Accelerating Digital Innovation w/ Data Science in the Cloud

Power of data. Simplicity of design.

Speed of innovation.

Data Science Workshop

Schedule:

9:00 – 10:00
Kickoff – Introduction
Overview of Data Science – Digital Acceleration

10:00 - 11:30
Lab 1 – Intro to Coding w/ Spark, Python and Juypter Notebooks

11:30 - 12:00
Overview of Machine Learning

12:00 – 1:00
Lunch

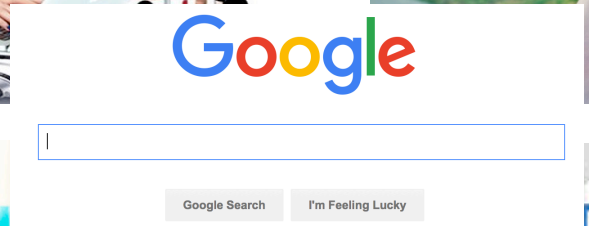
1:00 – 2:00
Lab 2 - Create ML Model & Deployment – Titanic Analysis

2:00 – 4:00
Lab 3 – Notebooks - Breast Cancer Analysis

4:00 – 4:30
Wrap up, Open Discussion

Material:
https://ibm.box.com/v/IBMDataScience-Workshop/

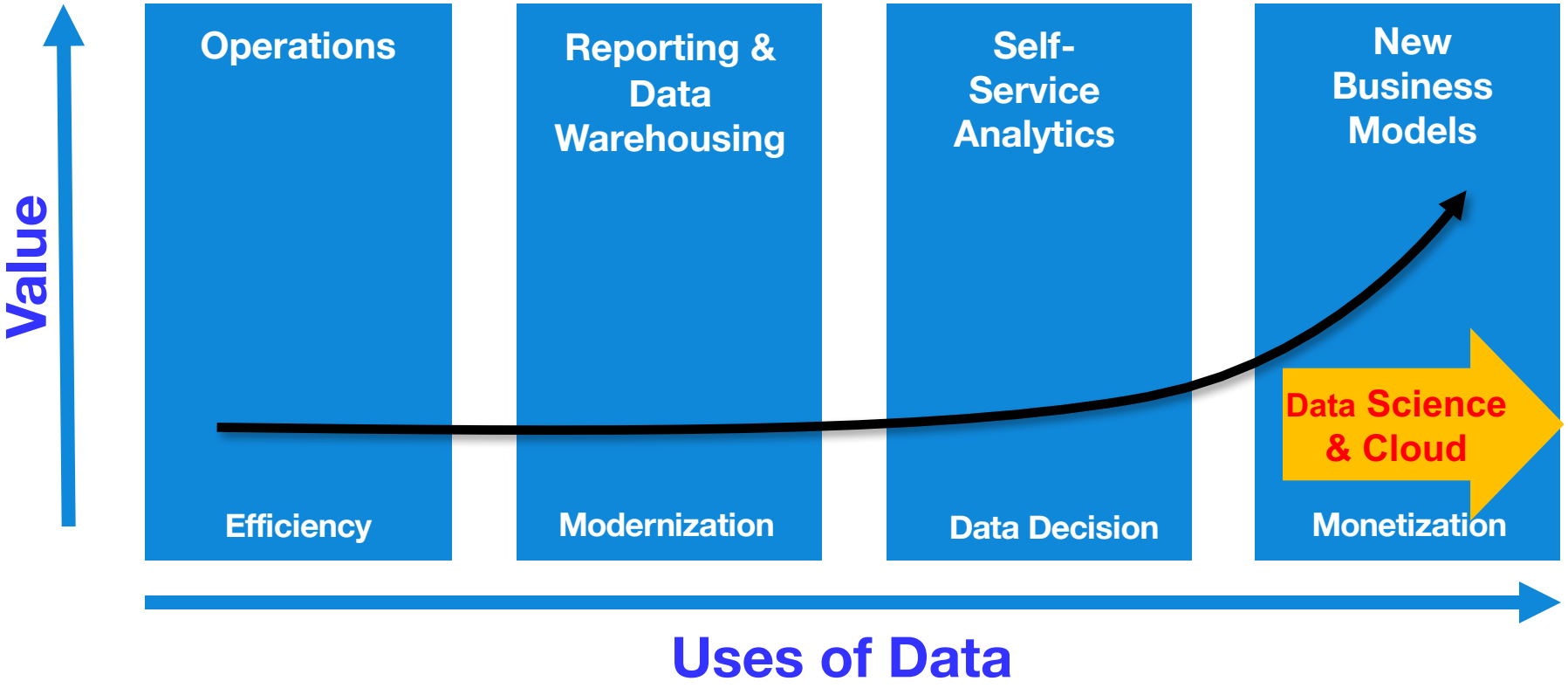
The digital age changed the way we Live, Play, Learn and Work…



