SPARK and PySPARK

Dr. Yudhishthir Raut

Why Spark?

The following four main reasons from Apache Spark™ official website a

1. The Speed are good enough to convince you to use Spark.

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Apache Spark has an advanced DAG execution engine that supports acyclic data flow and in-memory computing.

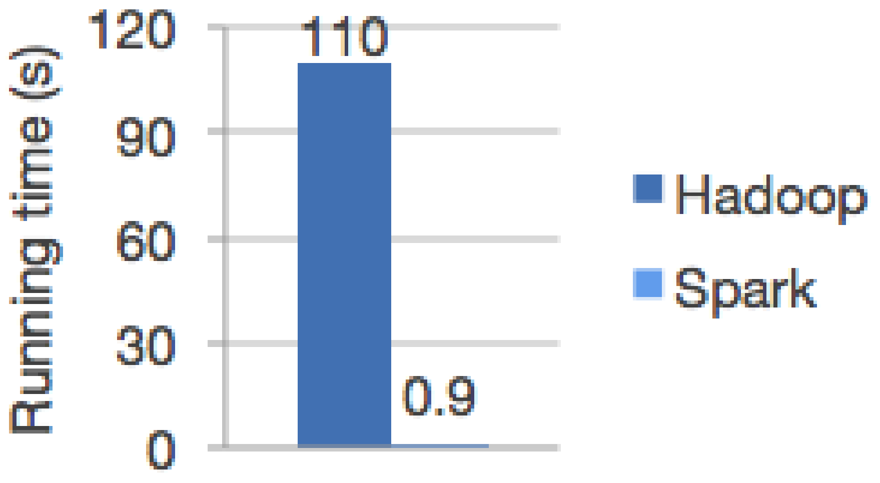


Fig. 1: Logistic regression in Hadoop and Spark

2. Ease of Use

- Write applications quickly in Java, Scala, Python, R. Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it interactively from the Scala, Python and R shells.

- Generality

Combine SQL, streaming, and complex analytics.

- Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning,
- GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.

