Spark

Bereket Temtime

Jonathan Ouegnin

Big Data

December 8th 2016

What is it?

- **Apache Spark** is a fast cluster computing framework and general engine for big data processing, with built-in modules for streaming, SQL, machine learning and graph processing

- Used to process, query and analyze big data

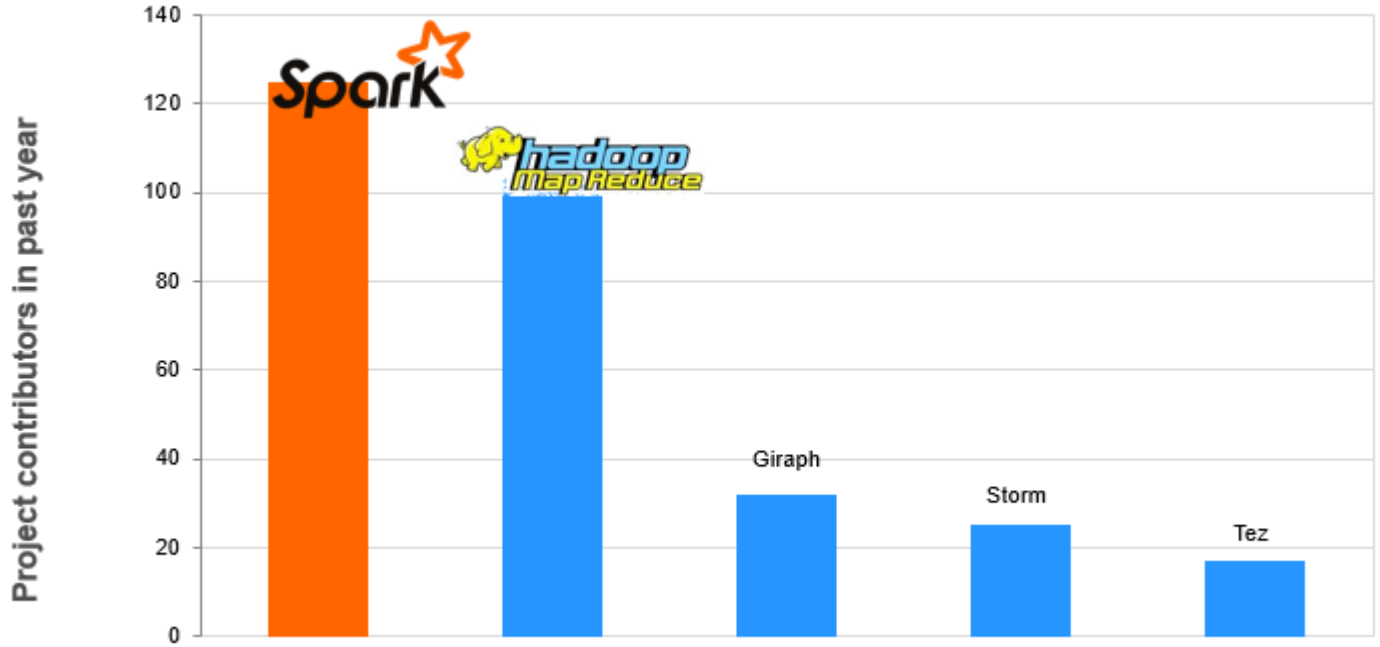
History of Apache Spark

- University of California Berkeley's AMP Lab
- Donated to Apache Software Foundation
- Became open source in 2010
- Now a top level project at Apache

