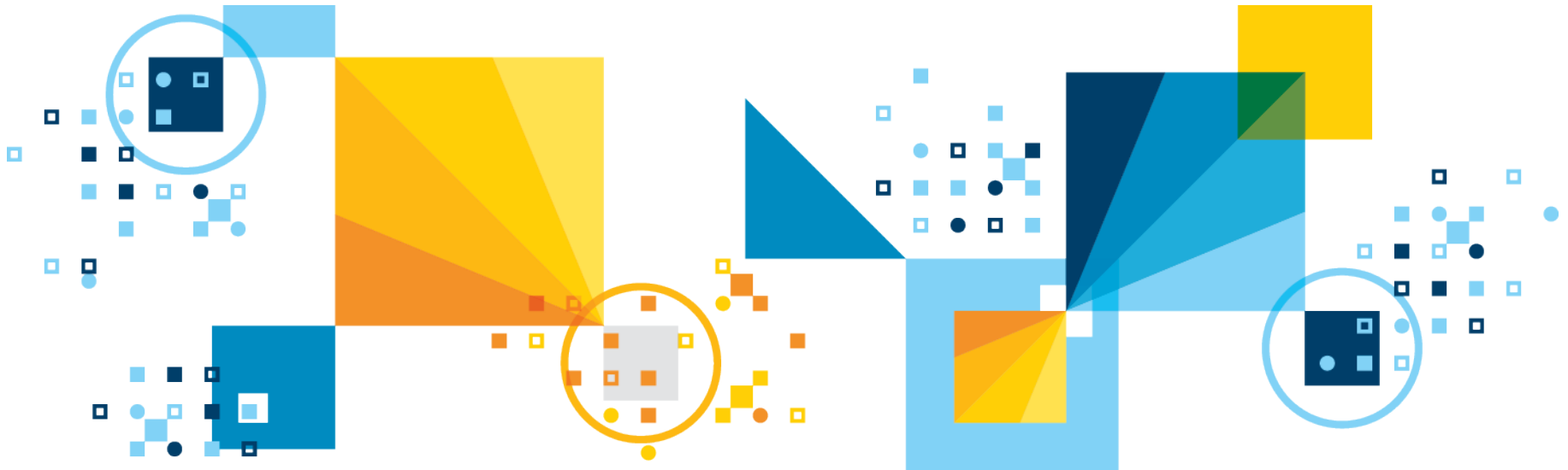


Understanding Spark



Agenda

- **Why Spark?**
- **IBM's commitment to Spark**
- **Spark background**
- **Spark details**
- **Spark: final thoughts**

Why *Spark*

Key Reasons for the Interest in Spark

Beware of the hype!

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, R
- **New tools** continually reduce skill barrier for access (e.g. SQL for analysts)

Leverages existing investments



- Works well within **existing Hadoop ecosystem**

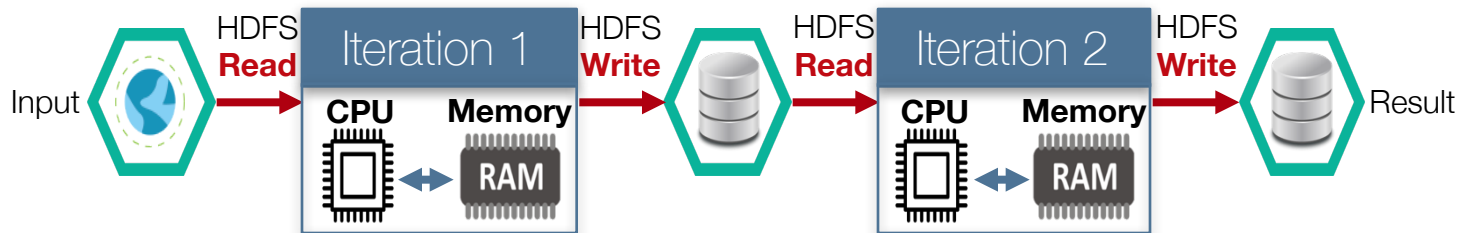
Improves with age



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

Motivation for Apache Spark

- **Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of **(slow) disk I/O****

