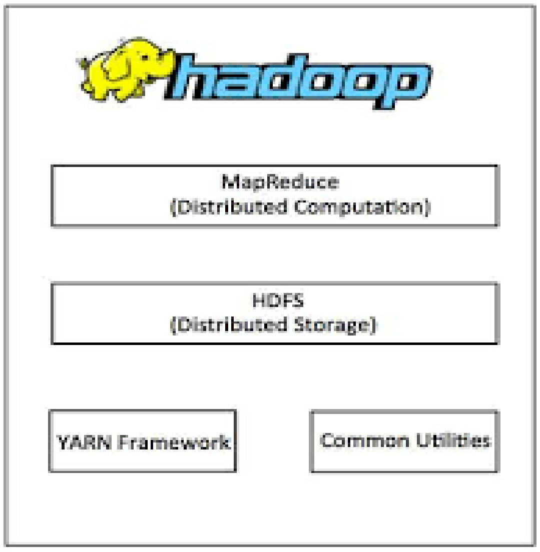
Introduction to Apache HADOOP

Big Data

- The main elements of Big Data are:
    - Volume - There is a massive amount of data generated every second.
    - Velocity - The speed at which data is generated, collected, and analyzed.
    - Variety - The different types of data: structured, semi-structured, unstructured.
    - Value - The ability to turn data into useful insights for your business.
    - Veracity - Trustworthiness in terms of quality and accuracy.

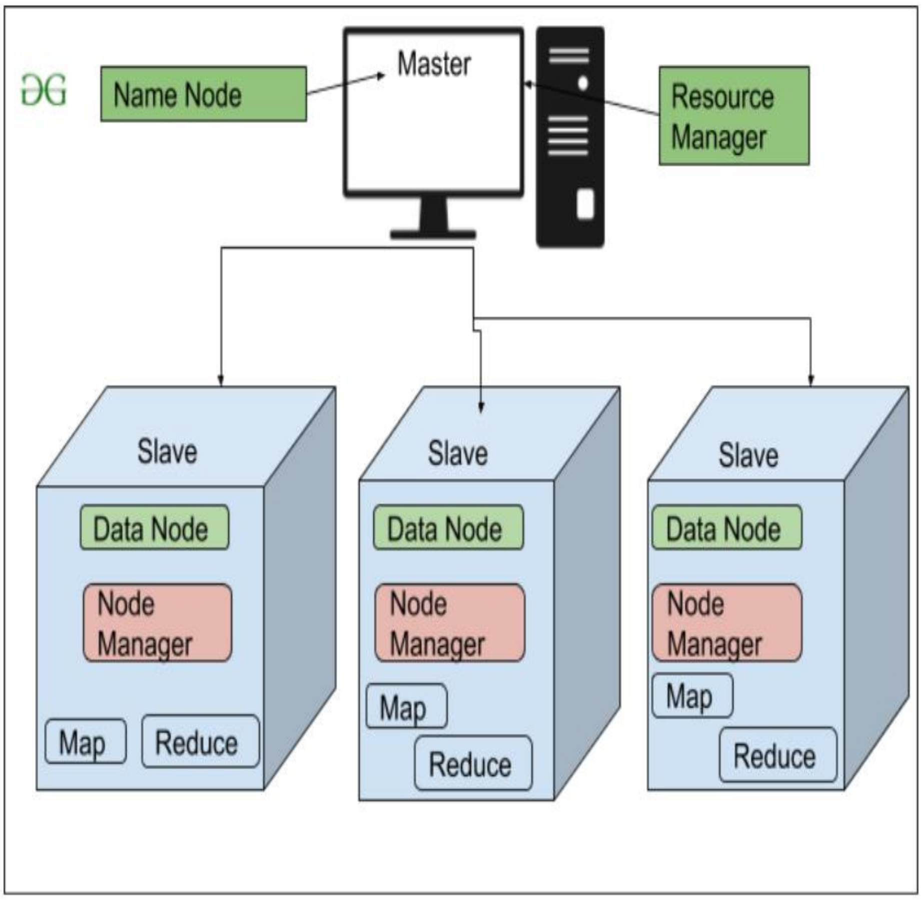
Hadoop Introduction

- Hadoop is Apache's open-source platform written in java which allows distributed computing of large datasets across clusters of commodity hardware using simple programming models.
- Hadoop architecture: It has mainly three components.
    - Hadoop distributed file system (HDFS) - storage layer.
    - Map-Reduce - Process layer.
    - Yarn - Resource management layer.



