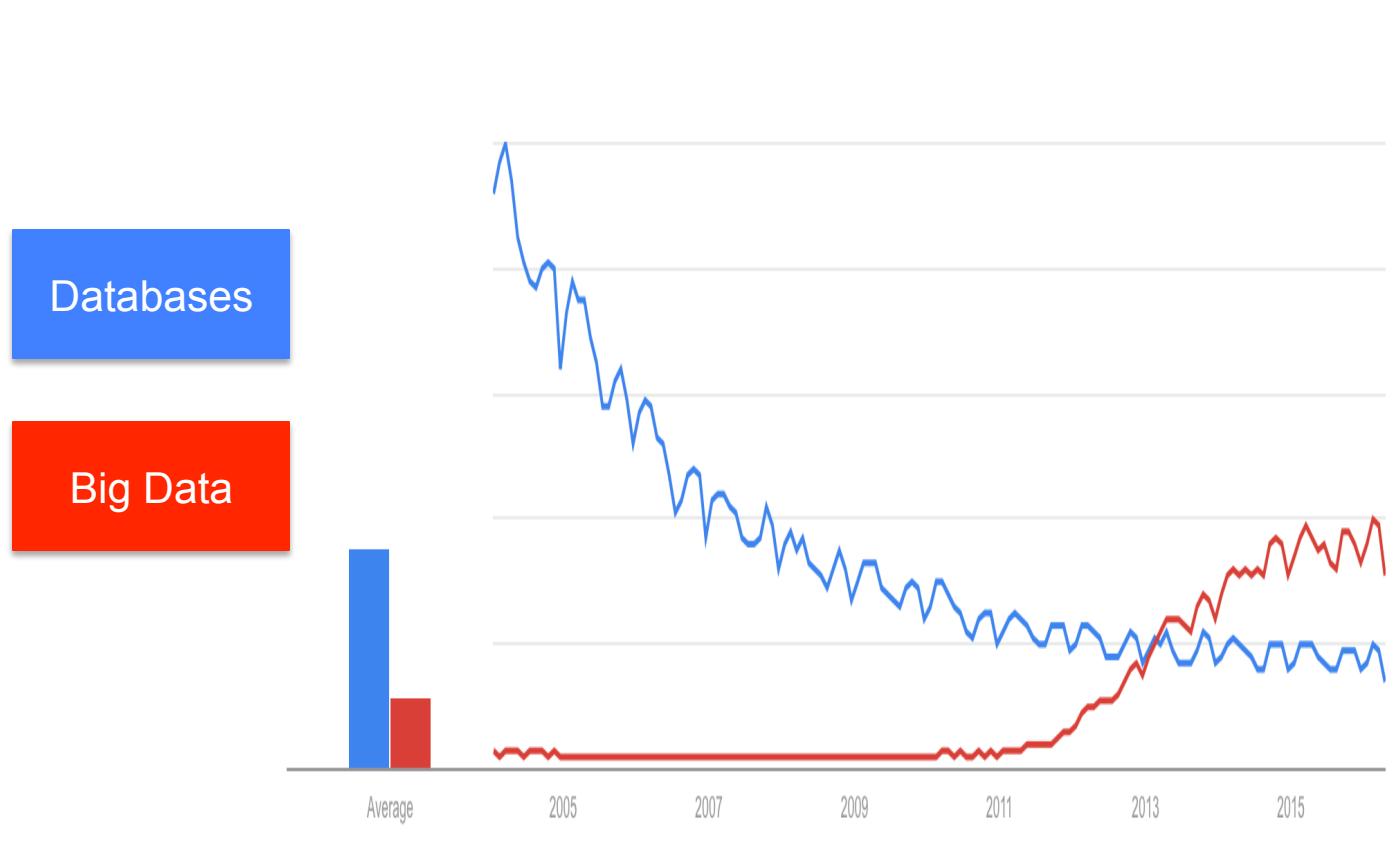




Power of data. Simplicity of design. Speed of innovation.

**Hands on Introduction to Apache Spark for
Data Engineers, Data Scientists, and Developers**

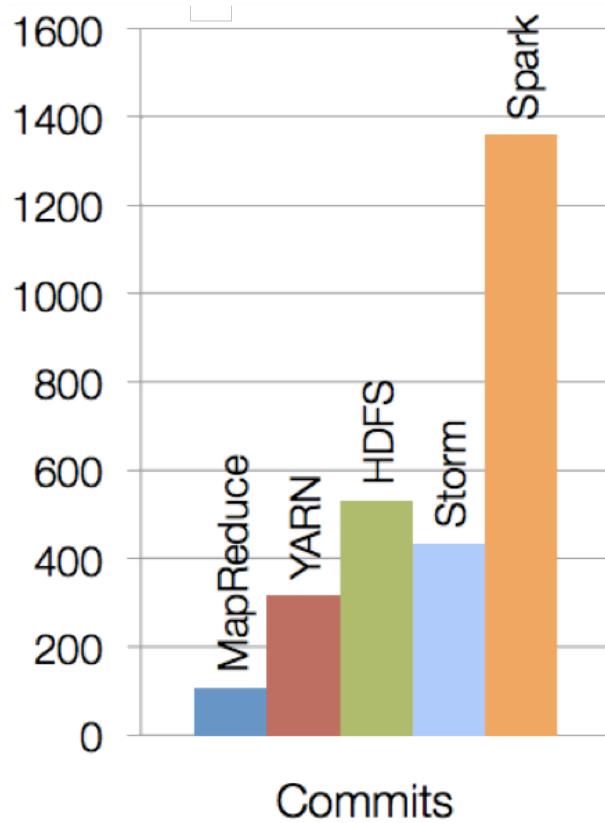
Market Evolution Databases to Big Data



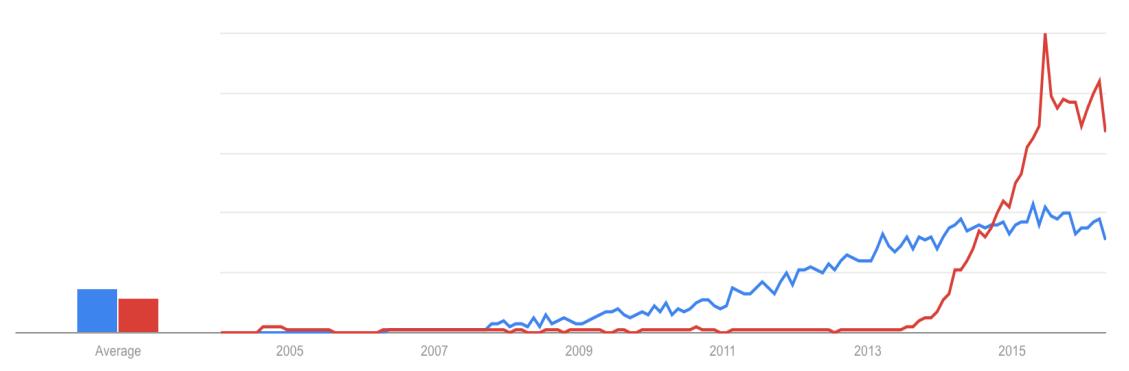
Source: Google Trends

Spark is new, but evolving quickly and has gained tremendous traction in a short amount of time

Spark is one of the most active open source projects



Hadoop vs Spark (Google Trends)



Spark Job Trends (Indeed.com)





What is Spark?

