# Apache Spark MLlib's past trajectory and new directions

Joseph K. Bradley

Spark Summit 2017

databricks

# About me

Software engineer at Databricks

Apache Spark committer & PMC member

Ph.D. Carnegie Mellon in Machine Learning

# About Databricks

## TEAM

Started Spark project (now Apache Spark) at UC Berkeley in 2009

## MISSION

Making Big Data Simple

## PRODUCT

Unified Analytics Platform

# Try Apache Spark in Databricks!

**UNIFIED ANALYTICS PLATFORM**

- Collaborative cloud environment
- Free version (community edition)

**DATABRICKS RUNTIME 3.0**

- Apache Spark - optimized for the cloud
- Caching and optimization layer - DBIO
- Enterprise security - DBES

Try for free today.
**databricks.com**

# 5 years ago…

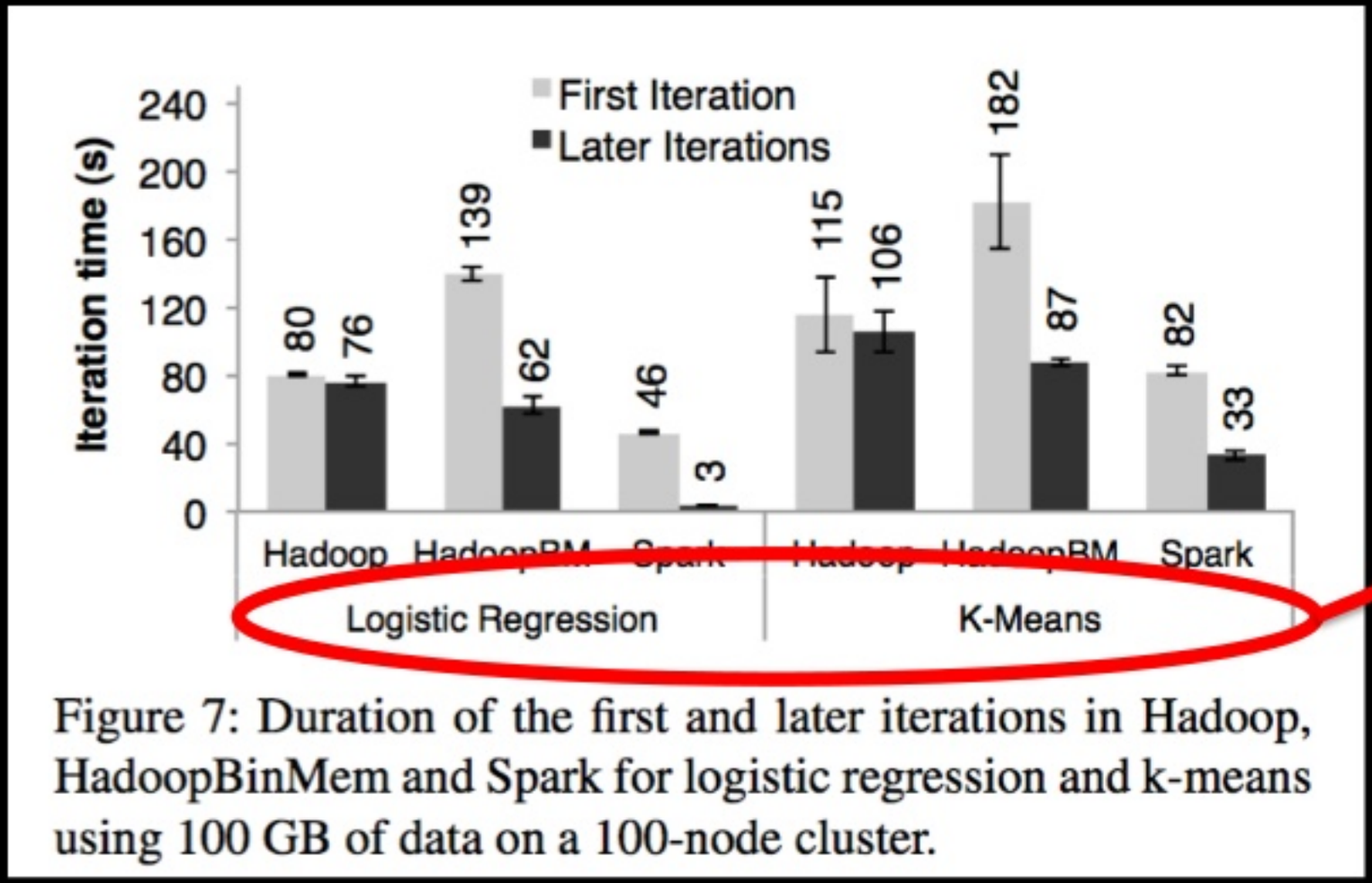



Figure 7: Duration of the first and later iterations in Hadoop, HadoopBinMem and Spark for logistic regression and k-means using 100 GB of data on a 100-node cluster.

*Zaharia et al. NSDI, 2012*

Scalable ML using Spark!

# 3 ½ years ago until today

## People added some algorithms…

**Classification**
- Logistic regression & linear SVMs
  - L1, L2, or elastic net regularization
- Decision trees
- Random forests
- Gradient-boosted trees
- Naive Bayes
- Multilayer perceptron
- One-vs-rest
- Streaming logistic regression

**Statistics**
- Pearson correlation
- Spearman correlation
- Online summarization
- Chi-squared test
- Kernel density estimation

**Frequent itemsets**
- FP-growth
- Prefix span

**Linear Algebra**
- Local & distributed dense & sparse matrices
- Matrix decompositions (PCA, SVD, QR)

**Regression**
- Least squares
  - L1, L2, or elastic net regularization
- Decision trees
- Random forests
- Gradient-boosted trees
- Isotonic regression
- Streaming least squares

**Recommendation**
- Alternating Least Squares (ALS)

**Feature extraction, transformation & selection**
- Binarizer
- Bucketizer
- Chi-Squared selection
- CountVectorizer
- Discrete cosine transform
- ElementwiseProduct
- Hashing term frequency
- Inverse document frequency
- MinMaxScaler
- Ngram
- Normalizer
- VectorAssembler
- VectorIndexer
- VectorSlicer
- Word2Vec

**Clustering**
- Gaussian mixture model
- K-Means
- Streaming K-Means
- Latent Dirichlet Allocatio
- Power Iteration Clusterin

# Today

Apache Spark is widely used for ML.

- Many production use cases
- 1000s of commits
- 100s of contributors
- 10,000s of users (on Databricks alone)










# This talk

**What?**

How has MLlib developed, and what has it become?

**So what?**

What are the implications for the dev & user communities?

**Now what?**

Where could or should it go?

# What?

How has MLlib developed, and what has it become?