HDFS (Hadoop Distributed File Structure)

- This is the storage component of Hadoop, which allows for the storage of large amounts of data across multiple machines.
- It has master-slave architecture.
- The data files are divided into multiple blocks.
- HDFS has two daemons:
    - Master daemons
        - NameNode
        - Secondary NameNode
        - Job Tracker
    - Slave daemons
        - DataNode
        - Task Tracker



NameNode

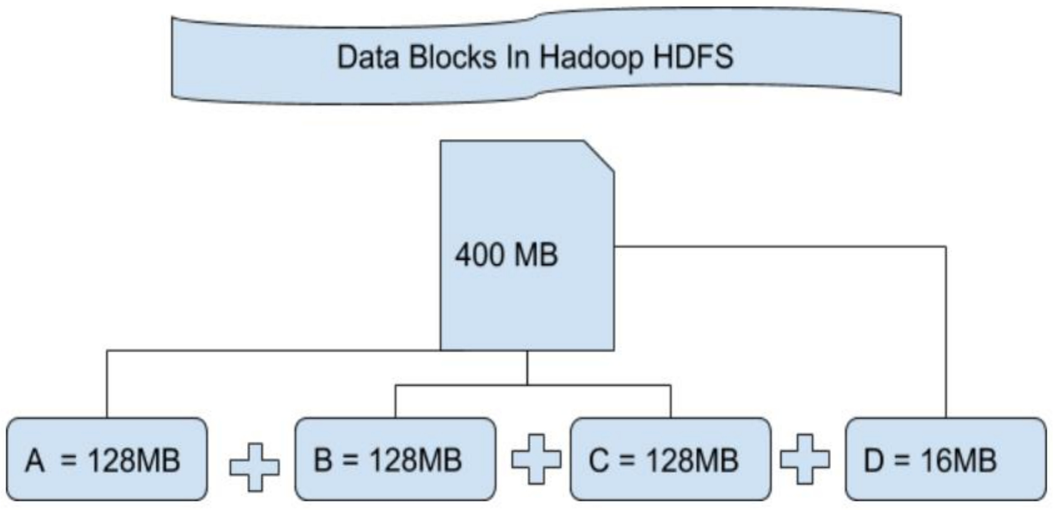
- NameNode works as a Master in a Hadoop cluster that guides the Datanode(Slaves).
- Namenode is mainly used for storing the Metadata i.e. the data about the data.
- Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication.
- Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

DataNode

- DataNodes works as a Slave DataNodes are mainly utilized for storing the data in a Hadoop cluster, the number of DataNodes can be from 1 to 500 or even more than that.
- The more number of DataNode, the Hadoop cluster will be able to store more data. So it is advised that the DataNode should have High storing capacity to store a large number of file blocks.

Note

- Data in HDFS is always stored in terms of blocks. So the single block of data is divided into multiple blocks of size 128MB which is default and we can also change it manually.



Replication in HDFS

- Replication ensures the availability of the data. Replication is making a copy of something and the number of times we make a copy of that particular thing can be expressed as it's Replication Factor. As we have seen in File blocks that the HDFS stores the data in the form of various blocks at the same time Hadoop is also configured to make a copy of those file blocks.
- By default, the Replication Factor for Hadoop is set to 3 which can be configured means we can change it manually as per our requirement like in above example we have made 4 file blocks which means that 3 Replica or copy of each file block is made means total of 4×3 = 12 blocks are made for the backup purpose.

Rack Awareness

- The rack awareness is a collection of 30-40 DataNodes connected using the same network switch.
- Rack awareness is the concept of selecting the closer DataNodes based on rack information.

Advantages of HDFS

- Fault tolerance
- Speed
- Compatibility and portability
- Scalable
- Data locality
- Cost effective
- Storage large amounts of data
- Flexible

Top HDFS Commands

ls	This command is used to list all the files.	hadoop fs -ls
mkdir	To create a directory.	hadoop fs -mkdir Folder1
touchz	It creates an empty file.	hadoop fs -touchz abc.txt
copyFromLocal (or) put	To copy files/folders from local file system to hdfs store.	hadoop fs -copyFromLocal abc.txt Folder1
cat	To print file contents.	hadoop fs -cat Folder1/abc.txt
copyToLocal (or) get	To copy files/folders from hdfs store to local file system.	hadoop fs copyToLocal Folder1/abc.txt -/Desktop/
cp	This command is used to copy files within hdfs.	hadoop fs -cp Folder1/abc.txt Folder2/
mv	This command is used to move files within hdfs.	hadoop fs -mv Folder1/abc.txt Folder2/
du (or) dus	It will give the size of each file in directory.	hadoop fs -du Folder2
moveFromLocal	Moves from local machine to hdfs directory.	hadoop fs -moveFromLocal a2.txt Folder2
getmerge	Merge content of two files into a single file.	hadoop fs -getmerge -nl destination/xyz.txt destination/xyz1.txt -/Desktop/mergeresult.txt
appendToFile	This command appends the contents of all the given local files to the provided destination file on the HDFS filesystem. All the content of q1 and q2 present in local file system will be added to the q3 which is present over hdfs.	hadoop fs -appendToFile q1.txt q2.txt q3.txt