Community of Spark

March 27th 2010 - February 15th 2014
Commits to master, excluding merge commits

Contribution Type: **Commits**

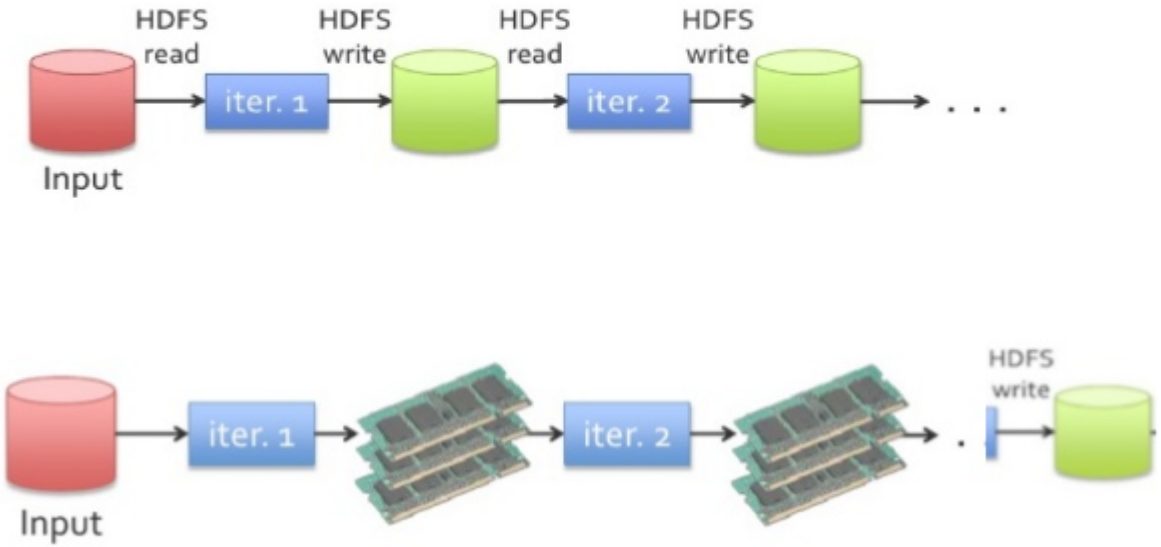


Major Advantage

- In-memory processing
  - It saves and loads data from RAM instead of the Hard disk
  - Disk operations are slower than RAM operations
  - Conversion of data from RAM to disk (Serialization, reverse: Deserialization)

- 100 times faster than Hadoop when using in-memory computation
- 10 times faster when using disk

HADOOP MAPREDUCE VS SPARK



Not a replacement of Hadoop – designer to run on top of it

Other Advantages

- Allows both batch and real time processes
- Faster map side shuffling and Reduce side shuffling
- Multiple types of transformations and actions
  - Not only MapReduce
- Has lazy operation features that don’t need to execute until we require results

Languages Supported

- Has APIs in these languages:
  - Java
  - Scala
  - Python
  - R
- Was coded mostly using Scala and Java
- More programming options for users
  - East to Develop
- Less Code

Python vs Scala

- Can use Python to some extent
- Use Scala to process some serious data across several machines and clusters
- Computation speed of Python is much slower than Scala in Spark
  - Scala is native language for Spark (because Spark itself written in Scala)
  - Scala is a compiled language where as Python is an interpreted language
  - Python has process based executors where as Scala has thread based executors
  - Python is not a JVM (java virtual machine) language

Data Source

- Local source
  - file:///opt/httpd/logs/access_log
- HDFS
  - Regular files, sequence files, any other Hadoop InputFormat
- Hbase
- S3

Machine Learning Algorithms Supported by Spark

- K-Means
- L_1 and L_2-regularized Linear Regression
- L_1 and L_2-regularized Logistic Regression
- Alternating Least Squares
- Naive Bayes
- Random Forest
- Stochastic Gradient Descent

- Mahout libraries are being imported to Spark
  - In progress

Data Representations of Spark

- Spark uses three different data representations:
  - Resilient Distributed Datasets (RDD)
  - Dataframe
  - Dataset
- It uses different APIs for each data representation

Resilient Distributed Datasets (RDD)

- Collection of elements that can be divided across multiple nodes in a cluster to run parallel processing
- Fault tolerant
- Information of how it was created, what the input sources were for, any transformations done on it can be known at anytime
- This is done in parallel
  - Scala collection is run in parallel
  - Records of files supported by Hadoop

RDD Operations

- Two types of operations can be applied to RDDs
  - Transformations
    - Creation of a new dataset from an existing (operation applied on RDD to create a new RDD)
    - map, filter, distinct, union, sample, groupByKey, join, etc…
  - Actions
    - Also applied on RDDs to return a value after running a computation
    - collect, count, first, takeSample, foreach, etc…

