Apache Spark

In-Memory Compute Engine

Challenges of MapReduce

- **Challenges of MapReduce**:
  - **Low Performance**: Many IO disk seeks.
  - **Complexity**: Requires extensive code, even for simple tasks.
  - **Batch Processing Only**: No real-time processing.
  - **Steep Learning Curve**: Difficult to master.
  - **Rigid Paradigm**: Must think in Map-Reduce terms.
  - **No Interactivity**: Lacks interactive mode.

What is Apache Spark?

- **MapReduce Bottleneck**: MapReduce was slow and required extensive coding, even for simple tasks.
- **Apache Spark to the Rescue**: Apache Spark is a:
  - General Purpose
  - In-Memory
  - Compute Engine

Apache Spark:

- **Plug and Play Compute Engine**: Used for distributed processing.
- **Requirements**:
  - **Storage**: HDFS, ADLS Gen2, Amazon S3, Google Cloud Storage.
  - **Resource Manager**: YARN, Mesos, Kubernetes.
- **Definition**: A multi-language engine for data engineering, data science, and machine learning on single-node or cluster setups.

Is Apache Spark a Replacement for Hadoop?

- **Explanation**: Hadoop provides 3 components:
  - Storage (HDFS)
  - Compute (MapReduce)
  - Resource Management (YARN)
- **Spark vs. Hadoop**:
  - Spark can replace MapReduce but not Hadoop as a whole.
  - Spark is a plug-and-play compute engine that requires Storage and a Resource Manager to function.
  - It is not bound to HDFS and YARN only.

Components Needed for Spark:

Storage:

Examples:

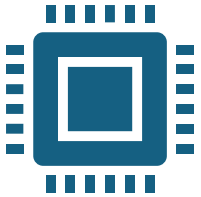
HDFS, Amazon S3, Azure ADLS Gen2, Google Cloud Storage, Local Storage, etc.

Resource Manager:

Examples:

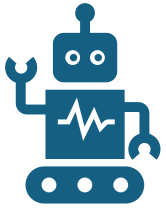
YARN, Mesos, Kubernetes

Spark Features



Spark Performance:

Spark is **10x to 100x faster** than traditional MapReduce due to its In-Memory data storage and processing.



Language Support for Spark:

Languages:

- Python
- Scala (Spark itself is written in Scala)
- Java
- R

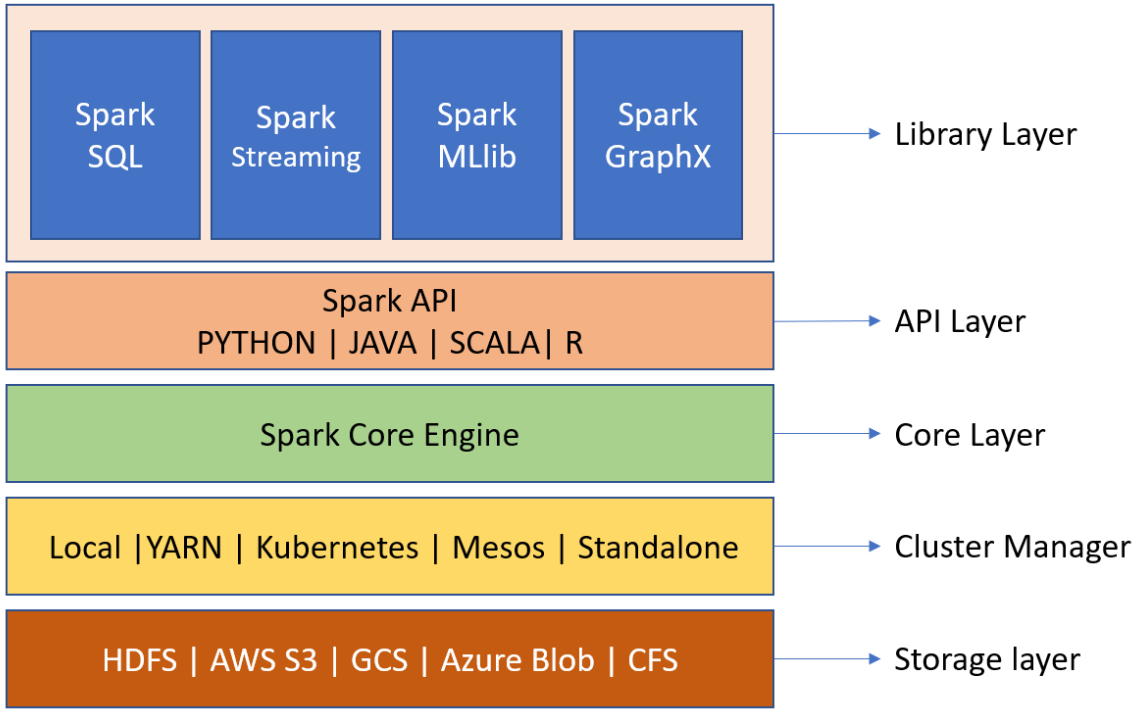
PySpark:

Spark with Python is referred to as PySpark.