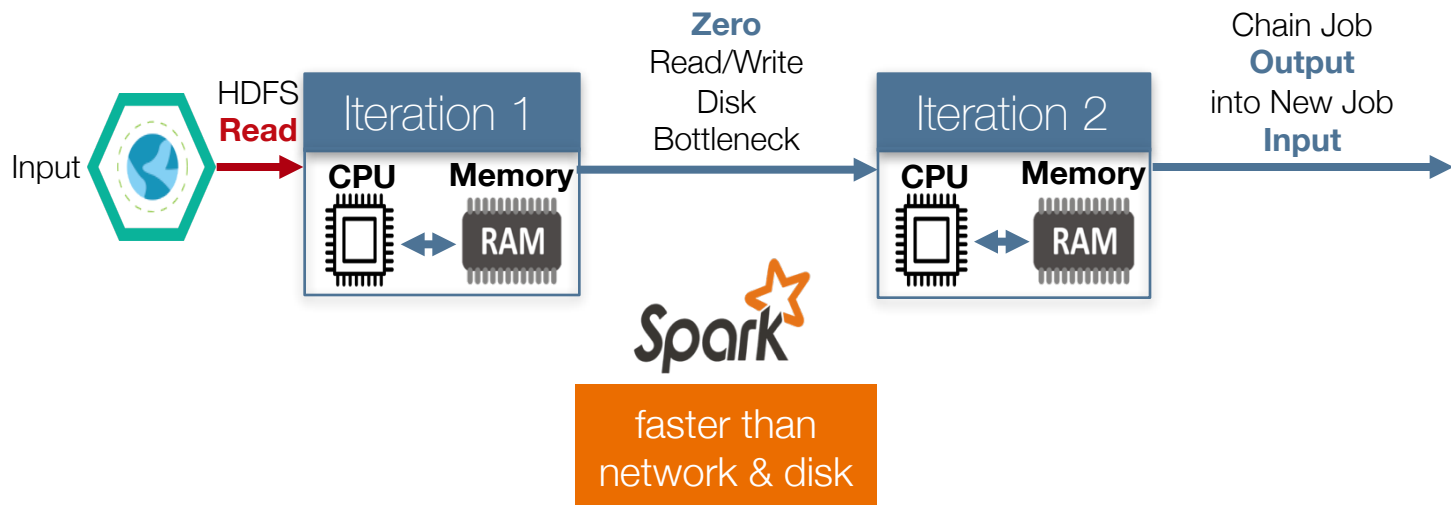


Motivation for Apache Spark

- Traditional Approach: MapReduce jobs for complex jobs, interactive query, and online event-hub processing involves lots of **(slow) disk I/O**



- Solution: Keep more data in-memory** with a new distributed execution engine



Spark Common Use Cases

Interactive Query

- Enterprise-scale data volumes accessible to interactive query for business intelligence (BI)
- Faster time to job completion allows analysts to ask the “next” question about their data & business

Large-Scale Batch

- Data cleaning to improve data quality (missing data, entity resolution, unit mismatch, etc.)
- Nightly ETL processing from production systems

Complex Analytics

- Forecasting vs. “Nowcasting” (e.g. Google Search queries analyzed en masse for Google Flu Trends to predict outbreaks)
- Data mining across various types of data

Event Processing

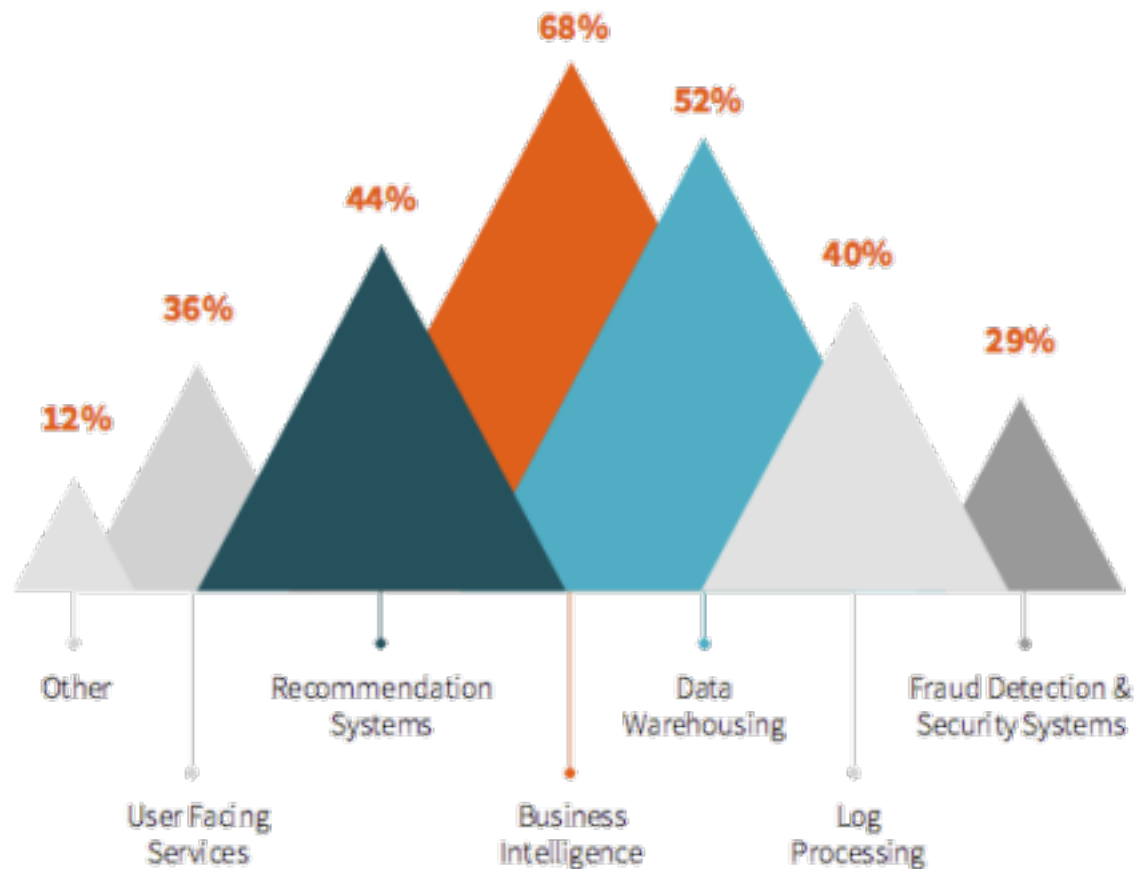
- Web server log file analysis (human-readable file formats that are rarely read by humans) in near-real time
- Responsive monitoring of RFID-tagged devices

