Big Data Viva
What is HDFS

Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application.

It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.

Where it use

- **Very Large Files:** Files should be of hundreds of megabytes, gigabytes or more.
- **Streaming Data Access:** The time to read whole data set is more important than latency in reading the first. HDFS is built on write-once and read-many-times pattern.
- **Commodity Hardware:**It works on low cost hardware.

Features of HDFS

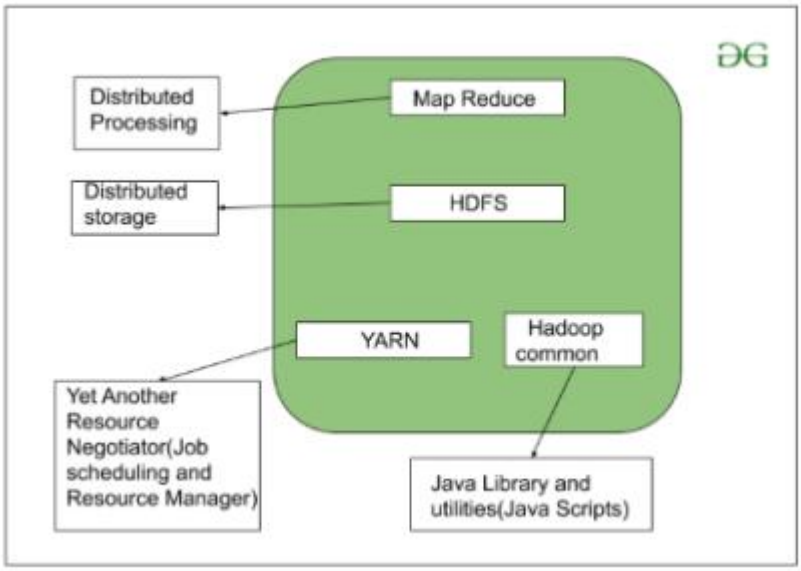
Features:

- *Distributed data storage.*
- *Blocks reduce seek time.*
- *The data is highly available as the same block is present at multiple datanodes.*
- *Even if multiple datanodes are down we can still do our work, thus making it highly reliable.*
- *High fault tolerance.*

HDFC Architecture

As we all know Hadoop is a framework written in Java that utilizes a large cluster of commodity hardware to maintain and store big size data. Hadoop works on MapReduce Programming Algorithm that was introduced by Google. Today lots of Big Brand Companies are using Hadoop in their Organization to deal with big data, eg. Facebook, Yahoo, Netflix, eBay, etc. The Hadoop Architecture Mainly consists of 4 components.

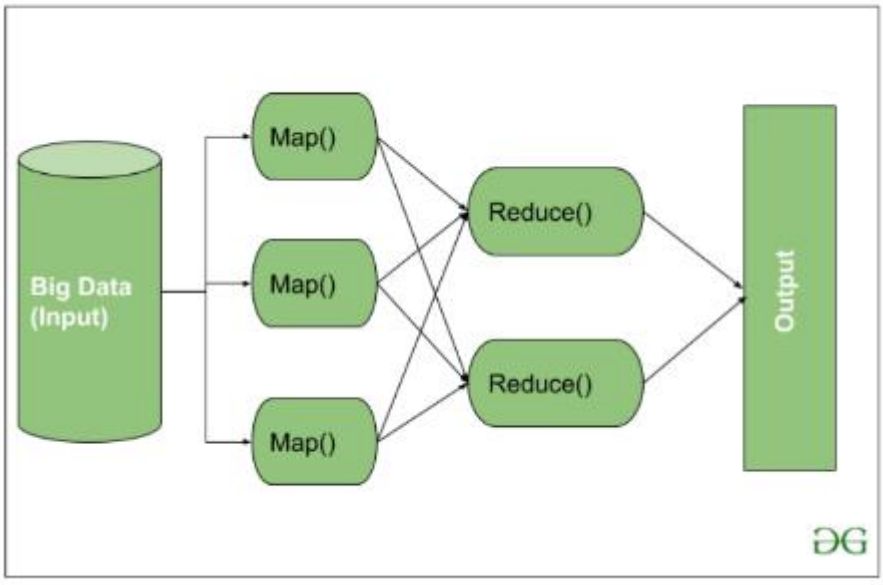
- MapReduce
- HDFS(Hadoop Distributed File System)
- YARN(Yet Another Resource Negotiator)
- Common Utilities or Hadoop Common