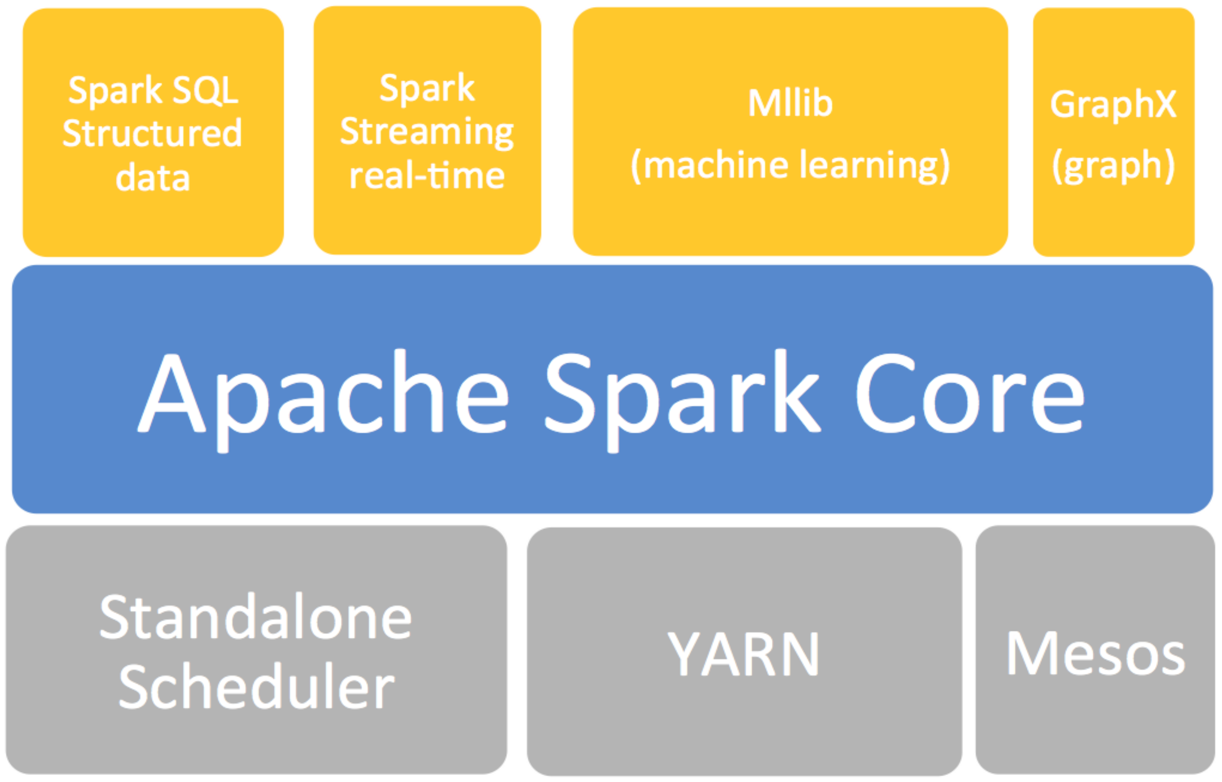


Fig. 2: The Spark stack

4. Runs Everywhere

- Spark runs on Hadoop, Mesos, standalone, or in the cloud.
- It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

Fig. 3: The Spark platform

Why Spark with Python (PySpark)?

No matter you like it or not, Python has been one of the most popular programming languages.

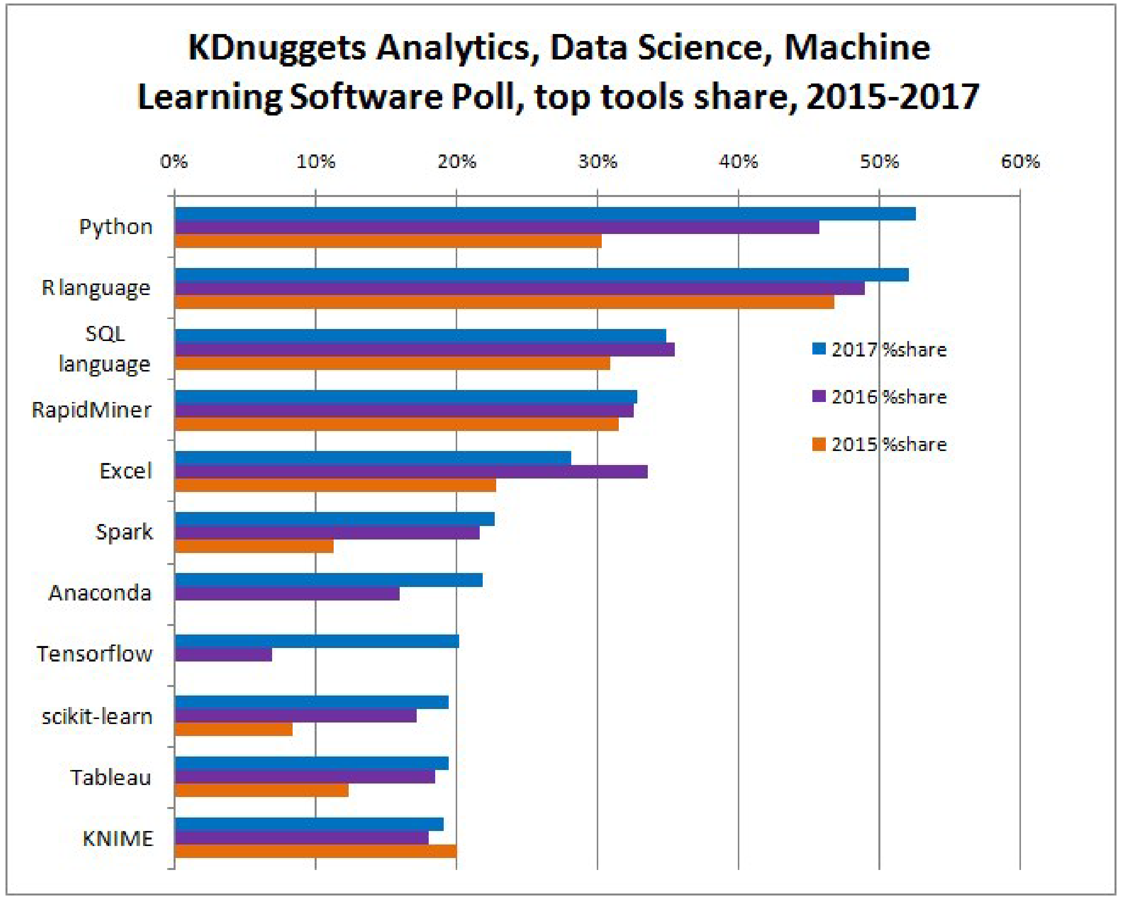


Fig. 4: KDnuggets Analytics/Data Science 2017 Software Poll from kdnuggets

The features of Spark

1. Spark provisions for creating applications that use the complex data. In-memory Apache Spark computing engine enables up to 100 times performance with respect to Hadoop.

