Apache Spark: Lightning Fast Data Analytics

Downloading and Installing Spark

1. Download Java SE
2. Dowload Python for Windows
    a. Anaconda Python - Scientific Computing
    b. Enthought Canopy
3. Download Apache Spark

References:

1. Apache Spark Channel - Youtube
https://www.youtube.com/user/TheApacheSpark

2. Apache Spark - Documentation
http://spark.apache.org/documentation.html

3. Spark Summit East 2015 : March 17-18

What is Spark?

Cluster-Computing Platform - designed to be fast and general purpose.

What purposes can we imagine:
Big-Data Analysis
Scientific Simulations. etc

Data-Science Tasks
Analyze and Model Data - with goal of answering a question or discovering insight.
Tackle problems with larger data-sizes compared to Python-Pandas or R.
Supports calling out external programs in Matlab or R.

Data-Processing Engine for Computer-Clusters::
Primarily a replacement for the Map-Reduce Paradigm of Hadoop ecosystem. It is NOT YET a replacement of the Hadoop ecosystem!

Speed and Efficiency : Efficient data-structures and algorithms implemented, to run complex applications ON DISK or IN MEMORY

==========

The essential difference lies in the way the two systems reads, processes and writes data. Hadoop does the following:

1. reads huge chunks of data into the Hard Disk,
2. processes the chunks as a batch - using a 2-stage Map-Reduce Paradigm
3. writes again into hard-disk

Spark however:

1. Does all the computations and write operations in the RAM
2. The Data can be read from any existing Distributed Data repository, and handled back.

Why consider it?

1. If you're new to Big Data Analytics - this is a good platform to start off with.
   Flexibility - Batch Jobs / Data Stream Processing
2. It is still under development,
   with active contribution from Industry and Open Source community.
3. Spark - Compatibility with existing Hadoop ecosystem - benefits of existing mature technologies.

Getting Hand's On - with Spark

1. Downloading and Unpacking
   a. All you need is to have java installed on your system PATH, or the JAVA_HOME environment variable pointing to a Java installation.
   b. Spark runs on Java 6+ and Python 2.6+. For the Scala API, Spark 1.2.0 uses Scala 2.10.
   c.
2. Spark Shells - Interactive shels that enable ad-hoc data analysis
   a. similar to Matlab shells or Python/R/Bash shells or the Windows CMD
   b. Interact and manipulate data - distributed on disk or in memory ACROSS MANY MACHINES, and distributes the processing too!.
   c. Spark supports many languages. However, the INTERACTIVE SHELLS are available only for PYTHON and SCALA. It's advisable to adopt either of the shells, to facilitate learning/exploration of the SPARK API for DataProcessing.

Spark - Unified Stack

Core: "Computational Engine" -

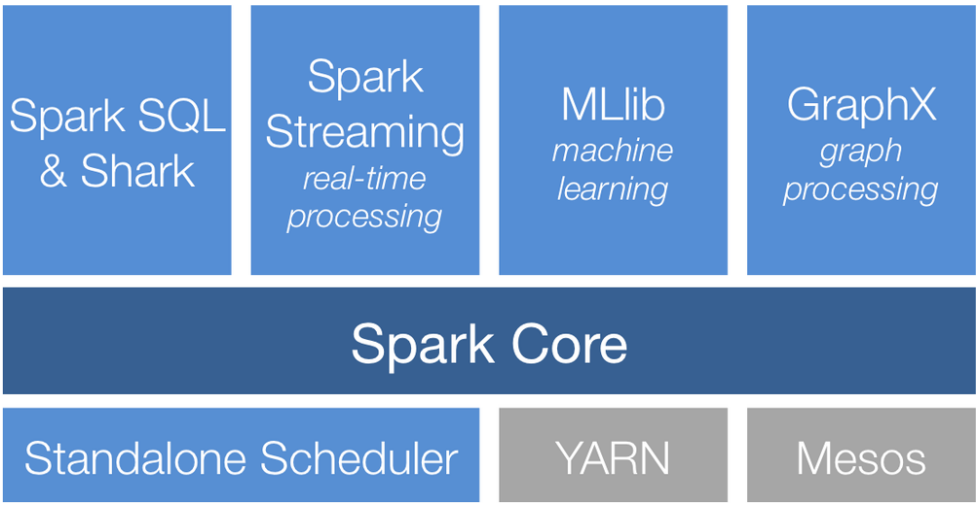
1. Scheduling, distributing, monitoring applications over a cluster, etc.
2. Powers multiple High-level components specialized for various work-loads - SQL, Machine-Learning, etc. These components inter-operate closely, and hence can be combined together as libraries, in a software project.

