

**Hadoop Architecture Overview**

Hadoop follows a Master Slave architecture for the transformation and analysis of large datasets using Hadoop MapReduce paradigm. The 3 important Hadoop components that play a vital role in the Hadoop architecture are -

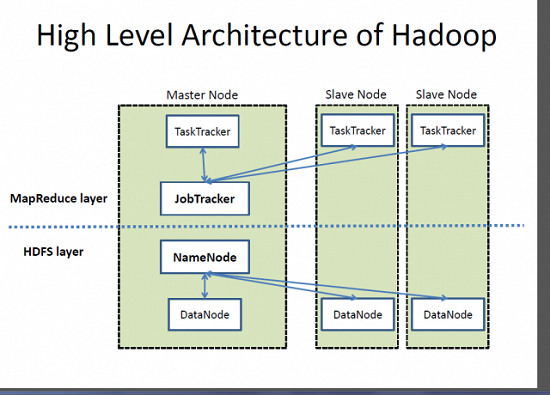
1. Hadoop Distributed File System (HDFS) – Patterned after the UNIX file system
2. Hadoop MapReduce
3. Yet Another Resource Negotiator (YARN)

Hadoop follows a master slave architecture design for data storage and distributed data processing using [HDFS](https://www.dezyre.com/hadoop-course/hdfs) and MapReduce respectively.

The master node for data storage is Hadoop HDFS is the Name Node.

The master node for parallel processing of data using Hadoop MapReduce is the Job Tracker.

The slave nodes in the Hadoop architecture are the other machines in the Hadoop cluster which store data and perform complex computations. Every slave node has a Task Tracker daemon and a Data Node that synchronizes the processes with the Job Tracker and Name Node respectively.



HDFS Architecture:

Hadoop Distributed File System (HDFS) stores the application data and file system metadata separately on dedicated servers. Name Node and Data Node are the two critical components of the Hadoop HDFS architecture. Application data is stored on servers referred to as Data Nodes and file system metadata is stored on servers referred to as Name Node.

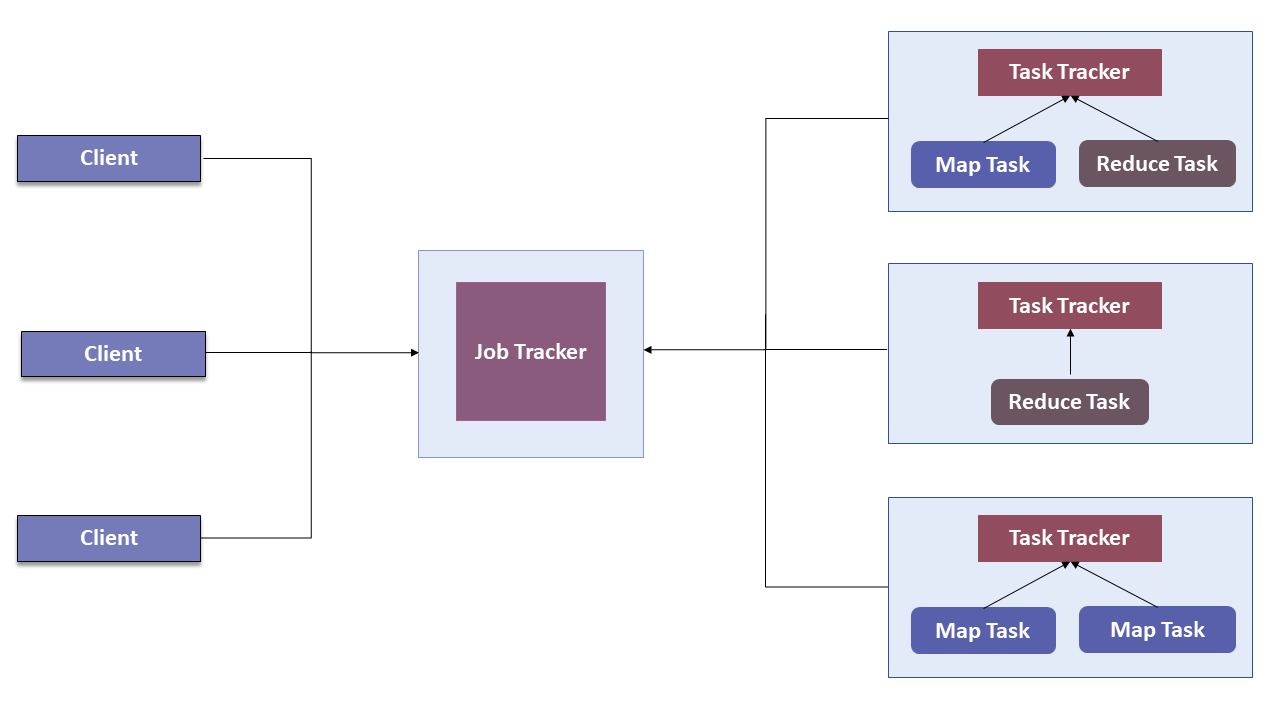
HDFS replicates the file content on multiple Data Nodes based on the replication factor to ensure reliability of data. The Name Node and Data Node communicate with each other using TCP based protocols.