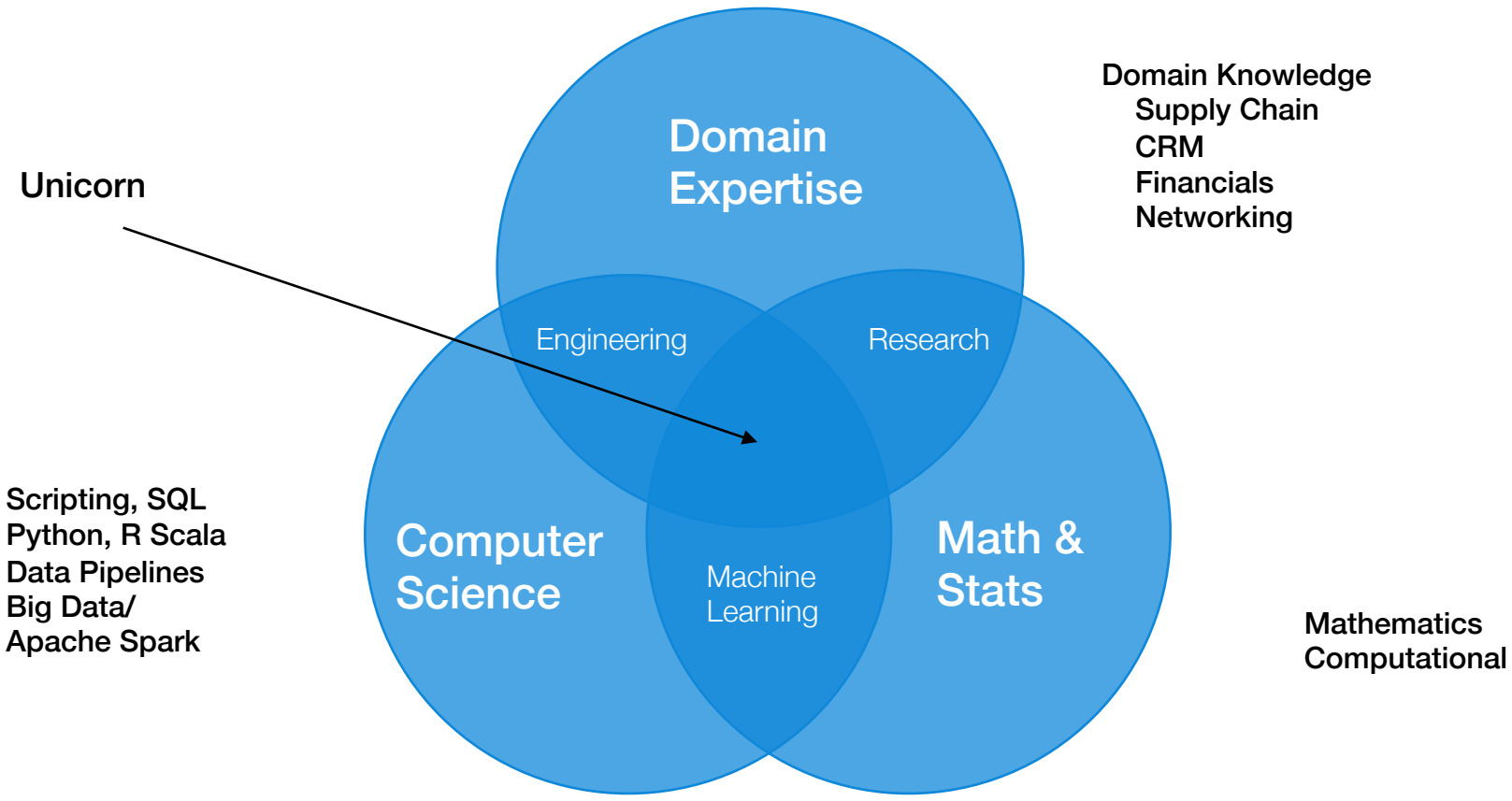
Transformation is Critical…

Companies must shift to a **Data-Driven** Business

Making the shift to a Data-Driven Organization Accelerated by Data Science in the Cloud

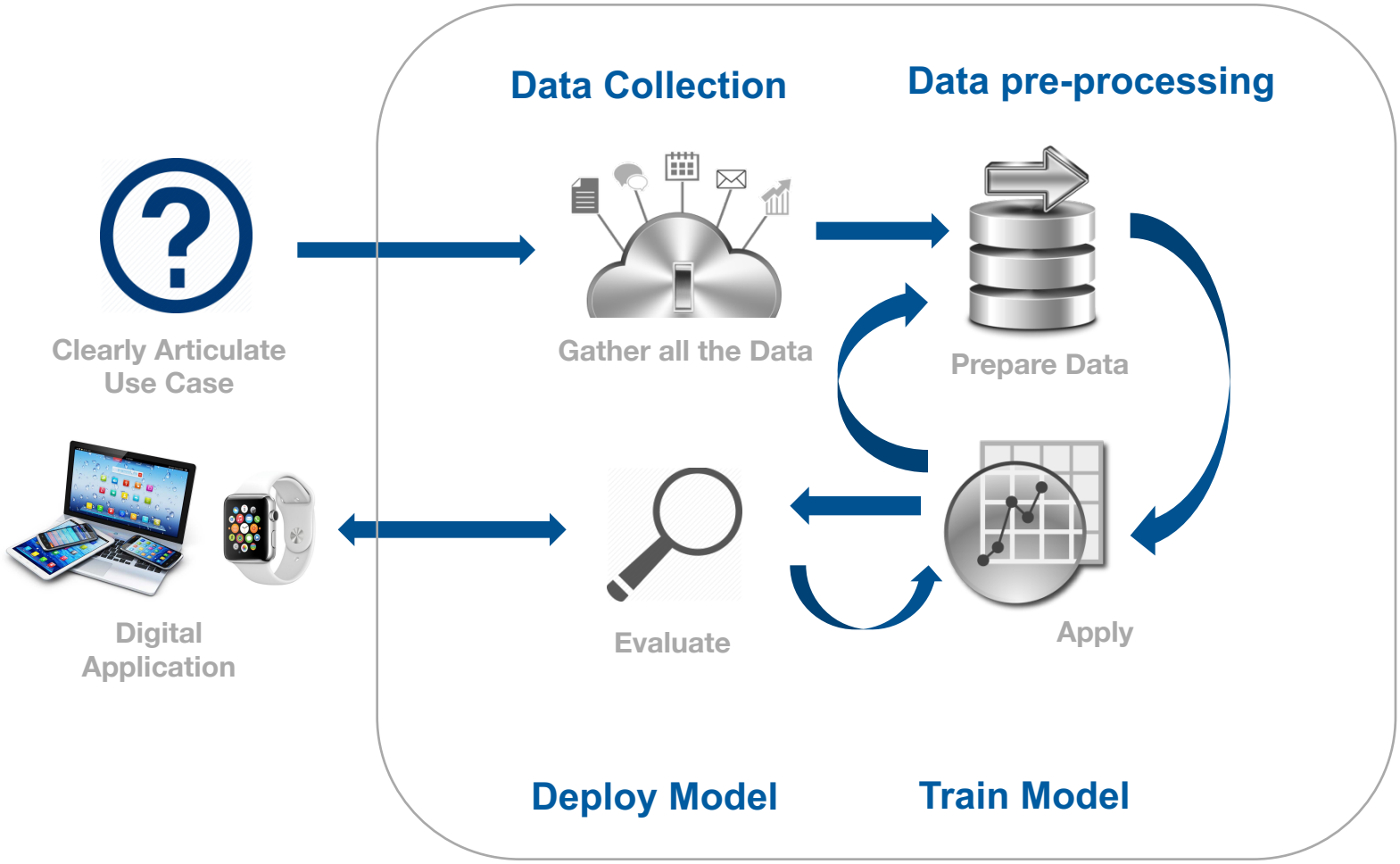


What is Data Science?