# Let’s do some data analytics

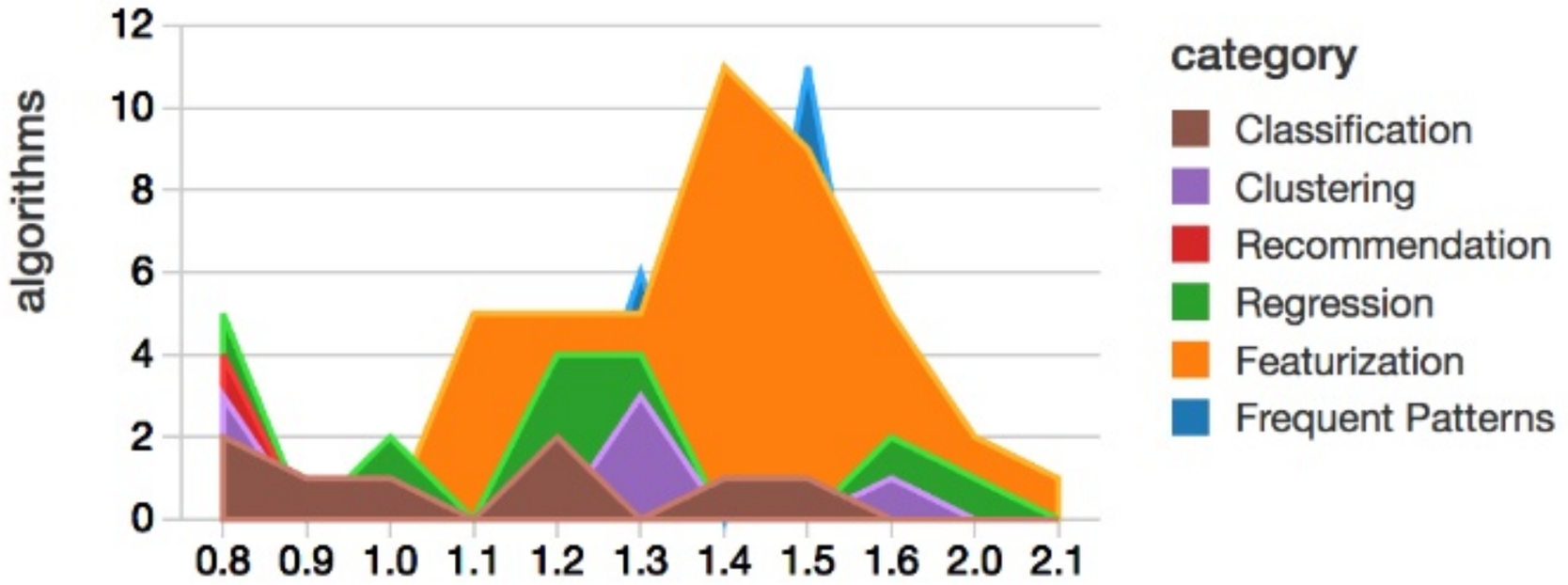
50+ algorithms and featurizers

Rapid development