Spark is an **open source**
in-memory
application framework for
distributed data processing and
iterative analysis
on **massive** data volumes

“Analytic Operating System”

Key reasons for interest in Spark

Performant



- In-memory architecture greatly reduces disk I/O
- Anywhere from **20-100x faster** for common tasks

Productive



- **Concise and expressive syntax**, especially compared to prior approaches
- **Single programming model** across a range of use cases and steps in data lifecycle
- **Integrated with common programming languages** – Java, Python, Scala
- **New tools** continually reduce skill barrier for access (e.g. SQL for analysts)

Leverages existing investments



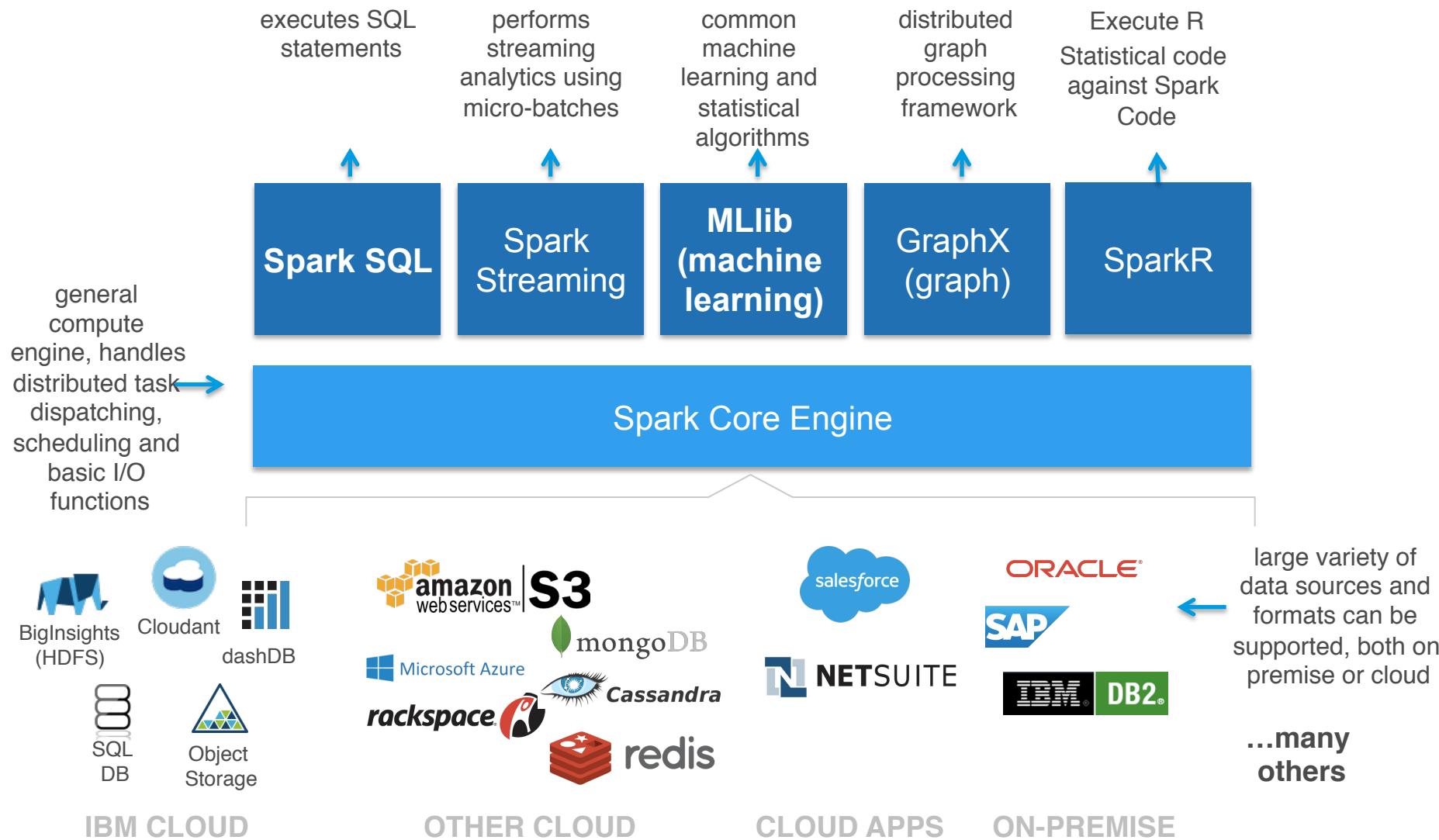
- Works well within **existing Hadoop ecosystem**

Improves with age



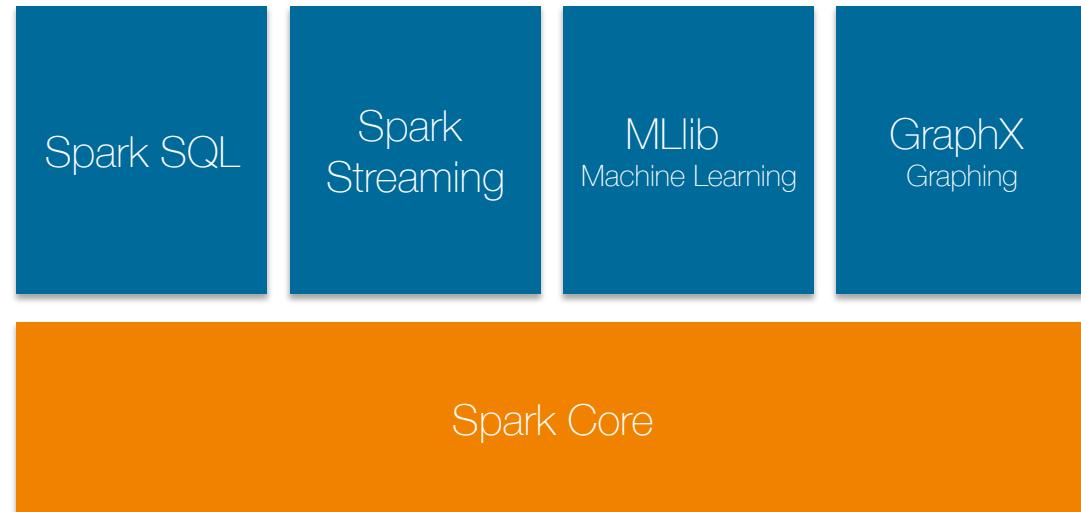
- **Large and growing community** of contributors continuously improve full analytics stack and extend capabilities

