Go to www.menti.com and use the code 90 79 41

Accelerating Digital Innovation w/ Data Science in Cloud

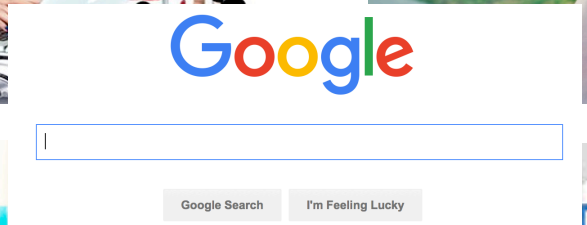
Power of data. Simplicity of design. Speed of innovation.

Agenda

9:00 – 10:00	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

https://github.com/joshishwetha/DataScience_Cloud_Workshop /

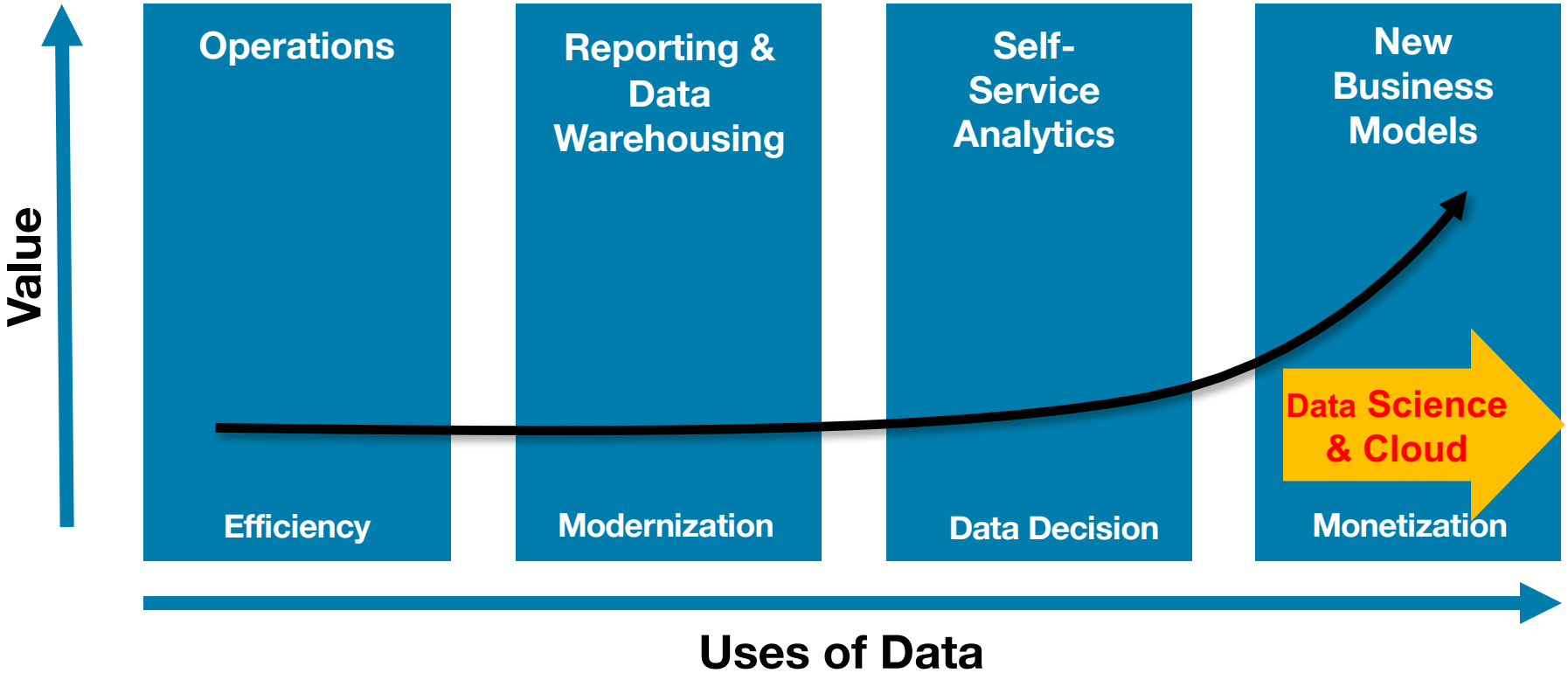
The digital age changed the way we Live, Play, Learn and Work…



Transformation is Critical…

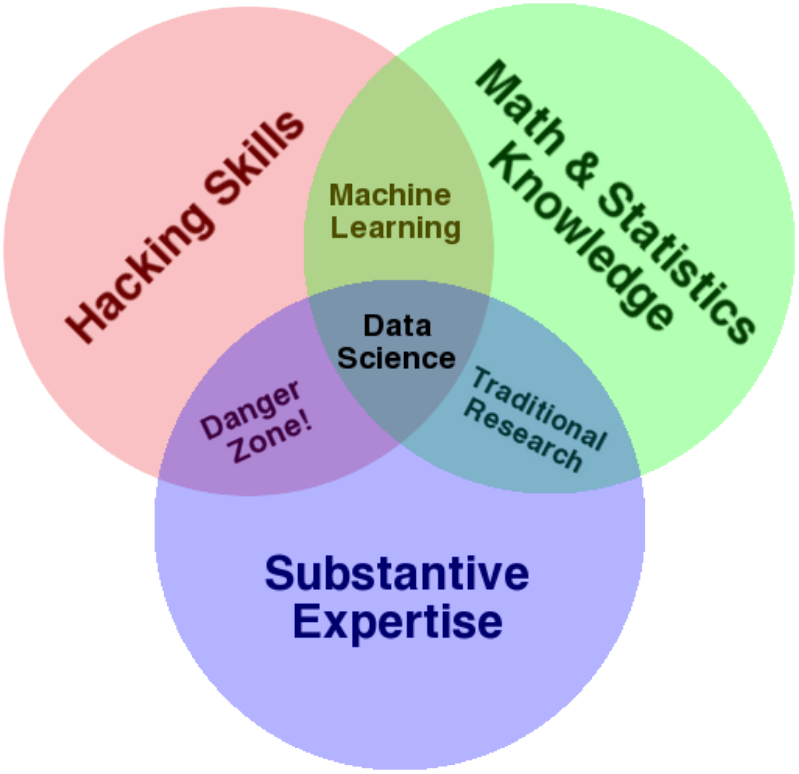
Companies must shift to a **Data-Driven** Business

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



What is data science?