2. Execution engine uses both in-memory and on-disk computing. Intermediate results save in-memory and spill over to disk.

3. Data uploading from an Object Store for immediate use as a Spark object instance. Spark service interface sets up the Object Store.

4. Provides high performance when an application accesses memory cache from the disk.

5. Contains API to define Resilient Distributed Datasets (RDDs). ROD is a programming abstraction. ROD is the core concept in Spark framework.

The features of Spark…

6. Processes any kind of data, namely structured, semi-structured or unstructured data arriving from various sources.

7. Supports many new functions in addition to Map and Reduce functions.

8. Optimizes data processing performance by slowing the evaluation of Big Data queries.

9. Provides concise and consistent APIs in Scala, Java and Python. Spark codes are in

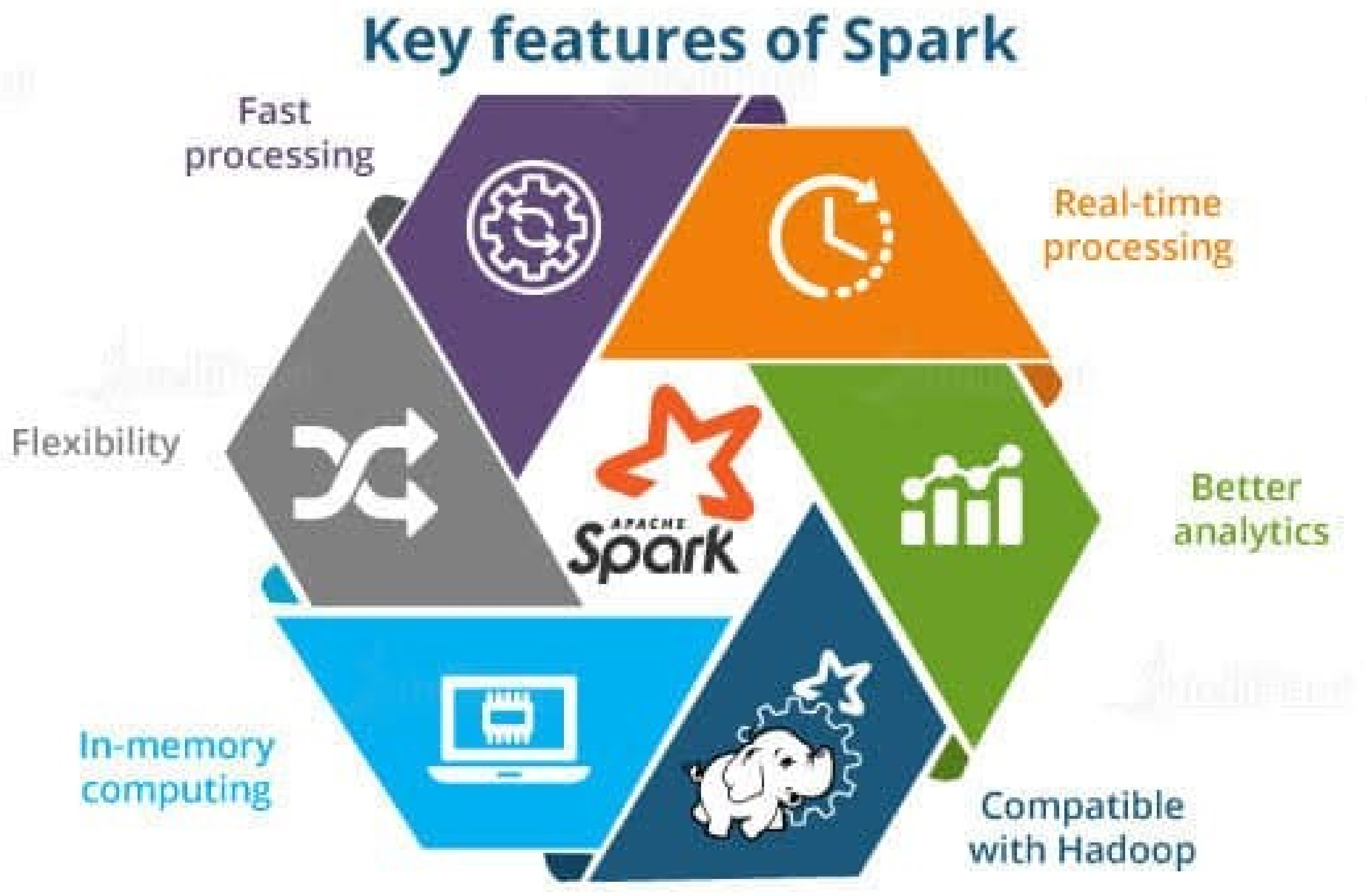
Scala and run on JVM environment.