Spark includes a set of core libraries that enable various analytic methods which can process data from many sources



Libraries Usage¹

- **SparkSQL – 69%**
- **DataFrames – 62%**
- **Spark Streaming – 58%**
- **MLlib/GraphX – 58%**



75% of users use more than one component

¹ From a survey done by Databricks, summer 2015

Spark Programming Languages

▪ Scala

- Functional programming
- Spark written in Scala
- Scala compiles into Java byte code

▪ Java

- New features in Java 8 makes for more compact coding (lambda expressions)

▪ Python

- Most widely used API with Spark today

▪ R

- Functional programming language used to create and manipulate functions

Language	2014	2015
Scala	84%	71%
Java	38%	31%
Python	38%	58%
R	unknown	18%

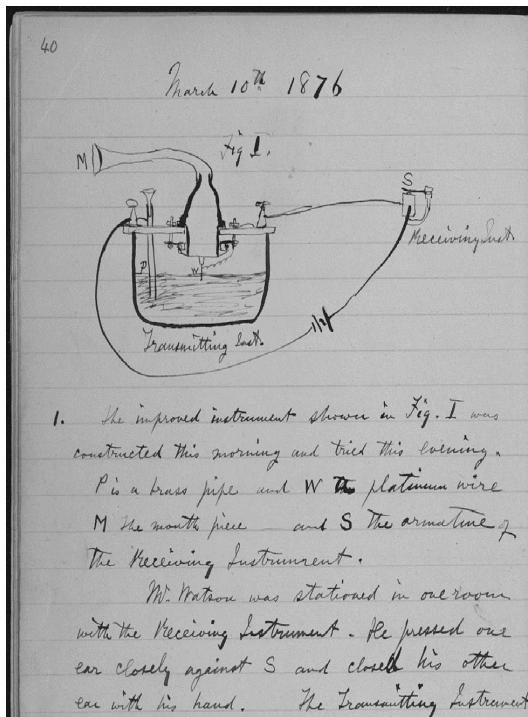
Survey done by Databricks,
Summer 2015

This probably means that more “data scientists” are starting to use Spark
DataFrames make all languages equally performant

What is a “Notebook”?

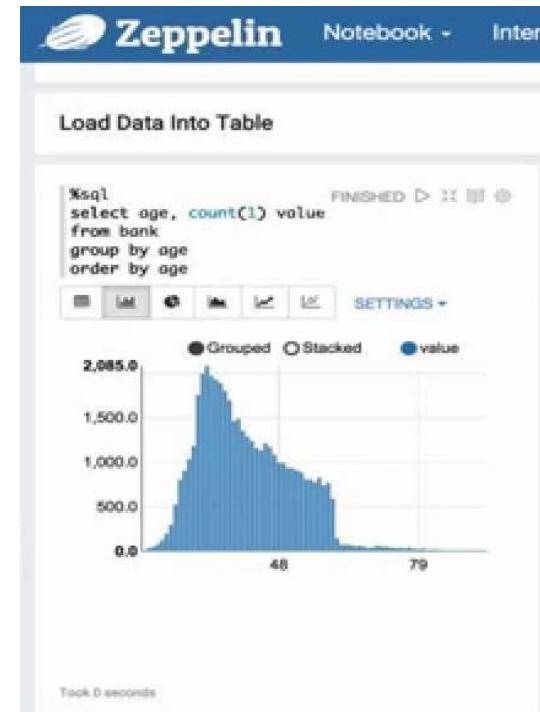
Pen and Paper

- Pen and paper has long provided the rich experience that scientists need to document progress through notes and drawings:
 - Expressive
 - Cumulative
 - Collaborative



Notebooks

- Notebooks are the digital equivalent of the “pen and paper” lab notebook, enabling data scientists to document reproducible analysis:
 - Markdown and visualization
 - Iterative exploration
 - Easy to share



Web-Based Notebooks...

▪ Notebooks:

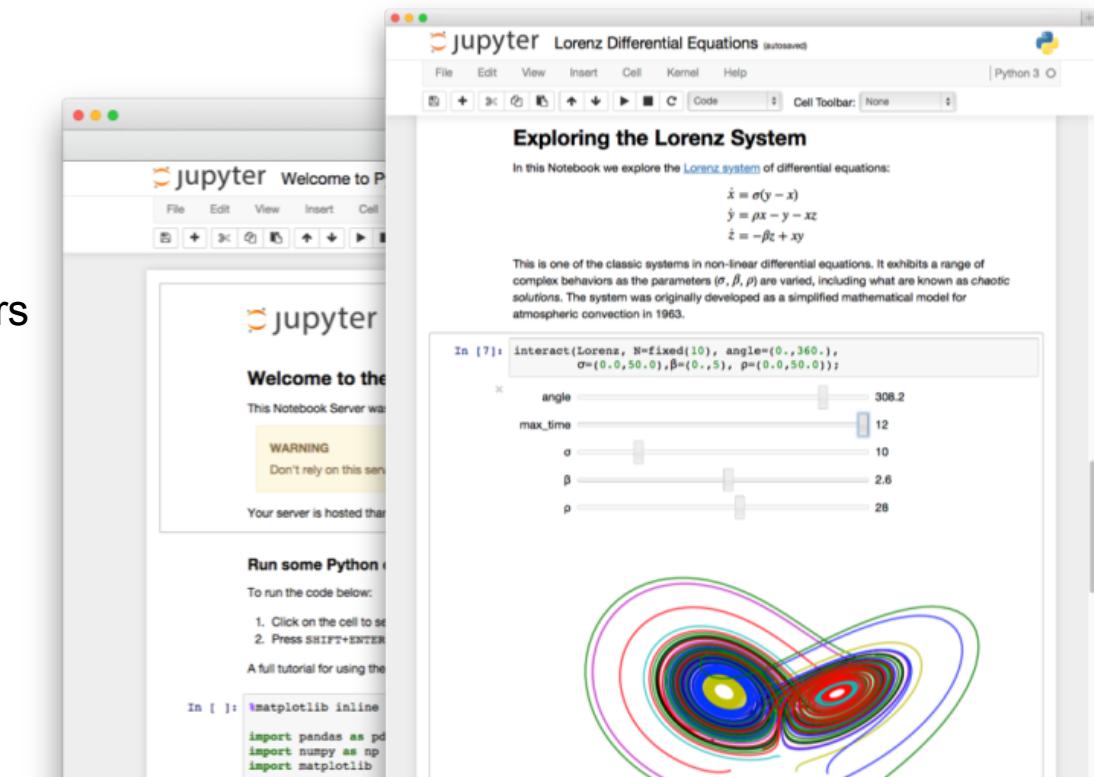
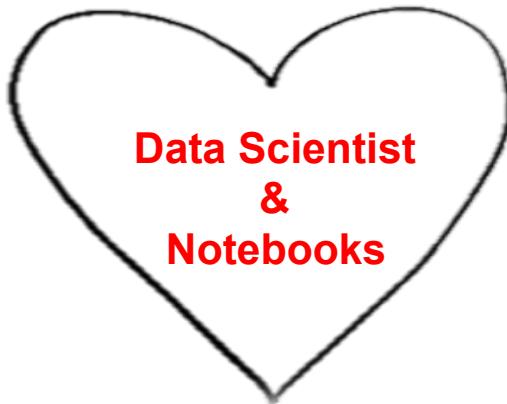
“interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media”

