Simplifying Big Data Analysis with Apache Spark

Matei Zaharia

April 27, 2015

What is Apache Spark?

Fast and general cluster computing engine interoperable with Apache Hadoop

Improves efficiency through:

- In-memory data sharing
- General computation graphs

→ Up to 100× faster

Improves usability through:

- Rich APIs in Java, Scala, Python
- Interactive shell

→ 2-5× less code

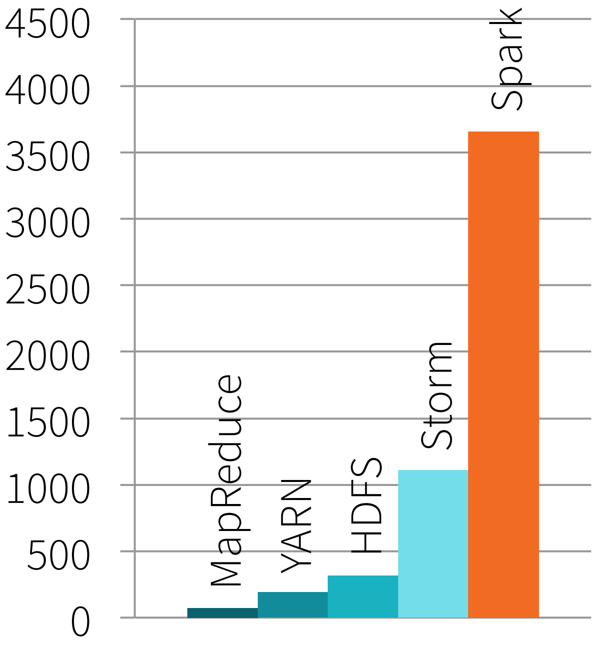
A General Engine

Spark SQL structured data	Spark Streaming real-time	MLlib machine learning	GraphX graph

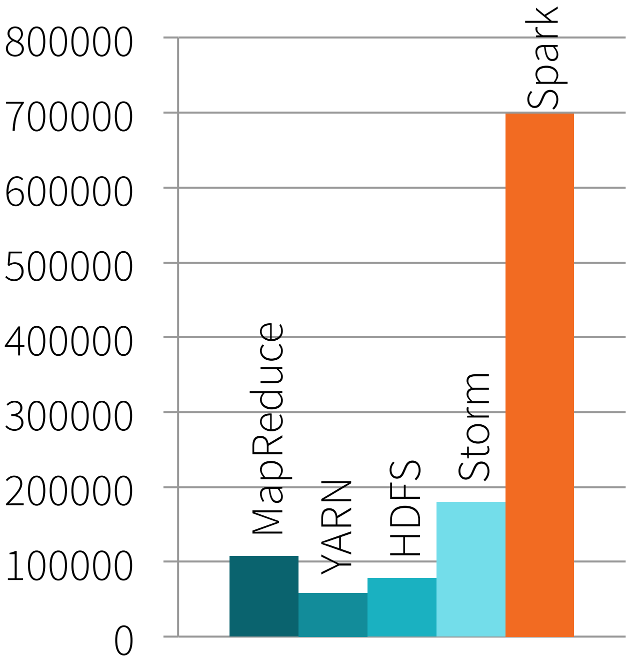
Spark Core

hadoop · amazon web services™ · cassandra · openstack™ · Google Compute Engine · MySQL® · …

A Large Community



Commits in past year



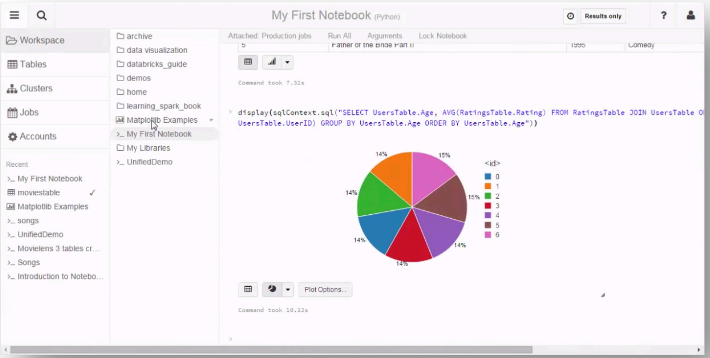
Lines of code changed in past year

About Databricks

Founded by creators of Spark and remains largest contributor

Offers a hosted service, Databricks Cloud

- Spark on EC2 with notebooks, dashboards, scheduled jobs



This Talk

Introduction to Spark

Built-in libraries

New APIs in 2015

- DataFrames
- Data sources
- ML Pipelines

Why a New Programming Model?