Question: What is the purpose of the `hdfs dfs` command?
Explanation: `hdfs dfs` is a command-line interface for interacting with HDFS. It is used to perform various file and directory operations on Hadoop Distributed File System.
Question: What does the `hdfs dfs -touchz` command do?
Explanation: `hdfs dfs -touchz <path>` is used to create an empty file at the specified HDFS path. It's similar to the Unix `touch` command.
Question: How is the `hdfs dfs -put` command used?
Explanation: `hdfs dfs -put <local-src> <hdfs-dest>` is used to copy a file or a directory from the local file system to HDFS.
Question: What is the purpose of the `hdfs dfs -test` command?
Explanation: `hdfs dfs -test` is used to perform tests on files or directories in HDFS. It allows checking for various conditions such as existence, non-existence, directory, file, etc.
Question: How can you create a directory in HDFS using the `hdfs dfs -mkdir` command?
Explanation: `hdfs dfs -mkdir <hdfs-dir>` is used to create a directory in HDFS.
Question: What does the `hdfs dfs -appendToFile` command do?
Explanation: `hdfs dfs -appendToFile <local-src> <hdfs-dest>` is used to append the contents of a local file to a file in HDFS.
Question: How is the `hdfs dfs -count` command used?

Explanation: `hdfs dfs -count <hdfs-path>` provides a summary of the number of directories, files, and total disk space used by the specified HDFS path.
Question: How can you get help on HDFS commands?
Explanation: `hdfs dfs -help` provides a list of available HDFS commands along with brief descriptions. It's a useful command for getting information on various options.
Question: What does the `hdfs dfs -expunge` command do?
Explanation: `hdfs dfs -expunge` is used to permanently delete all the files from the trash in HDFS. It helps in freeing up storage space.
What is Mapreduce

MapReduce nothing but just like an Algorithm or a data structure that is based on the YARN framework. The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use. MapReduce has mainly 2 tasks which are divided phase-wise:



As we can see that an Input is provided to the Map(), now as we are using Big Data. The Input is a set of Data. The Map() function here breaks this DataBlocks into **Tuples** that are nothing but a key-value pair. These key-value pairs are now sent as input to the Reduce(). The Reduce() function then combines this broken Tuples or key-value pair based on its Key

value and form set of Tuples, and perform some operation like sorting, summation type job, etc. which is then sent to the final Output Node. Finally, the Output is Obtained.

The data processing is always done in Reducer depending upon the business requirement of that industry. This is How First Map() and then Reduce is utilized one by one.

Reduce Task

- **Shuffle and Sort:** The Task of Reducer starts with this step, the process in which the Mapper generates the intermediate key-value and transfers them to the Reducer task is known as *Shuffling*. Using the Shuffling process the system can sort the data using its key value.

Once some of the Mapping tasks are done Shuffling begins that is why it is a faster process and does not wait for the completion of the task performed by Mapper.

-
- **Reduce:** The main function or task of the Reduce is to gather the Tuple generated from Map and then perform some sorting and aggregation sort of process on those key-value depending on its key element.
- **OutputFormat:** Once all the operations are performed, the key-value pairs are written into the file with the help of record writer, each record in a new line, and the key and value in a space-separated manner.

HDFS

HDFS(Hadoop Distributed File System) is utilized for storage permission. It is mainly designed for working on commodity Hardware devices(inexpensive devices), working on a distributed file system design. HDFS is designed in such a way that it

believes more in storing the data in a large chunk of blocks rather than storing small data blocks.