### **MapReduce in Hadoop Application Architecture Implementation**

The heart of the distributed computation platform Hadoop is its java-based programming paradigm Hadoop Map Reduce. Map or Reduce is a special type of directed acyclic graph that can be applied to a wide range of business use cases. Map function transforms the piece of data into key-value pairs and then the keys are sorted where a reduce function is applied to merge the values based on the key into a single output.

## ****Why YARN?****

In Hadoop version 1.0 which is also referred to as MRV1 (Map Reduce Version 1), Map Reduce performed both processing and resource management functions. It consisted of a Job Tracker which was the single master. The Job Tracker allocated the resources, performed scheduling and monitored the processing jobs. It assigned map and reduce tasks on a number of subordinate processes called the Task Trackers. The Task Trackers periodically reported their progress to the Job Tracker.



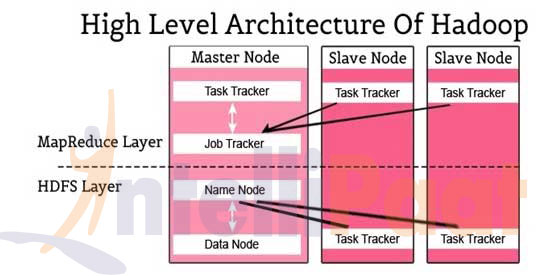
To overcome all these issues, YARN was introduced in Hadoop version 2.0 in the year 2012 by Yahoo and Hortonworks. The basic idea behind YARN is to relieve MapReduce by taking over the responsibility of Resource Management and Job Scheduling. YARN started to give Hadoop the ability to run non-MapReduce jobs within the Hadoop framework.

Apart from Resource Management, YARN also performs Job Scheduling. YARN performs all your processing activities by allocating resources and scheduling tasks. Apache Hadoop YARN Architecture consists of the following main components:

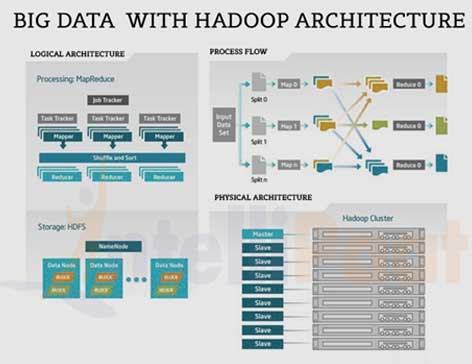
1. **Resource Manager:**Runs on a master daemon and manages the resource allocation in the cluster.
2. **Node Manager:**They run on the slave daemons and are responsible for the execution of a task on every single Data Node.
3. **Application Master:**Manages the user job lifecycle and resource needs of individual applications. It works along with the Node Manager and monitors the execution of tasks.
4. **Container:** Package of resources including RAM, CPU, Network, HDD etc on a single node.