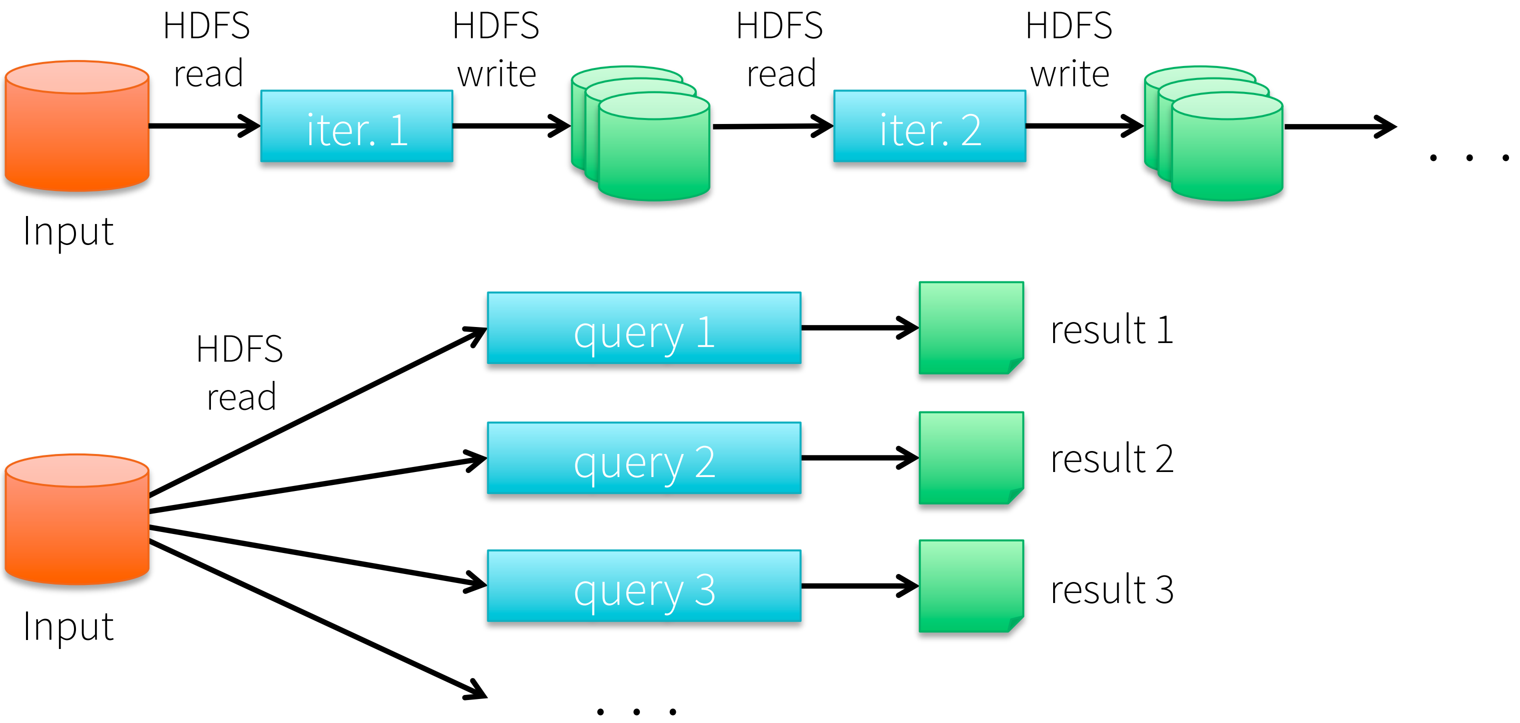
MapReduce simplified big data analysis

But users quickly wanted more:

- More complex, multi-pass analytics (e.g. ML, graph)
- More interactive ad-hoc queries
- More real-time stream processing

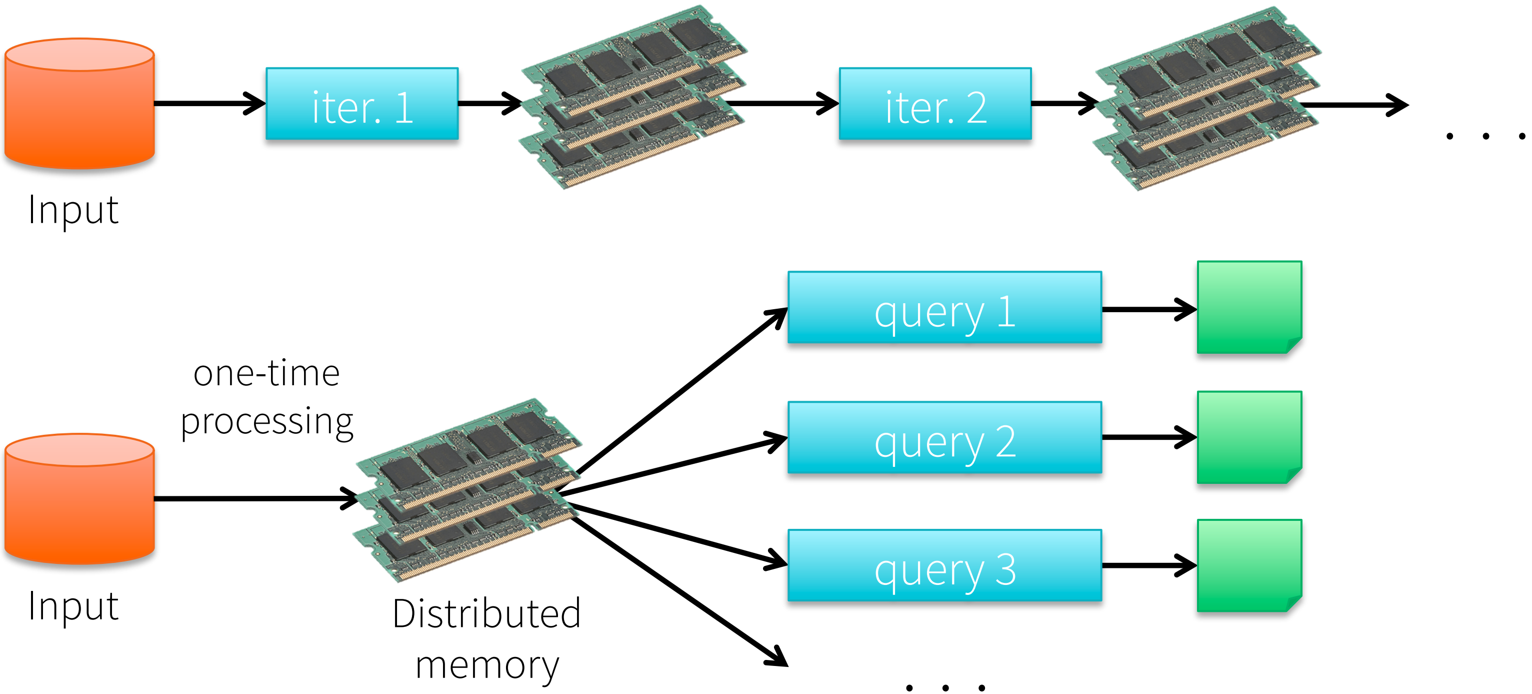
All 3 need faster data sharing in parallel apps