– Zeppelin

- Apache incubator project
- Supports multiple interpreters
 - Scala, Python, SparkSQL

– Jupyter

- Based on Ipython
- Supports multiple interpreters
 - Python, Scala



IBM is all-in on Spark

Contribute to the Core

Launch Spark Technology Cluster (STC), 300 engineers

Open source SystemML

Partner with Databricks

Foster Community

Educate 1M+ data scientists and engineers via online courses

Sponsor AMPLab, creators and evangelists of Spark

"It's like Spark just got blessed by the enterprise rabbi."

Ben Horowitz,
Andreessen Horowitz

Infuse the Portfolio

Integrate Spark throughout portfolio

3,500 employees working on Spark-related topics

Spark however customers want it – standalone, platform or products

IBM Contribute to core Apache Spark Project

WWW.SPARK.TC

IBM has the largest investment in Spark of any company in the world



IBM Spark Technology Center

- Launched in June of 2015
- Goal to hire 300 Engineers.
- Goal to Contribute to Apache Spark Apache community
- Contributed SystemML technology to Apache community
- STC continues to grow...

STC - JIRAs, check out what we are doing at apache.org

<https://issues.apache.org/jira/secure/Dashboard.jspa?selectPageId=12326761>

Issue Statistics

Statistics: STC Completed Spark JIRAs (Components)

Build	13	3%
Deploy	1	0%
Documentation	28	7%
Examples	6	2%
GraphX	1	0%
Input/Output	1	0%
ML	71	18%
MLlib	37	10%
Optimizer	2	1%
Project Infra	3	1%
PySpark	68	18%
Spark Core	22	6%
Spark Shell	4	1%
Spark Submit	1	0%
SparkR	12	~%
SQL	182	
Streaming	16	
Tests	6	
Web UI	15	
Windows	1	
YARN	1	
No component	2	

Total Issues: 493

Issue Statistics

Statistics: STC Completed Spark JIRAs Filter (Priority)

Blocker	2	1%
Critical	15	4%
Major	180	47%
Minor	144	37%
Trivial	45	12%

Total Issues: 386

Spark Technology Center (STC) - Completed JIRAs

Filter Results: STC Completed Spark JIRAs Filter

T	Key	P	Summary	Components	Assignee	Status	Resolved
↑	SPARK-13739	↑	Predicate Push Down Through Window Operator	SQL	Xiao Li	RESOLVED	25/Apr/16
↓	SPARK-14892	↓	Disable the HiveCompatibilitySuite test case for INPUTDRIVER and OUTPUTDRIVER	SQL	Xiao Li	RESOLVED	25/Apr/16
↑	SPARK-11337	↑	Make example code in user guide testable	Documentation	Xusen Yin	RESOLVED	25/Apr/16
+	SPARK-14433	↑	PySpark ml GaussianMixture	ML, PySpark	Miao Wang	RESOLVED	25/Apr/16
↑	SPARK-14768	↓	Remove expectedType arg for PySpark Param	ML, PySpark	Jason C Lee	RESOLVED	25/Apr/16
↑	SPARK-14548	↓	Support != and != operator in Spark SQL	SQL	Jia Li	RESOLVED	24/Apr/16
↑	SPARK-14691	↑	Simplify and Unify Error Generation for Unsupported Alter Table DDL	SQL	Xiao Li	RESOLVED	24/Apr/16
↑	SPARK-14838	↑	Implement statistics in SerializeFromObject to avoid failure when estimating sizeInBytes for ObjectType	SQL	Liang-Chi Hsieh	RESOLVED	24/Apr/16
↑	SPARK-14843	↑	Error while encoding: java.lang.ClassCastException with LibSVMRelation	ML, MLlib, SQL	Liang-Chi Hsieh	RESOLVED	22/Apr/16
↑	SPARK-13266	↑	Python DataFrameReader converts None to "None" instead of null	PySpark, SQL	Liang-Chi Hsieh	RESOLVED	22/Apr/16
↑	SPARK-14609	↑	SPARK-14118 / LOAD DATA	SQL	Liang-Chi Hsieh	RESOLVED	22/Apr/16
↑	SPARK-10001	↓	Allow Ctrl-C in spark-shell to kill running job	Spark Shell	Jakob Odersky	RESOLVED	22/Apr/16
↑	SPARK-7992	↑	Hide private classes/objects in generated Java API doc	Build, Documentation	Jakob Odersky	RESOLVED	21/Apr/16
↑	SPARK-14779	↓	Incorrect log message in Worker while handling KillExecutor message	Spark Co			
↑	SPARK-13419	↑	SPARK-13417 / SubquerySuite should use checkAnswer rather than ScalaTest's assertResult	SQL			
↑	SPARK-12457	↑	SPARK-12454 / Add ExpressionDescription to co				
↑	SPARK-14398	↑	SPARK-12362 / Audit non-reserved keyword list i				

Created vs Resolved Chart: STC Completed Spark JIRAs Filter

Issues: 39 created and 75 resolved
Period: last 30 days (grouped Daily)

Issue Statistics

Statistics: STC Completed Spark JIRAs Filter (Issue Type)

Bug	138	36%
Documentation	8	2%
Improvement	128	33%
New Feature	19	5%
Sub-task	87	23%
Task	2	1%
Test	3	1%
Umbrella	1	0%