checksum	Store hash value	hadoop fs -checksum destination/xyz1.txt
fsck	Used for status	hadoop -fsck - /
rm	Removes the file.	hadoop fs -rm destination/xyz.txt
stat	Status of file. %b - File Size in bytes will be printed %g - Group name of file will be printed %r - Replication factor will be printed %u - Username will be printed %y - When the file will modified last	hadoop fs -stat %b destination/xyz.txt hadoop fs -stat %g destination/xyz.txt hadoop fs -stat %r destination/xyz.txt hadoop fs -stat %u destination/xyz.txt hadoop fs -stat %y destination/xyz.txt
expunge	Deletes the trash files of hdfs.	hadoop fs -expunge
rmdir	Remove directory	Hadoop fs -rmdir /home/ab
setrep -R 2	To change replication factor	Hadoop fs -setrep -R 2 /home/user/

<u>**Note:**</u>

The process of map reduce execute (word count)

```
hadoop fs -mkdir /home/WordCount
```

```
hadoop fs -put "C:\hadoopsetup\ETP\Map-Reduce Programs\Word Count\WordCount.txt"
/home/WordCount/
```

```
hadoop jar "C:\hadoopsetup\ETP\Map-Reduce Programs\Word Count\WordCount2.jar"
/home/WordCount/Output/
```

- To print last few rows of file in Hadoop by hdfs command:

  ```
  hdfs dfs -tail /path/to/your/file
  hdfs dfs -tail -n 20 /path/to/your/file
  ```

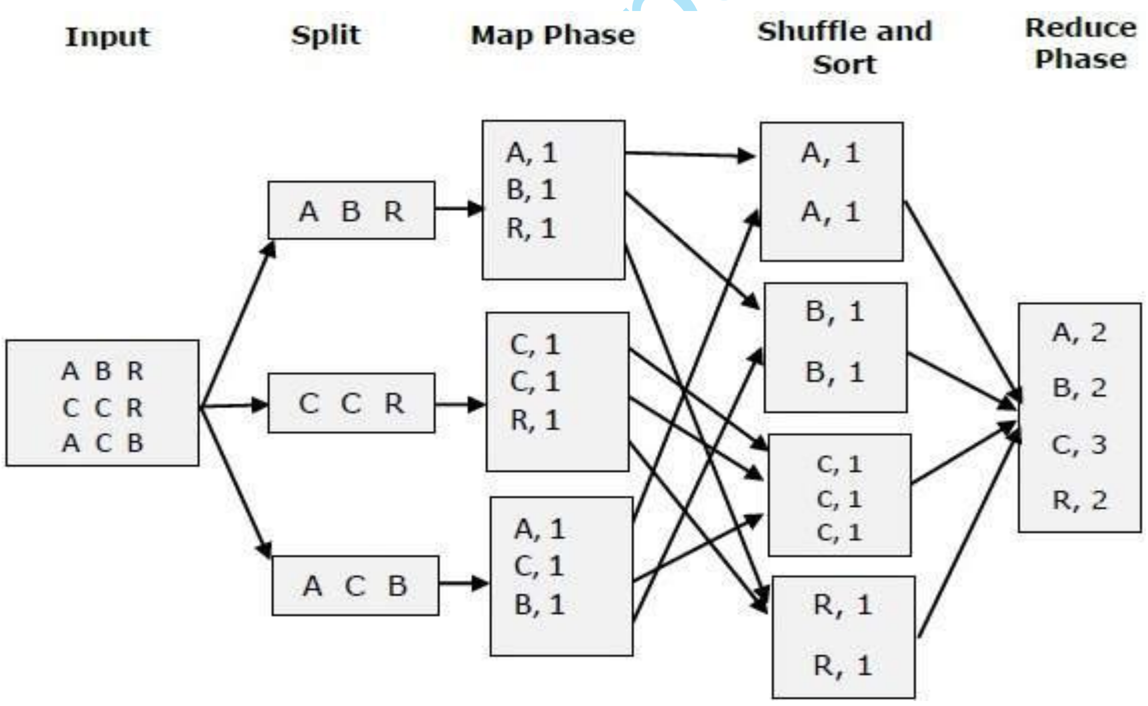
- To change the permission of file so that anyone can execute that in hadoop command:

  hdfs dfs -chmod o+x /path/to/your/file

- Permissions to read only
- hadoop fs -chmod 444 /user/abc/
- Permissions to read and write
- hadoop fs -chmod 777 /user/abc/