Check the documentation for a complete list

http://spark.apache.org/docs/latest/scala-programming-guide.html#rdd-operations

RDDs Use Shared Variable

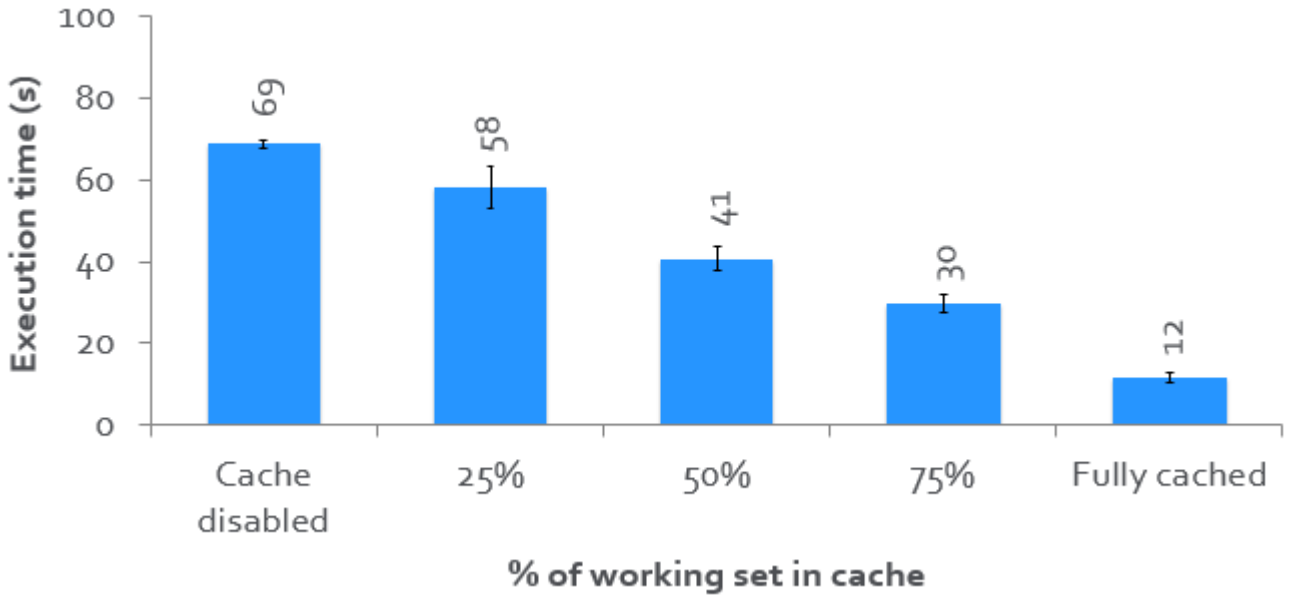
- The parallel operations in Apache Spark use shared variable.
  - Whenever a task is sent by a driver to executors program in a cluster, a copy of shared variable is sent to each node in a cluster, so that they can use this variable while performing task.

- Spark supports two types of shared variables
  - **Broadcast** variable to save the copy of data across all node
  - **Accumulator** variables are used for aggregating the information

RDD Persistence / Caching

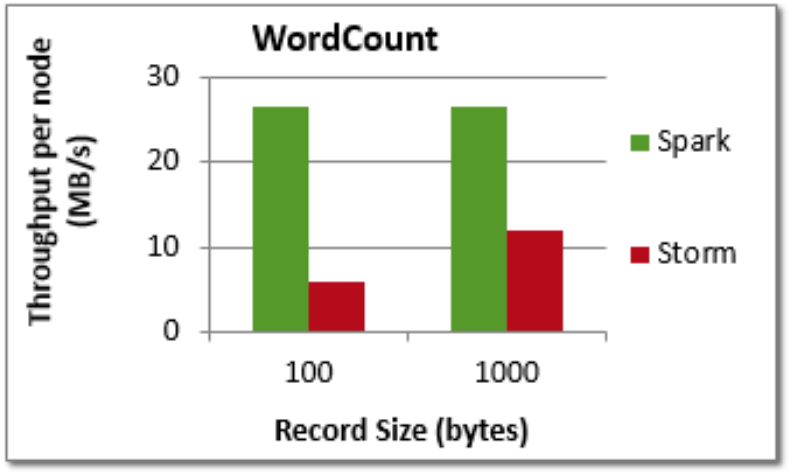
- Variety of storage levels
  - memory_only (default), memory_and_disk, etc…
- API Calls
  - persist(StorageLevel)
  - cache() – shorthand for persist(StorageLevel.MEMORY_ONLY)
- Considerations
  - Read from disk vs. recompute (memory_and_disk)
  - Total memory storage size (memory_only_ser)
  - Replicate to second node for faster fault recovery (memory_only_2)
    - Think about this option if supporting a web application