Scripting, SQL
Python, R Scala
Data Pipelines
Big Data/
Apache Spark

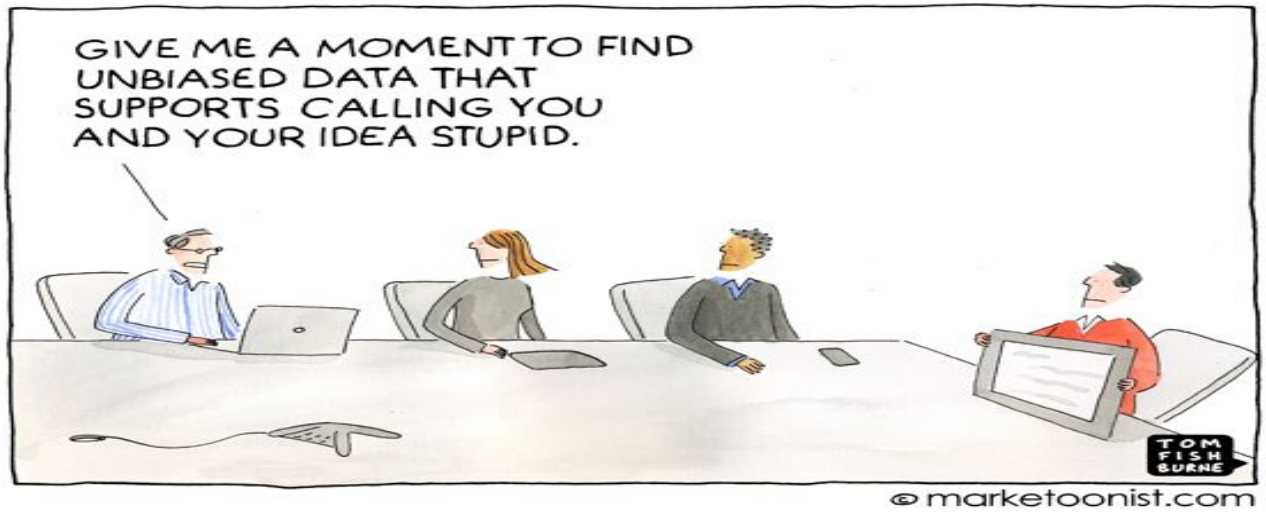


Mathematics
Computational

Domain Knowledge
Supply Chain
CRM
Financials
Networking

Why do we need it?

- We need Data Science to make valid, objective inferences from data, free from human bias – KDNuggets



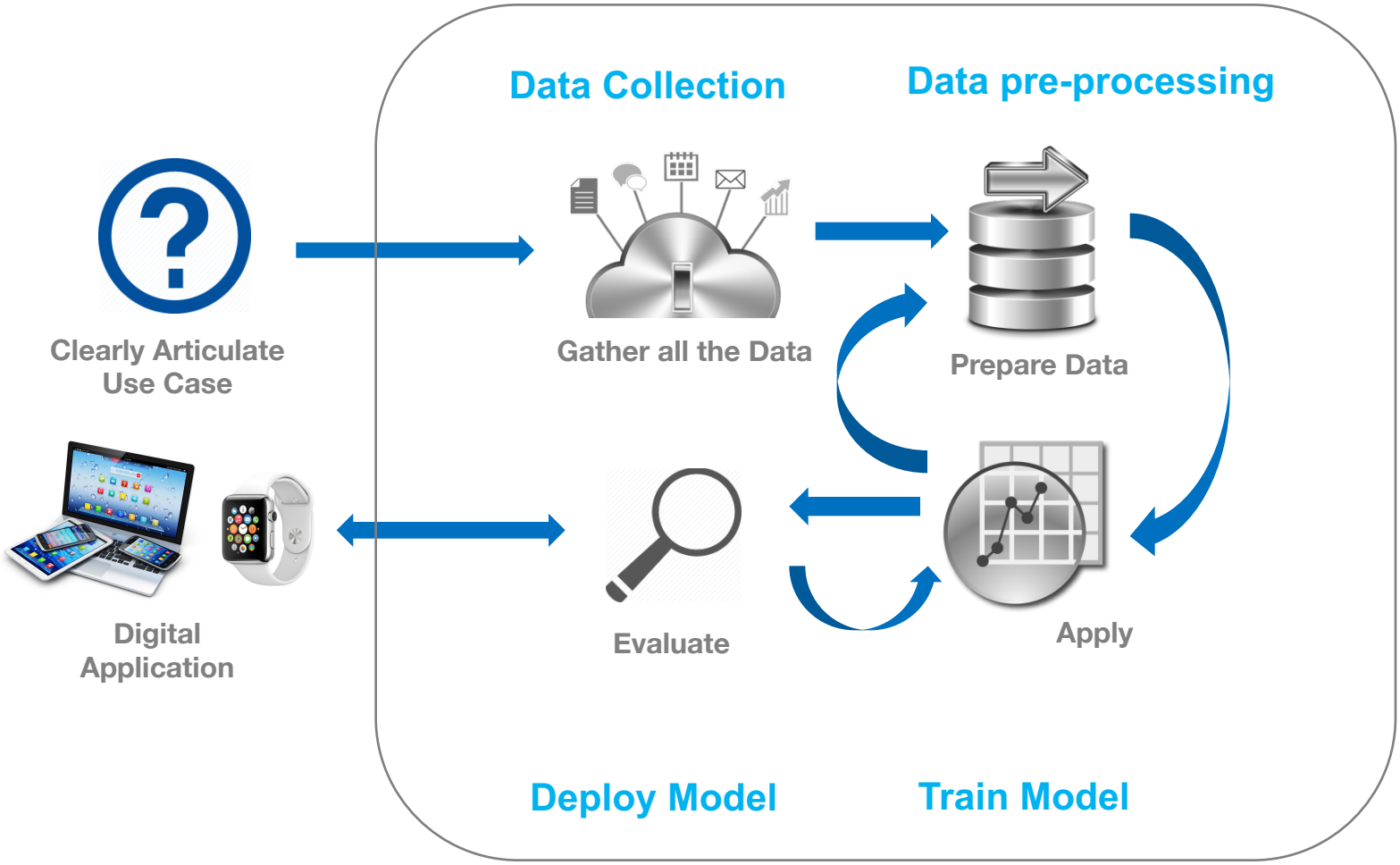
What do we need to practice data science?

- Technology / Technology stack
- Platform / Tool(s)
- Data
- Expertise

What do we need to practice data science?