Map-Reduce in Hadoop

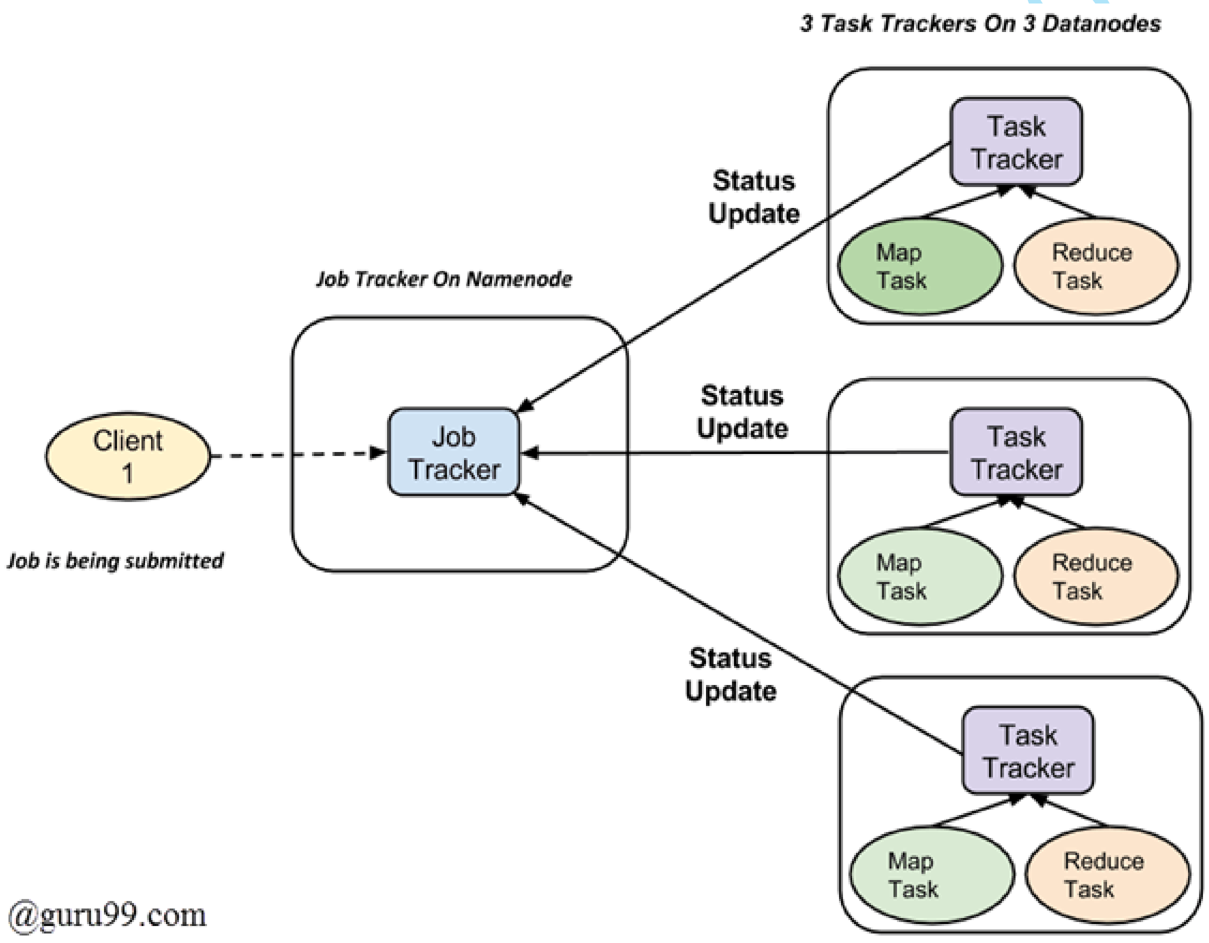
- Map-Reduce performs the processing of large data sets in distributed or parallel manner.
- Map-Reduce consists of two distinct tasks : Map and Reduce.
- Two essential daemons of Map-Reduce are : Job tracker and Task tracker.