Key Features:

- **General Purpose | In-Memory | Compute Engine**
- **High Performance**: Fewer IO disk seeks due to in-memory processing.

Spark Architecture



Spark Architecture:

Apache Spark Layers

Apache Spark Layers:

- **Spark Core**:
  - **Foundation**: Provides basic functionalities like task scheduling, memory management, and fault tolerance.
  - **Resilient Distributed Dataset (RDD)**: The fundamental data structure in Spark.
- **Spark Components**:
  - **Spark SQL**: For querying structured data.
  - **Spark Streaming**: For real-time data processing.
  - **MLlib**: For machine learning.
  - **GraphX**: For graph processing.

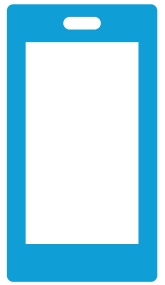
Spark Data Processing Steps

- **Load : Data Ingestion**
  - Collect and load data from various sources into Spark.
- **Transform : Data Processing**
  - **Transformations**: Apply operations such as filtering, mapping, and aggregating.
  - **Actions**: Trigger computations and return results, e.g., collect, count, or save.
- **Write : Data Output**
  - Write the processed data to storage or serve it to visualization tools.

Spark Basic Programming APIs : easy to difficulty

Data frame API
(java,scala,Python)