HDFS in Hadoop provides Fault-tolerance and High availability to the storage layer and the other devices present in that Hadoop cluster. Data storage Nodes in HDFS.

- NameNode(Master)
- DataNode(Slave)

YARN(Yet Another Resource Negotiator)

YARN is a Framework on which MapReduce works. YARN performs 2 operations that are Job scheduling and Resource Management. The Purpose of Job schedular is to divide a big task into small jobs so that each job can be assigned to various slaves in a Hadoop cluster and Processing can be Maximized. Job Scheduler also keeps track of which job is important, which job has more priority, dependencies between the jobs and all the other information like job timing, etc. And the use of Resource Manager is to manage all the resources that are made available for running a Hadoop cluster.

Features of YARN

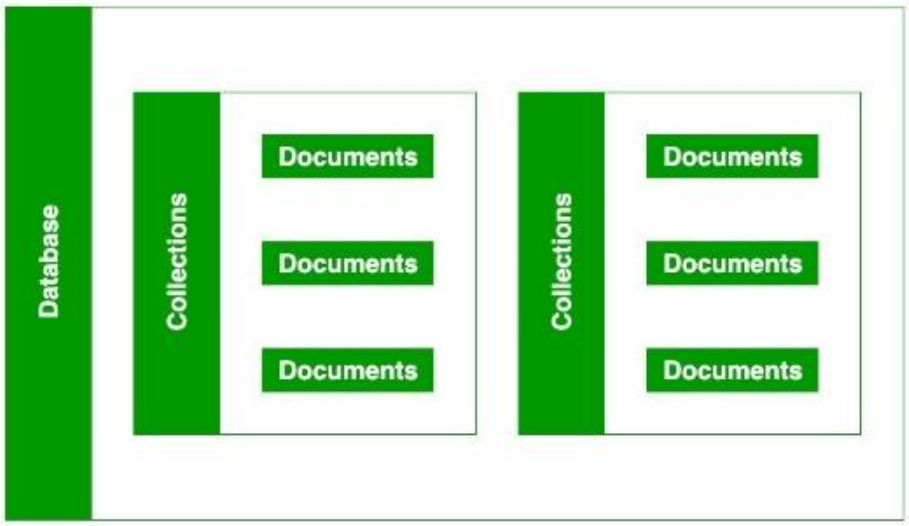
- Multi-Tenancy
- Scalability
- Cluster-Utilization
- Compatibility

What is MongoDB

MongoDB is an open-source document-oriented database that is designed to store a large scale of data and also allows you to work with that data very efficiently. It is categorized under the NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables.

The MongoDB database is developed and managed by MongoDB.Inc under SSPL(Server Side Public License) and initially released in February 2009. It also provides official driver support for all the popular languages like C, C++, C#, and .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. So, that you can create an application using any of these languages. Nowadays there are so many companies that used MongoDB like Facebook, Nokia, eBay, Adobe, Google, etc. to store their large amount of data.

How it work



- The MongoDB database contains collections just like the MYSQL database contains tables. You are allowed to create multiple databases and multiple collections.

- Now inside of the collection we have documents. These documents contain the data we want to store in the MongoDB database and a single collection can contain multiple documents and you are schema-less means it is not necessary that one document is similar to another.
- The documents are created using the fields. Fields are key-value pairs in the documents, it is just like columns in the relation database. The value of the fields can be of any BSON data types like double, string, boolean, etc.
- The data stored in the MongoDB is in the format of BSON documents. Here, BSON stands for Binary representation of JSON documents. Or in other words, in the backend, the MongoDB server converts the JSON data into a binary form that is known as BSON and this BSON is stored and queried more efficiently.
- In MongoDB documents, you are allowed to store nested data. This nesting of data allows you to create complex relations between data and store them in the same document which makes the working and fetching of data extremely efficient as compared to SQL. In SQL, you need to write complex joins to get the data from table 1 and table 2. The maximum size of the BSON document is 16MB.