10. Supports Scala,Java, Python, Clojure and R languages.

11. Provides powerful tool to analyze data interactively using shell which is available in either Scala (which runs on the Java VM and is thus a good way to use existing Java libraries) or Python.

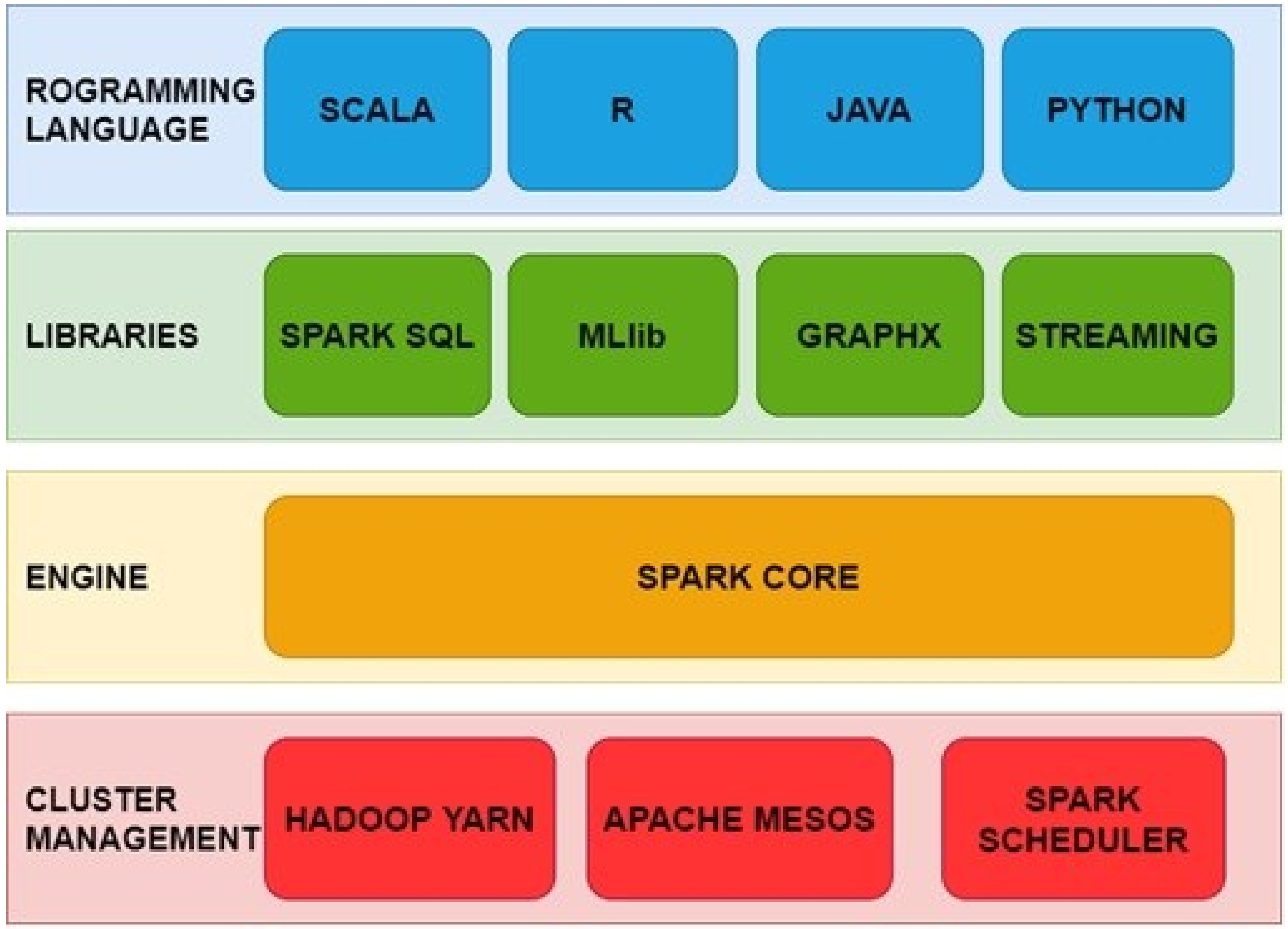
Key Features of SPARK

Apache Spark is an **open-source unified analytics engine for large-scale data processing**. Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance.



Spark Software Stack

The main components of Spark stack are SQL, Streaming, R, Graphx, MLib and Arrow at the applications support layer. Spark core is the processing engine. Data Store provides the data to the processing engine. Hadoop, YARN or Mesos facilitates the parallel running of the tasks and the management and scheduling of the resources.



Spark stack imbibes generality to Spark. Grouping of the following forms

- Spark stack: Spark SQL for the structured data. The SQL runs the queries on Spark data in the traditional business analytics and visualization applications.
- Spark SQL enables Spark datasets to use JDBC or ODBC APL HQL queries also run in Spark SQL.
- Runs UDFs for inline SQL, distributed DataFrames, Parquet, Hive and Cassandra Data Stores.

- Spark Streaming is for processing real-time streaming data. Processing is based on micro-batches style of computing and processing. Streaming uses the DStream which is basically a series of RDDs, to process the real-time data.
- SparkR is an R package used as light-weight front end for Apache Spark from R. Spark API uses SparkR through the RDD class. A user can interactively run the jobs from the R shell on a cluster. An RDD API is in the distributed lists in R.
- Spark Mlib is Spark's scalable machine learning library. It consists of common learning algorithms and utilities. MLib includes classification, regression, clustering, collaborative filtering, dimensionality reduction and optimization primitives. MLib applies in recommendation systems, clustering and classification using Spark