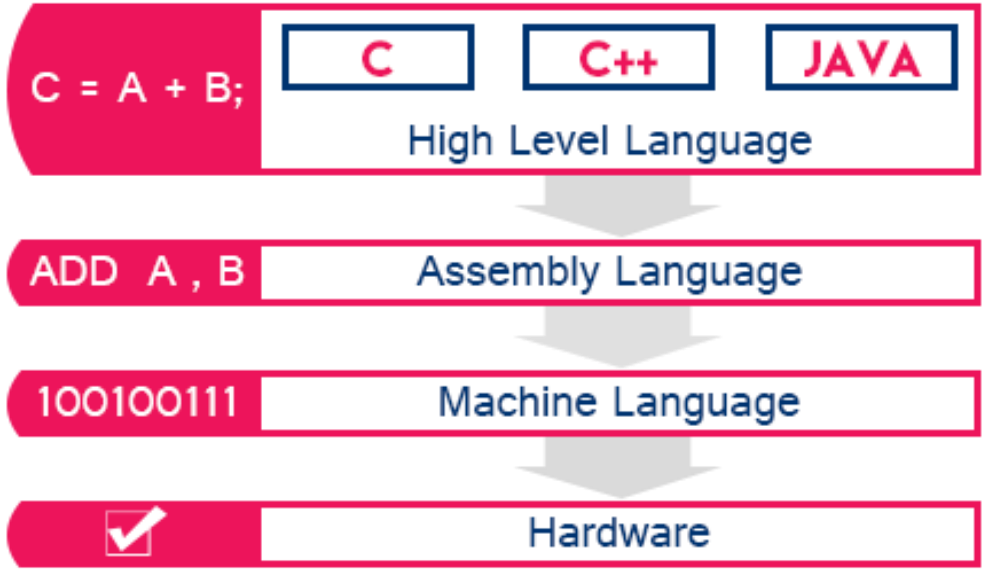
SQL API



Core API

Spark APIs : **Core API**

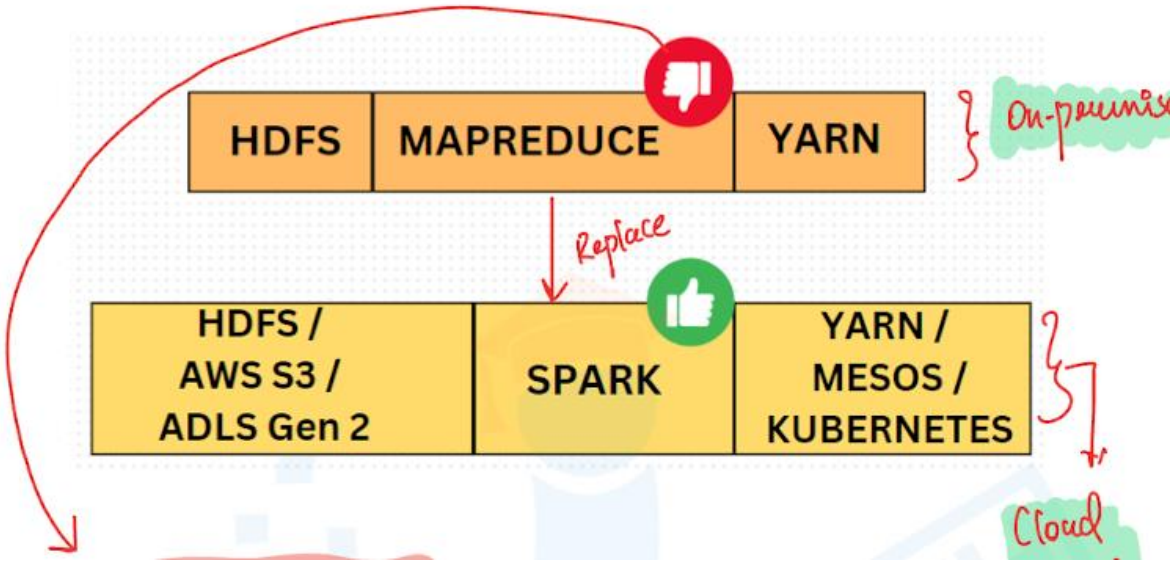
- **RDD (Resilient Distributed Dataset)**:
  - **Definition**: The fundamental data structure in Spark, providing fault-tolerant and distributed data processing.
  - **Operations**:
    - **Transformations**: map, filter, flatMap, reduceByKey, etc.
    - **Actions**: collect, count, saveAsTextFile, etc.
  - **Characteristics**: Low-level API, offers fine-grained control, and requires more coding.



Spark APIs: Difficulty Level

- **Easy to Use**:
  - **Spark SQL**: SQL-like queries for data manipulation.
  - **DataFrames**: High-level abstraction with named columns.
  - **MLlib**: Pre-built machine learning algorithms and pipelines.
  - **Spark Streaming**: Real-time processing with high-level abstractions.
- **Moderate Difficulty**:
  - **GraphX**: Graph processing and analytics.
- **Advanced**:
  - **RDD (Resilient Distributed Dataset)**: Low-level data structure requiring manual management of transformations and fault tolerance.

Apache spark



On-premises clusters/Mimic of On-premises

Most of cloud

Spark vs. Databricks:

APACHE SPARK: OPEN-SOURCE DISTRIBUTED PROCESSING FRAMEWORK.

DATABRICKS: SPARK ON THE CLOUD WITH ADDITIONAL FEATURES.

Databricks account creation

- Register Free account : https://www.databricks.com/try-databricks#account
- Login : https://community.cloud.databricks.com/login.html

Demo

Core API

Spark-Dataframe-&-SQL-API

Higher-level-APIs

Spark APIs : Higher-Level APIs

- **Spark SQL**:
  - **Definition**: Provides a SQL interface to interact with data.
  - **Features**:
    - **DataFrames**: Distributed collections of data organized into named columns.
    - **SQL Queries**: Execute SQL queries on data.
  - **Advantages**: Simplifies complex data queries and integrates with various data sources.
- **Spark Streaming**:
  - **Definition**: Enables real-time data processing.
  - **Advantages**: Processes data in near real-time.

Spark APIs : Higher-Level APIs

- **MLlib**:
  - **Definition**: Machine learning library.
  - **Features**:
    - **Algorithms**: Includes classification, regression, clustering, and recommendation algorithms.
    - **Pipelines**: Tools for building and evaluating machine learning pipelines.
- **GraphX**:
  - **Definition**: Graph processing library.
  - **Features**:
    - **Graph Computations**: Algorithms for graph processing and analysis.
    - **GraphFrames**: DataFrames with graph-specific operations.

Why Choose Higher-Level APIs Over Core API?

Ease of Use: Higher-level APIs (e.g., DataFrames, Spark SQL) offer simpler, more abstracted interfaces.

Reduced Complexity: Built-in functions reduce the need for extensive coding.

Performance Optimizations: Automatic optimizations (e.g., Catalyst, Tungsten) enhance performance.

SQL and ML Integration: Seamless integration with SQL queries and machine learning models.

Advanced Features: Includes features like real-time processing (Spark Streaming) and MLlib.

Increased Productivity: Simplifies development, saving time and effort.

Assignment

Compare Apache Spark and Databricks and write a blog on LinkedIn and tag.