Data Science Projects Require multiple Skills

Steps to apply Data Science in Cloud…



Google Trends - *Data Science Languages*

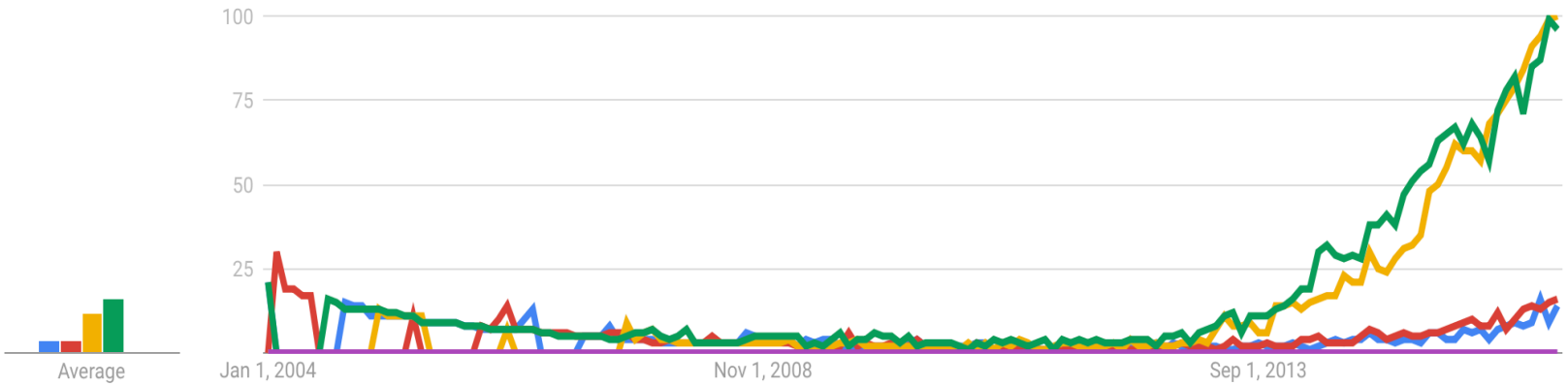
SPSS

SAS

Python

R

Scala



Convergence of Big Data and Data Science…

{   } R

{   } Big Data

{   } Python

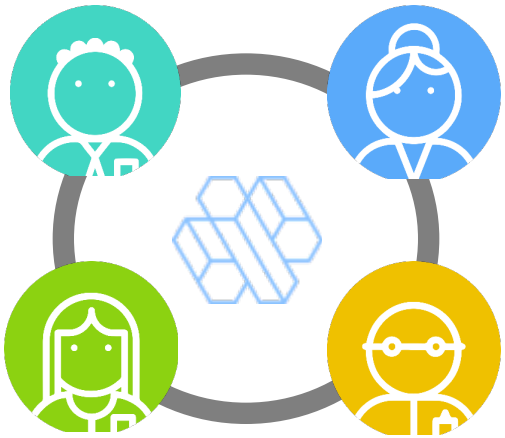
The Data Scientist - Sexiest Job in 21st Century, Also the toughest Jobs!

- **Rigid toolset**
  - Have to choose only one approach
  - Cannot easily connect all of the capabilities needed
  - Difficult to navigate between the various tools used
- **Fragmented and time consuming**
  - Using multiple disjointed environments
  - Separate on-ramp/community for each tool/environment
  - Does not have meta data or data lineage
- **Analytical Silo**
  - Difficult to maintain and version control project assets
  - Limited means of collaborating with team
  - Results are difficult to share

Data Science is a Team Sport.

Data Scientists

Business Analysts

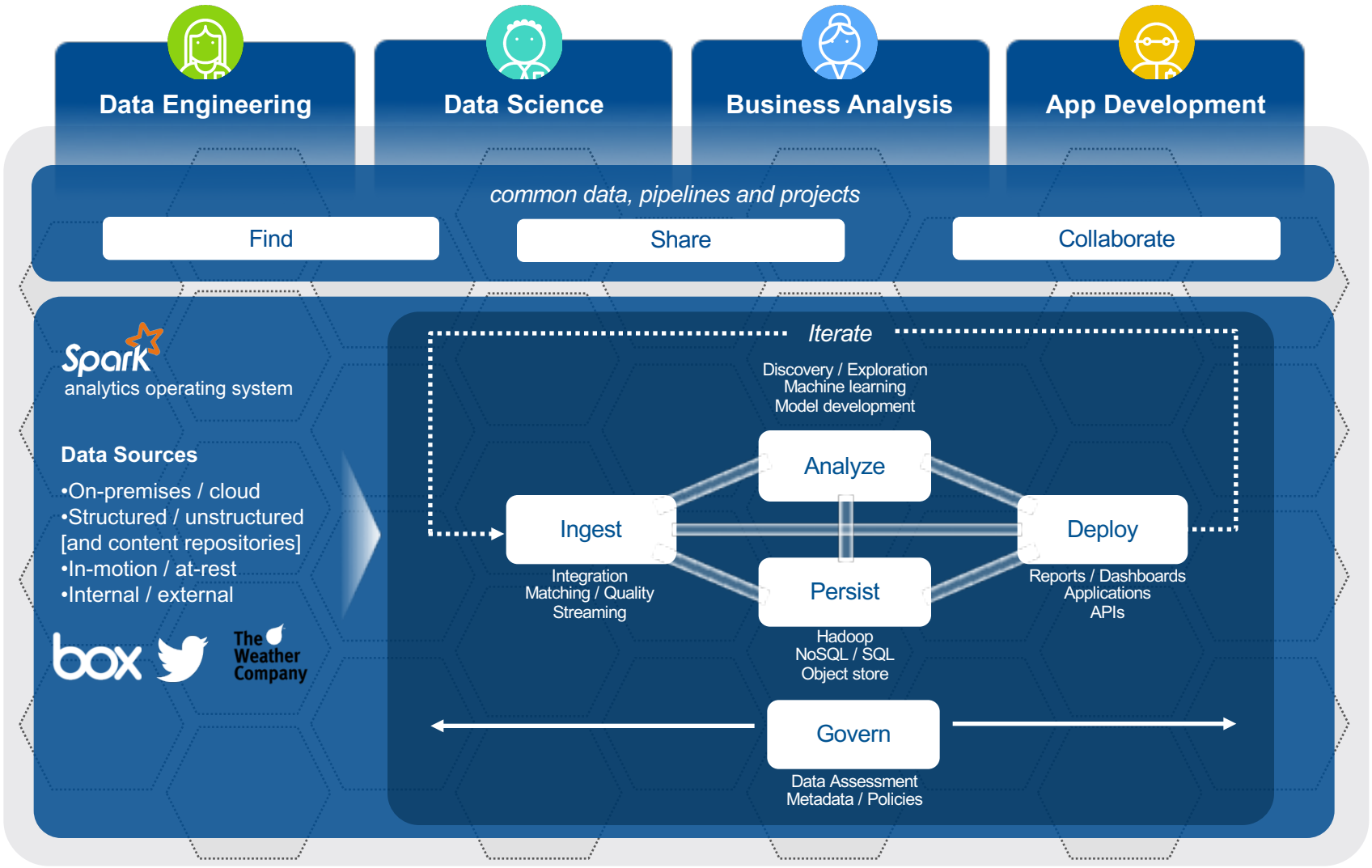


Data Engineers

Application Developers

IBM Watson Data Platform

Connects Users to Data and Analytics



IBM DATA SCIENCE EXPERIENCE

ALL YOUR TOOLS IN ONE PLACE

IBM Data Science Experience provides an environment that brings together everything that a Data Scientist needs. It includes the most popular Open Source tools and IBM unique value-add functionalities with community and social features, integrated as a first class citizen to make Data Scientists more successful.

Learn

Built-in learning to get started or go the distance with advanced tutorials

Create

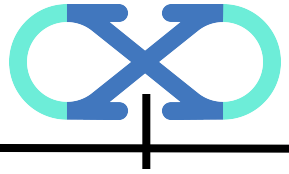
The best of open source and IBM value-add to create state-of-the-art data products

Collaborate

Community and social features that provide meaningful collaboration

URL: http://datascience.ibm.com

Core Attributes of the Data Science Experience



Community

- Find tutorials and datasets
- Connect with Data Scientists
- Ask questions
- Read articles and papers
- Fork and share projects

Open Source

- Code in Scala/Python/R/SQL
- Jupyter and Zeppelin* Notebooks
- RStudio IDE and Shiny apps
- Apache Spark
- Your favorite libraries

IBM Added Value

- Data Shaping/Pipeline UI *
- Auto-data preparation and modeling*
- Advanced Visualizations*
- Model management and deployment*
- Documented Model APIs*