Data Sharing in MapReduce



Slow due to data replication and disk I/O

What We'd Like



10-100× faster than network and disk

databricks

Spark Model

Write programs in terms of transformations on datasets

Resilient Distributed Datasets (RDDs)

- Collections of objects that can be stored in memory or disk across a cluster
- Built via parallel transformations (map, filter, …)
- Automatically rebuilt on failure

Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

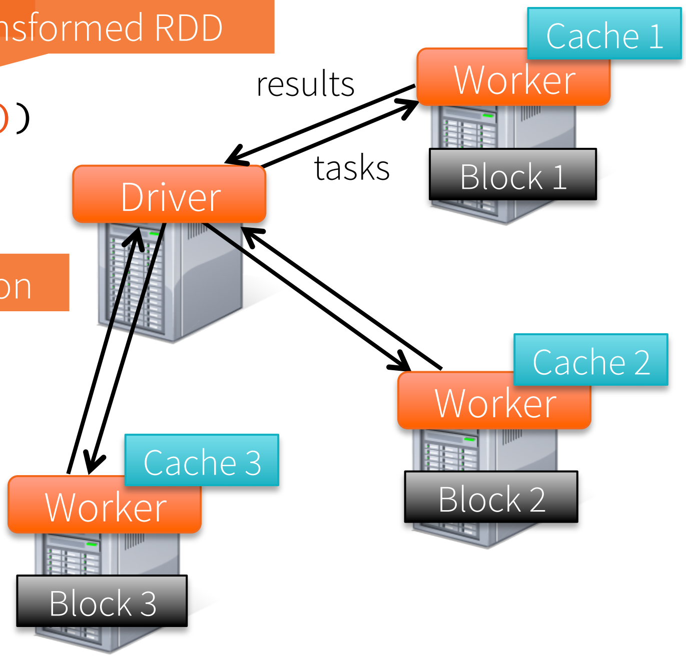
Base RDD

Transformed RDD

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split('\t')[2])
messages.cache()
```

Action

```
messages.filter(lambda s: "foo" in s).count()
messages.filter(lambda s: "bar" in s).count()
. . .
```



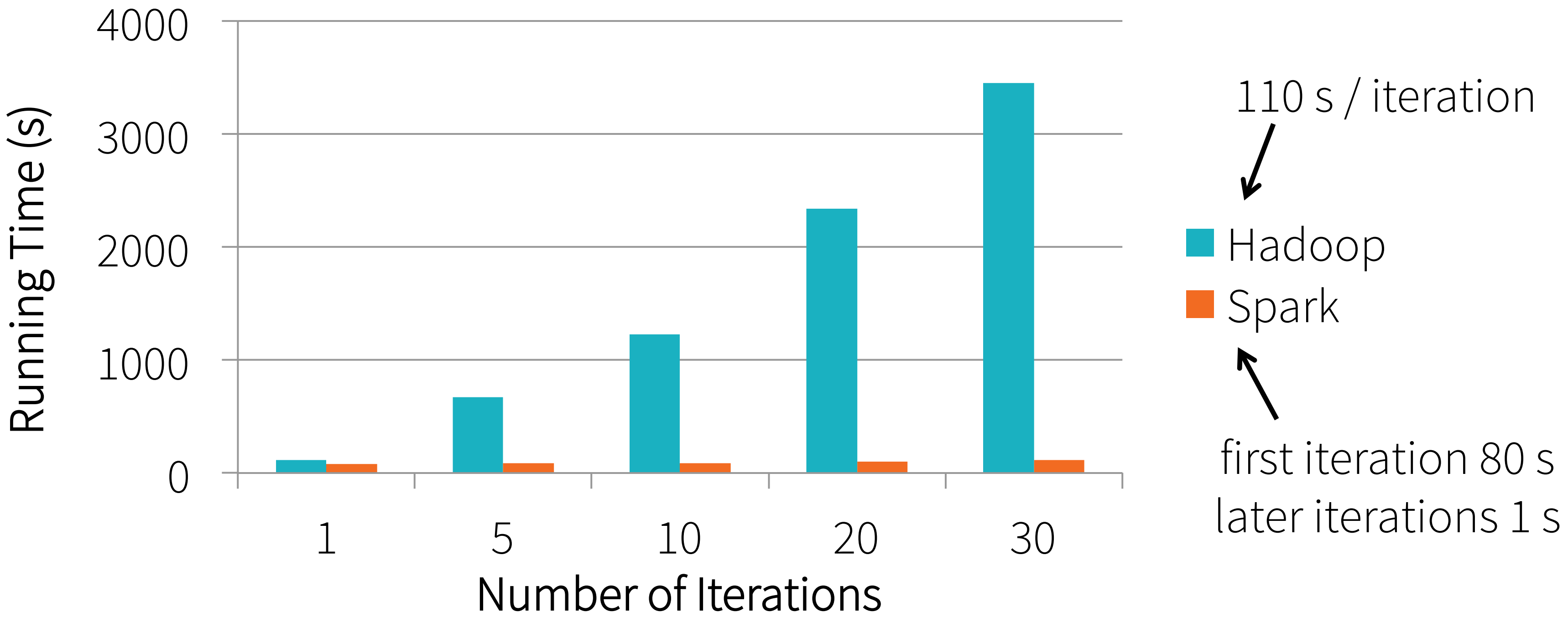
Full-text search of Wikipedia in <1 sec (vs 20 sec for on-disk data)

Fault Tolerance

RDDs track the transformations used to build them (their *lineage*) to recompute lost data