Model Building

- Predictive modeling answers questions of “what will happen?”
- Self-tuning machine learning, continually updating algorithms, and predictive modeling

Common Applications for Spark

- Spark is implemented inside many types of products, across a multitude of industries and organizations



IBM and *Spark*

IBM is all-in on Spark

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

Ben Horowitz,
Andreessen Horowitz

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

Infuse the Portfolio

Integrate Spark throughout portfolio

3,500 employees working on Spark-related topics

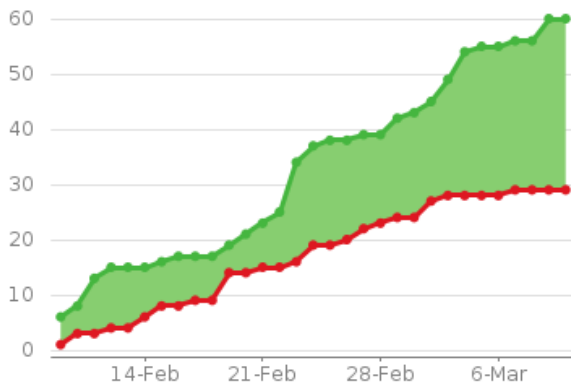
Spark however customers want it – standalone, platform or products

IBM Spark Technology Center – San Francisco, CA

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



Created vs Resolved Chart: STC Completed Spark JIRAs
Filter










Issues: 29 created and 60 resolved

Period: last 30 days (grouped Daily)

Issue Statistics

Statistics: STC Completed Spark JIRAs Filter (Issue Type)

 Bug	105	41%
 Documentation	8	3%
 Improvement	81	31%
 New Feature	7	3%
 Sub-task	54	21%
 Task	1	0%
 Test	3	1%

Total Issues: 259

As of March 10, 2016

Fixing lot's of issues
reported by others

See what we're up to ...

IBM Spark Technology Center

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

Spark Certification

<http://www.ibm.com/certify/tests/ovrC2090-103.shtml>

IBM Professional Certification Program

Certify your skills. Accelerate your career.

Overview

Products

Certifications

Tests

Mastery Tests

By Number

By Unit

Pricing

Updates & Revisions

Test Info

Test Series Changes

Test C2090-103: Apache Spark 1.6 Developer

Overview

Objectives

Test preparation

Sample / Assessment Test

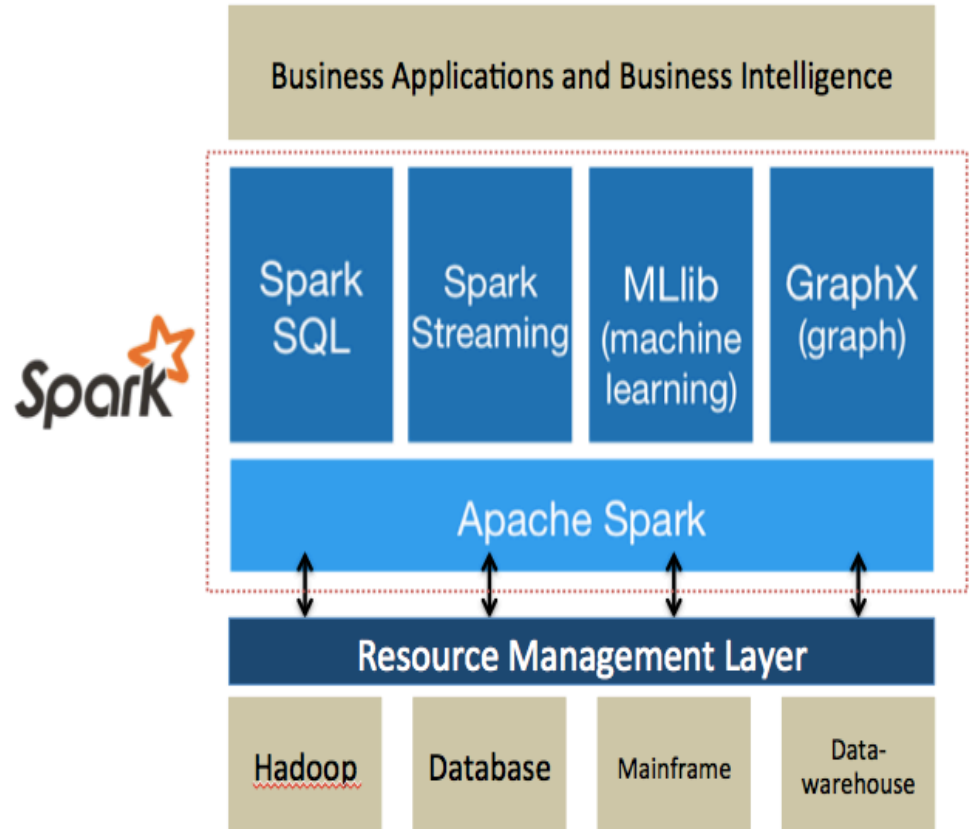
Test information:

- Number of questions: 60
- Time allowed in minutes: 120
- Required passing score: 65%
- Languages: English

IBM | SPARK – The Analytics Operating System

“Enabling New Classes of Intelligent Applications Embedded with Analytics”

- Spark unifies data, enabling real-time insights
- Spark processes and analyzes data from any data source
- Spark is complementary to Hadoop, but faster with in-memory performance
- Build models quickly. Iterate faster. Apply intelligence