Cache Scaling



Comparison to Storm

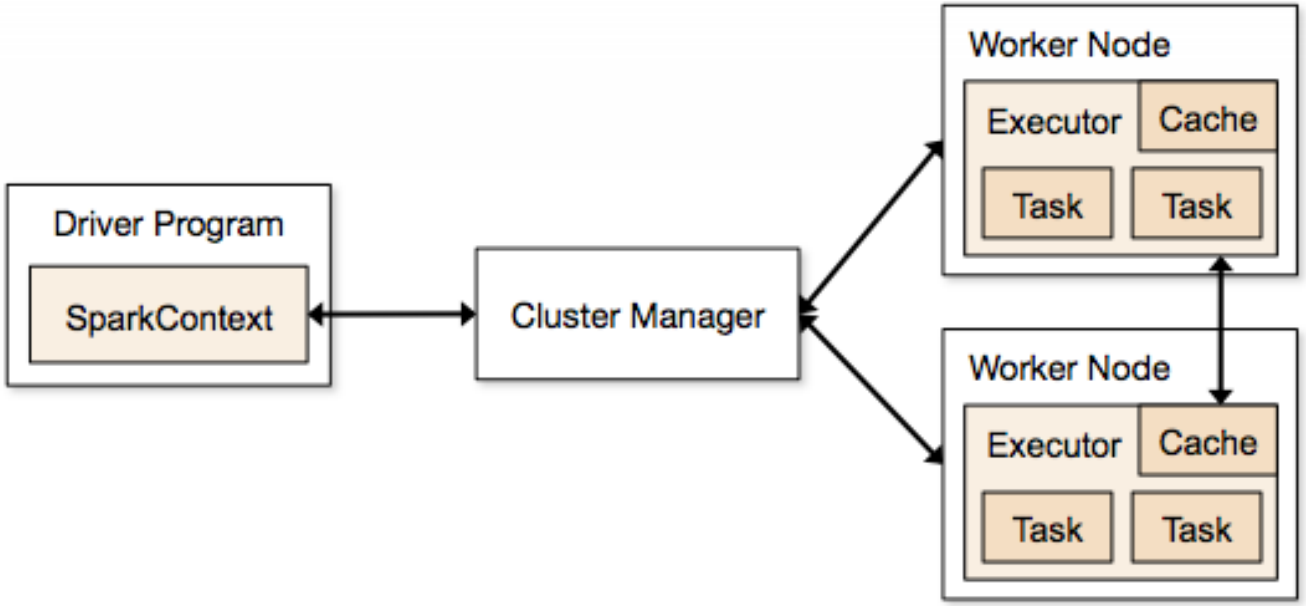
- Higher throughput than Storm
  - Spark Streaming: **670k** records/sec/node
  - Storm: **115k** records/sec/node
  - Commercial systems: **100-500k** records/sec/node



Interactive Shell

- Open the shell and ask questions
- Compile / save your code for scheduled jobs later

