Overview of Apache Spark and its Ecosystem

Presented By:
Nila Michelle
Jaime Joshka
Mithra Devi
Vandana Gupta
Sabrin Maria

Table of Contents

01

Introduction to Apache Spark

What is Apache Spark ?

- Spark is an open-source, distributed data processing framework created at UC Berkeley in 2009.
- Designed to handle large-scale data processing quickly and efficiently.
- Supports multiple data processing types: Batch Processing, Real-time Streaming, Machine Learning,Graph Processing.
- Used by major companies like Netflix, Uber, and Facebook for data analysis

Key Features Of Using Apache Spark

Speed: In-memory processing makes Spark 10-100 times faster

Ease of Use: Simple APIs available in Python, Java,Scala,and R.

Versatility: Handles batch, streaming, machine learning, and graph processing in one too

Less Coding: Reduces programming effort with built-in functions and libraries.

Integration: Works with HDFS, cloud storage, databases, and other data sources.

Scalability & Reliability: Scales easily, handles failures, prevents data loss.

Hadoop VS Apache Spark

Feature	Spark	Hadoop
Speed	In-memory processing (10-100x faster)	Disk-based processing (slower)
Simplicity	Easy-to-use APIs (Python, Java, Scala, R)	Complex MapReduce programming
Purpose	Multi-purpose: Batch, Streaming, ML, Graph	Mainly Batch Processing
Coding Effort	Requires less coding	Requires more coding effort
Integration	Works with HDFS, cloud storage, databases; can work alongside Hadoop	Primarily works with HDFS
Industry Trend	Increasing adoption by companies	Gradual shift towards Spark

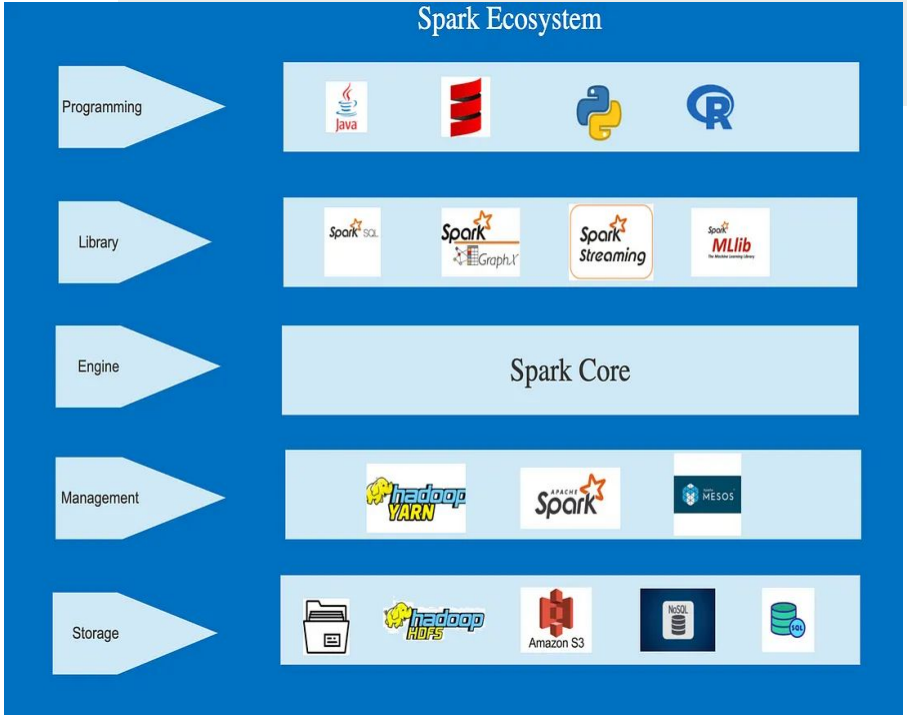
02

Spark Ecosystem

Spark Ecosystem

Spark Core component: -

- It is the foundation for parallel and distributed processing of large datasets.
- It is accountable for all the basic I/O functionalities, scheduling and monitoring the jobs on spark clusters, task dispatching, networking with different storage systems, fault recovery and efficient memory management
- **Scala** is the native language, offering the best performance.**Java** is widely used in enterprise applications due to its stability and scalability.**Python (PySpark)** is popular for data science, **R (SparkR)** is used for statistical analysis, and **SQL (Spark SQL)** allows querying structured data easily.