Total Issues: 386

© 2016 IBM Corporation

13

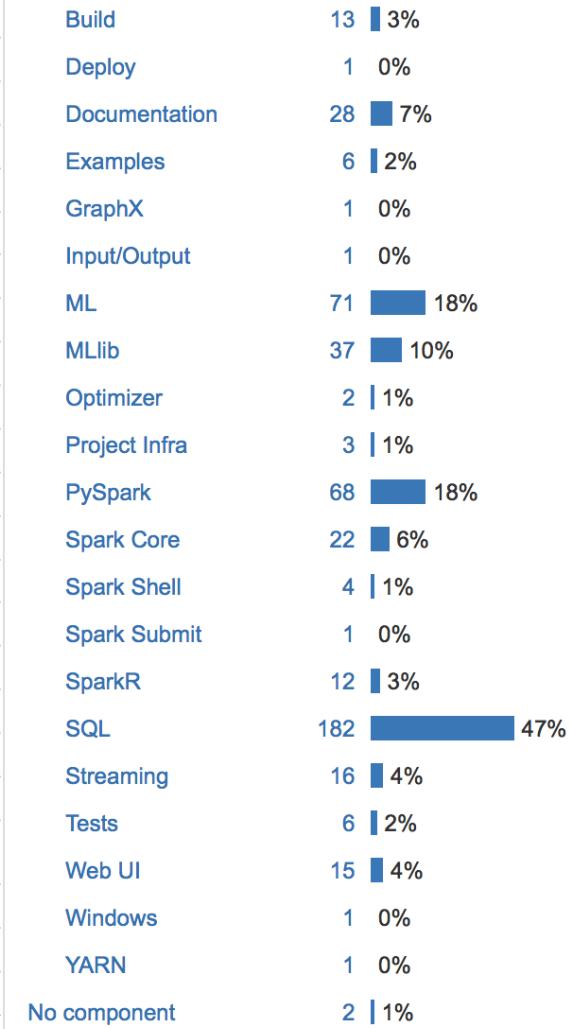
IBM driving SQL and Machine Learning innovation..

Filter Results: STC Completed Spark JIRAs Filter

T	Key	P	Summary	Components
	SPARK-13996	↑	Add more not null attributes for Filter codegen	SQL
	SPARK-14308	↓	SPARK-12326 / Remove unused mllib tree classes and move private classes to ML	ML, MLLib
	SPARK-13241	↓	add long-formatted timestamps to org.apache.spark.status.api.v1.ApplicationAttemptInfo	Web UI
	SPARK-13674	↑	Add wholestage codegen support to Sample	SQL
	SPARK-14191	↑	Fix Expand operator constraints	SQL
	SPARK-13995	↑	Extract correct IsNotNull constraints for Expression	SQL
+	SPARK-14264	↓	Add feature importances for GBTs in Pyspark	ML, PySpark
	SPARK-14182	↑	Parse DDL command: Alter View	SQL
	SPARK-11892	↑	SPARK-6725 / Model export/import for spark.ml: OneVsRest	ML
	SPARK-14181	↓	TrainValidationSplit should have HasSeed	ML
	SPARK-14124	↑	SPARK-14118 / Database related commands	SQL
+	SPARK-10570	↓	Add Spark version endpoint to standalone JSON API	Spark Core, Web
	SPARK-13963	↓	SPARK-13964 / Add binary toggle Param to ml.HashingTF	ML
	SPARK-14071	↓	SPARK-11939 / Change MLWritable.write to be a property	ML, PySpark
+	SPARK-11730	↑	Feature Importance for GBT	ML, MLLib
	SPARK-11893	↑	SPARK-6725 / Model export/import for spark.ml: TrainValidationSplit	ML
	SPARK-14156	↓	Use executedPlan for HiveComparisonTest	SQL
	SPARK-13742	↑	Add non-iterator interface to RandomSampler	Spark Core
	SPARK-14177	↑	Parse DDL command: "DESCRIBE DATABASE" and "ALTER DATABASE SET DBPROPERTIES"	SQL
	SPARK-14157	↑	Parse Drop Function DDL command	SQL
	SPARK-14161	↑	Parse Drop Database DDL command	SQL
	SPARK-12443	↑	encoderFor should support Decimal	SQL

Issue Statistics

Statistics: STC Completed Spark JIRAs Filter (Components)



Total Issues: 493

<http://www.spark.tc/blog/>

Foster Community - Free Education

Big Data University

<http://bigdatauniversity.com/>

The screenshot shows the Big Data University homepage. At the top, there's a navigation bar with links for File, Edit, View, History, Bookmarks, Tools, and Help. Below the navigation is a toolbar with links for bigdatauniversity.com, Most Visited, and various IBM-related items like w3 - Home, Inbox, IBM Travel - Travel Res..., Expense reimbursement, ZACS, Host Meetings, All files and folders - B..., and Faces.

The main content area features the "BIG DATA UNIVERSITY" logo and a banner with social media icons for Facebook, Twitter, Google+, and LinkedIn. The banner also contains the text "Analytics, Big Data, and Data Science Courses" and a "SIGN UP" button.

A large, circular network diagram at the bottom represents various data science and engineering concepts: NoSQL, Python, Spark, Scala, R, Data Engineering, Big Data, Social Good, and Hadoop. The network is interconnected by dashed lines.

In the center, there's a section titled "RECOMMENDED FOR YOU" featuring nine course cards:

- Big Data Fundamentals (red)
- Data Science Fundamentals (orange)
- Introduction to OpenRefine (green)
- Introduction to R (orange)
- SQL Fundamentals (pink)
- Introduction to NoSQL (blue)
- Introduction to Data Analysis Using R (green)
- Hadoop Fundamentals I (blue)
- Spark Fundamentals I (orange)

At the bottom left, the URL <https://w3-connections.ibm.com/wikis/home?lang=en#!wiki/Zone%20for%20Accelerating%20Client%20Success> is visible.

Foster Community – Free Platforms

Data Sciencist Work Bench

<https://datascientistworkbench.com/>

Spark as a Service

www.bluemix.net

IBM Software > Products > Big data > Hadoop system > IBM BigInsights >

IBM Open Platform with Apache Hadoop

Features What's new for Version 4.1

100% Apache Hadoop Open Source platform

No-charge download available → IBM Open Platform with Apache Hadoop

IBM released a new Open Platform for Apache Hadoop on Intel and Power platforms and IBM BigInsights v4.1 on Intel and Power Systems.

IBM Open Platform with Apache Hadoop

- Native support for rolling upgrades for Hadoop services
- Support for long-running applications within YARN for enhanced reliability & security
- Heterogeneous storage in HDFS for in-memory, SSD in addition to HDD
- Spark in-memory distributed compute engine for dramatic performance increases over MapReduce and simplifies developer experience, leveraging Java, Python & Scala languages

IBM Open Platform

<http://www-03.ibm.com/software/products/en/ibm-open-platform-with-apache-hadoop>

Data Scientist Workbench

Technology Preview

Prepare data. Analyze data. Get answers.