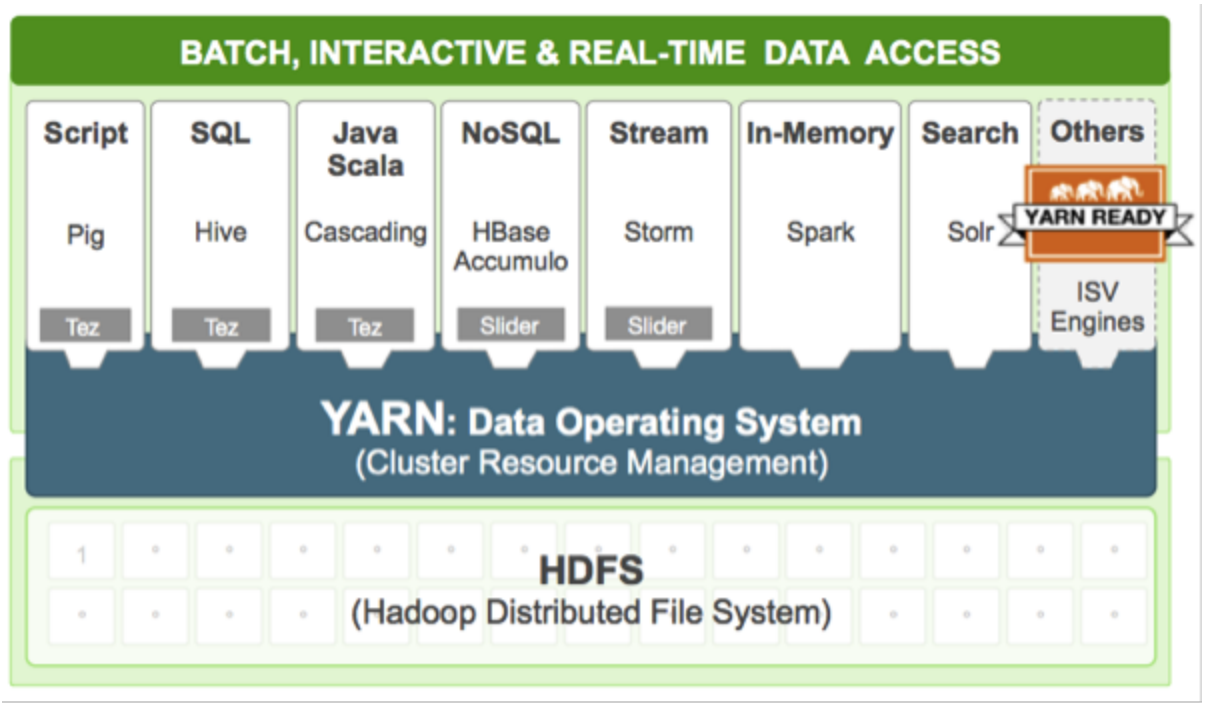
==========

Click here to provide Feedback for Gemmill Workshop: Apache Spark

Tight - Integration:

1. Enables the individual to run only one Software system, instead of 5-10 independent software systems.
2. Build applications combining different processing models - Real Time Data-streaming like Twitter Feeds + Batch data processing of Wikipedia database.





==========

Click here to provide Feedback for Gemmill Workshop: [Apache Spark]

The Magic inside Spark: Resilient Distributed Datasets(RDDs)
References : Prof. Matei Zaharia

Introduction to Core Spark Concepts

Driver Program (Master) → Worker Nodes(Slaves)

Driver Program
- Applications main() function
- Launches various parallel operations on a cluster.
- Defines Distributed Datasets on the Cluster and Applies operations on them.
- Accesses SPARK through a SPARK-CONTEXT(sc) -
  - representing a connection to a cluster.
  - In a shell(Driver Program) - the SparkContext is the variable **sc.**

The DriverProgram(DP) needs a SparkContext before it can Create and Operate on RDDs.
To run operations, the DP typically manage a number of nodes called EXECUTORS.

Python and Scala support Functional-Programming Paradigm, which is used extensively by the Spark API. Most of the API consists of passing functions to its Operators to run them on the cluster.

Example: Wordcount.py

==========