## The Hadoop High-level Architecture

**Hadoop Architecture based on the two main components namely MapReduce and HDFS**

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Hadoop-High-level-Architecture.jpg)

## Different Hadoop Architectures based on the Parameters chosen

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Big-data-With-Hadoop.jpg)

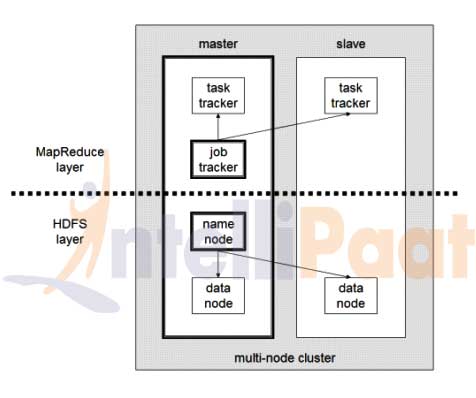
## The Apache Hadoop Module

**Hadoop Common:** Includes the common utilities which supports the other Hadoop modules

[HDFS](https://intellipaat.com/interview-question/hdfs-interview-questions/)**:** Hadoop Distributed File System provides unrestricted, high-speed access to the data application.

**Hadoop YARN:** This technology is basically used for scheduling of job and efficient management of the cluster resource.

[MapReduce](https://intellipaat.com/tutorial/mapreduce-tutorial/)**:** This is a highly efficient methodology for parallel processing of huge volumes of data.

[](https://cdn.intellipaat.com/mediaFiles/2015/07/Architecture-of-Hadoop.jpg)

***Then there are other projects included in the Hadoop module which are less used:***

[**Apache Ambari**](https://intellipaat.com/apache-ambari-training/)**:**It is a tool for managing, monitoring and provisioning of the Hadoop clusters. Apache Ambari supports the HDFS and MapReduce programs. Major highlights of Ambari are:

* Managing of the Hadoop framework is highly efficient, secure and consistent.
* Management of cluster operations with an intuitive web UI and a robust API
* The installation and configuration of Hadoop cluster are simplified effectively.
* It is used to support automation, smart configuration and recommendations
* Advanced cluster security set-up comes additional with this tool kit.
* The entire cluster can be controlled using the metrics, heat maps, analysis and troubleshooting
* Increased levels of customization and extension make this more valuable.

[**Cassandra**](https://intellipaat.com/cassandra-training/)**: It** is a distributed system to handle extremely huge amount of data which is stored across several commodity servers. The [database management system](https://intellipaat.com/database-architect-training/) (DBMS) is highly available  with no single point of failure.

[**HBase**](https://intellipaat.com/hbase-training/)**:**it is a non-relational, distributed database management system that works efficiently on sparse data sets and it is highly scalable.

[**Apache Spark**](https://intellipaat.com/tutorial/spark-tutorial/)**:**This is highly agile, scalable and secure the Big Data compute engine, versatile the sufficient work on a wide variety of applications like real-time processing, machine learning, ETL and so on.

**Hive:** It is a data warehouse tool basically used for analyzing, querying and summarizing of analyzed data concepts on top of the Hadoop framework.

**Pig:** Pig is a high-level framework which ensures us to work in coordination either with Apache Spark or MapReduce to analyze the data. The language used to code for the frameworks are known as Pig Latin.

**Sqoop:** This framework is used for transferring the data to Hadoop from relational databases. This application is based on a [command-line interface](https://intellipaat.com/tutorial/hbase-tutorial/installation/).

**Oozie:** This is a scheduling system for workflow management, executing workflow routes for successful completion of the task in a Hadoop.

**Zookeeper:** Open source centralized service which is used to provide coordination between distributed applications of Hadoop. It offers the registry and synchronization service on a high level.