Core Attributes of the Data Scientist Experience



IBM 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*
- Spark as a Service *

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

* DSX product roadmap items

Spark  **background**

Spark Background

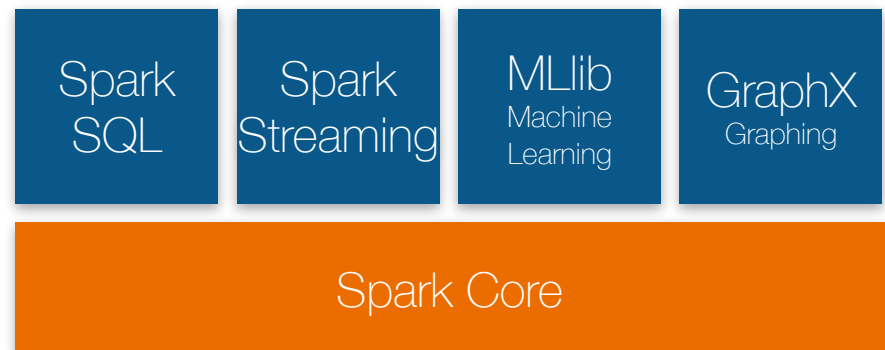


- **Started as a research project in 2009, open source in 2010**
 - General purpose cluster computing system
 - **Generalizes MapReduce**
 - Batch oriented processing
 - Main concept:
Immutable Resilient Distributed Datasets (RDDs)
- **Apache incubator project in June 2013**
 - Apache top level project Feb 27, 2014
- **Current version 2.1.0 (December 28, 2016)**
 - Languages supported: Java, Scala, Python, R (Java 7+, Python 2.6+/3.4, R 3.1+)
 - Still includes changes of behavior from previous versions

- [Spark 2.1.0](#) (Dec 28 2016)
- [Spark 2.0.2](#) (Nov 14 2016)
- [Spark 2.0.1](#) (Oct 03 2016)
- [Spark 2.0.0](#) (Jul 26 2016)
- [Spark 1.6.3](#) (Nov 07 2016)
- [Spark 1.6.2](#) (Jun 25 2016)
- [Spark 1.6.1](#) (Mar 09 2016)
- [Spark 1.6.0](#) (Jan 04 2016)
- [Spark 1.5.2](#) (Nov 09 2015)
- [Spark 1.5.1](#) (Oct 02 2015)
- [Spark 1.5.0](#) (Sep 09 2015)
- [Spark 1.4.1](#) (Jul 15 2015)
- [Spark 1.4.0](#) (Jun 11 2015)
- [Spark 1.2.1](#) (Feb 09 2015)
- [Spark 1.2.0](#) (Dec 18 2014)
- [Spark 1.1.1](#) (Nov 26 2014)
- [Spark 1.1.0](#) (Sep 11 2014)
- [Spark 1.0.2](#) (Aug 05 2014)
- [Spark 1.0.1](#) (Jul 11 2014)
- [Spark 1.0.0](#) (May 30 2014)

Libraries Usage¹

Library	2015	2016
SparkSQL	69%	88%
DataFrames	62%	N/A
Streaming	58%	71%
MLLib/GraphX	58%	71% ²



¹ From surveys done by Databricks, summer/fall 2015 and 2016

² MLLib only

Spark Programming Languages

▪ Scala

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

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

▪ Java

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

Surveys done by Databricks,
summer 2015, 2016

▪ Python

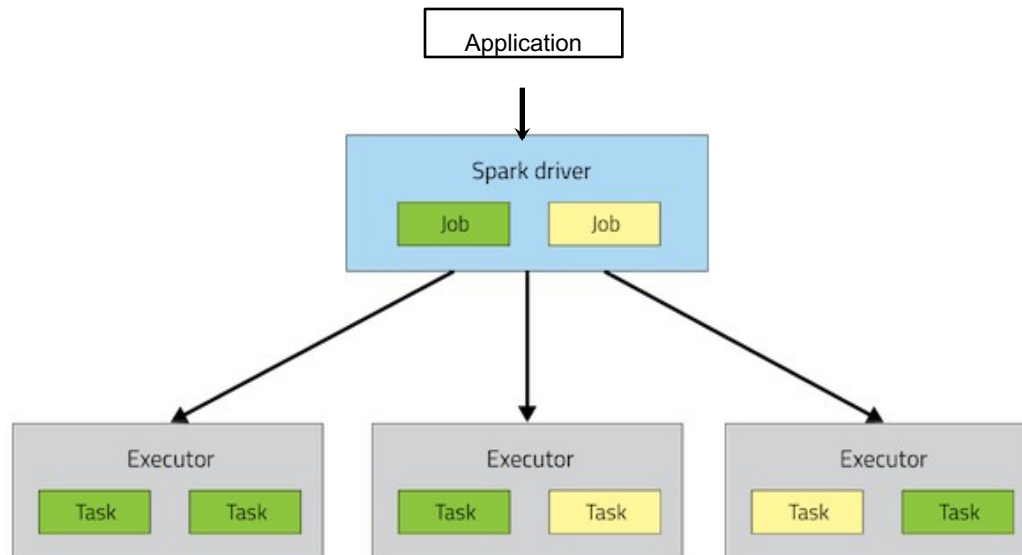
- Always a bit behind Scala in functionality. Becoming the language of choice.