Powered by IBM **IBM Watson Platform** in the Cloud

* DSX product roadmap items

IBM Data Science Experience

Click here to watch video

IBM is Leading Data Science…

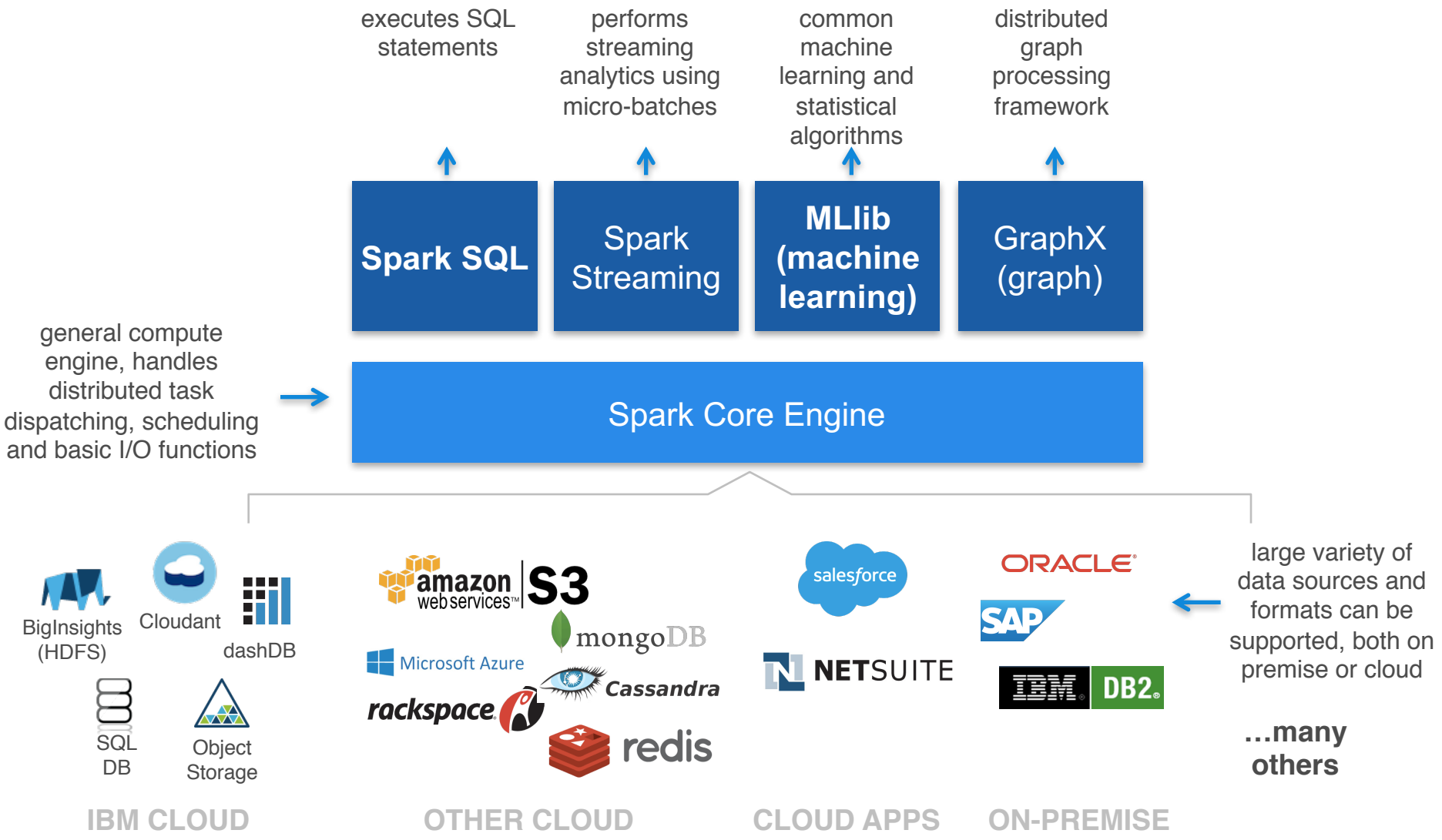
Lab 1 – Coding with Apache Spark
Deeper look at Apache Spark

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**"