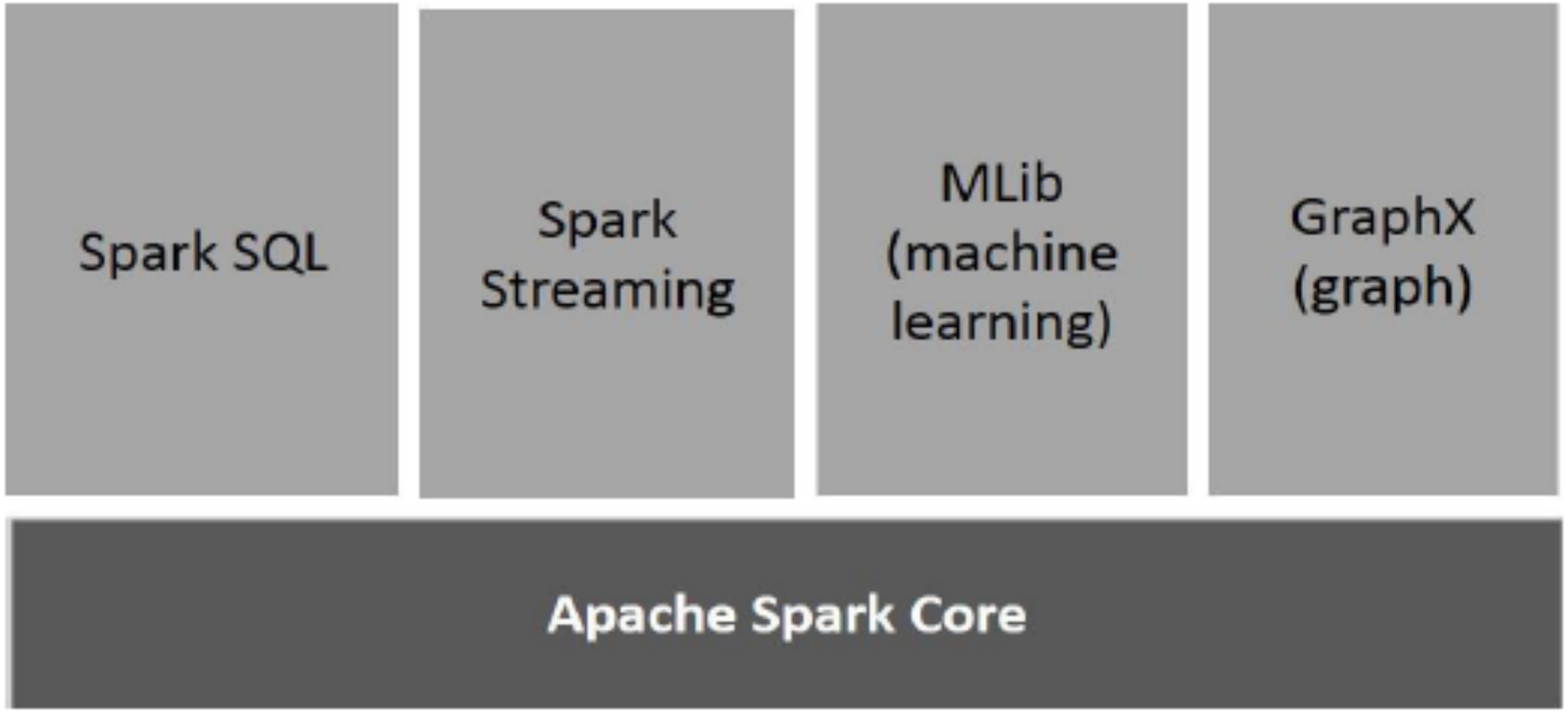
```
mbo@mbo-ubuntu-vbox:~/mbo/spark$ MASTER=spark://localhost:7077 ./spark-shell
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 0.9.0-SNAPSHOT
      /_/

Using Scala version 2.9.3 (Java HotSpot(TM) Client VM, Java 1.6.0_45)
Initializing interpreter...
Creating SparkContext...
Spark context available as sc.
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

Cluster Manager

- Apache supports different types of cluster managers
  - Standalone
  - Mesos
  - YARN
- Different scheduling, security and monitoring

Components of Spark



Spark Core

- Spark Core is the foundation of the overall project
- Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon.
- Accessed though API of the different languages

```
val conf = new SparkConf().setAppName("wiki_test") // create a spark config object
val sc = new SparkContext(conf) // Create a spark context
val data = sc.textFile("/path/to/somedir") // Read files from "somedir" into an RDD of (filename, content) pairs.
val tokens = data.flatMap(_.split(" ")) // Split each file into a list of tokens (words).
val wordFreq = tokens.map((_, 1)).reduceByKey(_ + _) // Add a count of one to each token, then sum the counts per word type.
wordFreq.sortBy(s => -s._2).map(x => (x._2, x._1)).top(10) // Get the top 10 words. Swap word and count to sort by count.
```

Spark SQL

- Spark SQL is a component on top of Spark Core that introduced new data abstractions such as SchemaRDD and Dataframe
- provides support for structured and semi-structured data
- Spark SQL provides a domain-specific language (DSL) to manipulate DataFrames in Scala, Java, or Python