Features of MongoDB

- **Schema-less Database:** It is the great feature provided by the MongoDB. A Schema-less database means one collection can hold different types of documents in it. Or in other words, in the MongoDB database, a single collection can hold multiple documents and these documents may consist of the different numbers of fields, content, and size. It is not necessary that the one document is similar to another

document like in the relational databases. Due to this cool feature, MongoDB provides great flexibility to databases.

- **Document Oriented**: In MongoDB, all the data stored in the documents instead of tables like in RDBMS. In these documents, the data is stored in fields(key-value pair) instead of rows and columns which make the data much more flexible in comparison to RDBMS. And each document contains its unique object id.
- **Indexing**: In MongoDB database, every field in the documents is indexed with primary and secondary indices this makes easier and takes less time to get or search data from the pool of the data. If the data is not indexed, then database search each document with the specified query which takes lots of time and not so efficient.
- **Scalability**: MongoDB provides horizontal scalability with the help of sharding. Sharding means to distribute data on multiple servers, here a large amount of data is partitioned into data chunks using the shard key, and these data chunks are evenly distributed across shards that reside across many physical servers. It will also add new machines to a running database.
- **Replication**: MongoDB provides high availability and redundancy with the help of replication, it creates multiple copies of the data and sends these copies to a different server so that if one server fails, then the data is retrieved from another server.
- **Aggregation**: It allows to perform operations on the grouped data and get a single result or computed result. It is similar to the SQL GROUPBY clause. It provides three different aggregations i.e, aggregation pipeline, map-reduce function, and single-purpose aggregation methods

- **High Performance:** The performance of MongoDB is very high and data persistence as compared to another database due to its features like scalability, indexing, replication, etc.

MongoDB	RDBMS
It is a non-relational and document-oriented database.	It is a relational database.
It is suitable for hierarchical data storage.	It is not suitable for hierarchical data storage.
It has a dynamic schema.	It has a predefined schema.
It centers around the CAP theorem (Consistency, Availability, and Partition tolerance).	It centers around ACID properties (Atomicity, Consistency, Isolation, and Durability).
In terms of performance, it is much faster than RDBMS.	In terms of performance, it is slower than MongoDB.

Advantages of MongoDB :

- It does not support join operation.
- It provides great flexibility to the fields in the documents.
- It contains heterogeneous data.
- It provides high performance, availability, scalability.
- It supports Geospatial efficiently.

Disadvantages of MongoDB :

- It uses high memory for data storage.
- You are not allowed to store more than 16MB data in the documents.
- The nesting of data in BSON is also limited you are not allowed to nest data more than 100 levels.

What is HIVE

Hive is a data warehouse system which is used to analyze structured data. It is built on the top of Hadoop. It was developed by Facebook.

Hive provides the functionality of reading, writing, and managing large datasets residing in distributed storage. It runs SQL like queries called HQL (Hive query language) which gets internally converted to MapReduce jobs.

Using Hive, we can skip the requirement of the traditional approach of writing complex MapReduce programs. Hive supports Data Definition Language (DDL), Data Manipulation Language (DML), and User Defined Functions (UDF).

Features of Hive

- Hive is fast and scalable.
- It provides SQL-like queries (i.e., HQL) that are implicitly transformed to MapReduce or Spark jobs.
- It is capable of analyzing large datasets stored in HDFS.
- It allows different storage types such as plain text, RCFile, and HBase.
- It uses indexing to accelerate queries.
- It can operate on compressed data stored in the Hadoop ecosystem.
- It supports user-defined functions (UDFs) where user can provide its functionality.

Limitations of Hive

- Hive is not capable of handling real-time data.
- It is not designed for online transaction processing.
- Hive queries contain high latency.

Basic of Pig

Pig is a high-level data flow platform for executing Map Reduce programs of Hadoop. It was developed by Yahoo. The language for Pig is pig Latin.

Our Pig tutorial includes all topics of Apache Pig with Pig usage, Pig Installation, Pig Run Modes, Pig Latin concepts, Pig Data Types, Pig example, Pig user defined functions etc.