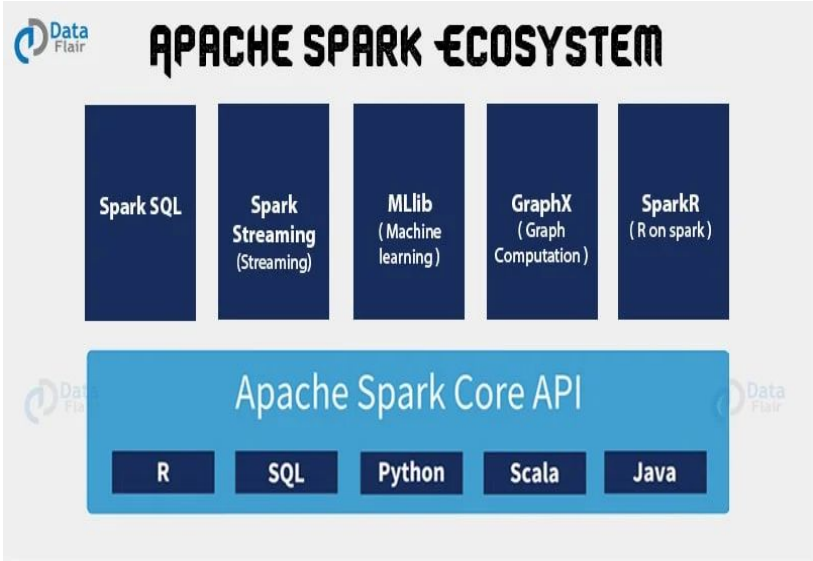
Components of Spark Ecosystem

- **Spark SQL:** This component is a distributed framework for structured data processing. Using Spark SQL, Spark gets more information about the structure of data and the computation. With this information, Spark can perform extra optimization

- **Spark Streaming:** It is an add-on to the core Spark API that enables scalable, fault-tolerant stream processing of live data streams from sources like Kafka, Flume, Kinesis, or TCP sockets. It operates using various algorithms, delivering processed data to file systems and live dashboards, using micro-batching for real-time streaming.

- **GraphX**:GraphX in Spark is API for graphs and graph parallel execution. It is network graph analytics engine and data store. *Clustering, classification, traversal, searching, and pathfinding* is also possible in graphs.

- **ML Lib**: Library to run machine learning algorithms on large data sets in a native distributed environment. MLlib in Spark is a scalable machine learning library designed for high performance and ease of use, providing algorithms for clustering, regression, classification, and collaborative filtering



03

Resilient Distributed Datasets (RDDs)

Introduction to RDD (Resilient Distributed Datasets)

- **Resilient Distributed Datasets (RDDs)** are the **fundamental data structure** in Apache Spark.
- They enable **fault-tolerant, distributed, and parallel processing** of large datasets.
- RDDs are **immutable, partitioned, and lazily evaluated**, making Spark highly efficient.
- They serve as the **building blocks** for higher-level Spark APIs like **DataFrames and Datasets**.
- Designed to work efficiently with both **in-memory and disk-based** computations.

Key Features of RDDs

Immutable	Once created, RDDs cannot be changed. Instead, transformations create new RDDs
Partitioned & Distributed	Data is automatically split across multiple machines for parallel processing.
Lazy Evaluation	Computations are run only when an action is triggered
Fault Tolerance	If a node fails, RDDs restore data by recomputing from lineage.
In-Memory Processing	Spark caches RDDs in memory for faster computation.
Transformations and Actions	Supports transformations (map, filter, reduce) and actions (collect, count).
Schema-Free (Flexible Data Handling)	Can store structured, semi-structured, and unstructured data without predefined schemas.

How to Create RDDs in Apache Spark

1

Parallelizing an In-Memory Collection

Used for **small datasets** stored in memory.
Example:
val rdd = sc.parallelize(Seq(1, 2, 3, 4, 5))
Parallelism is determined by `spark.default.parallelism` (or manually set).

2

From External Storage (HDFS, Local Files, etc.)

Reads data from **files, databases, or distributed storage systems**.
Example:
val rdd = sc.textFile("hdfs://path/to/file.txt")
File is split into partitions based on Hadoop's block size (modifiable using a second argument).