```
import org.apache.spark.sql.SQLContext

val url = "jdbc:mysql://yourIP:yourPort/test?user=yourUsername;password=yourPassword" // URL for your database server.
val sqlContext = new org.apache.spark.sql.SQLContext(sc) // Create a sql context object

val df = sqlContext
  .read
  .format("jdbc")
  .option("url", url)
  .option("dbtable", "people")
  .load()

df.printSchema() // Looks the schema of this DataFrame.
val countsByAge = df.groupBy("age").count() // Counts people by age
```

Spark Streaming

- Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics.
- It ingests data in mini-batches and performs RDD transformations on those mini-batches of data

Mlib (Machine Learning Library)

- MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture
- Due in large part to the distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout
- Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning pipelines, including:
  - Regression, Classification, Random Forest, Cluster Analysis, PCA, Dimensionality Reduction Techniques, etc..

GraphX

- GraphX is a distributed graph-processing framework on top of Spark
- It provides an API for expressing graph computation that can model the user-defined graphs
- Uses API called Pregel abstraction, and it provides an optimized runtime to do the abstraction
- Facebook is using it

Spark Demonstration

ubuntu

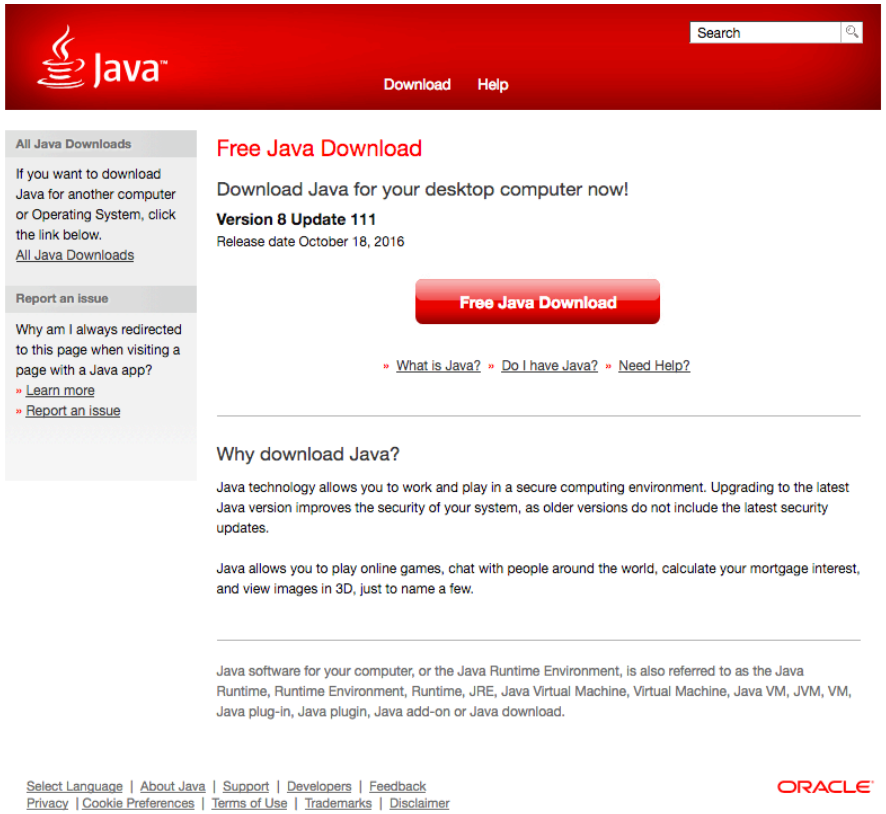
MAC OS X

Installation of Apache Spark with PySpark

- Java
- Python
- Scala
- Spark

Java

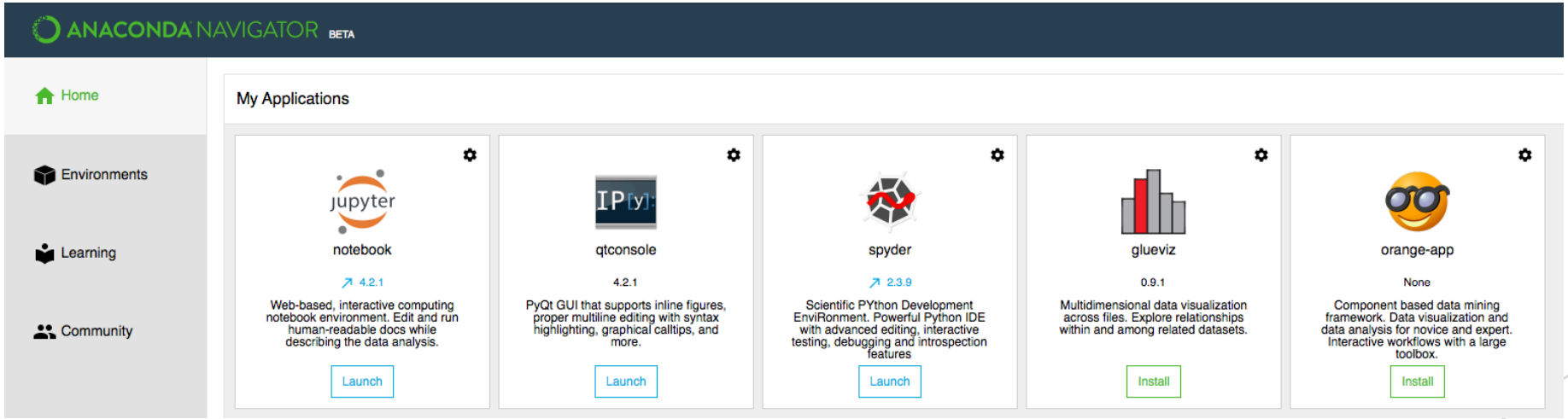
https://www.java.com/en/download/



Python

Anaconda

- https://www.continuum.io/downloads (Mac and Windows)
- Open source version of Anaconda is a high performance distribution of Python and R.
- Includes over 100 of the most popular Python, R and Scala packages for data science.



Terminal: iphyton

Scala

Homebrew

- Run git and ruby

- Homebrew installs packages to their own directory and then links their files into /usr/local.