What is Apache Pig

Apache Pig is a high-level data flow platform for executing MapReduce programs of Hadoop. The language used for Pig is Pig Latin.

The Pig scripts get internally converted to Map Reduce jobs and get executed on data stored in HDFS. Apart from that, Pig can also execute its job in Apache Tez or Apache Spark.

Pig can handle any type of data, i.e., structured, semi-structured or unstructured and stores the corresponding results into Hadoop Data File System. Every task which can be achieved using PIG can also be achieved using java used in MapReduce.

Features of Apache Pig

Let's see the various uses of Pig technology.

1) Ease of programming

Writing complex java programs for map reduce is quite tough for non-programmers. Pig makes this process easy. In the Pig, the queries are converted to MapReduce internally.

2) Optimization opportunities

It is how tasks are encoded permits the system to optimize their execution automatically, allowing the user to focus on semantics rather than efficiency.

3) Extensibility

A user-defined function is written in which the user can write their logic to execute over the data set.

4) Flexible

It can easily handle structured as well as unstructured data.

5) In-built operators

It contains various type of operators such as sort, filter and joins.

Advantages of Apache Pig

- Less code - The Pig consumes less line of code to perform any operation.
- Reusability - The Pig code is flexible enough to reuse again.
- Nested data types - The Pig provides a useful concept of nested data types like tuple, bag, and map.

Differences between Apache MapReduce and PIG

Apache MapReduce	Apache PIG
It is a low-level data processing tool.	It is a high-level data flow tool.
Here, it is required to develop complex programs using Java or Python.	It is not required to develop complex programs.
It is difficult to perform data operations in MapReduce.	It provides built-in operators to perform data operations like union, sorting and ordering.
It doesn't allow nested data types.	It provides nested data types like tuple, bag, and map.

Differences between Hive and Pig

Hive	Pig
Hive is commonly used by Data Analysts.	Pig is commonly used by programmers.
It follows SQL-like queries.	It follows the data-flow language.
It can handle structured data.	It can handle semi-structured data.
It works on server-side of HDFS cluster.	It works on client-side of HDFS cluster.
Hive is slower than Pig.	Pig is comparatively faster than Hive.

Partitioning in Hive

The partitioning in Hive means dividing the table into some parts based on the values of a particular column like date, course, city or country. The

advantage of partitioning is that since the data is stored in slices, the query response time becomes faster.

As we know that Hadoop is used to handle the huge amount of data, it is always required to use the best approach to deal with it. The partitioning in Hive is the best example of it.

Let's assume we have a data of 10 million students studying in an institute. Now, we have to fetch the students of a particular course. If we use a traditional approach, we have to go through the entire data. This leads to performance degradation. In such a case, we can adopt the better approach i.e., partitioning in Hive and divide the data among the different datasets based on particular columns.

The partitioning in Hive can be executed in two ways -

- Static partitioning
- Dynamic partitioning

Static Partitioning

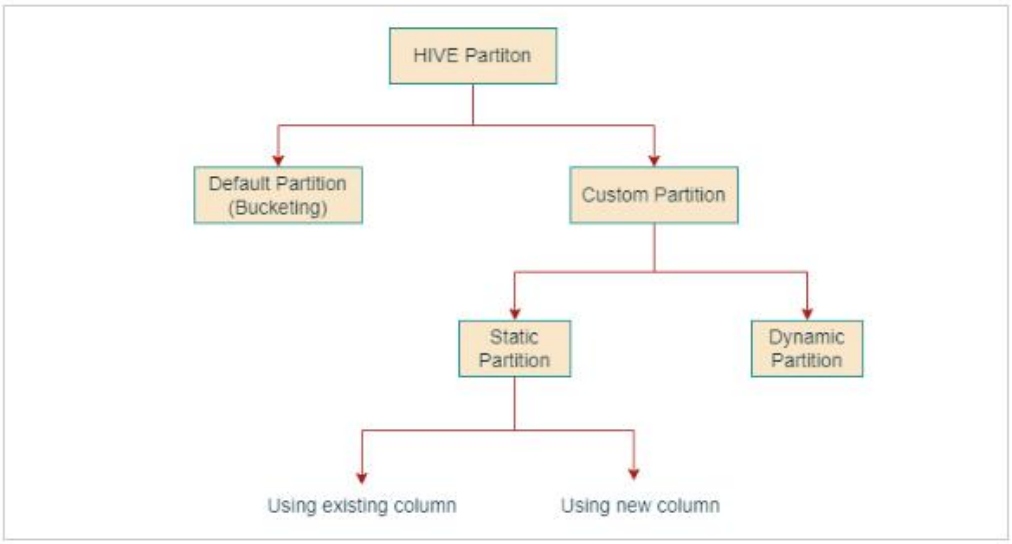
In static or manual partitioning, it is required to pass the values of partitioned columns manually while loading the data into the table. Hence, the data file doesn't contain the partitioned columns.