Get started now

IBM Bluemix

The Digital Innovation Platform

BUILD EXTEND SCALE INTEGRATE FEATURED

Build your apps, your way.

Instant Runtimes App-centric runtime environments based on Cloud Foundry

IBM Containers Portable and consistent delivery of your app without having to manage an OS.

Virtual Machines Get the most flexibility and control over your environment with VMs

openstack POWERED

Apache Spark Under the hood!

- **Context (Connection):**

- Represents a connection to the Spark cluster. The Application which initiated the context can submit one or several jobs, sequentially or in parallel, batch or interactively.

- **Driver (Coordinator agent)**

- The program or process running the Spark context. Responsible for running jobs over the cluster and converting the App into a set of tasks

- **Job (Query / Query plan):**

- A piece of logic (code) which will take some input from HDFS (or the local filesystem), perform some computations (transformations and actions) and write some output back.

- **Stage (Subplan)**

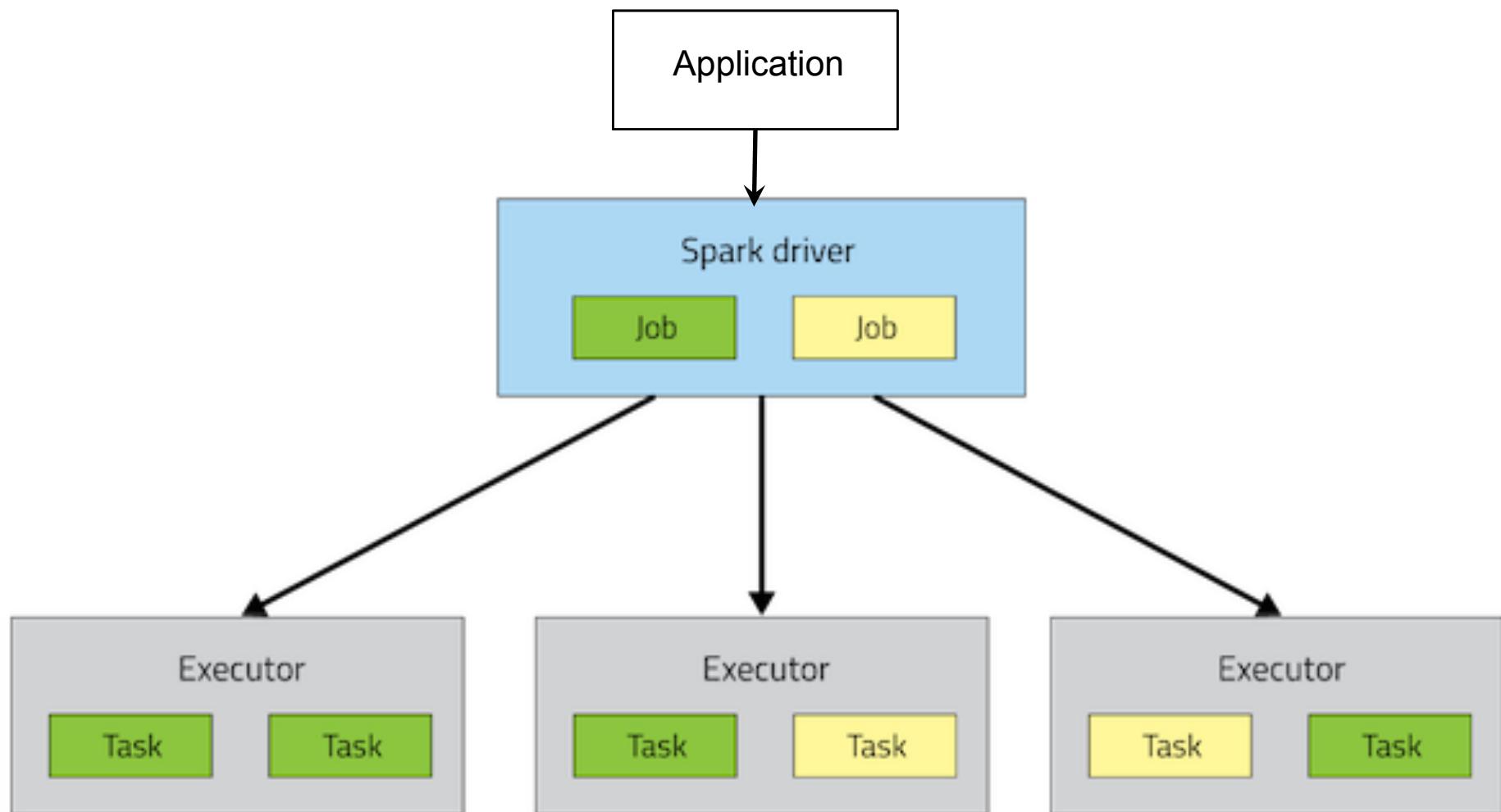
- Jobs are divided into stages

- **Tasks (Sub section)**

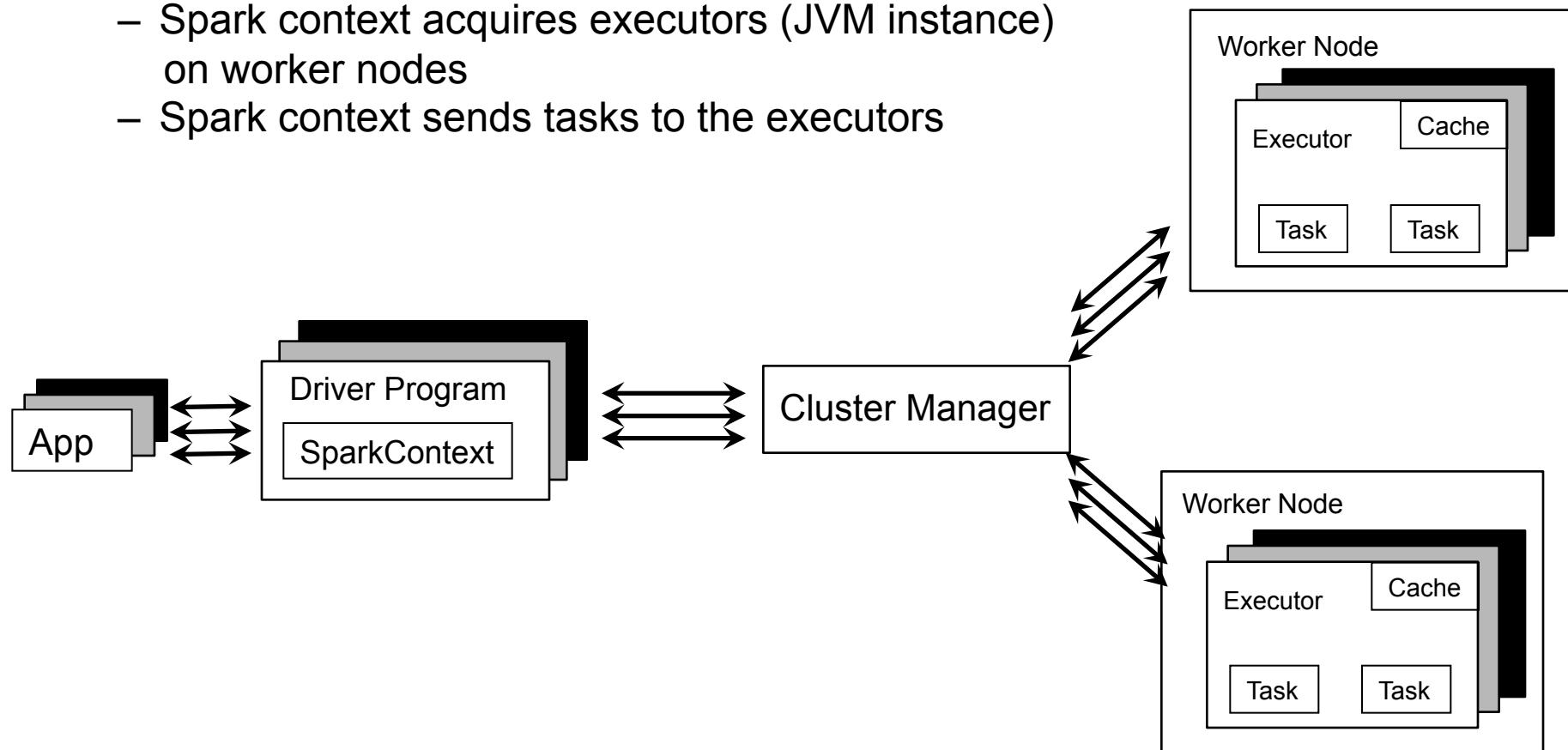
- Each stage is made up of tasks. One task per partition. One task is executed on one partition (of data) by one executor

- **Executor (Sub agent)**

- The process responsible for executing a task on a worker node

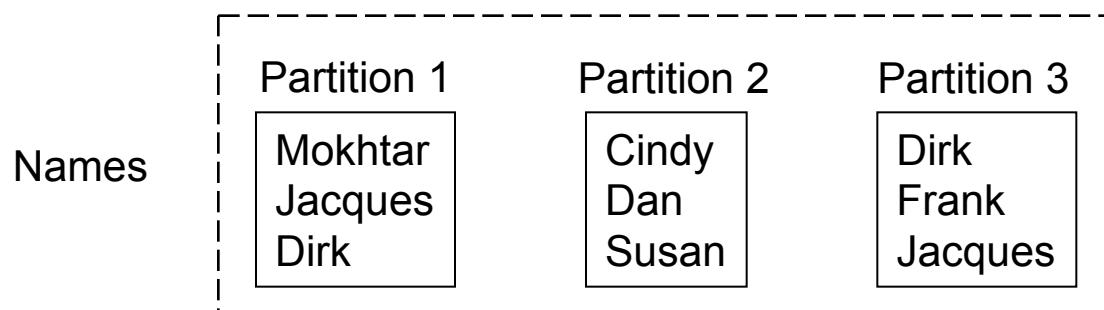


- A Spark application is initiated from a driver program
- Each Spark application runs as a set of processes coordinated by the Spark context object (driver program)
 - Spark context connects to Cluster Manager (standalone, Mesos/Yarn)
 - Spark context acquires executors (JVM instance) on worker nodes
 - Spark context sends tasks to the executors



Resilient Distributed Datasets

- **RDD is a distributed collection of Scala/Python/Java objects of the same type:**
 - Strings
 - Integers
 - (key, value) pairs
 - class Java/Python/Scala objects
- **RDDs are immutable**
Modifications create new RDDs
- **Physically distributed across cluster, but manipulated as one logical entity:**
 - Spark will “distribute” any required processing to all partitions where the RDD exists and perform necessary redistributions and aggregations as well.
 - Example: Consider a distributed RDD “Names” made of names



Resilient Distributed Datasets

- Suppose we want to know the number of names in the RDD “Names”