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



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%

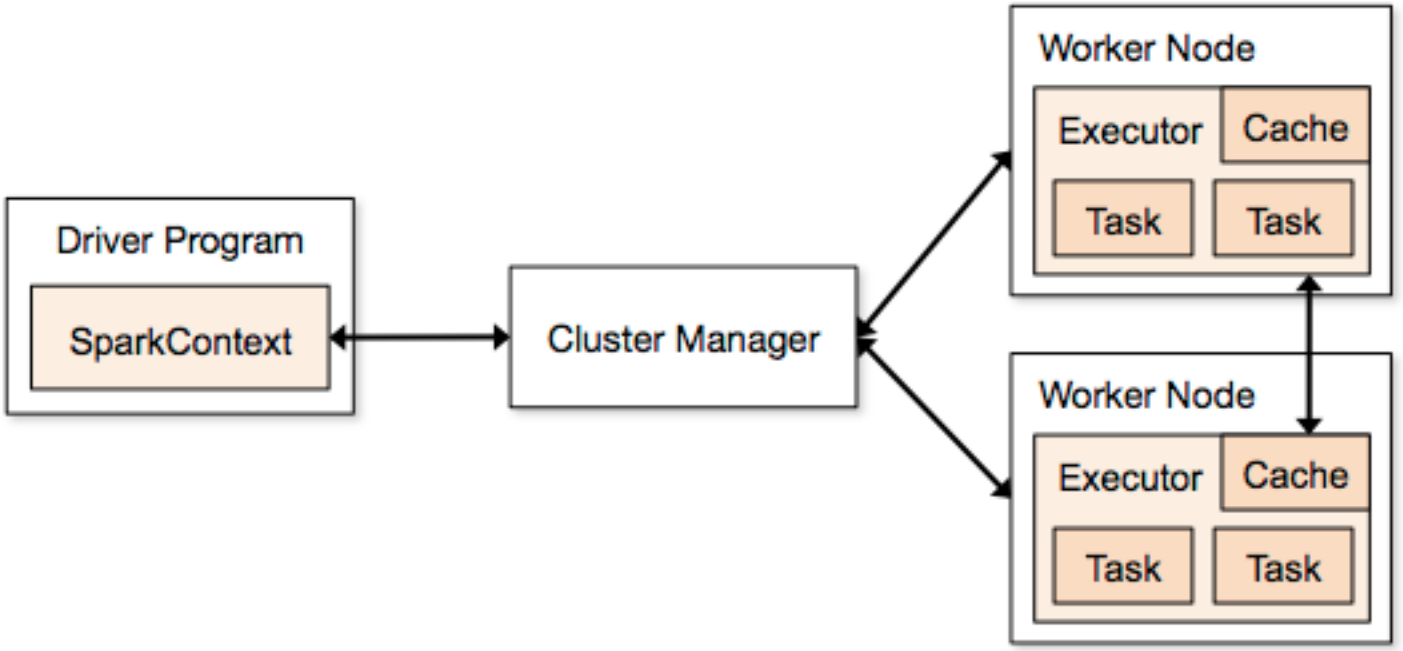
Survey done by Databricks, Summer 2015

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

Spark Application Architecture

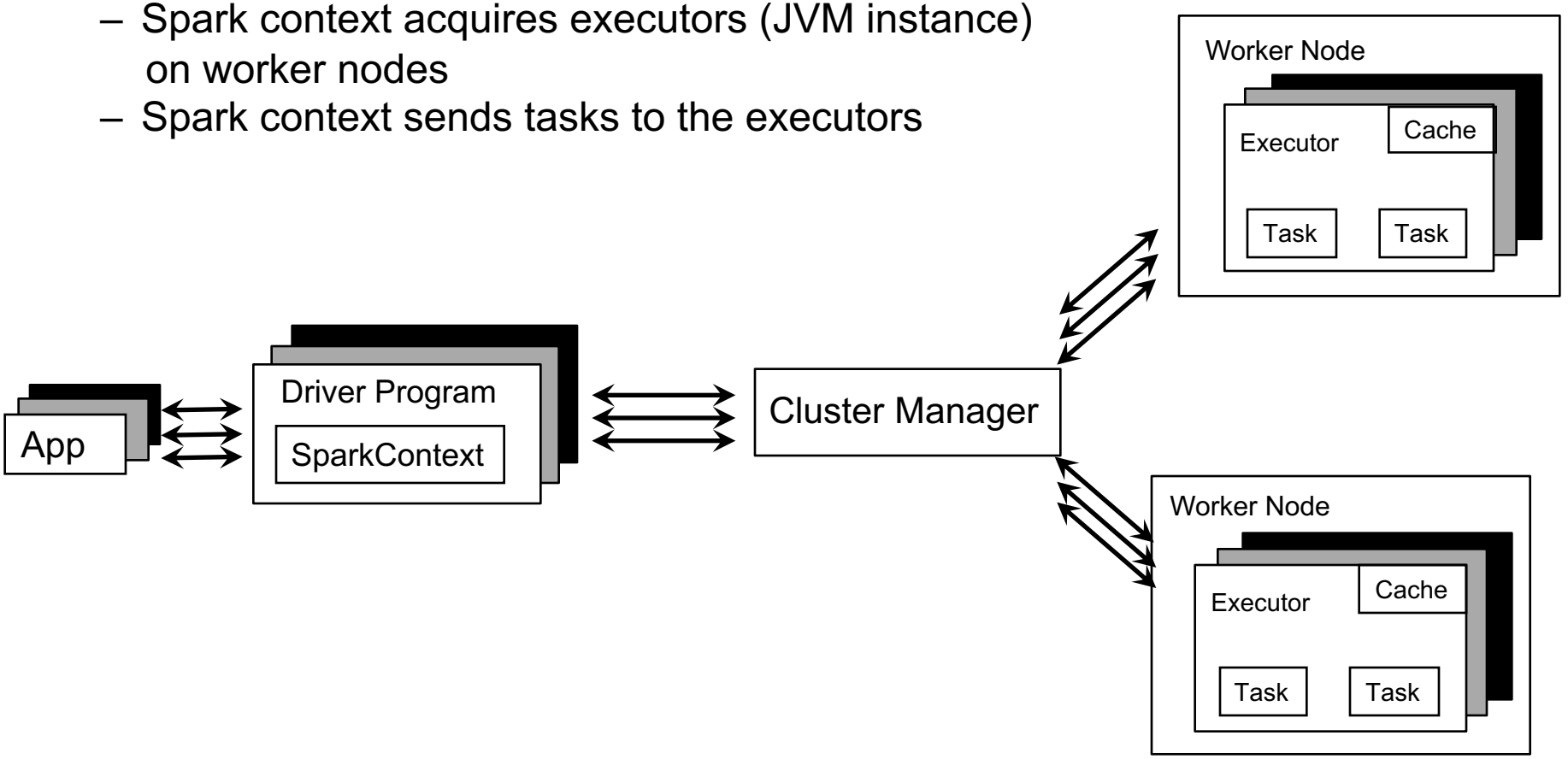
- **A Spark application is initiated from a driver program**

- **Spark execution modes:**
  - Standalone with the built-in cluster manager
  - Use Mesos as the cluster manager
  - Use YARN as the cluster manager
  - Standalone cluster on any cloud (BlueMix, IBM Softlayer, Amazon, Azure, …)



Showing multiple applications

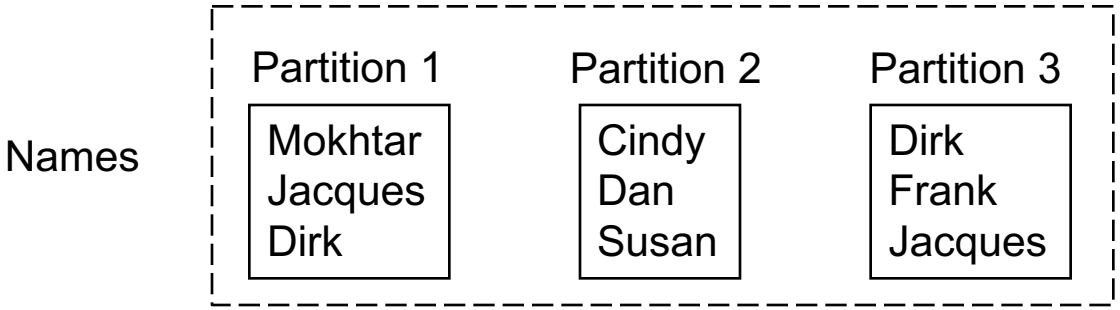
- Each Spark application runs as a set of processes coordinated by the Spark context object (driver program)
  - Spark context connects to Cluster Manager (standalone, Mesos/Yarn)
  - Spark context acquires executors (JVM instance) on worker nodes
  - Spark context sends tasks to the executors



Resilient Distributed Datasets

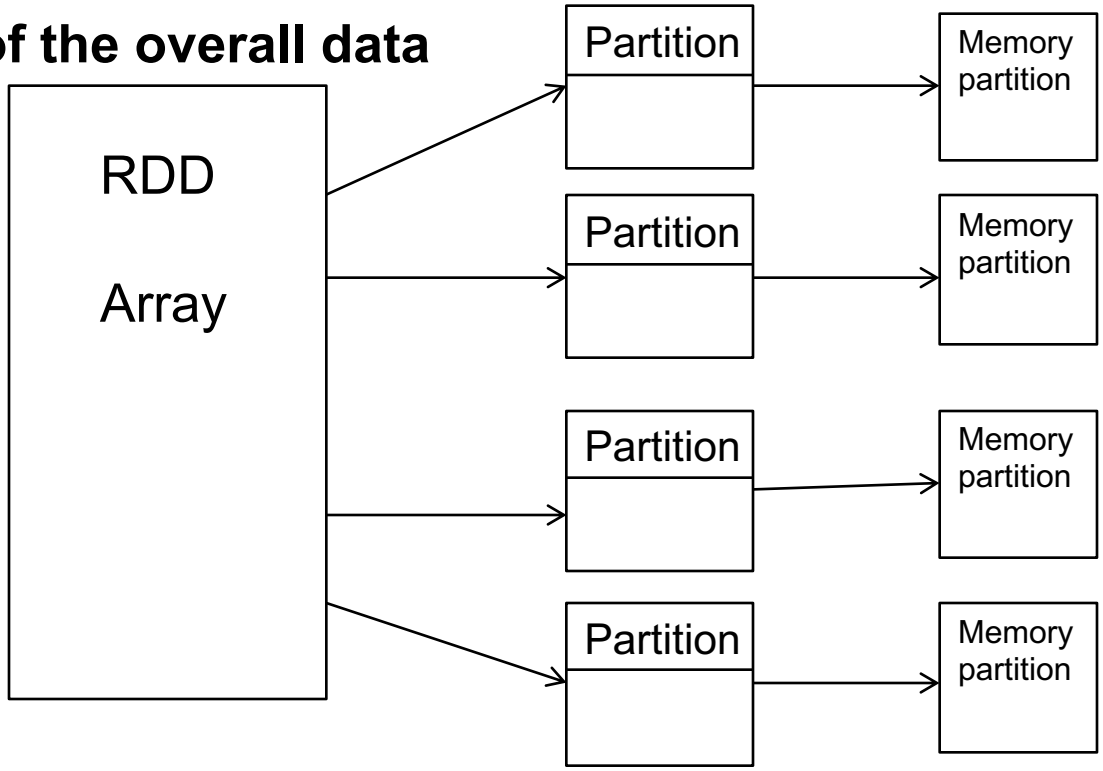
- **An RDD is a distributed collection of Scala/Python/Java objects of the same type:**
  - RDD of strings
  - RDD of integers
  - RDD of (key, value) pairs
  - RDD of class Java/Python/Scala objects
- **It can help to think of an RDD as a view**