* **Hadoop MapReduce (Processing/Computation layer) –**MapReduce is a parallel programming model mainly used for writing large amount of data distribution applications devised from Google for efficient processing of large amounts of datasets, on large group of clusters.
* **Hadoop HDFS (Storage layer) –**Hadoop Distributed File System or HDFS is based on the Google File System (GFS) which provides a distributed file system that is especially designed to run on commodity hardware. It reduces the faults or errors and helps incorporate low-cost hardware. It gives high level processing throughput access to application data and is suitable for applications with large datasets.
* **Hadoop YARN –**Hadoop YARN is a framework used for job scheduling and cluster resource management.
* **Hadoop Common –**This includes Java libraries and utilities which provide those java files which are essential to start Hadoop.
* **Task Tracker –**It is a node which is used to accept the tasks such as shuffle and MapReduce form job tracker.
* **Job Tracker –**It is a service provider which runs MapReduce jobs on cluster.
* **Name Node –**It is a node where Hadoop stores all file location information (data stored location) in Hadoop distributed file system.
* **Data Node –** The data is stored in the Hadoop distributed file system.
* **Data Node –**It stores data in the Hadoop distributed file system.

## How does Hadoop Work?

Hadoop helps to execute large amount of processing where the user can connect together multiple commodity computers to a single-CPU, as a single functional distributed system and have the particular set of clustered machines that reads the dataset in parallel and provide intermediate, and after integration gets the desired output.

Hadoop runs code across a cluster of computers and performs the following tasks:

* Data are initially divided into files and directories. Files are divided into consistent sized blocks ranging from 128M and 64M.
* Then the files are distributed across various cluster nodes for further processing of data.
* Job tracker starts its scheduling programs on individual nodes.
* Once all the nodes are done with scheduling then the output is return back.

**Word Count - Hadoop Map Reduce Example – How it works?**

Hadoop Wordcount operation occurs in 3 stages –

1. Mapper Phase
2. Shuffle Phase
3. Reducer Phase

### **Hadoop Word Count Example- Mapper Phase Execution**

The text from the input text file is tokenized into words to form a key value pair with all the words present in the input text file. The key is the word from the input file and value is ‘1’.

For instance if you consider the sentence “An elephant is an animal”. The mapper phase in the Wordcount example will split the string into individual tokens i.e. words. In this case, the entire sentence will be split into 5 tokens (one for each word) with a value 1 as shown below –

Key-Value pairs from Hadoop Map Phase Execution-

(an,1)

(elephant,1)

(is,1)

(an,1)

(animal,1)

### ​**Hadoop Word Count Example- Shuffle Phase Execution**

After the map phase execution is completed successfully, shuffle phase is executed automatically wherein the key-value pairs generated in the map phase are taken as input and then sorted in alphabetical order. After the shuffle phase is executed from the Wordcount example code, the output will look like this -

(an,1)

(an,1)

(animal,1)

(elephant,1)

(is,1)

In the reduce phase, all the keys are grouped together and the values for similar keys are added up to find the occurrences for a particular word. It is like an aggregation phase for the keys generated by the map phase. The reducer phase takes the output of shuffle phase as input and then reduces the key-value pairs to unique keys with values added up. In our example “An elephant is an animal.” is the only word that appears twice in the sentence. After the execution of the reduce phase of MapReduce Word Count example program, appears as a key only once but with a count of 2 as shown below –

(an,2)

(animal,1)

(elephant,1)

(is,1)

This is how the MapReduce word count program executes and outputs the number of occurrences of a word in any given input file. An important point to note during the execution of the Wordcount example is that the mapper class in the Word Count program will execute completely on the entire input file and not just a single sentence. Suppose if the input file has 15 lines then the mapper class will split the words of all the 15 lines and form initial key value pairs for the entire dataset. The reducer execution will begin only after the mapper phase is executed successfully.