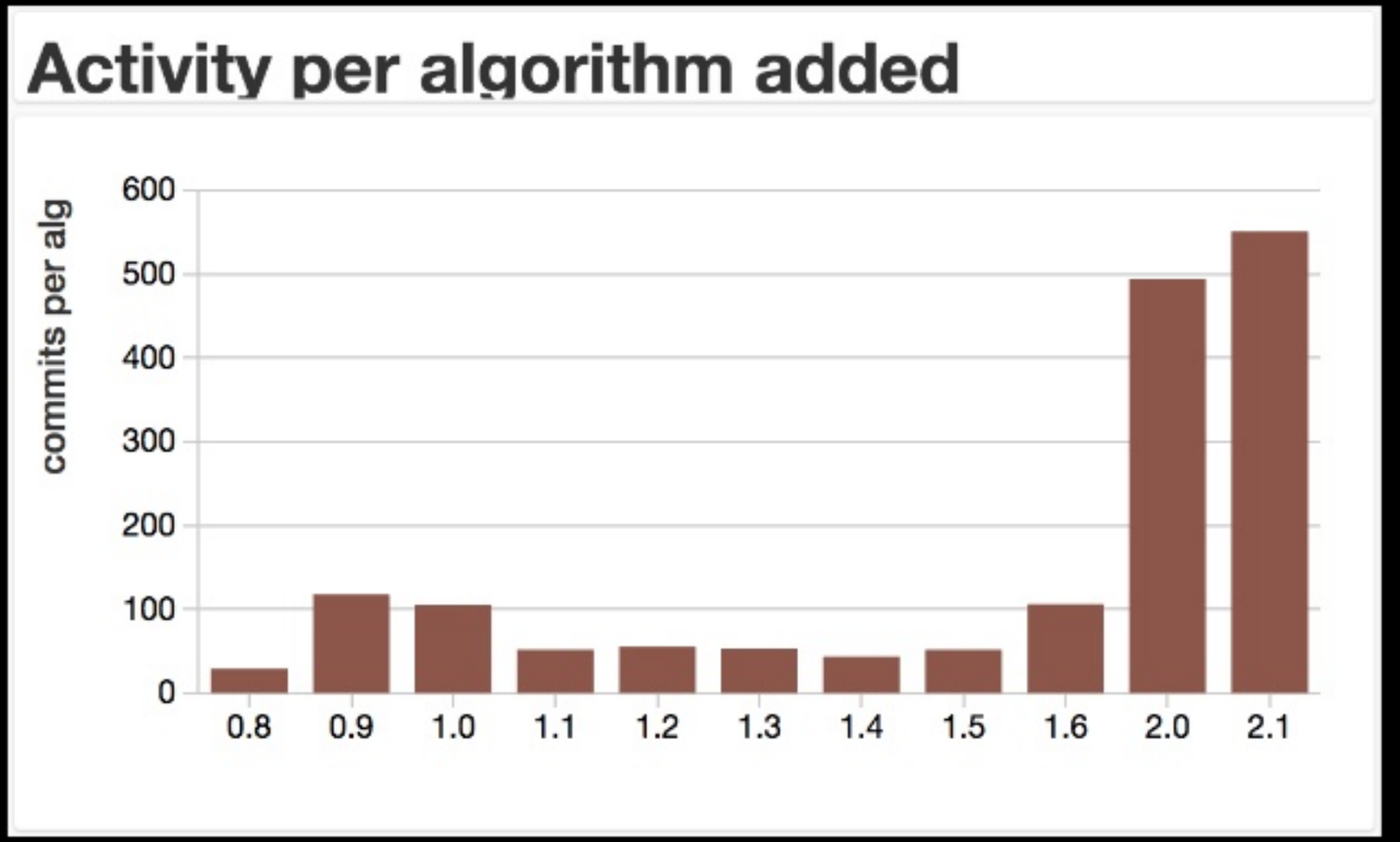
- Shift from adding algorithms to improving algorithms & infrastructure