- **An RDD is physically distributed across the 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 Dataset

- **RDDs are immutable**
  - Modifications create new RDDs
- **Fault tolerance**
  - If data in memory is lost it will be recreated from lineage
- **Holds references to partition objects**
- **Each partition is a subset of the overall data**
- **Partitions are assigned to nodes on the cluster**
- **Partitions are in memory by default**
- **RDDs keep information on their lineage**



Resilient Distributed Datasets

- **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**
    - 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**
  - **Actions ~ Select (Select * From V2…)**
    - **rddNumbers2.collect(): Array [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]**
    - Performs list of transformations and THE action
    - Returns a value (or write to a file)
- **Fault tolerance**
  - If data in memory is lost it will be recreated from lineage

Code Execution (1)

- 'spark-shell' provides Spark context as 'sc'

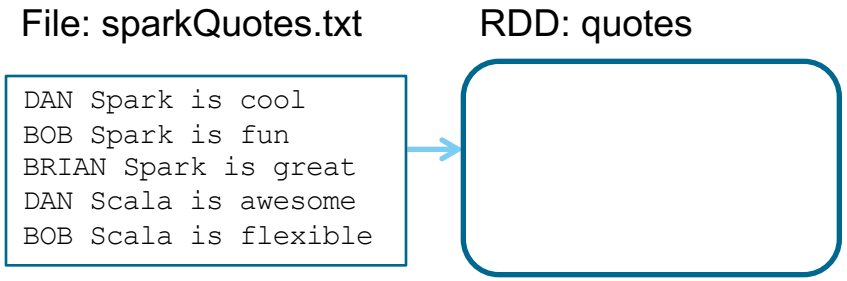
```
// Create RDD
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")
// Transformations
val danQuotes = quotes.filter(_.startsWith("DAN"))
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))
// Action
danSpark.filter(_.contains("Spark")).count()
```

File: sparkQuotes.txt

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

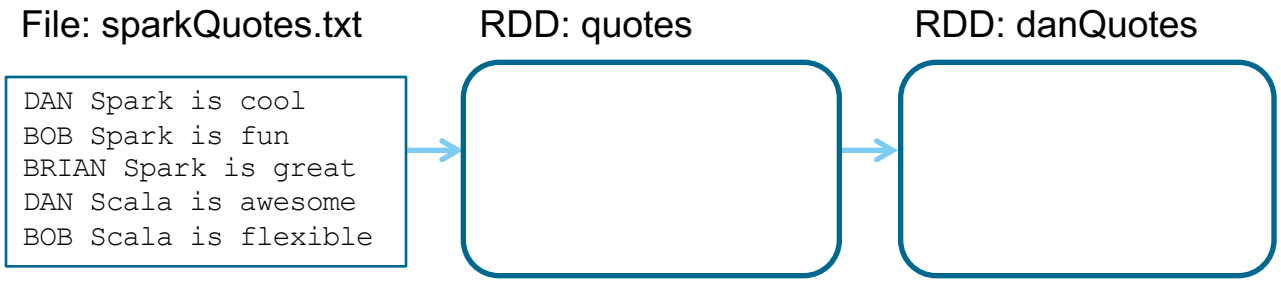
Code Execution (2)

```
// 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()
```



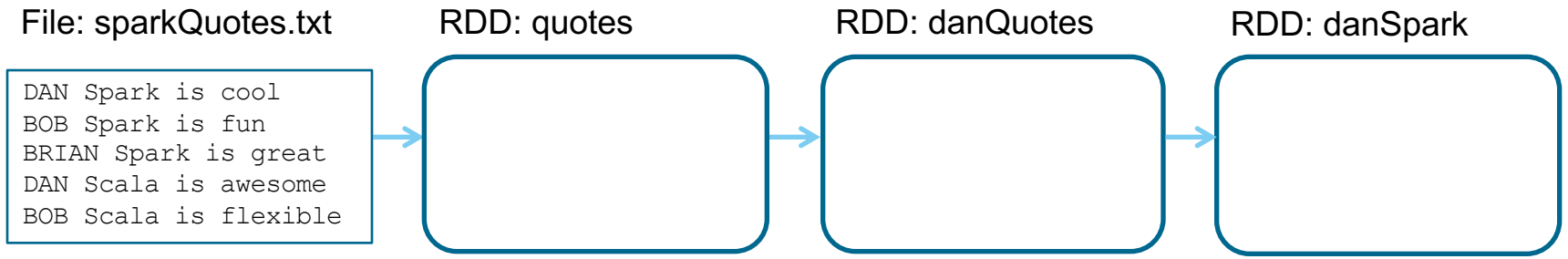
Code Execution (3)

```
// 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()
```



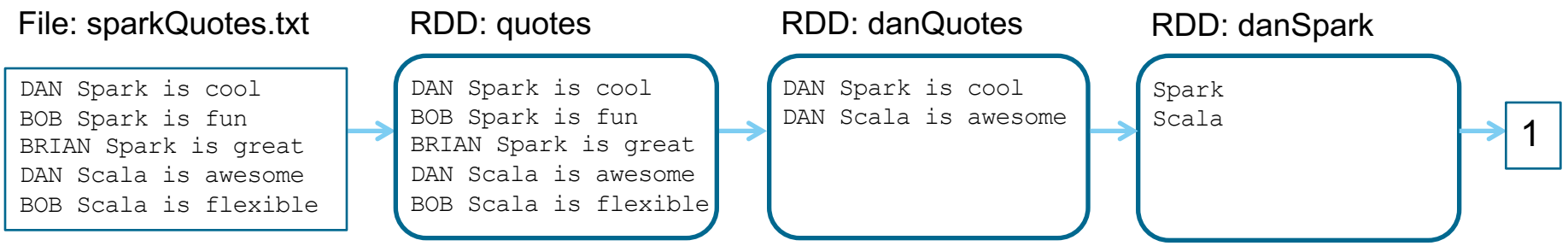
Code Execution (4)

```
// 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()
```