```
messages = textFile(...).filter(lambda s: “ERROR” in s)
                        .map(lambda s: s.split(“\t”)[2])
```

Example: Logistic Regression



databricks

On-Disk Performance

Time to sort 100TB

2013 Record:
Hadoop

2100 machines

72 minutes

2014 Record:
Spark

207 machines

23 minutes

Source: Daytona GraySort benchmark, sortbenchmark.org

Supported Operators

map	reduce	take
filter	count	first
groupBy	fold	partitionBy
union	reduceByKey	pipe
join	groupByKey	distinct
leftOuterJoin	cogroup	save
rightOuterJoin	flatMap	...

Spark in Scala and Java

```
// Scala:

val lines = sc.textFile(...)
lines.filter(s => s.contains(“ERROR”)).count()



// Java:

JavaRDD<String> lines = sc.textFile(...);
lines.filter(s -> s.contains(“ERROR”)).count();
```

User Community

Over 500 production users

Clusters up to 8000 nodes, processing 1 PB/day

Single jobs over 1 PB

This Talk

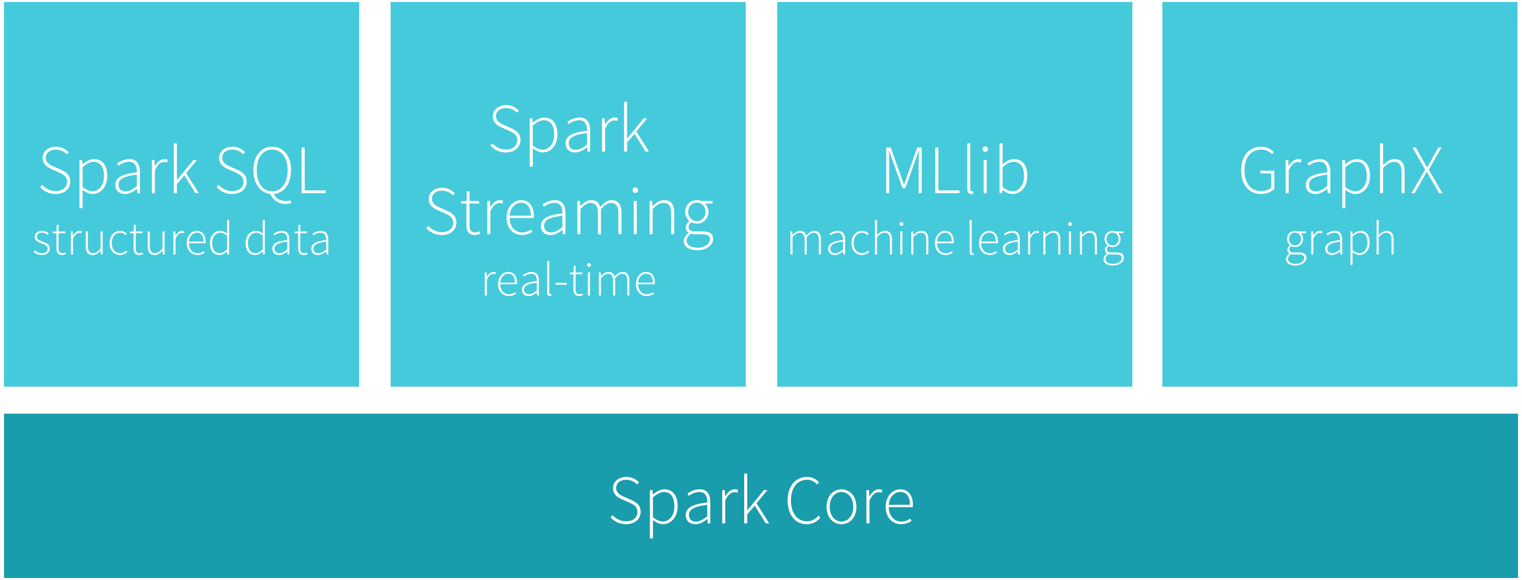
Introduction to Spark

Built-in libraries

New APIs in 2015

- DataFrames
- Data sources
- ML Pipelines

Built-in Libraries



Key Idea

Instead of having separate execution engines for each task, all libraries work directly on RDDs

Caching + DAG model is enough to run them efficiently

Combining libraries into one program is much faster

Spark SQL

Represents tables as RDDs

Tables = Schema + Data

From Hive:

