**Apache Spark🡪**

1. Open-source engine for data processing on computer clusters
2. Standard tool among the developers and data scientist
3. Supports variety of language like Python, Java, Scala, R
4. Includes libraries for variety of tasks such as ETL, streaming, ML, graph processing and analytics.
5. Can run on a laptop as well as cluster of 1000s of servers.
6. Spark can be deployed on Mesos, Hadoop via YARN, or its own cluster manager
7. Framework based component that can process all types of data like unstructured, semi structured and structured data
8. Alternative to Hadoop and Map-Reduce architecture.
9. Resilient Distributed Dataset (RDD) is Spark’s fundamental data structure/storage used for storage
10. Directed Acyclic Graph (DAG) is Spark’s processing framework which is used for processing the data
11. Spark architecture has 4 main components-
12. Spark Driver
13. Executors
14. Cluster Administrators
15. Worker Nodes
16. Spark uses Dataset and Dataframes as the fundamental data storage mechanism.

**Features of Spark🡪**

1. Spark is used to apply transformation in Big Data
2. Spark can be used efficiently on Big Data
3. Spark is used when we want to accelerate the calculation
4. Spark can perform parallel processing hence it is Very Fast and the speed is 100 times faster than the Hadoop Map Reduce
5. It is also able to divide the data into chunks in a controlled way
6. Powerful caching and disk persistence capabilities are offered by a simple programming layer

Caching is like having a quick-access memory that stores frequently used data. Caching makes data processing faster because the system can quickly retrieve the data from the cache.

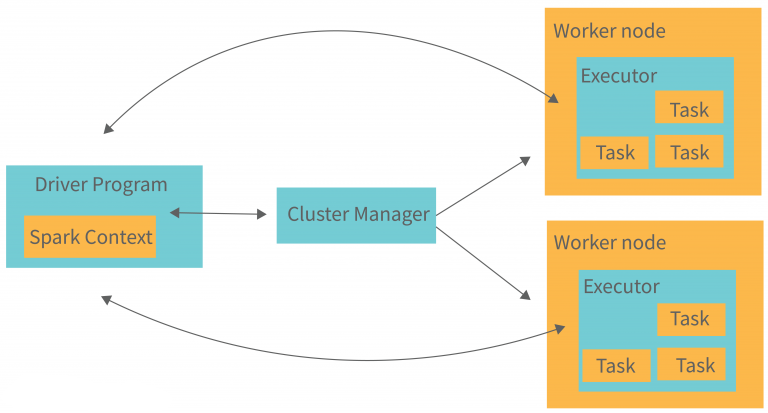
Disk persistence means that the system can save data to a more permanent storage location like HDD or SSD. This ensures that the data is not lost.

1. Deployment is easy via Mesos, Hadoop via YARN, Spark’s own cluster manager
2. Because of In-Memory processing, it offers real time computation and low latency
3. In-Memory processing means that Spark stores and manipulates data in computer memory (RAM).
4. Real-time computation means that Spark can process data as it arrives, almost instantly. Spark keeps data in memory and continuously analyses and processes it in real-time.
5. Low latency means that Spark can quickly respond to data inputs without significant delays.
6. Polyglot meaning ability to support multiple programming languages like Python, Java, R, Scala.
7. It provides a command line interface in Scala(spark-shell) and Python(pyspark)

**Spark Vs Map Reduce🡪**

1. Spark is faster as it processes data in RAM (memory) while Hadoop reads and writes files to HDFS (on disk)
2. Spark is easier to use than Map Reduce because of the support of multiple programming languages.
3. Spark is well-suited for iterative algorithms, such as machine learning algorithms, where data needs to be processed multiple times.
4. Spark is more fault tolerant because of its ability to recompute data using Lineage information.
5. Spark is optimized for better parallelism , CPU utilization , and faster startup
6. Spark offers a broader range of data processing models, including batch processing, real-time stream processing, interactive queries, and machine learning. MapReduce is primarily designed for batch processing.
7. Spark can run on various cluster managers like Apache Mesos, Hadoop YARN, and Kubernetes, making it easy to integrate into existing big data ecosystems. MapReduce typically runs on the Hadoop Distributed File System (HDFS).

**Spark Architecture🡪**



Has 4 components-

1. Driver - In Apache Spark architecture, the driver plays a central and critical role. The driver is the main control program responsible for managing the entire Spark application and coordinating the execution of tasks across the Spark cluster.