Dynamic Partitioning

In dynamic partitioning, the values of partitioned columns exist within the table. So, it is not required to pass the values of partitioned columns manually.

Partitioning in Hive

Partitioning is the process of dividing a large dataset into smaller and more manageable parts. There are types of partitioning in Hive: Static and dynamic partitioning.



Different Join operation in Pig

The **JOIN** operator is used to combine records from two or more relations. While performing a join operation, we declare one (or a group of) tuple(s) from each relation, as keys. When these keys match, the two particular tuples are matched, else the records are dropped. Joins can be of the following types –

- Self-join
- Inner-join
- Outer-join – left join, right join, and full join

Self – join

Self-join is used to join a table with itself as if the table were two relations, temporarily renaming at least one relation.

Generally, in Apache Pig, to perform self-join, we will load the same data multiple times, under different aliases (names). Therefore let us load the contents of the file **customers.txt** as two tables

Syntax

```
grunt> Relation3_name = JOIN Relation1_name BY key, Relation2_name BY key ;
```

Inner Join

Inner Join is used quite frequently; it is also referred to as **equijoin**. An inner join returns rows when there is a match in both tables.

It creates a new relation by combining column values of two relations (say A and B) based upon the join-predicate. The query compares each row of A with each row of B to find all pairs of rows which satisfy the join-predicate. When the join-predicate is satisfied, the column values for each matched pair of rows of A and B are combined into a result row.

Syntax

```
grunt> result = JOIN relation1 BY columnname, relation2 BY columnname;
```

Outer Join: Unlike inner join, **outer join** returns all the rows from at least one of the relations. An outer join operation is carried out in three ways −

- Left outer join
- Right outer join
- Full outer join

Left Outer Join

The **left outer Join** operation returns all rows from the left table, even if there are no matches in the right relation.

Syntax

```
grunt> Relation3_name = JOIN Relation1_name BY id LEFT OUTER,
Relation2_name BY customer_id;
```

Right Outer Join

The **right outer join** operation returns all rows from the right table, even if there are no matches in the left table.

Syntax

```
grunt> outer_right = JOIN customers BY id RIGHT, orders BY customer_id;
```

Full Outer Join

The **full outer join** operation returns rows when there is a match in one of the relations.

Syntax

```
grunt> outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;
```

What is Apache Spark

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools

Features of Apache Spark

Apache Spark has following features.

-

Speed – Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.

-
-

Supports multiple languages – Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.