Install Homebrew:

/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)”

Scala

In terminal:

- brew install scala

bash_profile is a configuration file for bash shell. When bash is invoked as an interactive login shell it first reads and executes commands from ~/.bash_profile.

- nano .bash_profile
- *export SCALA_HOME=/usr/local/bin/scala*
- *export PATH=$PATH:$SCALA_HOME/bin*

Save and Exit

Spark

Download at :

http://spark.apache.org/downloads.html

Navigate to where you downloaded the file

- mv spark-VERSION spark
- sudo mv spark /usr/local/

Navigate to the /usr/local/spark directory

- bin/spark-shell

Why Jupyter Notebooks?

The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

Example

Jupyter Notebook - Pyspark

- nano .bash_profile

- export SPARK_HOME=/usr/local/spark/
  export PATH="$SPARK_HOME/bin:$PATH"
  export PYTHONPATH="$SPARK_HOME/python:$PYTHONPATH"
  export PYSPARK_DRIVER_PYTHON=ipython
  export PYSPARK_PYTHON=python3
  export PYSPARK_DRIVER_PYTHON_OPTS='notebook'

- source .bash_profile

Machine Learning Example

- A retail company wants to understand customers purchase behavior (specifically, purchase amount).
- Customers data
- Excel

Variable	Definition
User_ID	User ID
Product_ID	Product ID
Gender	Sex of User
Age	Age in bins
Occupation	Occupation (Masked)
City_Category	Category of the City (A,B,C)
Stay_In_Current_City_Years	Number of years stay in current city
Marital_Status	Marital Status
Product_Category_1	Product Category (Masked)
Product_Category_2	Product may belongs to other category also (Masked)
Product_Category_3	Product may belongs to other category also (Masked)
Purchase	Purchase Amount (Target Variable)

spark-csv

- https://spark-packages.org/package/databricks/spark-csv
- Make sure you copy the directory in PATH/spark

  /usr/local/spark
- pyspark --packages com.databricks:spark-csv_2.10:1.3.0
- Details: https://github.com/databricks/spark-csv

Questions?

Thank you