Growing dev community

# Algorithms added per Spark release



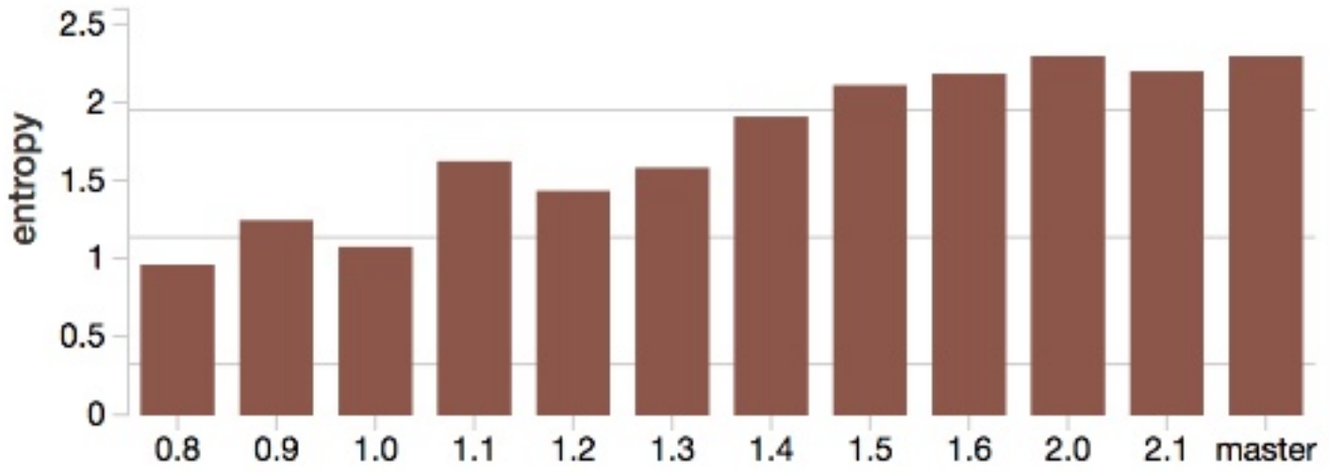
# Activity per algorithm added



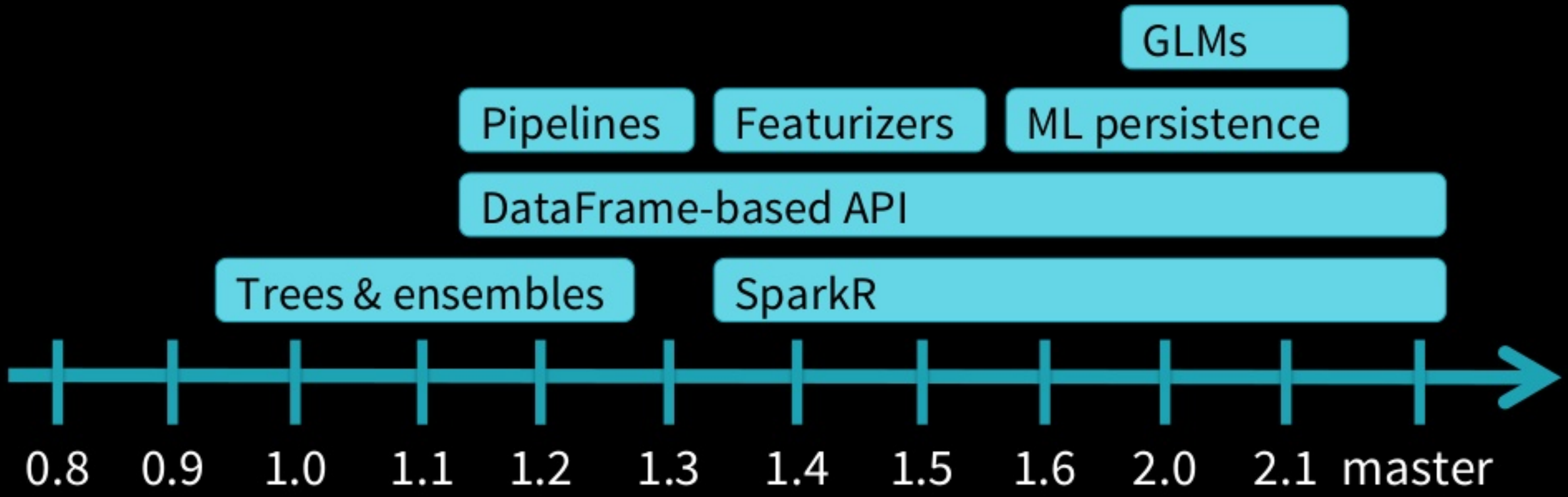
# Diversity of contributing organizations

Measured using entropy of distribution over organizations' commits

(Higher = more diverse)



# Major projects




*Note: These are unofficial "project" labels based on JIRA/PR activity.*

# So what?

What are the implications for the dev & user communities?

# Integrating ML with the Big Data world

Seamless integration with SQL, DataFrames, Graphs, Streaming, both within MLlib…

**Data sources**
CSV, JSON, Parquet, …

**DataFrames**
Simple, scalable pre/post-processing

**Graphs**
Graph-based ML implementations

**Streaming**
ML models deployed with Streaming