- User simply requests: `Names.count()`
 - Spark will “distribute” count processing to all partitions so as to obtain:
 - Partition 1: Mokhtar(1), Jacques (1), Dirk (1) → 3
 - Partition 2: Cindy (1), Dan (1), Susan (1) → 3
 - Partition 3: Dirk (1), Frank (1), Jacques (1) → 3
 - Local counts are subsequently aggregated: $3+3+3=9$
- To lookup the first element in the RDD: `Names.first()`
- To display all elements of the RDD: `Names.collect()` (careful with this)

Names		
Partition 1	Partition 2	Partition 3
Mokhtar Jacques Dirk	Cindy Dan Susan	Dirk Frank Jacques

RDDs

- **Two types of operations**

- **Transformations ~ DDL (Create View V2 as...)**

- val rddNumbers = sc.parallelize(1 to 10): Numbers from 1 to 10
 - val rddNumbers2 = rddNumbers.map (x => x+1): Numbers from 2 to 11
 - The LINEAGE on how to obtain rddNumbers2 from rddNumber is recorded
 - It's a Directed Acyclic Graph (DAG)
 - No actual data processing does take place → **Lazy evaluations, no Execution**

- **Actions ~ Select (Select * From V2...)**

- rddNumbers2.collect(): Array [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
 - Performs transformations and action
 - Returns a value (or write to a file) → **Executes on Cluster**

- **Fault tolerance**

- If data in memory is lost it will be recreated from lineage

Code Execution (1)

- ‘spark-shell’ provides Spark context as ‘sc’

```
// Create RDD
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")
// Transformations
val danQuotes = quotes.filter(_.startsWith("DAN"))
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))
// Action
danSpark.filter(_.contains("Spark")).count()
```

File: sparkQuotes.txt

```
DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible
```

Code Execution (2)

```
// Create RDD  
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")  
  
// Transformations  
val danQuotes = quotes.filter(_.startsWith("DAN"))  
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))  
  
// Action  
danSpark.filter(_.contains("Spark")).count()
```

File: sparkQuotes.txt

DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible

RDD: quotes



Code Execution (3)

```
// Create RDD  
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")  
  
// Transformations  
val danQuotes = quotes.filter(_.startsWith("DAN"))  
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))  
  
// Action  
danSpark.filter(_.contains("Spark")).count()
```