▪ R

This probably means that more “data scientists” are starting to use Spark

Spark details

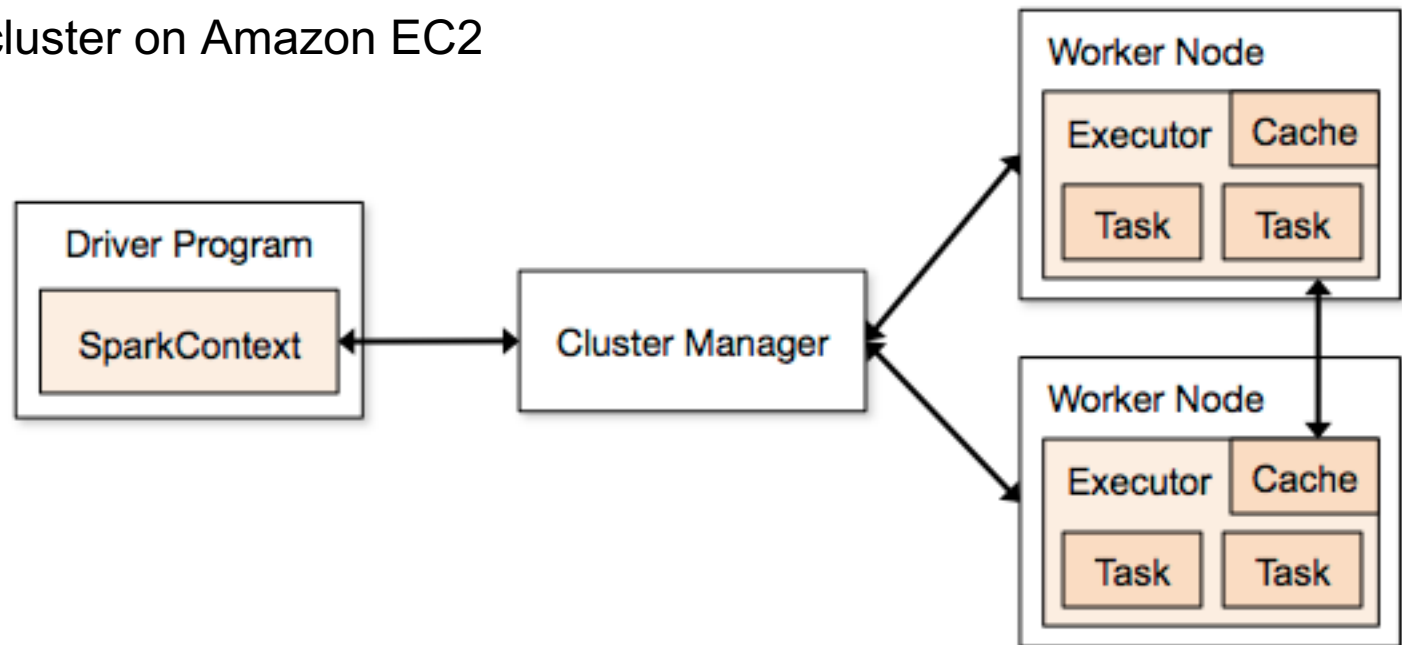
Spark Architecture



Spark Application Architecture

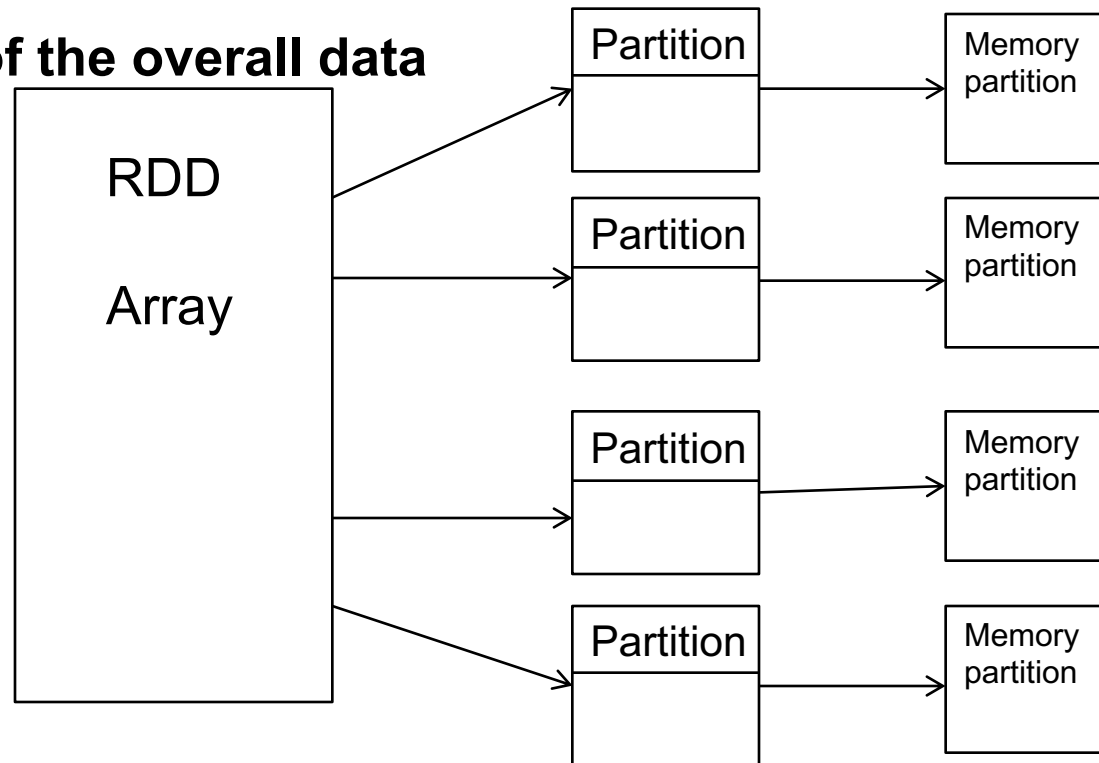


- A Spark application is initiated from a driver program
- Spark execution modes:
 - Standalone with the built-in cluster manager
 - Use Mesos as the cluster manager
 - Use YARN as the cluster manager
 - Standalone cluster on Amazon EC2