- **Technology / Technology stack**
  - Python, Spark
- **Platform / Tool(s)**
  - Data Science Experience
- **Data**
  - UCI
- **Expertise**
  - Data Scientists, Data Engineers, Application Developers

Steps to apply Data Science in Cloud…



Google Trends - *Data Science Languages*

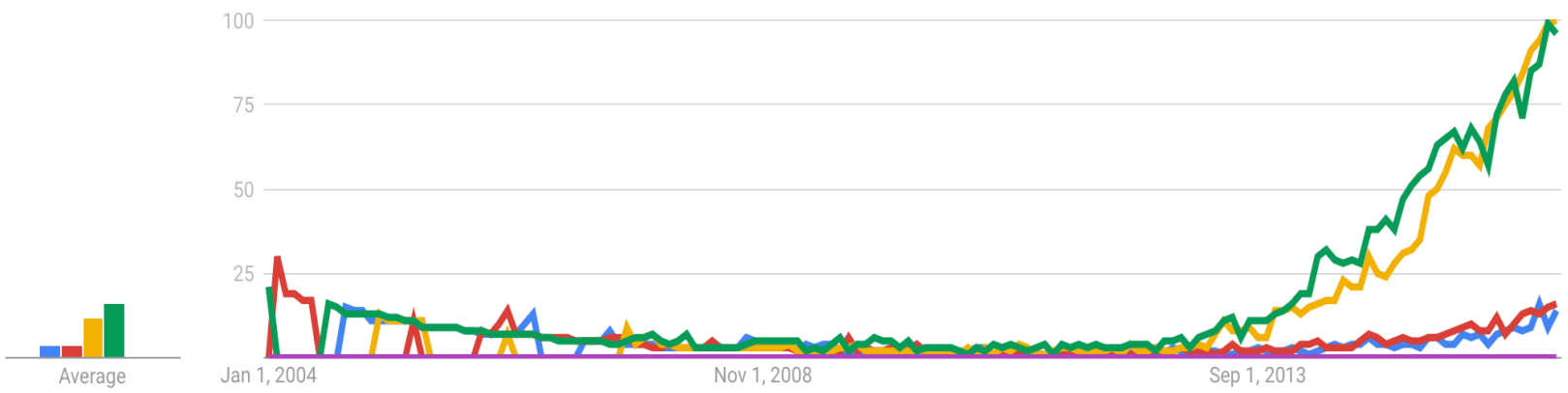
SPSS

SAS

Python

R

Scala



Trends in Google Searches (September 2nd 2016)

Convergence of Big Data and Data Science…

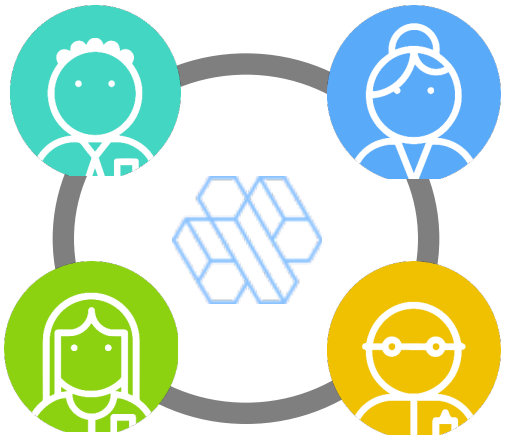
{   } R

{   } Big Data

{   } Python

Data Science is a Team Sport.

Data Scientists



Business Analysts

Data Engineers

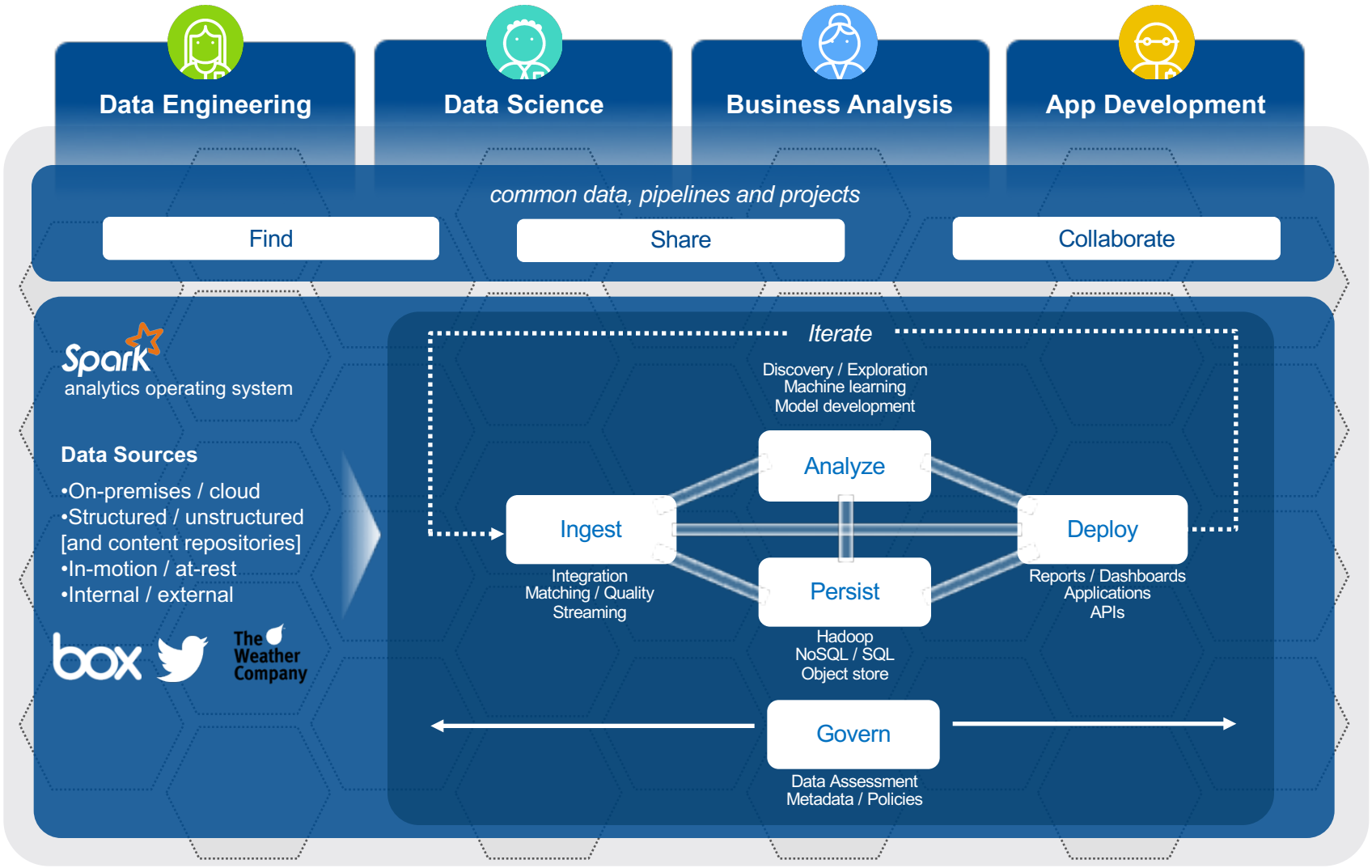
Application Developers

Data Scientist - Current Challenges and Pain Points

- **Rigid toolset**
  - Difficult to navigate between the various tools used
- **Fragmented and time consuming**
  - Using multiple disjoint environments
  - Separate on-ramp/community for each tool/environment
- **Analytical Silo**
  - Difficult to maintain and version control project assets