File: sparkQuotes.txt

DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible

RDD: quotes



RDD: danQuotes



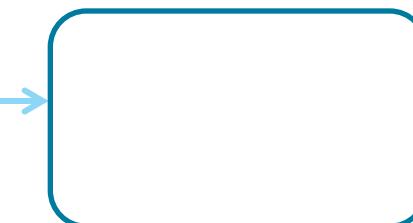
Code Execution (4)

```
// Create RDD  
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")  
  
// Transformations  
val danQuotes = quotes.filter(_.startsWith("DAN"))  
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))  
  
// Action  
danSpark.filter(_.contains("Spark")).count()
```

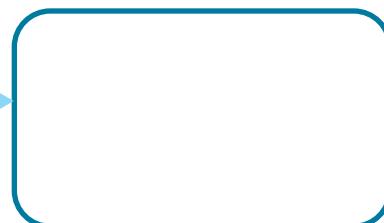
File: sparkQuotes.txt

DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible

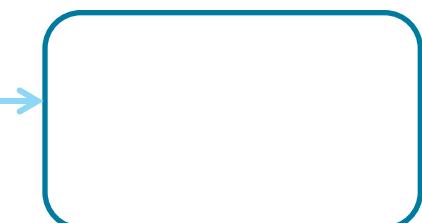
RDD: quotes



RDD: danQuotes



RDD: danSpark



Code Execution (5)

```
// Create RDD  
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")  
  
// Transformations  
val danQuotes = quotes.filter(_.startsWith("DAN"))  
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))  
  
// Action  
danSpark.filter(_.contains("Spark")).count()
```

File: sparkQuotes.txt

DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible

RDD: quotes

DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible

RDD: danQuotes

DAN Spark is cool
DAN Scala is awesome

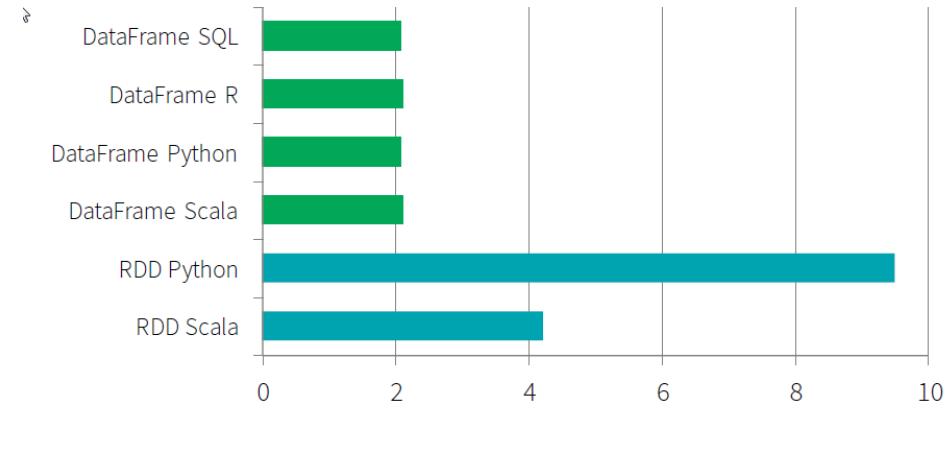
RDD: danSpark

Spark
Scala

1

SparkSQL, DataFrames and DataSets

- **A rich set of functionality that allows “Database-like” processing**
- **Share single optimizer, called “Catalyst” (at the driver)**
 - An open-source extensible query optimizer
- **Because it is the same engine, it has exactly the same performance for different APIs**
 - And performance is much better than for RDD
- **Much less code**
- **All SparkSQL, DF, and DataSets are essentially using the same engine**



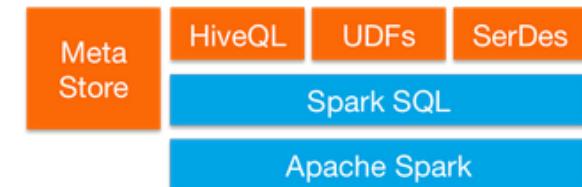
Time to aggregate 10 million integer pairs (in seconds)

3

Picture credit: databricks.com

SparkSQL

- Provide for relational queries expressed in SQL, HiveQL using Scala, Python, and Java API's
- Seamlessly mix SQL queries with Spark programs
- Dataframes provide a single interface for efficiently working with structured data including Apache Hive, Parquet and JSON files
- Graduated from alpha status with Spark 1.3
 - DataFrames API marked as experimental in 2013
- Standard connectivity through JDBC/ODBC



Spark Streaming



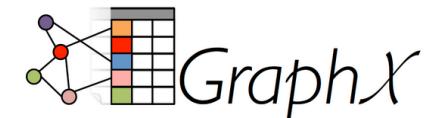
- **Component of Spark**
 - Project started in 2012
 - First alpha release in Spring 2013
 - Out of alpha with Spark 0.9.0
 - More enhancements targeted for Spark 2.0
- **Discretized Stream (DStream) programming abstraction**
 - Represented as a sequence of RDDs (micro-batches)
 - RDD: set of records for a specific time interval
 - Supports Scala, Java, and Python (with limitations)
- **Fundamental architecture: batch processing of datasets**



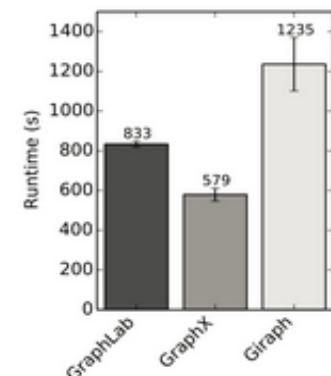
Spark MLlib

- **Spark MLlib for machine learning library**
 - Marked as under active development
- **Provides common algorithm and utilities**
 - Classification
 - Regression
 - Clustering
 - Collaborative filtering
 - Dimensionality reduction
- **Leverages iteration and yields better results than one-pass approximations sometimes used with MapReduce**

Spark GraphX



- **Flexible Graphing**
 - GraphX unifies ETL, exploratory analysis, and iterative graph computation
 - You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms with the API
 - GraphFrames - graph library based on Data Frames
- **Speed**
 - Comparable performance to the fastest specialized graph processing systems.
- **Algorithms**
 - Choose from a growing library of graph algorithms
 - In addition to a highly flexible API, GraphX comes with a variety of graph algorithms



Spark R

- **Spark R is an R package that provides a light-weight front-end to use Apache Spark from R**
- **Spark R exposes the Spark API through the RDD class and allows users to interactively run jobs from the R shell or RStudio on a cluster.**
- **Goals**
 - Make Spark R production ready
 - Efforts from AlteryX and DataBricks
 - Integration with MLlib
 - Consolidations to the data frame and RDD concepts