```
c = HiveContext(sc)
rows = c.sql("select text, year from hivetable")
rows.filter(lambda r: r.year > 2013).collect()
```

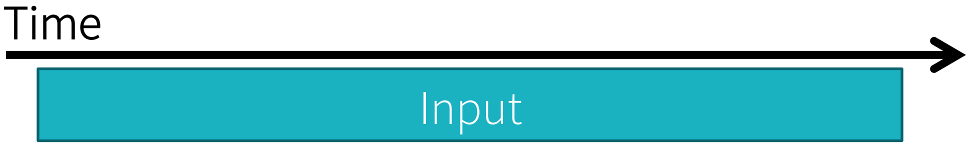
From JSON:

```
c.jsonFile("tweets.json").registerTempTable("tweets")
c.sql("select text, user.name from tweets")
```

tweets.json

```
{"text": "hi",
 "user": {
  "name": "matei",
  "id": 123
}}
```

Spark Streaming



Spark Streaming

Represents streams as a series of RDDs over time

```
sc.twitterStream(...)
  .map(lambda t: (t.username, 1))
  .reduceByWindow("30s", lambda a, b: a + b)
  .print()
```

MLlib

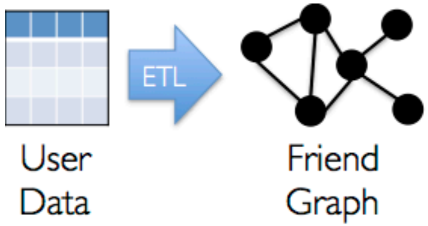
Vectors, Matrices = RDD[Vector]

Iterative computation

```
points = sc.textFile("data.txt").map(parsePoint)
model = KMeans.train(points, 10)