- Spark Graphx is an API for graphs. Graphx extends the Spark RDD by introducing the Resilient Distributed Property. GraphX computations use fundamental operators (e.g., subgraph, joinVertices and aggregateMessages).
- GraphX uses a collection of graph algorithms for programming. Graph analytics tasks are created with ease using GraphX.
- Spark Arrow for columnar in-memory analytics and enabling usages of vectorised UDFs (VUDFs). The Arrow enables high performance Python UDFs for SerDe and data pipelines.

Data Analysis with PySpark

"Analysis of data is a process of inspecting, cleaning, transforming and modeling data with the goal of discovering useful information, suggesting conclusions and supporting decision-making."

Following are the steps for analyzing the data:

1. Data Storage: Store of data from the multiple sources after acquisition. The Big Data storage may be in HDFS compatible files, Cassandra, Hive, HDFS or S3.

2. Data pre-processing: This step requires:

   (a) dropping out of range, inconsistent and outlier values,

   (b) filtering unreliable, irrelevant and redundant information, (c) data cleaning, editing, reduction and/or wrangling,

   (d) data-validation, transformation or transcoding.

3. Extract, transform and Load (ETL) for the analysis

4. Mathematical and statistical analysis of the data obtained after querying relevant data needing the analysis,

5. Applications of analyzed data, for example, descriptive, predictive and prescriptive analytics, business processes (BPs), business process automation (BPA), business intelligence (BI), decision modelling and knowledge discovery.

PySpark — An Effective ETL Tool?

Many of you may be curious about **ETL Tools** and the use of the ETL process in the world of data hubs where data plays a significant role. Today, we will examine this more closely.

What is ETL?

- ETL (which stands for *Extraction, Transform* and *Load)* is the generic process of extracting data from one or more systems and loading it into a data warehouse or databases after performing some intermediate transformations.
- There are many ETL tools available in the market that can carry out this process.