# Integrating ML with the Big Data world

Seamless integration with SQL, DataFrames, Graphs, Streaming, both within MLlib…and outside MLlib

Framework for integrating non-“big data” or specialized ML libraries

E.g., on spark-packages.org

- scikit-learn integration package (“spark-sklearn”)
- Stanford CoreNLP integration (“spark-corenlp”)

Deep Learning libraries

- Deep Learning Pipelines (“spark-deep-learning”)
- BigDL
- TensorFlowOnSpark
- Distributed training with Keras (“dist-keras”)

# Scaling

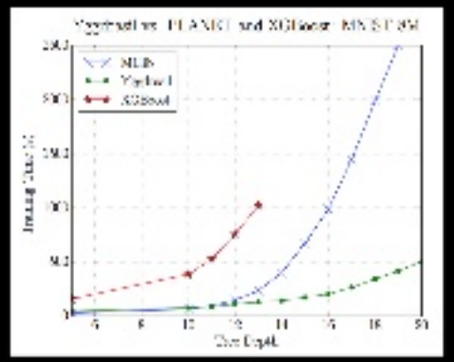
## Workflow scalability

Same code on laptop with small data → cluster with big data

## Big data scalability

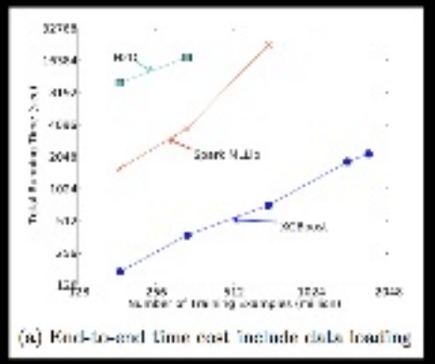
Scale-out implementations

4x faster than xgboost



Abuzaid et al. (2016)

