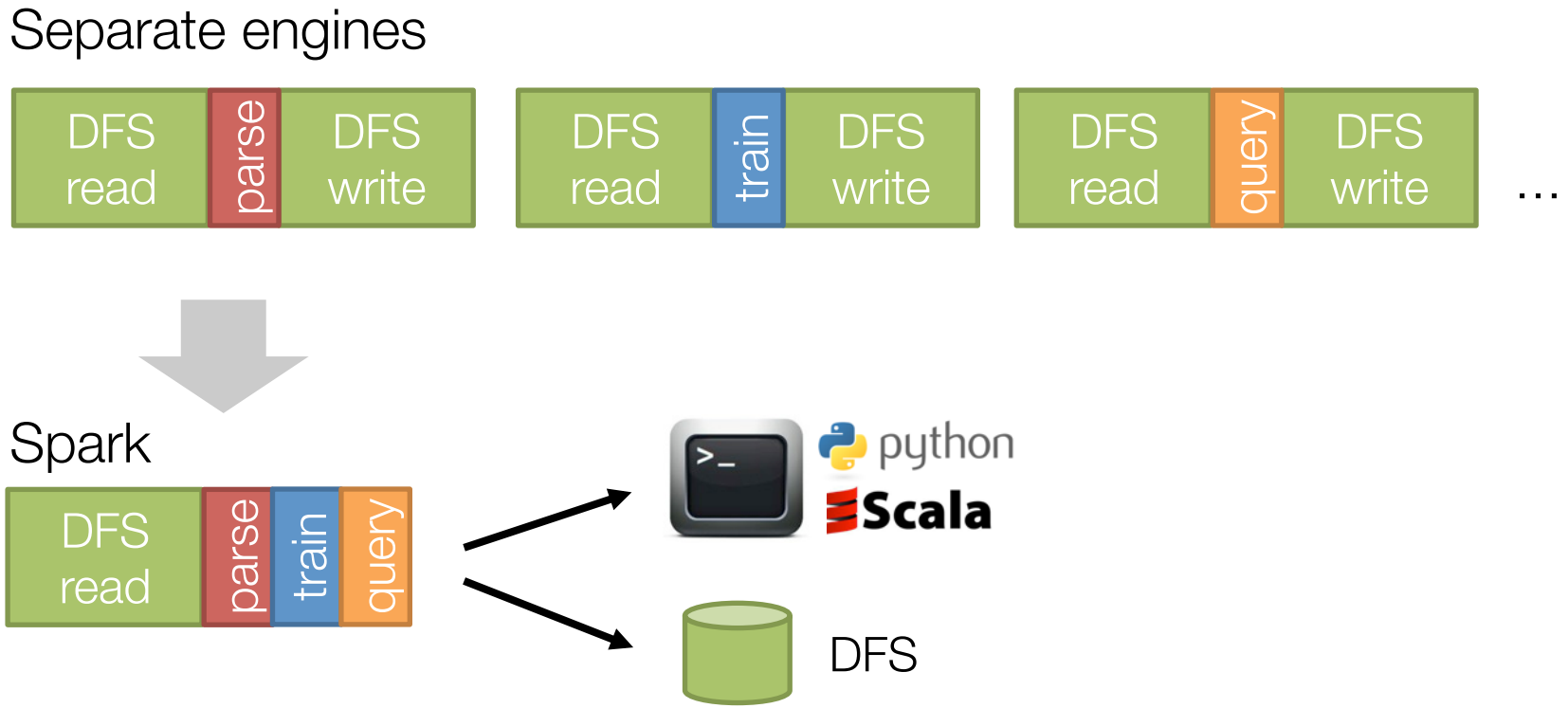
MLlib + {Streaming, GraphX, SQL}

A General Platform



Benefit for Users

Same engine performs data extraction, model training and interactive queries



MLlib + Streaming

As of Spark 1.1, you can train linear models in a streaming fashion

Model weights are updated via SGD, thus amenable to streaming

More work needed for decision trees

MLlib + SQL

```
points = context.sql(“select latitude, longitude from tweets”)
model = KMeans.train(points, 10)
```

MLlib + GraphX

```
// assemble link graph
val graph = Graph(pages, links)
val pageRank: RDD[(Long, Double)] = graph.staticPageRank(10).vertices

// load page labels (spam or not) and content features
val labelAndFeatures: RDD[(Long, (Double, Seq((Int, Double))))] = ...
val training: RDD[LabeledPoint] =
  labelAndFeatures.join(pageRank).map {
    case (id, ((label, features), pageRank)) =>
      LabeledPoint(label, Vectors.sparse(features ++ (1000, pageRank))
}

// train a spam detector using logistic regression
val model = LogisticRegressionWithSGD.train(training)
```