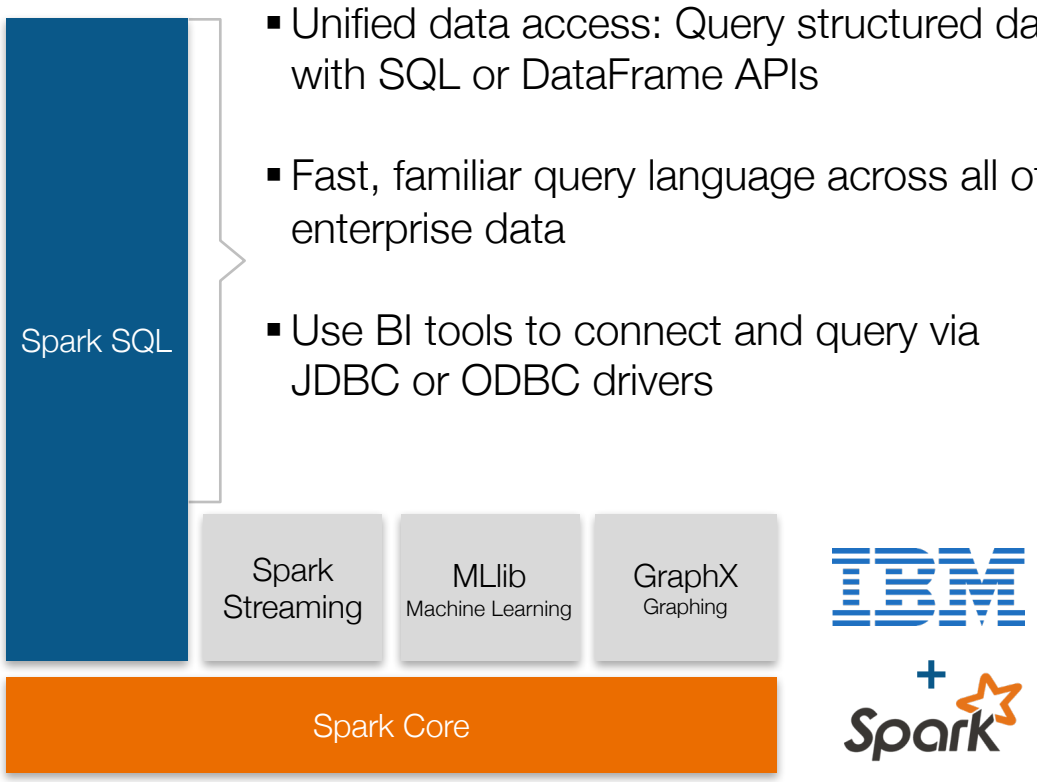


Code Execution (5)

```
// 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()
```



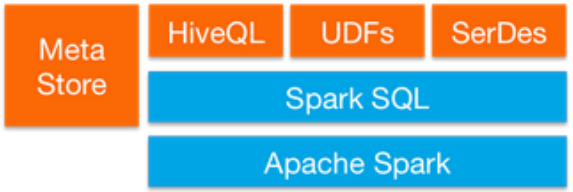
Closer Look at APIs - SQL

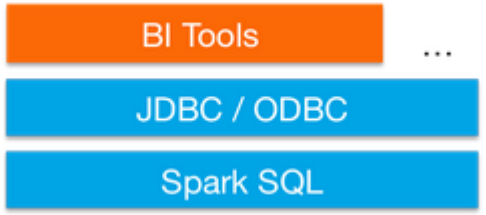


- Unified data access: Query structured data sets with SQL or DataFrame APIs
- Fast, familiar query language across all of your enterprise data
- Use BI tools to connect and query via JDBC or ODBC drivers

SparkSQL

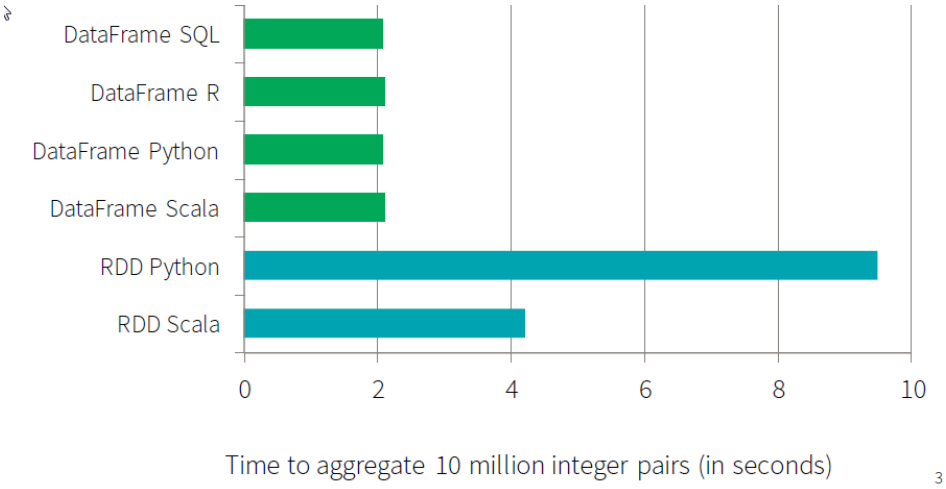
- **Provide for relational queries expressed in SQL, HiveQL using Scala, Python, and Java API's**
- **Seamlessly mix SQL queries with Spark programs**
- **Dataframes provide a single interface for efficiently working with structured data including Apache Hive, Parquet and JSON files**
- **Graduated from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





SparkSQL, DataFrames and DataSets

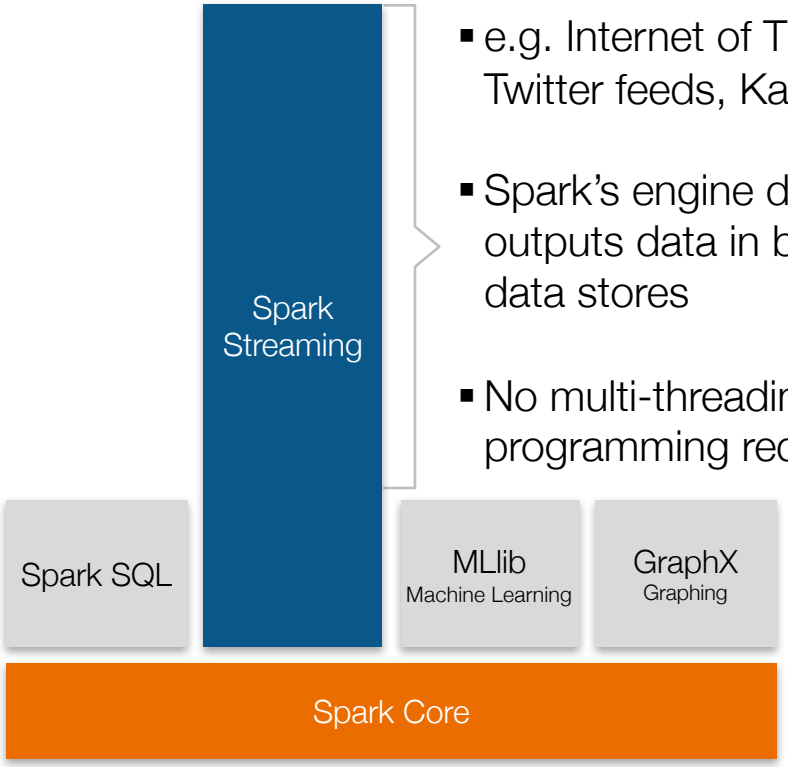
- **A rich set of functionality that allows "Database-like" processing**
- **Share single optimizer, called "Catalyst" (at the driver)**
  - An open-source extensible query optimizer
- **Because it is the same engine, it has exactly the same performance for different APIs**
  - And performance is much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