IBM Watson Data Platform

Connects Users to Data and Analytics



Data Science Experience

Learn

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

Create

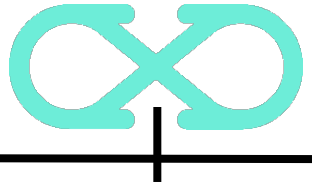
The best of open source and IBM value-add to create state-of-the-art data products

Collaborate

Community and social features that provide meaningful collaboration

http://datascience.ibm.com

Core Attributes of the Data Science Experience



IBM Data Science Experience

Community	Open Source	IBM Added Value
• Find tutorials and datasets	• Code in Scala/Python/R/SQL	• IBM Machine Learning*
• Connect with Data Scientists	• Jupyter Notebooks	• SPSS Modeler Canvas*
• Ask questions	• RStudio IDE and Shiny	• Prescriptive Analytics - DOcplexcloud
• Read articles and papers	• Apache Spark	• Projects and Version Control
• Fork and share projects	• Your favorite libraries	• Managed Spark Service

Powered by IBM **Watson Data Platform**

* Closed beta

IBM Data Science Experience

Click here to watch video

IBM is Leading Data Science…

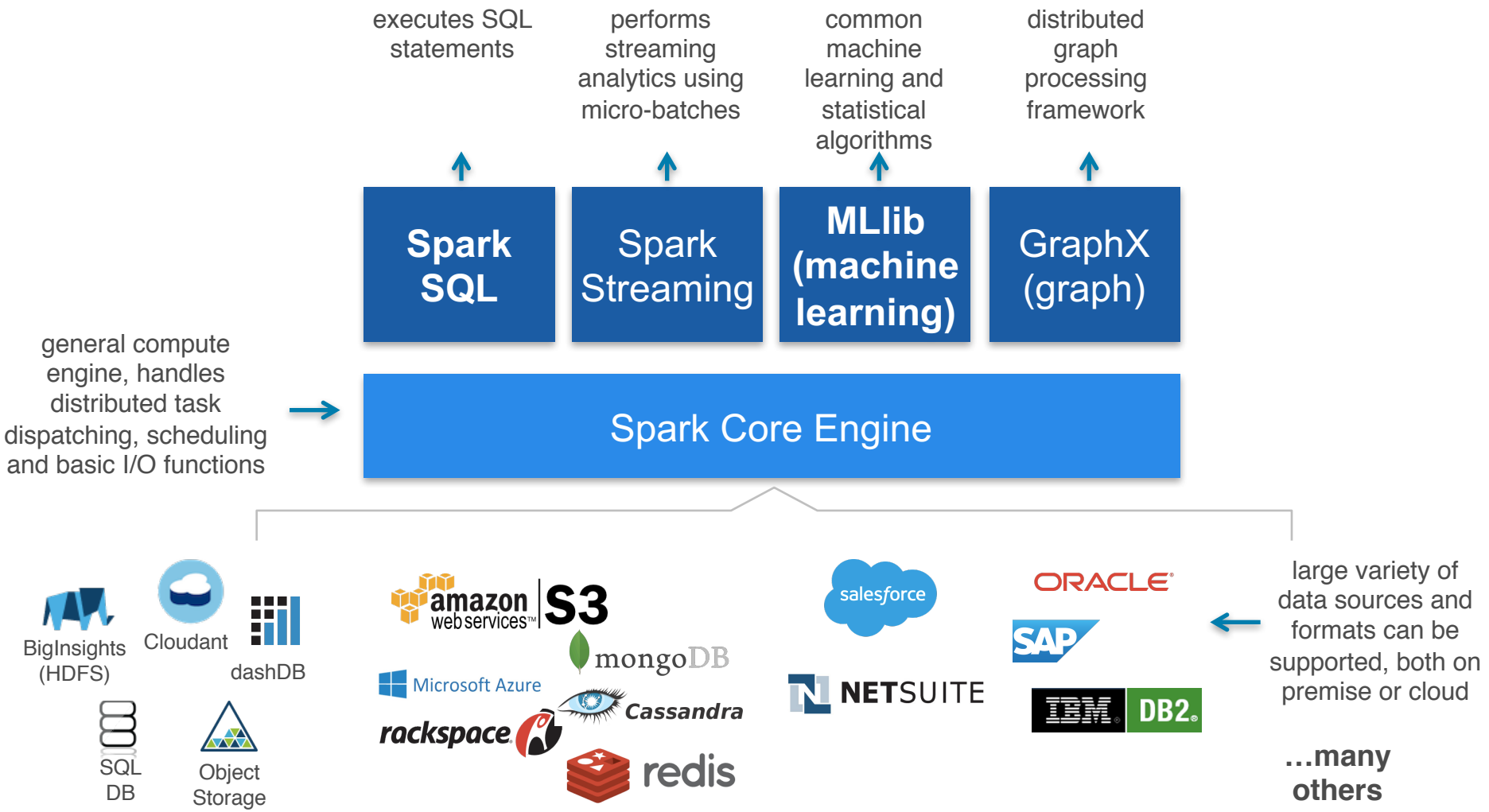
Lab 1 – Coding with Apache Spark
Deeper look at Apache Spark

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

Language	2014	2015	2016
Scala	84%	71%	65%
Java	38%	31%	29%
Python	38%	58%	62%
R	unknown	18%	20%