3

Transforming an Existing RDD

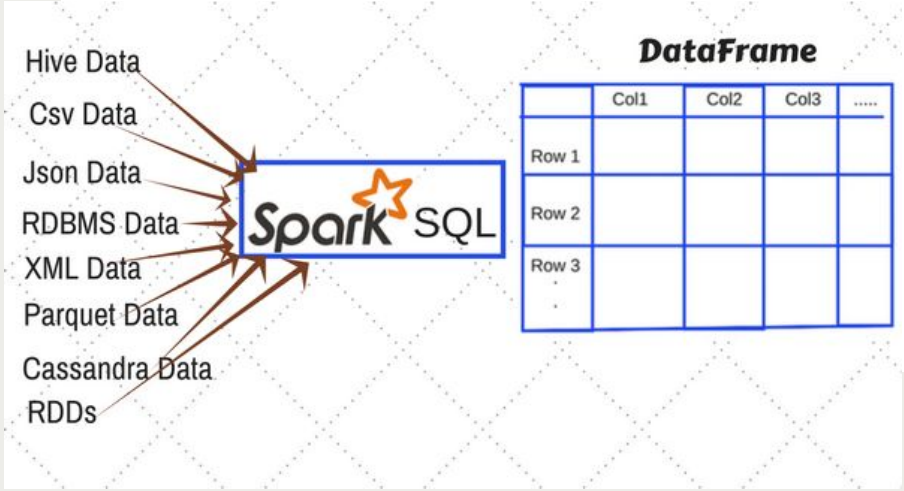
RDDs are **immutable**, so transformations like `map()`, `filter()` create **new RDDs**.
Example:
val newRDD = existingRDD.map(x => x * 2)
This maintains **RDD lineage**, allowing fault recovery.

04

Dataframe

DATAFRAME

- It is a distributed collection of data, which is organized into named columns.
- Equivalent to relational tables with good optimization techniques.
- Can be constructed from an array of different sources such as Hive tables, Structured Data files, external databases, or existing RDDs.
- Designed for modern Big Data and data science applications taking inspiration from DataFrame in R Programming and Pandas in Python.
- Simplifies and optimizes big data processing, making it a key component of Apache Spark for large-scale data analytics.



FEATURES OF DATAFRAME

1

Support for various data formats: Hive, CSV, XML, JSON, RDDs, Cassandra, Parquet, etc

2

Support for integration with various Big Data tools & APIs for Java, R, Python, and Spark

3

The ability to process kilobytes of data on smaller machines and petabytes on clusters

4

Catalyst optimizer for efficient data processing across multiple languages

5

Structured data handling through a schematic view of data

6

Custom memory management to reduce overload and improve performance compared to RDDs

05

RDD vs Dataframe

RDD vs DATAFRAME

FEATURES	RDD(Resilient Distributed Dataset	DATAFRAMES
Data Structure	Low-level distributed collection of objects	High-level abstraction with named columns, like SQL tables.
Performance	Slower due to no built-in optimizations.	Faster execution using Catalyst Optimizer.
Ease of Use	Requires functional transformations like `map`, `filter`.	Supports SQL-like queries for simpler data handling
Memory Management	High memory usage, no advanced optimization.	Uses Tungsten for efficient memory usage.
Interoperability	Limited support for external data sources.	Works with multiple formats like CSV, JSON, Parquet, and Hive.
Error Handling	More control over error handling.	Handles errors efficiently but with less granularity than RDDs.

When to Use RDD vs DataFrame?

Use RDD when:

- You need **low-level transformations** and fine-grained control over data.
- Your data is **unstructured** or **semi-structured** (text files, logs).
- You prefer **strict type safety** during operations.
- Performance is **not a major concern** but control over execution is.

Use Dataframe when:

- You work with **structured or semi-structured data** that fits into rows and columns.
- You need **high performance** and efficient memory usage.
- You want to use **SQL-like queries** for data analysis.
- You work with **multiple data sources** like Hive, JSON, Parquet, and CSV.

CONCLUSION:

- For most applications, **DataFrames are recommended** due to their **performance, optimization, and ease of use**. However, if you require **low-level control** and work with **raw data**, RDDs may still be useful.

THANK YOU