Resilient Distributed Dataset (RDD)

- RDDs are **immutable**
 - Modifications create new RDDs
- 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



Available RDD Types

▪ RDD

- Base class for RDDs
- Properties:
 - List of partitions
 - Function for computing each partition
 - List of dependencies on other RDDs
 - Optional partitioned for key-value RDDs
 - Optional list of preferred location to compute each partition

▪ Known subclasses

- CoGroupedRDD, EdgeRDD, EdgeRDDImpl, HadoopRDD, JdbcRDD, NewHadoopRDD, PartitionPruningRDD, ShuffledRDD, UnionRDD, VertexRDD, VertexRDDImpl
- [Extra functions based on RDD content](#):
DoubleRDDFunctions, OrderedRDDFunctions, PairRDDFunctions, SequenceFilesRDDFunctions

▪ Custom RDDs can be created

- An RDD can consist of objects (such as Row object)

RDD Classes

- **Different classes for different languages (Scala, Java)**
 - RDD has 87 value members
- **Multiple types of RDD**
 - Multiple additional members in other classes
- **Separate Python API**
 - Limited functionality

org.apache.spark.api.java	hide	focus
JavaDoubleRDD		
JavaHadoopRDD		
JavaNewHadoopRDD		
JavaPairRDD		
JavaRDD		
JavaRDDLike		
org.apache.spark.graphx	hide	focus
EdgeRDD		
VertexRDD		
org.apache.spark.graphx.impl	hide	focus
EdgeRDDImpl		
VertexRDDImpl		
	hide	focus
org.apache.spark.mllib.random		
RandomRDDs		
org.apache.spark.mllib.rdd	hide	focus
RDDFunctions		
org.apache.spark.rdd	hide	focus
AsyncRDDActions		
CoGroupedRDD		
DoubleRDDFunctions		
HadoopRDD		
JdbcRDD		
NewHadoopRDD		
OrderedRDDFunctions		
PairRDDFunctions		
PartitionPruningRDD		
RDD		
SequenceFileRDDFunctions		
ShuffledRDD		
UnionRDD		
org.apache.spark.scheduler	hide	focus
SparkListenerUnpersistRDD		
org.apache.spark.storage	hide	focus
RDDBlockId		
RDDInfo		

Spark Programming Model

- **Operations on RDDs (datasets)**

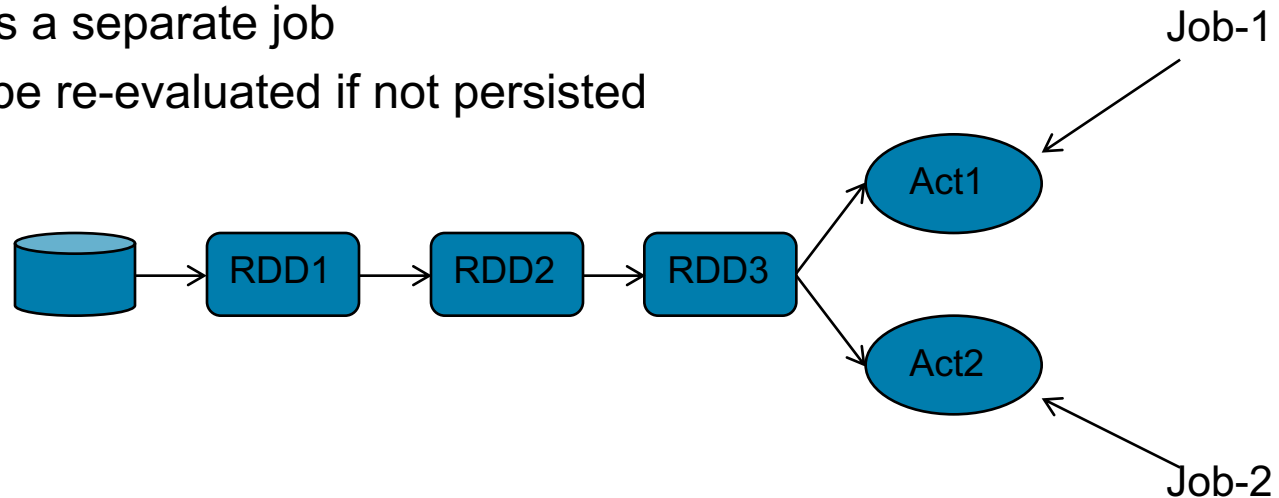
- Transformation
- Action

- **Transformations use lazy evaluation**

- Executed only if an action requires it

- **An application consist of a directed acyclic graph (DAG)**

- Each action results in a separate batch job
- Parallelism is determined by the number of RDD partitions
- Each action is a separate job
- RDDs could be re-evaluated if not persisted