Note:

- Hadoop divides the job into tasks. There are two types of tasks:
  - Map tasks (Splits & Mapping)
  - Reduce tasks (Shuffling, Reducing)

- The complete execution process (execution of Map and Reduce tasks, both) is controlled by two types of entities called as:
    - Jobtracker: Acts like a master (responsible for complete execution of submitted job).
    - Multiple Task Trackers: Acts like slaves, each of them performing the job.
- For every job submitted for execution in the system, there is one Jobtracker that resides on Namenode and there are multiple tasktrackers which reside on Datanode.

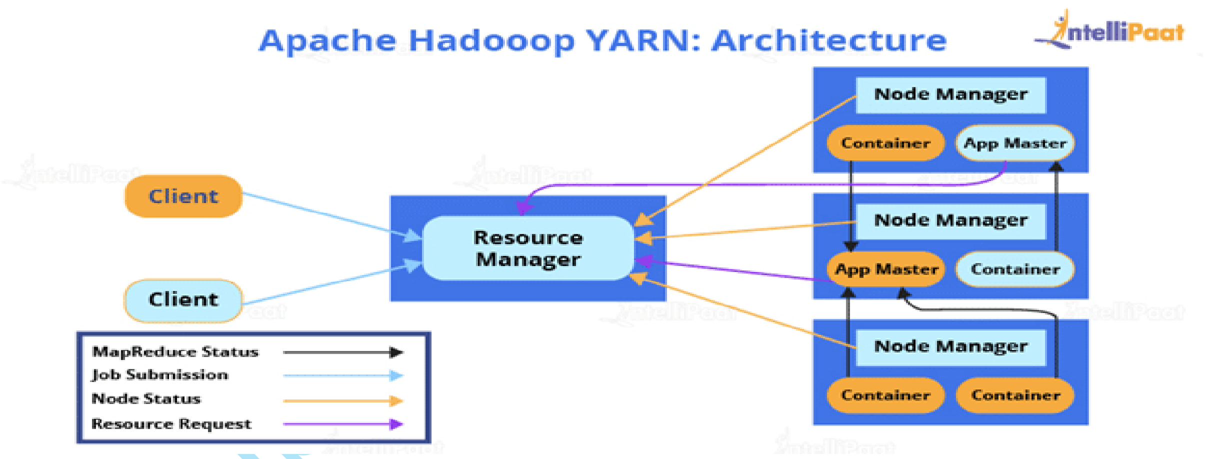


- A job is divided into multiple tasks which are then run onto multiple data nodes in a cluster.
- It is the responsibility of job tracker to coordinate the activity by scheduling tasks to run on different data nodes.
- Execution of individual task is then to look after by task tracker, which resides on every data node executing part of the job.
- Task tracker's responsibility is to send the progress report to the job tracker.

- In addition, task tracker periodically sends 'heartbeat' signal to the Jobtracker so as to notify him of the current state of the system.
- Thus job tracker keeps track of the overall progress of each job. In the event of task failure, the job tracker can reschedule it on a different task tracker.

Yarn in Hadoop

- Apache Hadoop YARN (Yet Another Resource Negotiator) is a resource management layer in Hadoop. YARN came into the picture with the introduction of Hadoop 2.x. It allows various data processing engines such as interactive processing, graph processing, batch processing, and stream processing to run and process data stored in HDFS (Hadoop Distributed File System).
- YARN framework contains a Resource Manager (master daemon), Node Manager (slave daemon), and an Application Master.



Resource Manager

- It is responsible for managing several other applications, along with the global assignments of resources such as CPU and memory. It is used for job scheduling. Resource Manager has two components:
    - Scheduler &
    - Application Manager

Node Manager

- Node Manager has to monitor the container's resource usage, along with reporting it to the Resource Manager.
- The health of the node on which YARN is running is tracked by the Node Manager.
- It keeps the data in the Resource Manager updated.
- Node Manager can also destroy or kill the container if it gets an order from the Resource Manager to do so.

Application Master

- Every job submitted to the framework is an application, and every application has a specific Application Master associated with it.
- It negotiates resources from the Resource Manager.

Container

- A container is a set of physical resources (CPU cores, RAM, disks, etc.) on a single node. The tasks of a container `are listed below:
    - It grants the right to an application to use a specific amount of resources (memory, CPU, etc.) on a specific host.

How is an application submitted in Hadoop YARN?

a. Submit the job

b. Get an application id

c. Retrieval of the context of application submission

    i. Start Container Launch

    ii. Launch Application Master

d. Allocate Resources

    i. Container

    ii. Launching

e. Executing