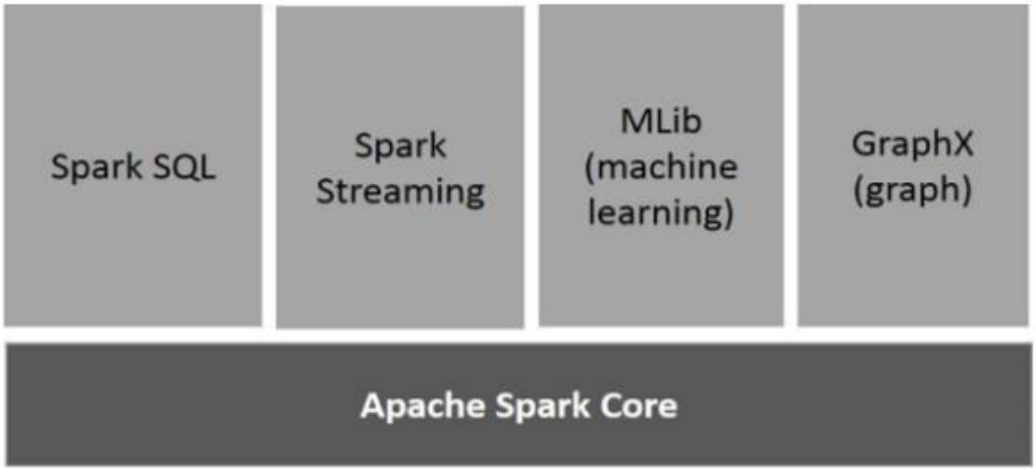
-
-

Advanced Analytics – Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

-

Components of Spark

The following illustration depicts the different components of Spark.



Spark Architecture

The Spark follows the master-slave architecture. Its cluster consists of a single master and multiple slaves.

The Spark architecture depends upon two abstractions:

- Resilient Distributed Dataset (RDD)
- Directed Acyclic Graph (DAG)

Resilient Distributed Datasets (RDD)

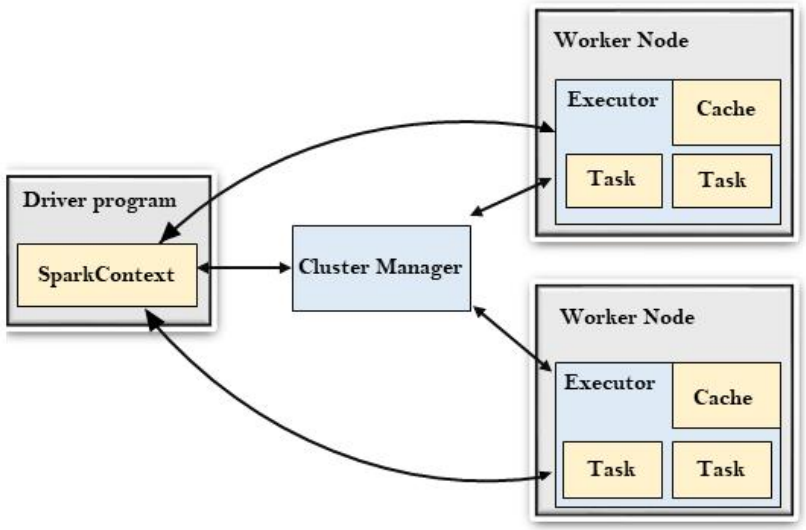
The Resilient Distributed Datasets are the group of data items that can be stored in-memory on worker nodes. Here,

- Resilient: Restore the data on failure.
- Distributed: Data is distributed among different nodes.
- Dataset: Group of data.

Directed Acyclic Graph (DAG)

Directed Acyclic Graph is a finite direct graph that performs a sequence of computations on data. Each node is an RDD partition, and the edge is a transformation on top of data. Here, the graph refers the navigation whereas directed and acyclic refers to how it is done.

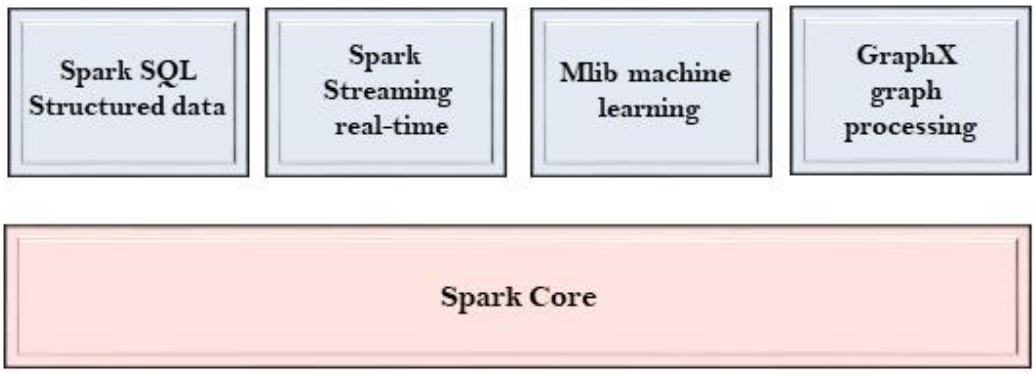
Let's understand the Spark architecture.