Spark Operations

■ Transformation operations (set operations)

- Convert RDDs into other RDDs
- Operations availability depends on the type of RDD processed
- Examples of operations:

++ (union)	aggregate	cache/persist	cartesian
coalesce	count	distinct	filter
flatMap/map	groupby	intersection	reduceByKey
subtract	union	zip	

■ Action operations

- Return a value to the driver
- Examples of actions:

reduce	collect	count	first
take_	saveAs_	countByKey	foreach

Spark: Miscellaneous Information

- **Applications run jobs sequentially by default**
 - Can run in parallel if they are submitted from separate threads

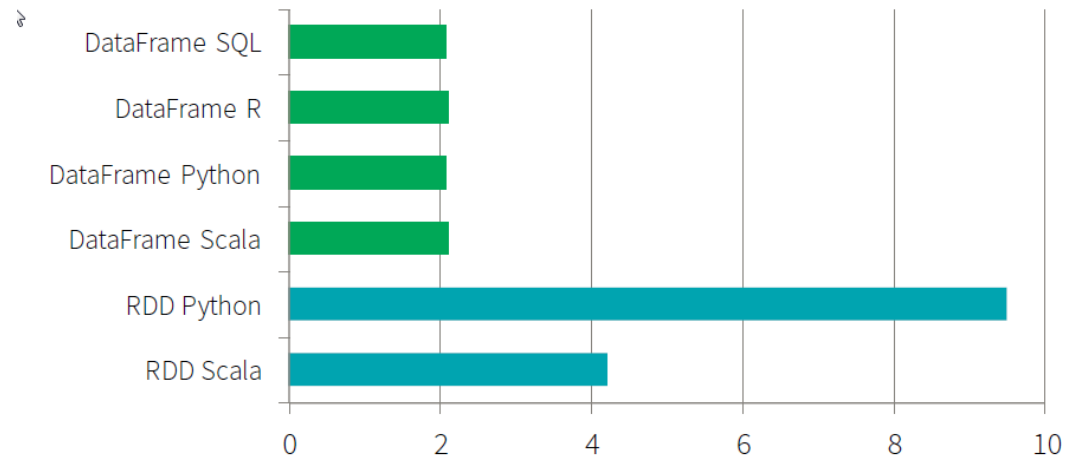
- **RDD can be persisted for reuse**
 - Using the `persist()` function
 - Removes the need to recreate an RDD from its origin
 - Optimizes processing in a multi-job DAG

- **Stages are determined by the shuffling of data between partitions**
 - The number of stages determines the number of tasks
 - More stages does not always mean slower performance

- **Spark tooling is currently limited**
 - Spark UI to monitor jobs
 - Eclipse plug-ins for programming
 - No graphical interface to build DAGs

SparkSQL, DataFrames and Datasets

- **A rich set of functionality that allows “Database-like” processing**
- **Share single optimizer, called “Catalyst”**
 - 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)

Picture credit: databricks.com

© 2016 IBM Corporation

Spark Streaming



▪ Component of Spark

- Project started in 2012
- First alpha release in Spring 2013
- Out of alpha with Spark 0.9.0

▪ 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



Current Spark Streaming I/O

▪ Input Sources

- Kafka, Flume, Twitter, ZeroMQ, MQTT, TCP sockets
- Basic sources: sockets, files, Akka actors
- Other sources require receiver threads

▪ Output operations

- Print(), saveAsTextFiles(), saveAsObjectFiles(), saveAsHadoopFiles(), foreachRDD()
- foreachRDD can be used for message queues, DB operations and more



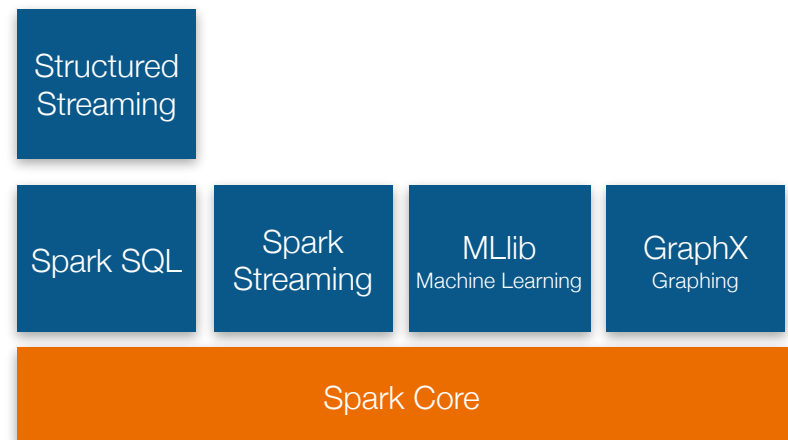
Structured Streaming

- **New streaming engine built on the Spark SQL engine**

- Alpha release in Spark 2.0, still alpha in 2.1.0 (experimental)
- Reduces programming differences between Spark Streaming and Spark
- Includes improvements on state maintenance