Future of MLlib

Research Goal: General Distributed Optimization

Distribute CVX by backing CVXPY with PySpark

Easy-to-express distributable convex programs

Need to know less math to optimize complicated objectives

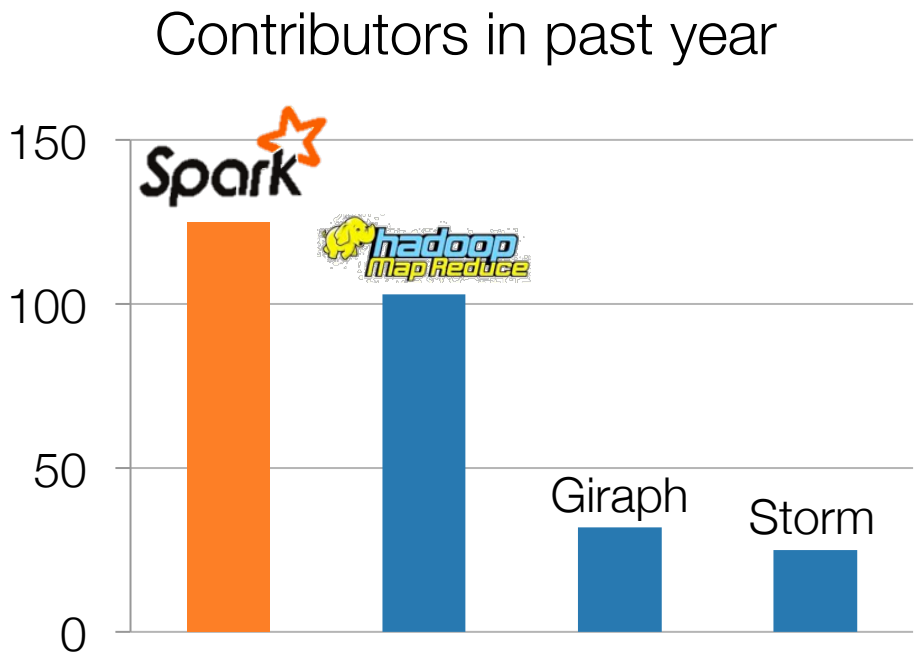
```
from cvxpy import *

# Create two scalar optimization variables.
x = Variable()
y = Variable()

# Create two constraints.
constraints = [x + y == 1,
               x - y >= 1]

# Form objective.
obj = Minimize(square(x - y))

# Form and solve problem.
prob = Problem(obj, constraints)
prob.solve()  # Returns the optimal value.
print "status:", prob.status
print "optimal value", prob.value
print "optimal var", x.value, y.value
```

```
status: optimal
optimal value 0.999999989323
optimal var 0.999999998248 1.75244914951e-09
```

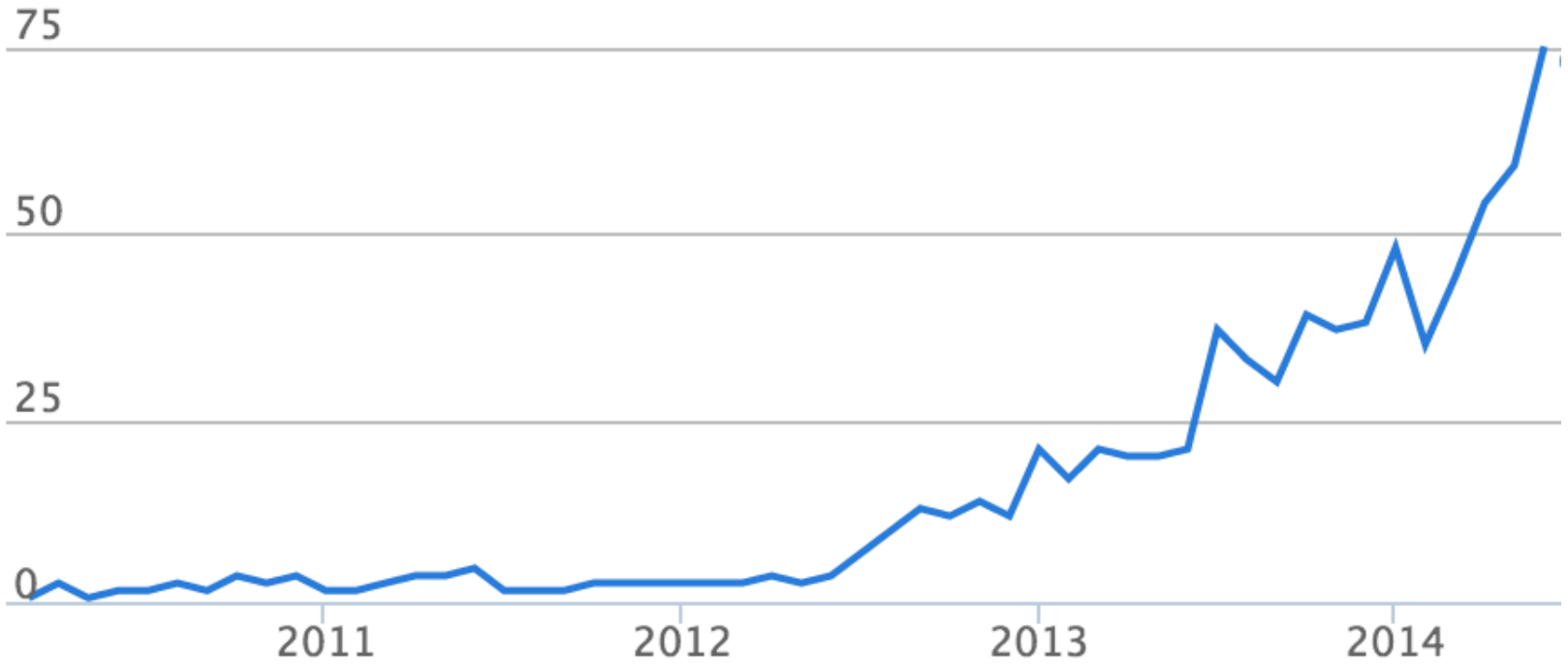
Spark Community

Most active open source community in big data

200+ developers, 50+ companies contributing



Continuing Growth



Contributors per month to Spark

source: ohloh.net

Spark and ML

Spark has all its roots in research, so we hope to keep incorporating new ideas!