model.predict(newPoint)
```

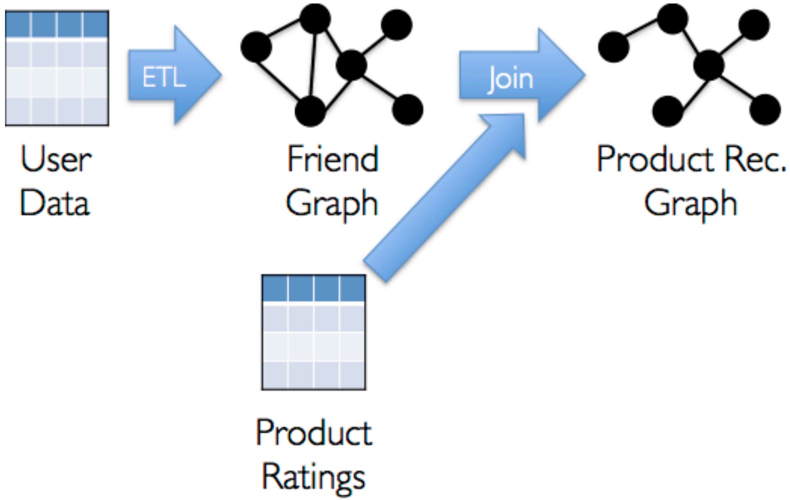
GraphX

Represents graphs as RDDs of vertices and edges

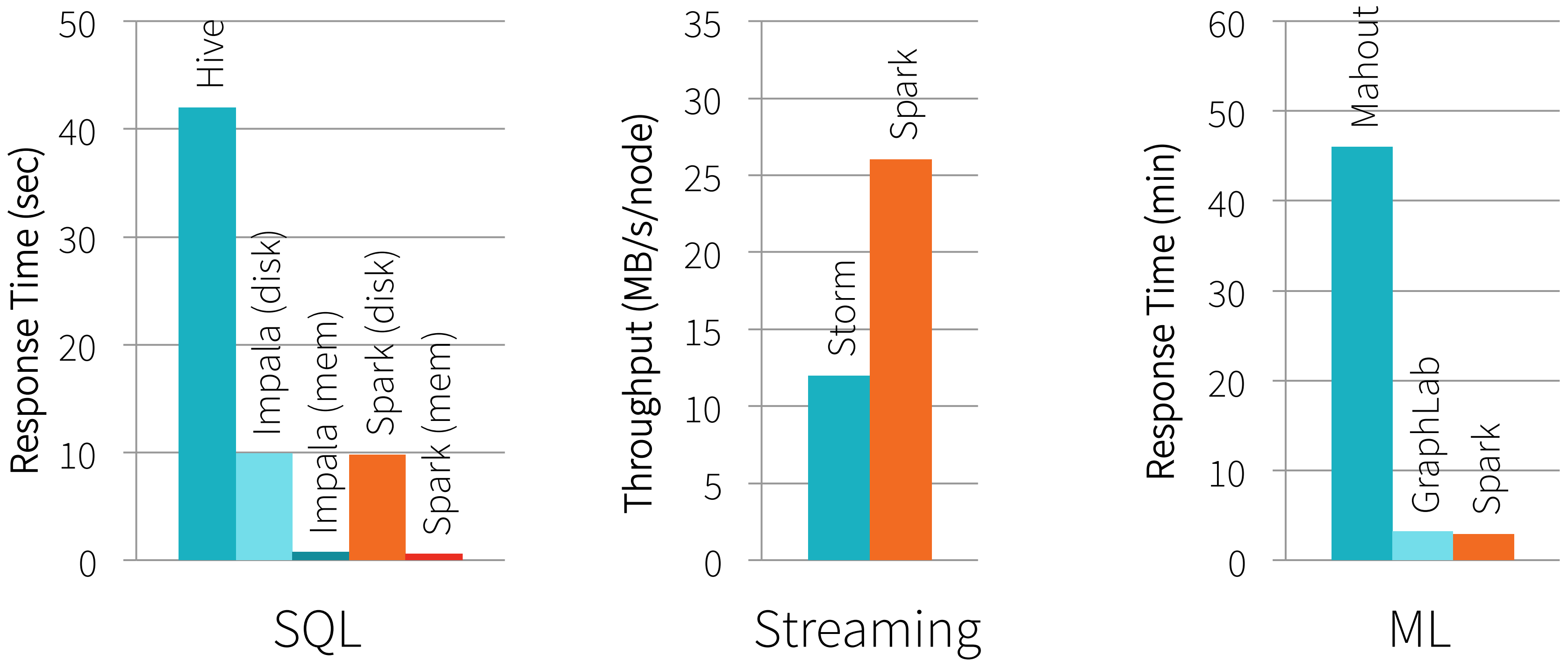


GraphX

Represents graphs as RDDs of vertices and edges



Performance vs. Specialized Engines



databricks

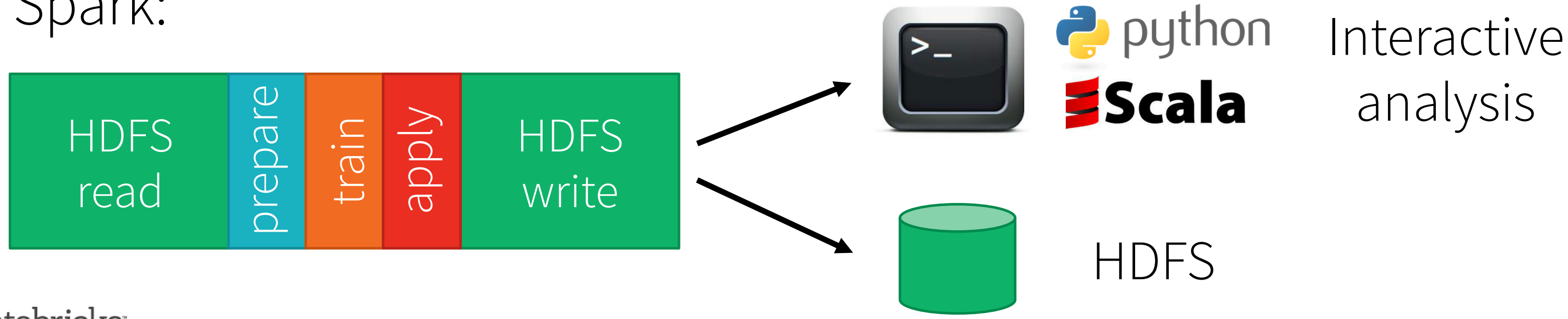
Combining Processing Types

```
// Load data using SQL
points = ctx.sql(“select latitude, longitude from hive_tweets”)

// Train a machine learning model
model = KMeans.train(points, 10)

// Apply it to a stream
sc.twitterStream(...)
  .map(lambda t: (model.predict(t.location), 1))
  .reduceByWindow(“5s”, lambda a, b: a + b)
```

Combining Processing Types

Separate engines:

Spark:



databricks

This Talk

Introduction to Spark

Built-in libraries

New APIs in 2015

- DataFrames
- Data sources
- ML Pipelines

Main Directions in 2015

Data Science

Making it easier for
wider class of users

Platform Interfaces

Scaling the ecosystem

From MapReduce to Spark

```java
public static class WordCountMapClass extends MapReduceBase
  implements Mapper<LongWritable, Text, Text, IntWritable> {

  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value,
                  OutputCollector<Text, IntWritable> output,
                  Reporter reporter) throws IOException {
    String line = value.toString();
    StringTokenizer itr = new StringTokenizer(line);
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      output.collect(word, one);
    }
  }
}

public static class WorkdCountReduce extends MapReduceBase
  implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                     OutputCollector<Text, IntWritable> output,
                     Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

```scala
val file = spark.textFile("hdfs://...")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://...")
```