8x slower than xgboost



Chen & Guestrin (2016)

# Building workflows


Simple concepts: Transformers, Estimators, Models, Evaluators

- Unified, standardized APIs, including for algorithm parameters

Pipelines

- Repeatable workflows across Spark deployments and languages
- DataFrame APIs and optimizations
- Automated model tuning

# Now what?

Where could or should it go?

# Top items from the dev@ list…

- Continued speed & scalability improvements
- Enhancements to core algorithms
- ML building blocks: linear algebra, optimization
- Extensible and customizable APIs

Goals

- Scale-out ML
- Standard library
- Extensible API

These are unofficial goals based on my experience + discussions on the dev@ mailing list.

# Scale-out ML

General improvements

- DataFrame-based implementations

E.g., 10x scaling for Connected Components in GraphFrames

Targeted improvements

- Gradient-boosted trees: feature subsampling
- Linear models: vector-free L-BFGS for billions of features
- …

# Standard library

General improvements

- NaN/null handling
- Instance weights
- Warm starts / initial models
- Multi-column feature transforms

Algorithm-specific improvements

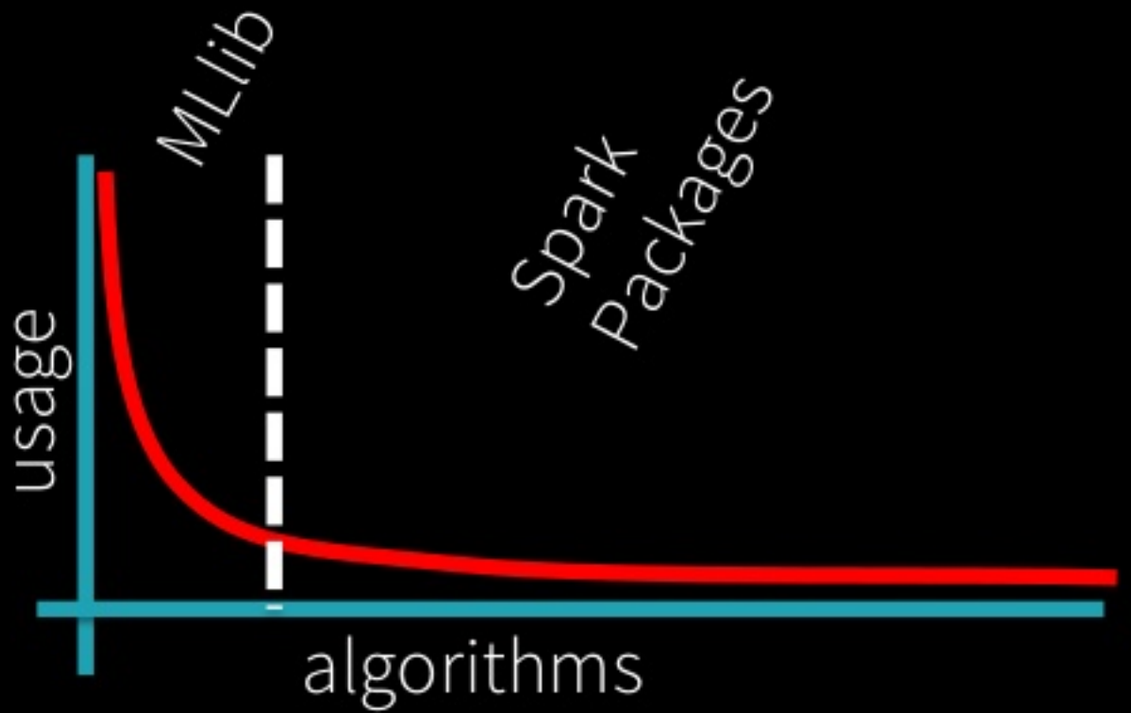
- Trees: access tree structure from Python
- …

# Extensible and customizable APIs

MLlib has ~50 algorithms.