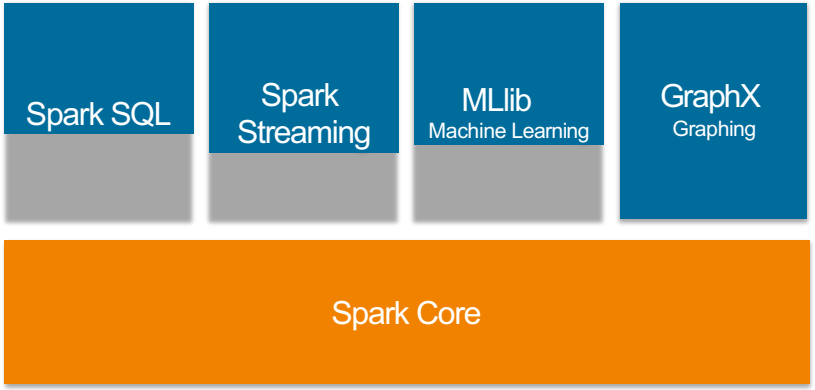
Libraries Usage

SparkSQL – 69%

DataFrames – 62%

Spark Streaming – 58%

MLlib/GraphX – 58%



75% of users use more than one component

Spark Terminology

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

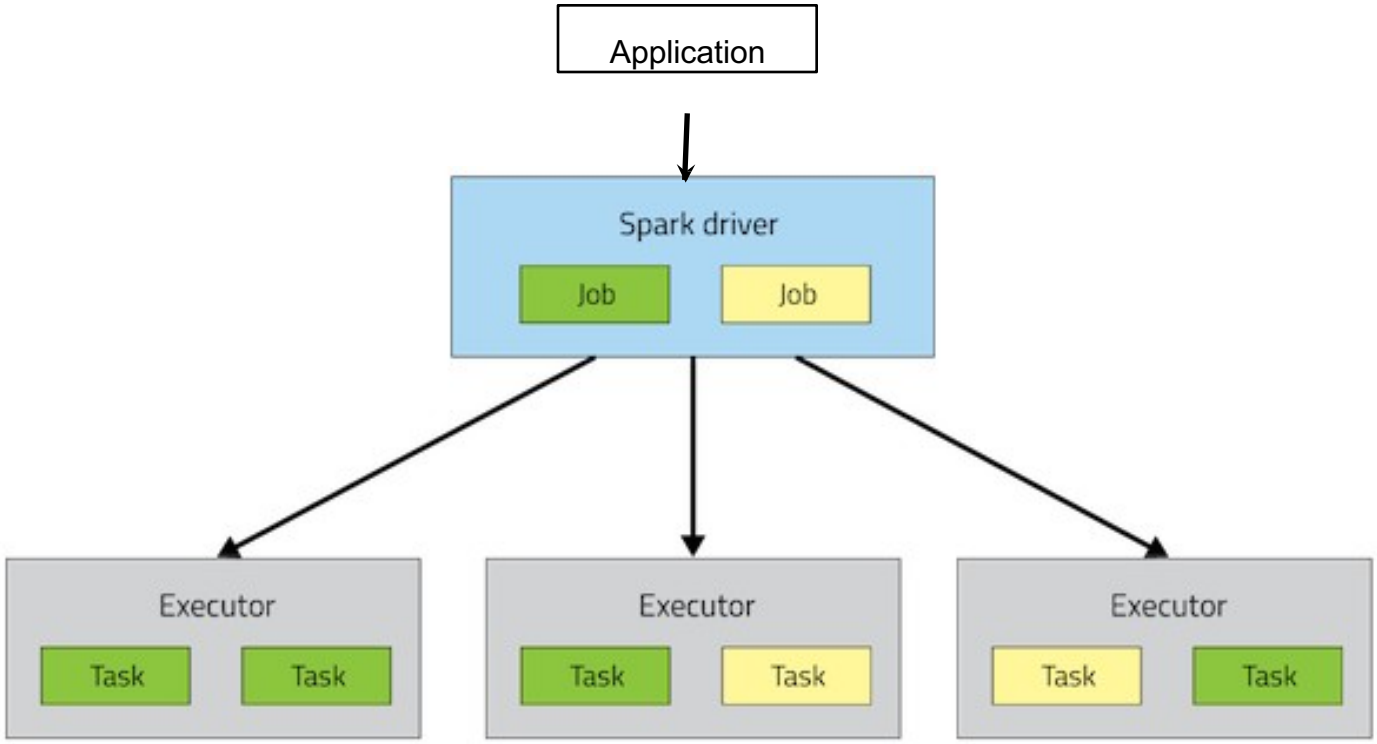
Job (Query / Query plan):
- A piece of logic (code) which will take some input from HDFS (or the local filesystem), perform some computations (transformations and actions) and write some output back.

Stage (Subplan)
- Jobs are divided into stages

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

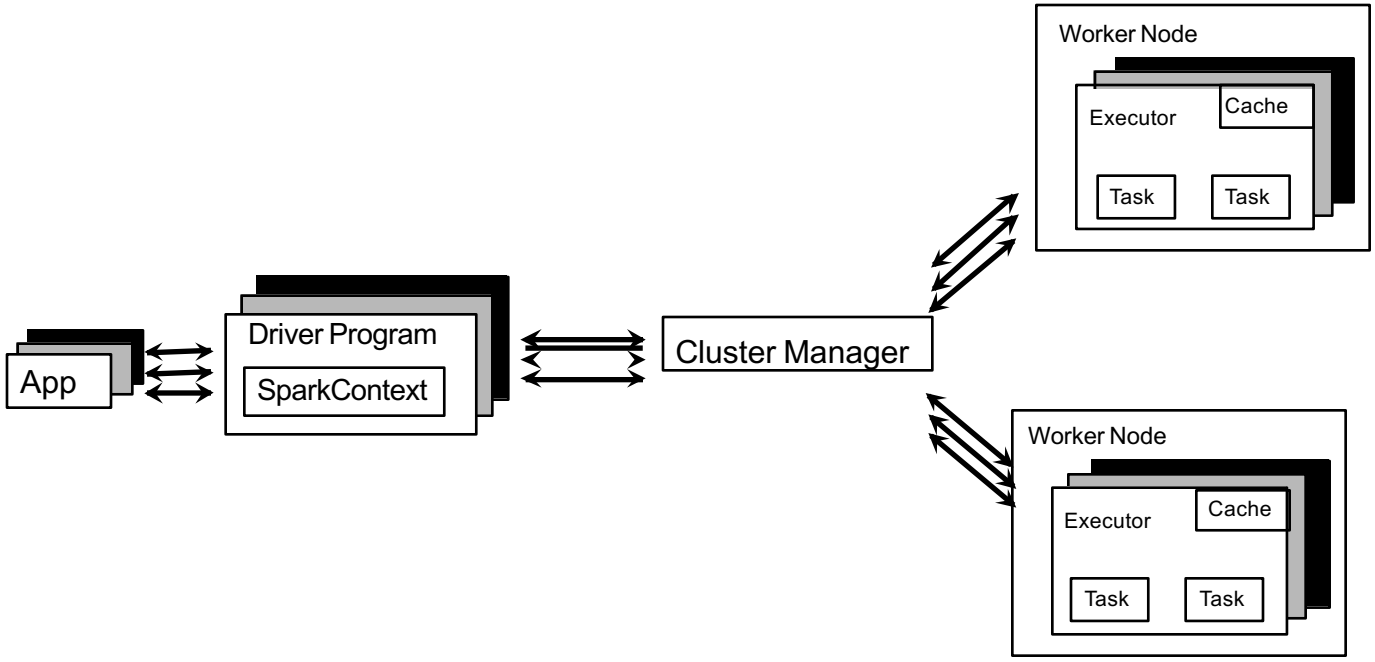
Executor (Sub agent)
- The process responsible for executing a task on a worker node