Beyond MapReduce Experts

Early adopters

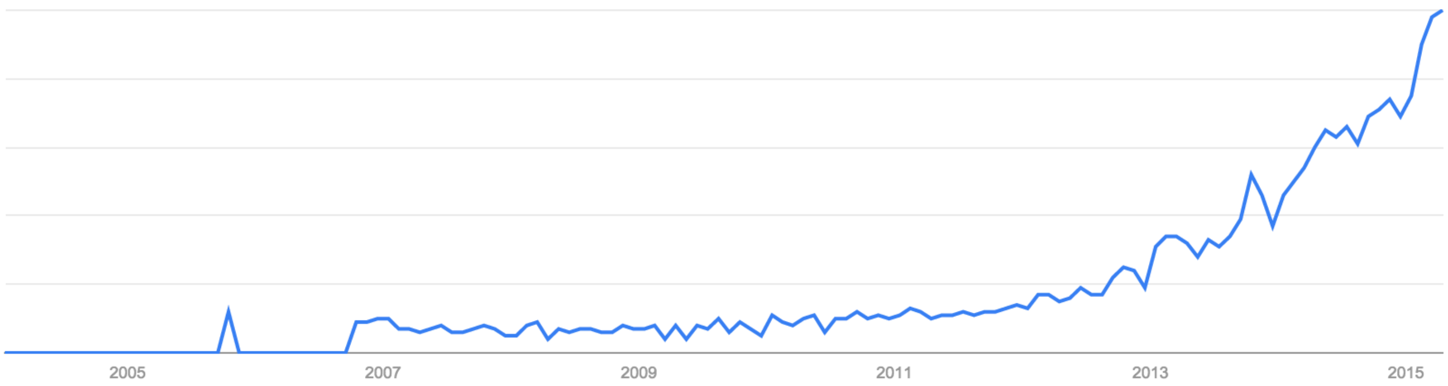
users

understands
MapReduce &
functional APIs

Data Scientists
Statisticians
R users
PyData
…

Data Frames

De facto data processing abstraction for data science
(R and Python)



Google Trends for “dataframe”

databricks

From RDDs to DataFrames

```
pdata.map(lambda x: (x.dept, [x.age, 1])) \
    .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
    .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
    .collect()
```

```
data.groupBy("dept").avg("age")
```

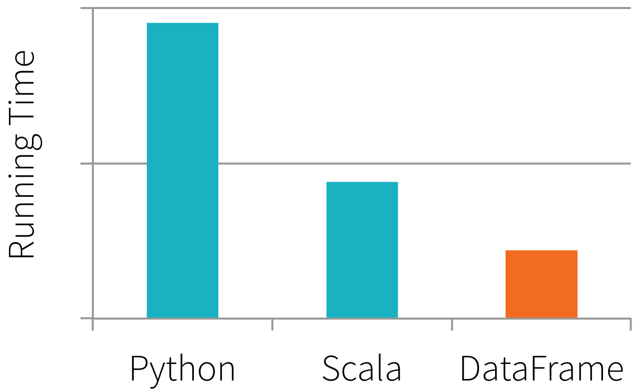
Spark DataFrames

Collections of structured data similar to R, pandas

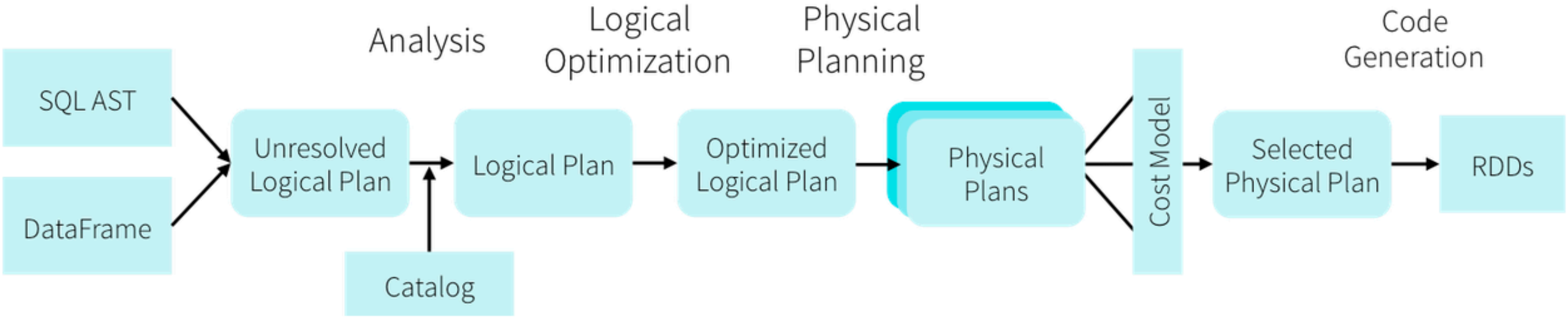
Automatically optimized via Spark SQL

- Columnar storage
- Code-gen. execution

```
df = jsonFile("tweets.json")

df[df["user"] == "matei"]
  .groupBy("date")
  .sum("retweets")
```



Optimization via Spark SQL



DataFrame expressions are relational queries, letting Spark inspect them

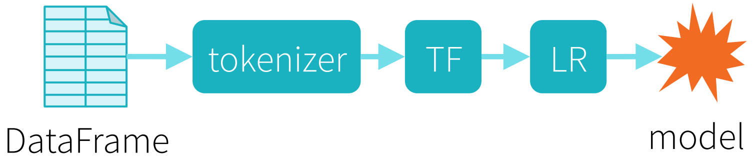
Automatically perform expression optimization, join algorithm selection, columnar storage, compilation to Java bytecode

Machine Learning Pipelines

High-level API similar to SciKit-Learn

Operates on DataFrames

Grid search to tune params across a whole pipeline




```
tokenizer = Tokenizer()
tf = HashingTF(numFeatures=1000)
lr = LogisticRegression()

pipe = Pipeline([tokenizer, tf, lr])
model = pipe.fit(df)
```