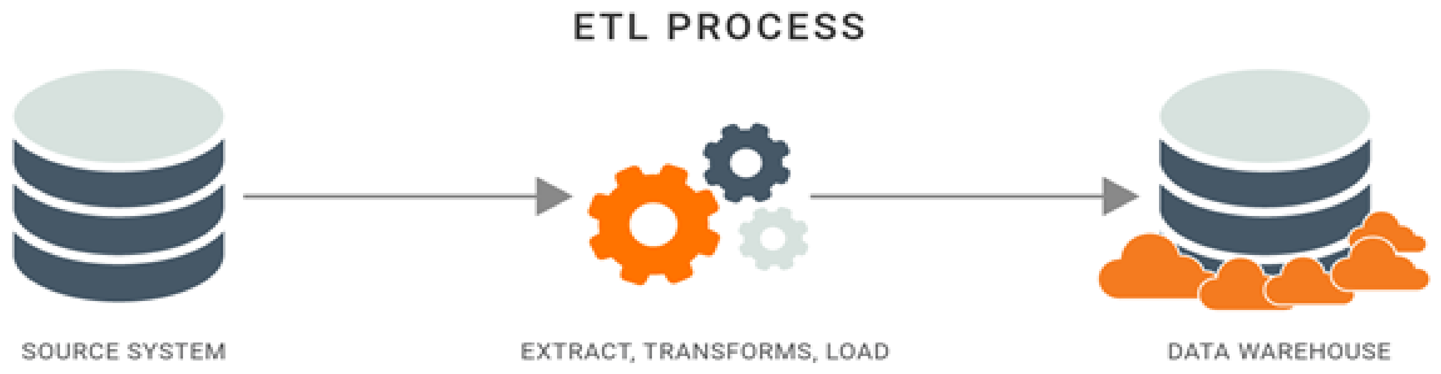
A standard ETL tool like *PySpark*, supports all basic data transformation features like *sorting*, *mapping*, *joins*, *operations*, etc. PySpark's ability to rapidly process massive amounts of data is a key advantage.

Some tools perform a complete ETL implementation while some tools help us create a custom ETL process from scratch, and there are a few those fall somewhere in between. Before going into the detail of **PySpark**, let's first understand some important features that an ETL tool should have.

Features of ETL Tools

ETL comprises of 3 processes which follow a sequence, beginning with extraction and ending with load. Let us look at these steps more closely:

Extract
This is the first step of ETL and it is the process of extracting or fetching data from various data sources. This can include most databases (RDBMS/NOSql) and file formats like JSON, CSV, XML, and XLS.



Transform

- In this process, all the extracted data is kept in a staging area where raw data is transformed into a structured format and into a meaningful form for storing it into a data warehouse.
- A standard ETL tool like PySpark that we will look at later, supports all basic data transformation features like sorting, mapping, joins, operations, etc.

Load

- This is the last process of the ETL tool in which the transformed data is loaded into the target zone or target warehouse database. This stage is a little challenging because a huge amount of data needs to be loaded in a short period.

Enter PySpark

- PySpark is a combination of Python and Apache Spark. It is a python API for spark which easily integrates and works with RDD using a library called ‘py4j’. It is the version of Spark which runs on Python.

- As per their official website, “Spark is a unified analytics engine for large-scale data processing”.

- The Spark core not only provides many robust features apart from creating ETL pipelines, but also provides support for machine learning (MLib), data streaming (Spark Streaming), SQL (Spark Sql), and graph processing (Graph X).

Advantages of PySpark

- **Speed:** It is 100 times faster than traditional large-scale any data processing frameworks.
- **Real-Time Computation:** The main key feature is its in-memory processing in the PySpark framework, it shows low latency.
- **Caching and Disk Persistence:** Simple programming layer provides powerful caching and disk persistence capabilities.
- **Deployment:** It can be deployed through Hadoop via Yarn, Mesos or Spark’s own cluster manager.