Spark Architecture



- **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

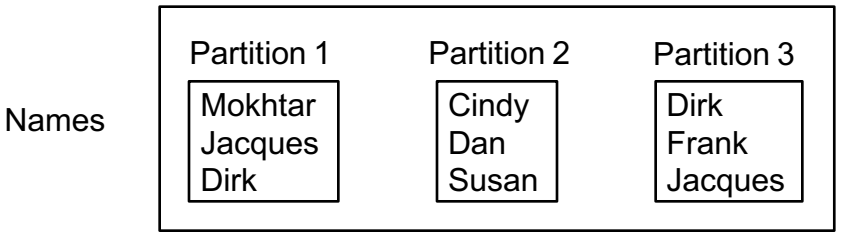


Resilient Distributed Datasets

- **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()**

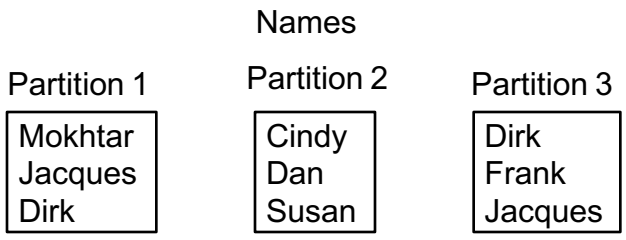
  Spark will "distribute" count processing to all partitions so as to obtain:

  Partition 1: Mokhtar(1), Jacques (1), Dirk (1) è 3
  Partition 2: Cindy (1), Dan (1), Susan (1) è 3
  Partition 3: Dirk (1), Frank (1), Jacques (1) è 3

  Local counts are subsequently aggregated: 3+3+3=9

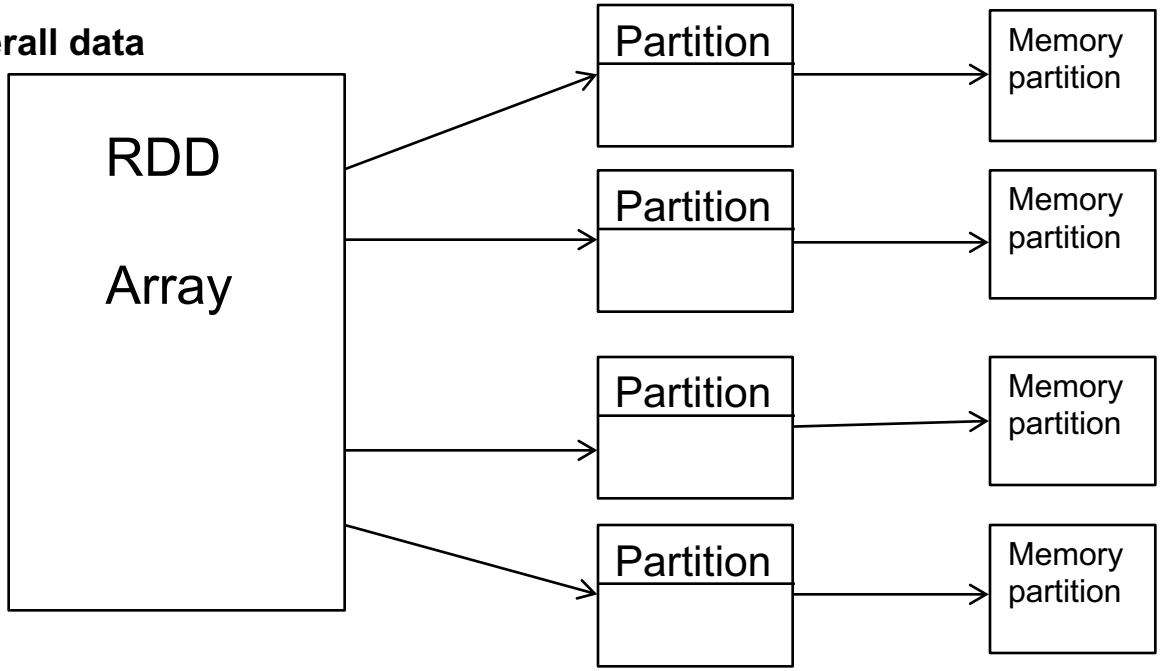
- **To lookup the first element in the RDD: Names.first()**

- **To display all elements of the RDD: Names.collect() (careful with this)**



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)

```
// 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

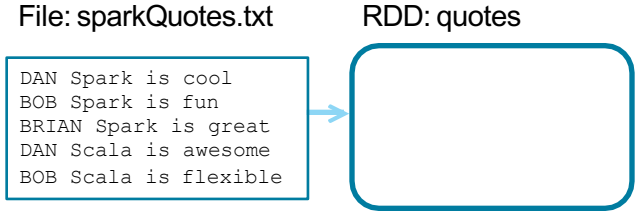
```
DAN Spark is cool
BOB Spark is fun
BRIAN Spark is great
DAN Scala is awesome
BOB Scala is flexible
```

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()
```

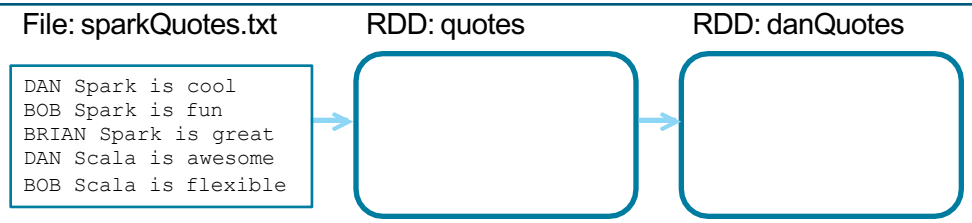


Code Execution (3)

```
// 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()
```