Picture credit: databricks.com

Closer Look at APIs - Streaming

- Micro-batch event processing for near-real time analytics
- e.g. Internet of Things (IoT) devices, Twitter feeds, Kafka (event hub), etc.
- Spark's engine drives some action or outputs data in batches to various data stores
- No multi-threading or parallel process programming required



Spark Streaming

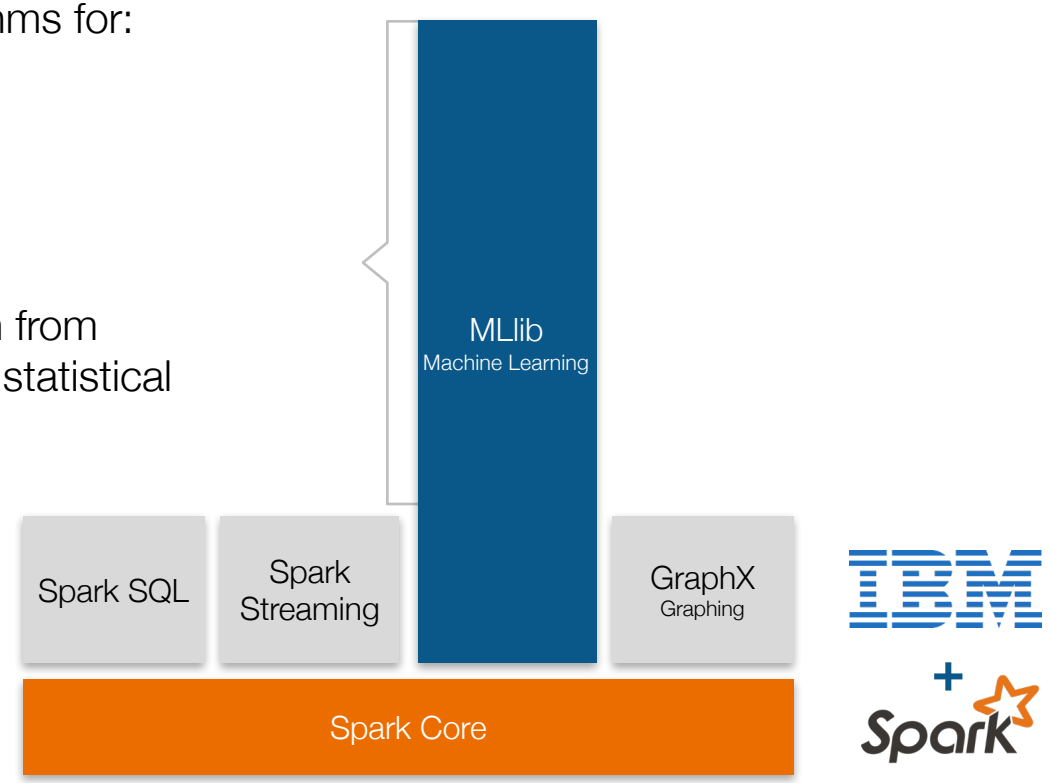
- **Component of Spark**
  - Project started in 2012
  - First alpha release in Spring 2013
  - Out of alpha with Spark 0.9.0
  - More enhancements targeted for Spark 2.0

- **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**

Closer Look at APIs – Machine Learning

- Predictive and prescriptive analytics
- Machine learning algorithms for:
  - Clustering
  - Classification
  - Regression
  - etc.
- Smart application design from pre-built, out-of-the-box statistical and algorithmic models

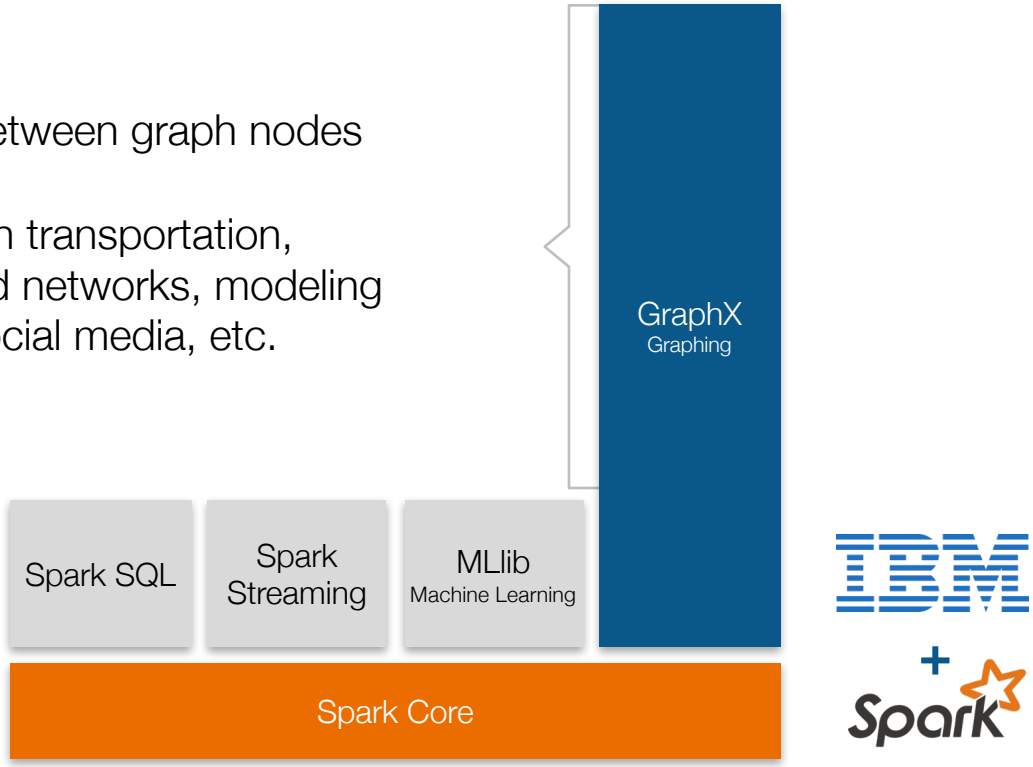


Spark MLlib

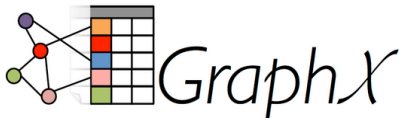
- **Spark MLlib for machine learning library**
  - Marked as under active development
- **Provides common algorithm and utilities**
  - Classification
  - Regression
  - Clustering
  - Collaborative filtering
  - Dimensionality reduction
- **Leverages iteration and yields better results than one-pass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

Closer Look at APIs – Graph DB

- Represent and analyze systems represented by graph nodes
- Trace interconnections between graph nodes
- Applicable to use cases in transportation, telecommunications, road networks, modeling personal relationships, social media, etc.



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