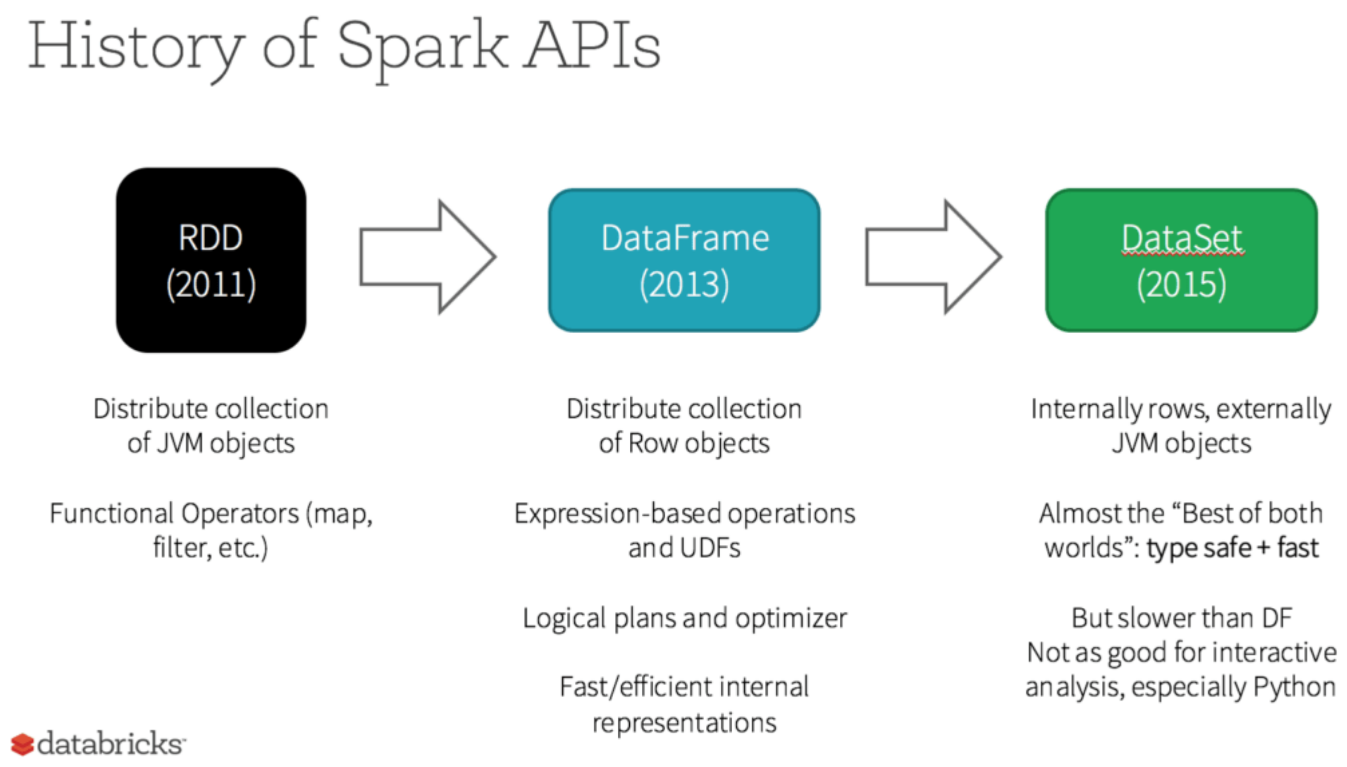
Why PySpark?

- PySpark’s ability to rapidly process massive amounts of data is a key advantage. If you are looking to create an ETL pipeline to process a huge amount of data quickly or process streams of data, PySpark offers a worthy solution.

- It is not an ETL solution out of the box, but it would be the best one for an ETL pipeline deployment.

RDD ?

Resilient Distributed Dataset (RDD) is **the fundamental data structure of Spark**. They are immutable Distributed collections of objects of any type. As the name suggests is a Resilient (Fault-tolerant) records of data that resides on multiple nodes.


History of Spark APIs
RDD (2011)
DataFrame (2013)
DataSet (2015)
Distribute collection of JVM objects
Functional Operators (map, filter, etc.)
Distribute collection of Row objects
Expression-based operations and UDFs
Logical plans and optimizer
Fast/efficient internal representations
Internally rows, externally JVM objects
Almost the "Best of both worlds": type safe + fast
But slower than DF
Not as good for interactive analysis, especially Python
databricks

Programming with RDDs

Spark Resilient Distributed Dataset (RDD) is a collection of objects distributed on many computing nodes. Each RDD can split into multiple partitions, which may be computed in parallel on different nodes of a cluster.

Characteristics of RDDs

1. fault-tolerant abstraction which enables In-Memory cluster computations,

2. immutable (thus read-only) and partitioned distributed collection of objects,

3. have interface which enables transformations that apply the same to many data objects,

4. create only through the deterministic operations on either (i) data in stable Data store such as file or (ii) operations on other RDDs,

5. are parallel data structures,

6. enable efficient execution of iterative algorithms,

7. enable efficient execution of interactive data-mining,