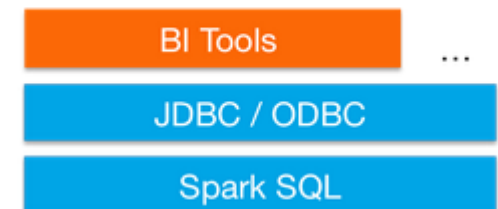
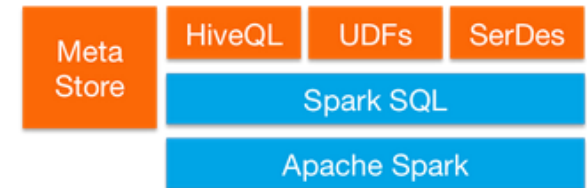
- **Too early to evaluate**

- Lack of functionality
- Still micro-batch paradigm



SparkSQL

- Provide for relational queries expressed in SQL, HiveQL and Scala
- Seamlessly mix SQL queries with Spark programs
- DataFrame/Dataset provide a single interface for efficiently working with structured data including Apache Hive, Parquet and JSON files
- Leverages Hive frontend and metastore
 - Compatibility with Hive data, queries, and UDFs
 - HiveQL limitations may apply
 - Not ANSI SQL compliant
 - Little to no query rewrite optimization, automatic memory management or sophisticated workload management
- Graduated from alpha status with Spark 1.3
 - DataFrames API marked as experimental
- Standard connectivity through JDBC/ODBC



Spark ML

- **Spark ML for machine learning library**

- RDD-based package `spark.mllib` now in maintenance mode
- The primary API is now the DataFrame-based package `spark.ml`
- Parity of `spark.ml` estimated for Spark 2.2

- **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 MLlib

Basic Statistics	Clustering	Frequent Pattern Mining
Summary Statistics	K-means	FP-growth
Correlations	Gaussian mixture	Association rules
Stratified sampling	Power iteration clustering	PreficSpan
Hypothesis testing	Latent Dirichlet allocation	Optimization (developer)
Streaming significant testing	Bisecting k-means	Stochastic gradient descent
Random data generation	Streaming k-means	Limited memory BFGS
Classification&Regression	Collaborative Filtering	Others
Linear models	Alternating least squares	Evaluation metrics
Naïve Bayes	Dimensionality Reduction	PMML model export
Decision trees	Singular value decomposition	Feature extraction and transformation
Ensembles trees	Principal component analysis	
Isotonic regression		

Spark ML

Classification/Regression	Clustering	Feature Extractors
Logistic regression	K-means	TF-IDF
Decision tree classifier	Gaussian mixture	Word2Vec
Random forest (class/reg)	Latent Dirichlet allocation	CountVectorizer
Gradient boosted tree	Bisecting k-means	Feature Transformers
Multilayer perceptron classifier	Collaborative Filtering	Tokenizer
One-vs-rest classifier	Alternating least squares	StopWordRemover
Linear regression	Feature Selectors	N-gram
Generalized linear reg.	VectorSlicer	Binarizer
Naïve Bayes	RFormula	PCA
Decision trees (class/reg)	ChiSqSelector	PolynomialExpansion
Survival regression		StringIndexer
Isotonic regression		---- 13 more ----

Spark GraphX

▪ Flexible Graphing

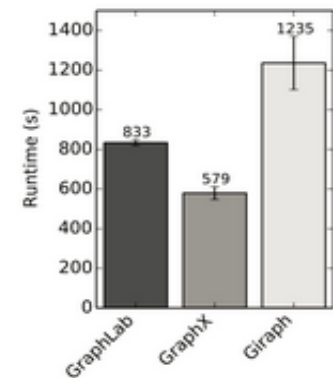
- GraphX unifies ETL, exploratory analysis, and iterative graph computation
- You can view the same data as both graphs and collections, transform and join graphs with RDDs efficiently, and write custom iterative graph algorithms with the API

▪ 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



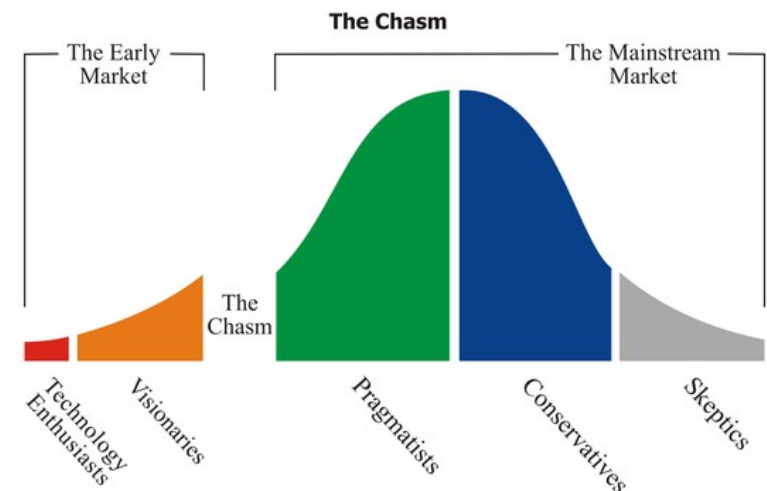
Spark R

- **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 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

Spark: Final Thoughts



- **Spark is a good replacement for MapReduce**
 - Higher performance
 - Framework makes it easier to use than MapReduce (M/R)
 - Powerful RDD concept and DataFrame/Dataset higher abstraction
- **Programming in Spark is still difficult**
 - M/R model where programmers plug-in their code for set processing
 - Great for “innovators” and “early adopters”¹
- **Spark success lays in higher APIs**
 - Such as SparkSQL, SparkR and so on



¹ See Geoffrey Moore “Crossing the Chasm”

Thank You