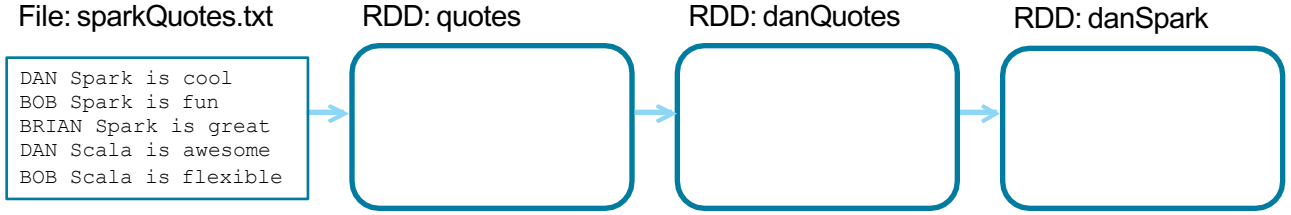


Code Execution (5)

```
// 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()
```

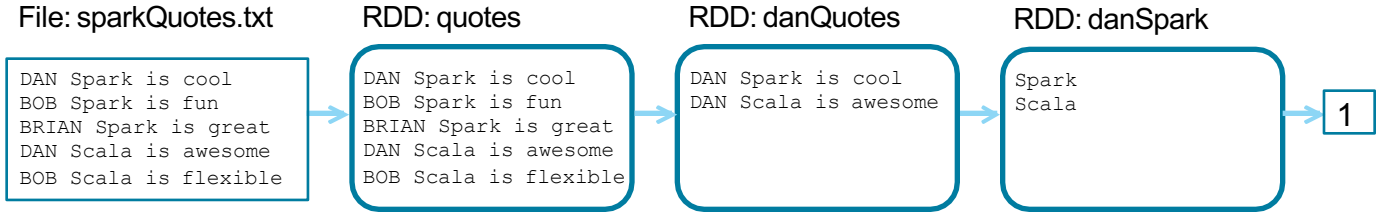


Code Execution (5)

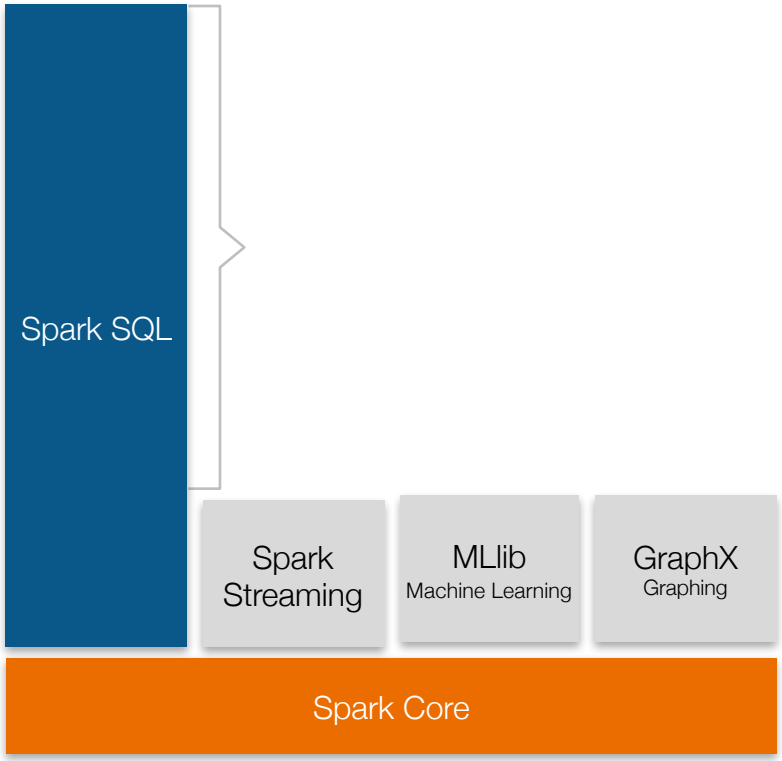
```
// Create RDD
val quotes = sc.textFile("hdfs:/sparkdata/sparkQuotes.txt")

// Transformations
val danQuotes = quotes.filter(_.startsWith("DAN"))
val danSpark = danQuotes.map(_.split(" ")).map(x => x(1))

// Action
danSpark.filter(_.contains("Spark")).count()
```



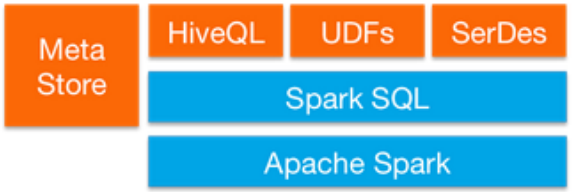
Closer Look at APIs - SQL

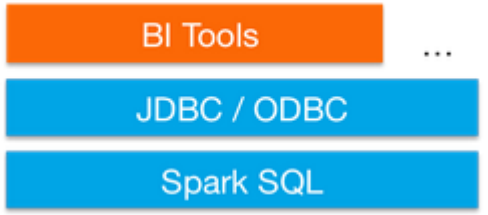


- Unified data access: Query structured data sets with SQL or DataFrame APIs
- Fast, familiar query language across all of your enterprise data
- Use BI tools to connect and query via JDBC or ODBC drivers

SparkSQL

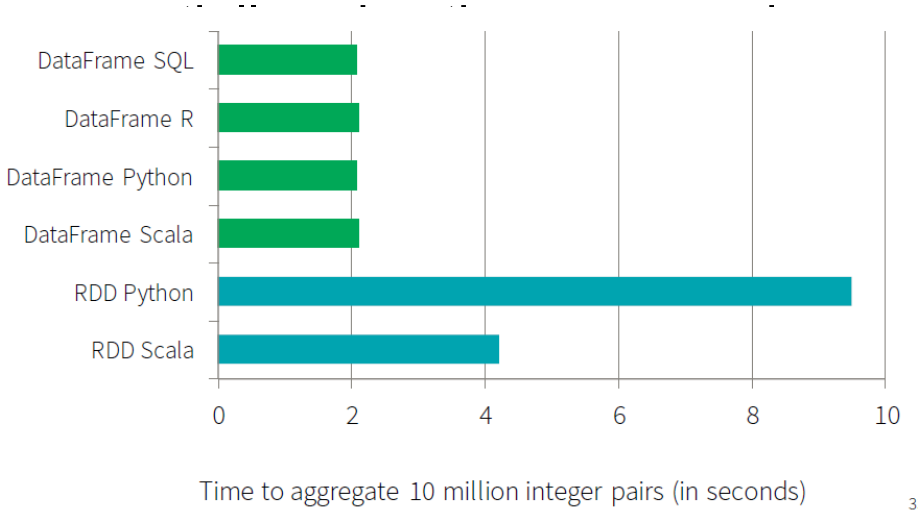
- Provide for relational queries expressed in SQL, HiveQL using Scala, Python, and Java API's
- Seamlessly mix SQL queries with Spark programs
- Dataframes provide a single interface for efficiently working with structured data including Apache Hive, Parquet and JSON files
- Graduated from alpha status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