CRAN lists 10,750 packages.

→ Users must be able to modify algorithms or write their own.



# Spark Packages

spark-packages.org

340+ packages written for Spark

80+ packages for ML and Graphs

E.g.:

- GraphFrames (“graphframes”): DataFrame-based graphs
- Bisecting K-Means (“bisecting-kmeans”): now part of MLlib
- Stanford CoreNLP wrapper (“spark-corenlp”): UDFs for NLP

# How to write a custom algorithm

You need to:
- Extend an API like Transformer, Estimator, Model, or Evaluator

MLlib provides some help:
- UnaryTransformer
- DefaultParamsWritable/Readable
- defaultCopy

You get:
- Natural integration with DataFrames, Pipelines, model tuning

# Challenges in customization

Modify existing algorithms with custom:

- Loss functions
- Optimizers
- Stopping criteria
- Callbacks

Implement new algorithms without worrying about:

- Boilerplate
- Python API
- ML building blocks
- Feature attributes/ metadata

# Example of MLlib APIs

Deep Learning Pipelines for Apache Spark

spark-packages.org/package/databricks/spark-deep-learning

APIs are great for users:

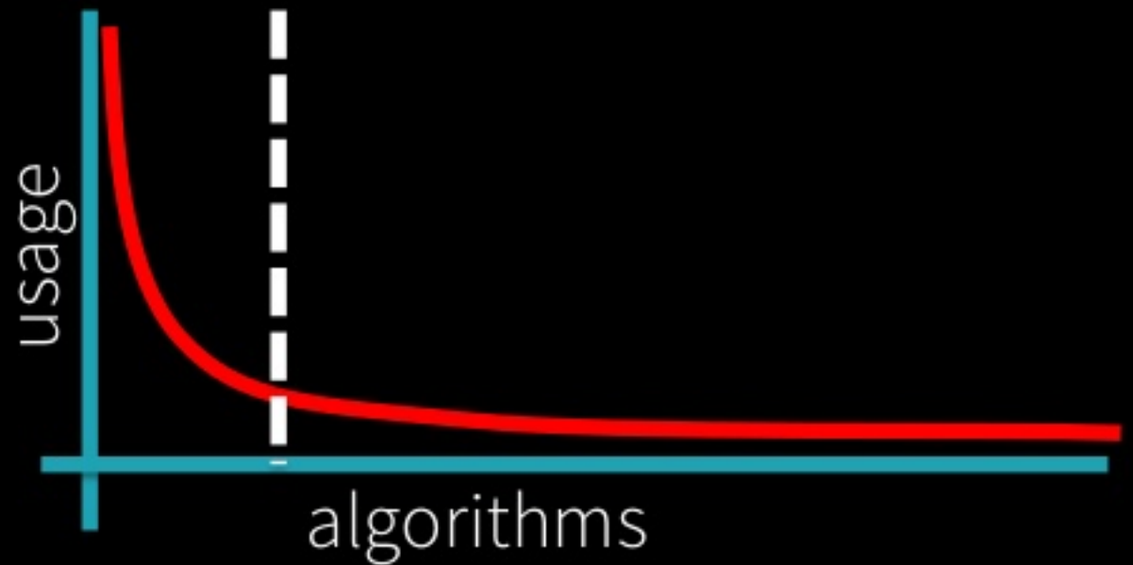
- Plug & play algorithms
- Standard API

But challenges remain:

- Development requires expertise
- Community under development

# My challenge to you

Can we fill out the long tail of ML use cases?



# Resources

Spark Packages: spark-packages.org

- Plugin for building packages: https://github.com/databricks/sbt-spark-package
- Tool for generating package templates: https://github.com/databricks/spark-package-cmd-tool

Example packages:

- scikit-learn integration (“spark-sklearn”) – Python-only
- GraphFrames – Scala/Java/Python

# Thank you!

Office hours today @ 3:50pm at Databricks booth

Twitter: @jkbatcmu