Spark R Interface

Exposes DataFrames and ML pipelines in R

Parallelize calls to R code

Target: Spark 1.4 (June)

```
df = jsonFile("tweets.json")

summarize(
  group_by(
    df[df$user == "matei",],
    "date"),
  sum("retweets"))
```

databricks

Main Directions in 2015

Data Science

Making it easier for
wider class of users

Platform Interfaces

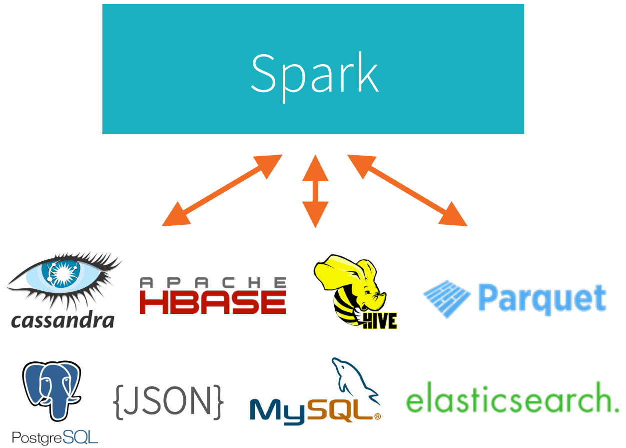
Scaling the ecosystem

Data Sources API

Allows plugging smart data sources into Spark

Returns DataFrames usable in Spark apps or SQL

Pushes logic into sources



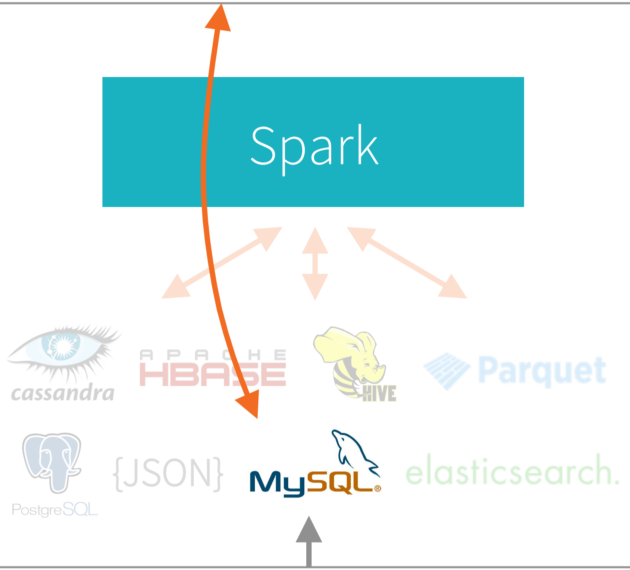
Data Sources API

Allows plugging smart data sources into Spark

Returns DataFrames usable in Spark apps or SQL

Pushes logic into sources

```
SELECT * FROM mysql_users u JOIN
              hive_logs h
         WHERE u.lang = "en"
```




```
SELECT * FROM users WHERE lang="en"
```

Current Data Sources

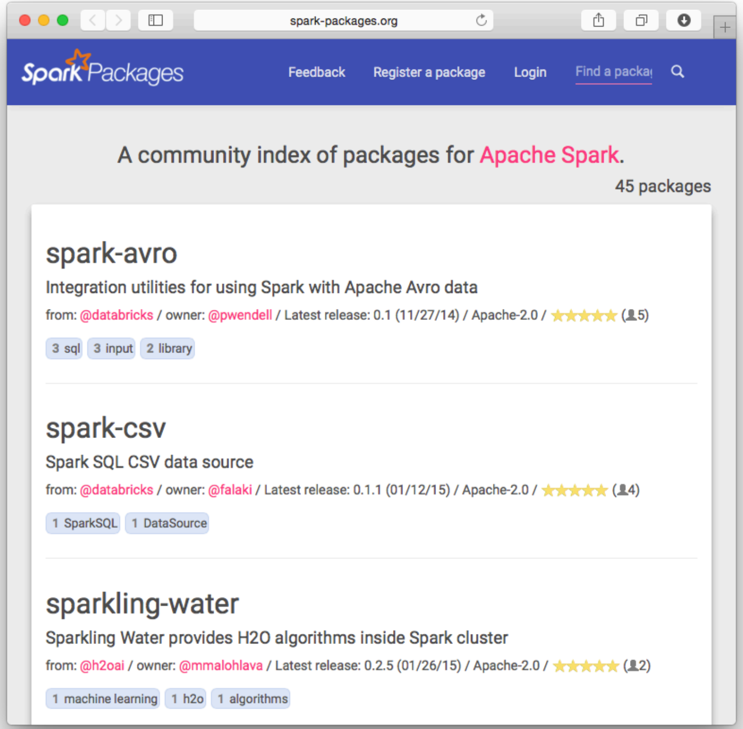
Built-in:

Hive, JSON, Parquet, JDBC

Community:

CSV, Avro, ElasticSearch, Redshift, Cloudant, Mongo, Cassandra, SequoiaDB

List at spark-packages.org



Goal: unified engine across data sources, workloads and environments

To Learn More

Downloads & docs: spark.apache.org

Try Spark in Databricks Cloud: databricks.com

Spark Summit: spark-summit.org