Tasks performed by Driver are 🡪

1. Driver program is a process that runs the main function of the application
2. Driver creates a SparkContext object.
3. SparkContext coordinates the spark applications which run as independent sets of processes on a cluster
4. In order to run on a cluster, SparkContext connects to different type of cluster managers
5. Once connected, it(SparkContext) performs following tasks-
   1. Acquires Executors on the nodes in the cluster.
   2. Sends application code to the Executors.
   3. Sends tasks to the executors to run.

Role of Driver🡪

1. Job Submissions – Driver is responsible for receiving the application code and dividing it into smaller chunks known as Jobs(Sequence of transformations and actions in the application).
2. Task Scheduling – Driver breaks down Jobs into smaller units called Tasks which represent individual data processing operations
3. Resource Management – Driver interacts with the cluster manager to request resources
4. Distributed Dataset Management – Driver keeps track of the how RDDs(Resilient Distributed Datasets) are derived and RDDs represents the distributed data across the cluster
5. Task Monitoring and Fault Tolerance - During task execution, the driver monitors the progress of each task and collects status updates from the worker nodes. If any task fails due to node failure or other issues, the driver uses the lineage information to recompute the lost data and reschedules the failed tasks on other available nodes, ensuring fault tolerance.
6. Result Collection – Driver collects information from all the worker nodes, aggregates them to present it to the user.
7. SparkContext - The driver creates and maintains the SparkContext, which serves as the entry point to Spark functionality. SparkContext allows the driver to communicate with the cluster manager and coordinate task execution across the nodes.
8. Cluster Manager –

Spark consists of various types of cluster manager like Apache Mesos, Hadoop via YARN, Standalone Scheduler.

Resource Allocation - Role of cluster manager is to allocate resources across applications.

Node Management - Cluster Manager keeps track of the available nodes in the cluster.

Task Scheduling - The cluster manager schedules Spark application tasks on the available worker nodes in the cluster.

It ensures that tasks are evenly distributed across nodes and maximizes the parallel processing capabilities of Spark.

Fault Tolerance – When a node fails, cluster manager is responsible for detecting failure and take actions by assigning tasks to other healthy nodes

Task Monitoring – Cluster manager monitors the tasks running on each worker node.

1. Worker Nodes -

Worker node is a slave node

Individual machines in the Spark cluster

Worker Nodes are individual machines or instances that provide the computing resources for executing Spark tasks

Role of worker node is to run the application code in the cluster

**Data Storage**: Worker nodes store the data partitions of the Resilient Distributed Datasets (RDDs) that are processed by Spark. RDDs are distributed across the worker nodes, and each node holds a subset of the overall data.

**Data Caching**: Spark supports data caching, where intermediate results or frequently accessed data can be cached in memory on worker nodes.

**Task Monitoring and Reporting**: Worker nodes continuously monitor the progress of the tasks they are executing and status updates to the cluster manager.

1. Executors –

Computational unit running on the worker nodes

Actual processes responsible for task execution

Executor is a process launched in the application on a worker node

It is responsible for running tasks parallely and keep data in memory or disk storage across them

Executors manage the data partitions of Resilient Distributed Datasets (RDDs) and store intermediate results in memory or on disk

Responsible for reading and writing to the external source

Every application contains an executor

Executors perform the actual data processing tasks on the distributed data

**Spark Components 🡪**

1. The Spark project consists of different types of tightly integrated components.
2. Apache Spark is an open-source, distributed computing system designed for big data processing and analytics.
3. It provides a unified data processing engine that supports batch processing, real-time streaming, machine learning, and graph processing

They are :

**Spark Core –**

It is the heart of the Spark architecture.

Provides the fundamental functionality like distributed task scheduling, memory management, fault recovery and interaction with storage systems.

It includes RDDs, data structure which is a collection of objects that can be processed in parallel.

**Spark SQL –**

Spark SQL allows users to run SQL-like queries on structured data including data in RDDs, or external data sources like Apache Hive, Apache HBase, Apache Parquet.

It provides DataFrame and Dataset APIs that offer a more user-friendly and optimized way of manipulating structured data.

It also supports various sources of data like Hive tables, Parquet, and JSON

It supports JDBC and ODBC connections.

**Spark Streaming –**

Supports scalability and fault tolerance

Its design ensures that the applications written for streaming data can be reused to analyze batches.

This component enables the processing of real-time data streams.

Spark Streaming divides incoming data streams into small batches, which are then processed using Spark's parallel processing capabilities.

**Mlib –**

MLlib is a scalable machine learning library that provides algorithm and tools for ML tasks such as classification, regression, clustering, PCA, Hypothesis testing.

It is built on top of Spark and can process large datasets efficiently.

**GraphX –**

GraphX is a graph processing library built on top of Spark.

Provides tools for manipulating graphs and performing graph computations.

Performs graph parallel computations.