Spark Components

The Spark project consists of different types of tightly integrated components. At its core, Spark is a computational engine that can schedule, distribute and monitor multiple applications.

Let's understand each Spark component in detail.



Spark Core

- The Spark Core is the heart of Spark and performs the core functionality.
- It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.

Spark SQL

- The Spark SQL is built on the top of Spark Core. It provides support for structured data.
- It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL?called the HQL (Hive Query Language).
- It supports JDBC and ODBC connections that establish a relation between Java objects and existing databases, data warehouses and business intelligence tools.
- It also supports various sources of data like Hive tables, Parquet, and JSON.

Spark Streaming

- Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data.
- It uses Spark Core's fast scheduling capability to perform streaming analytics.

- It accepts data in mini-batches and performs RDD transformations on that data.
- Its design ensures that the applications written for streaming data can be reused to analyze batches of historical data with little modification.
- The log files generated by web servers can be considered as a real-time example of a data stream.

Resilient Distributed Dataset (RDD)

An RDD is a collection of fault-tolerant elements that can be distributed and processed in parallel across multiple nodes within a cluster. RDDs are the basic architecture of Apache Spark.

Spark loads the data either by referencing the data source or using SparkContext parallelization to process the existing collection in an RDD. When you load data into the RDD, Spark will perform transformations and operations on the in-memory RDD. This is the essence of Spark's speed. Spark stores data in memory by default, though the user can choose to write data to disk to ensure persistence, because the system might run out of memory.

Each data set in an RDD is partitioned into logical components and can run on different nodes in the cluster. Users can also perform two RDD activities: transformations and jobs. A transform is an operation

that creates a new RDD. Jobs tell Apache Spark to apply computations and return results to the driver.

Spark supports various operations and transformations in RDD. Spark handles this deployment, so users don't have to calculate the correct deployment.

Directed Acyclic Graph (DAG)

Unlike MapReduce's two-step execution process, Spark creates a directed acyclic graph (DAG) to schedule jobs and orchestrate worker nodes across a cluster. As Spark processes and transforms data during job execution, the DAG scheduler adjusts the cluster's worker nodes to increase efficiency. This job tracking enables fault tolerance by re-applying logged actions to data in a previous state.

Apache Spark in the Cloud

Spark on AWS

Amazon supports Apache Spark as part of the Amazon Elastic Map/Reduce (EMR) service. EMR

makes it possible to quickly create a managed Spark cluster from the AWS Management Console, AWS CLI, or Amazon EMR API.

EMR features and integrated Amazon services

The managed cluster can benefit from additional Amazon EMR features, including:

- Fast connection to Amazon S3 using the EMR File System (EMRFS)
- Integration with the Amazon EC2 spot market to leverage low-cost spot instances
- AWS Glue Data Catalog
- EMR managed extensions

Spark on EMR can also benefit from AWS Lake Formation, which provides fine-grained access control for data lakes, and integration with AWS Step Functions to orchestrate data pipeline processes.

Spark development with EMR Studio and Zeppelin

Developers can leverage Amazon EMR Studio, an integrated development environment (IDE) for data scientists and data engineers. EMR Studio enables development, visualization, and debugging for

applications written in R, Python, Scala and PySpark.

EMR Studio provides fully managed Jupyter Notebooks and tools like the Spark UI and YARN timeline service to simplify debugging of Spark applications and make it easier to build applications with Spark. Alternatively, developers can use Apache Zeppelin to create interactive, collaborative notebooks for data exploration with Spark.