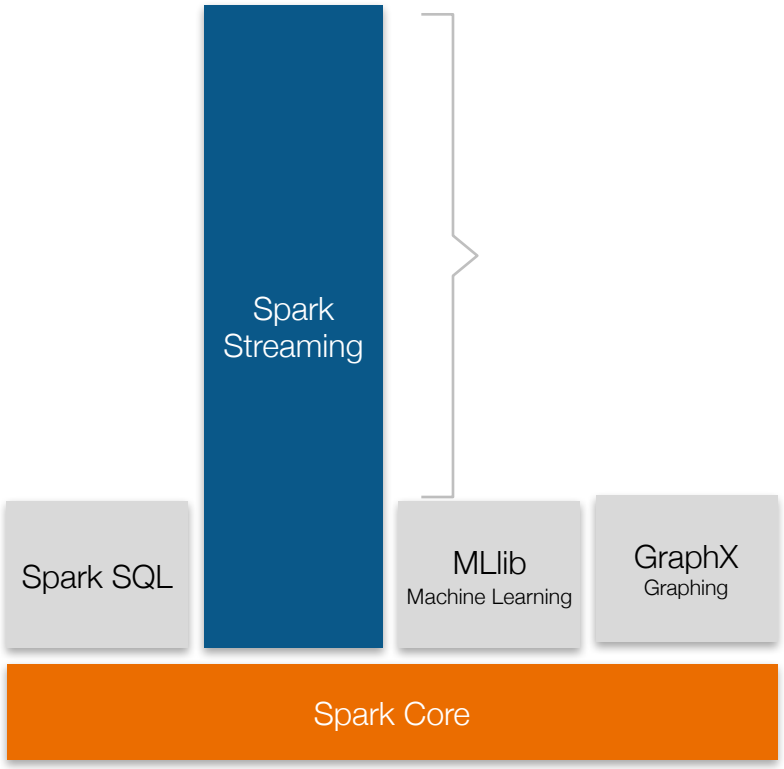
SparkSQL, DataFrames and DataSets

- 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 a



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

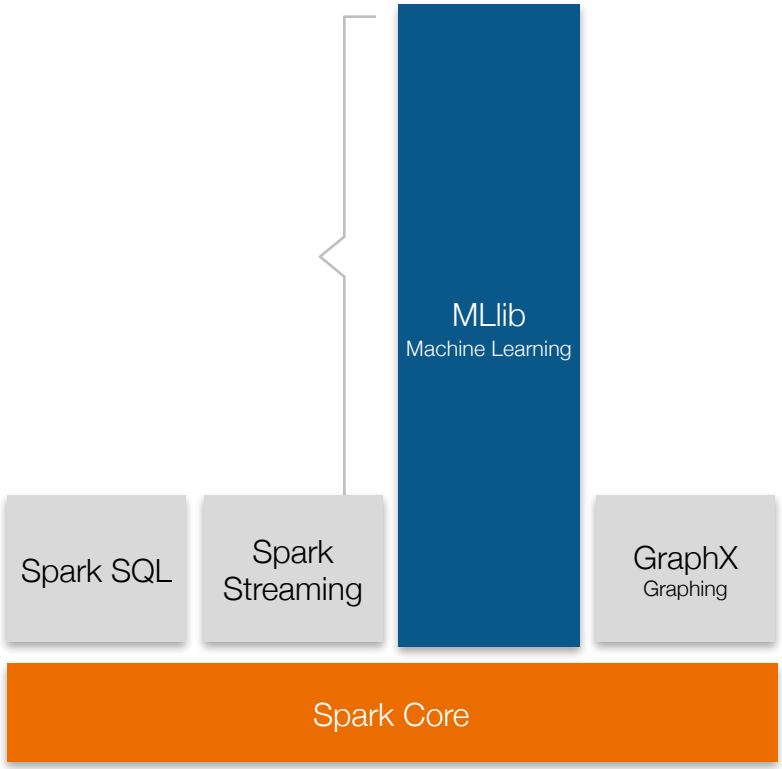
- **Component of Spark**
  - Project started in 2012
  - First alpha release in Spring 2013
  - Out of alpha with Spark 0.9.0
  - More enhancements targeted for Spark 2.0

- **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

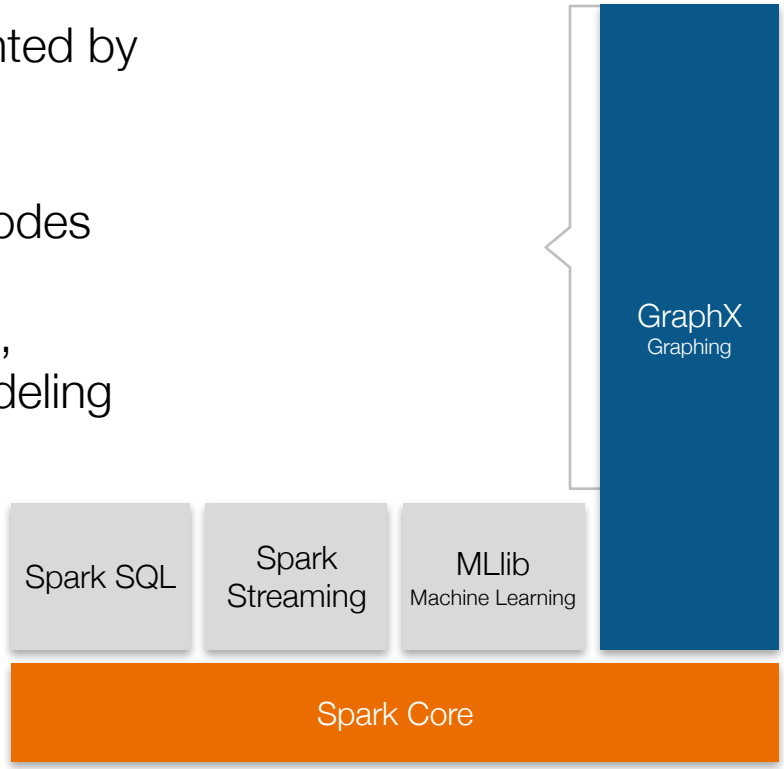


Spark MLlib

- **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 algorithms