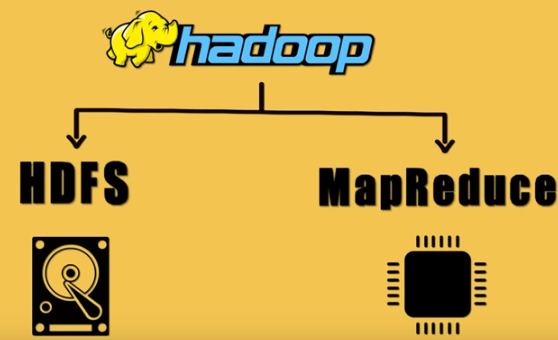
**List the Components of Hadoop 2.x**

**Hadoop** consists of MapReduce, the **Hadoop** distributed file system (HDFS).

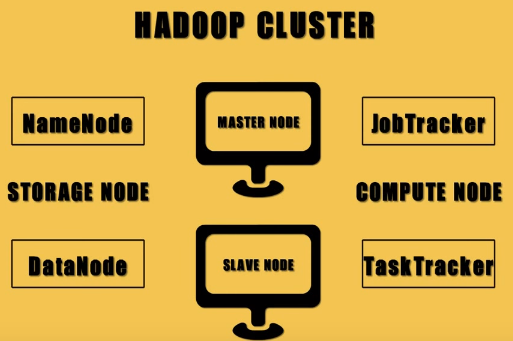


**What is HDFS?**

**Hadoop Distributed File System** is built to support applications with large data sets, including individual files that reach into the terabytes. It uses a master/slave architecture, with each cluster consisting of a single NameNode that manages file system operations and supporting DataNodes that manage data storage on individual compute nodes.

**Hadoop cluster**

A **Hadoop cluster** is a special type of computational **cluster** designed specifically for storing and analyzing huge amounts of unstructured data in a distributed computing environment.



**Namenode**

* NameNode is the centerpiece of HDFS.
* NameNode is also known as the Master
* NameNode only stores the metadata of HDFS – the directory tree of all files in the file system, and tracks the files across the cluster.
* NameNode does not store the actual data or the dataset. The data itself is actually stored in the DataNodes.
* NameNode knows the list of the blocks and its location for any given file in HDFS. With this information NameNode knows how to construct the file from blocks.
* NameNode is so critical to HDFS and when the NameNode is down, HDFS/Hadoop cluster is inaccessible and considered down.
* NameNode is a single point of failure in Hadoop cluster.
* NameNode is usually configured with a lot of memory (RAM). Because the block locations are help in main memory.

**DataNode**

* DataNode is responsible for storing the actual data in HDFS.
* DataNode is also known as the Slave
* NameNode and DataNode are in constant communication.
* When a DataNode starts up it announce itself to the NameNode along with the list of blocks it is responsible for.
* When a DataNode is down, it does not affect the availability of data or the cluster. NameNode will arrange for replication for the blocks managed by the DataNode that is not available.
* DataNode is usually configured with a lot of hard disk space. Because the actual data is stored in the DataNode.

**Job Tracker**

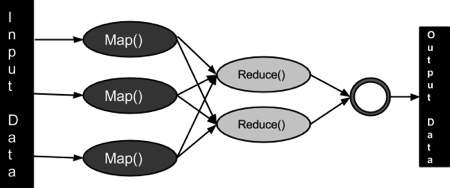
* JobTracker process runs on a separate node and not usually on a DataNode.
* JobTracker is an essential Daemon for MapReduce execution in MRv1. It is replaced by ResourceManager/ApplicationMaster in MRv2.
* JobTracker receives the requests for MapReduce execution from the client.
* JobTracker talks to the NameNode to determine the location of the data.
* JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data) and the available slots to execute a task on a given node.
* JobTracker monitors the individual TaskTrackers and the submits back the overall status of the job back to the client.
* JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.
* When the JobTracker is down, HDFS will still be functional but the MapReduce execution can not be started and the existing MapReduce jobs will be halted.

**TaskTracker**

* TaskTracker runs on DataNode. Mostly on all DataNodes.
* TaskTracker is replaced by Node Manager in MRv2.
* Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.
* TaskTrackers will be assigned Mapper and Reducer tasks to execute by JobTracker.
* TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.
* TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

## MapReduce

* MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.
* The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.
* Generally MapReduce paradigm is based on sending the computer to where the data resides!
* MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
* Map stage : The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
* Reduce stage : This stage is the combination of the Shufflestage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
* During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
* The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
* Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
* After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



## YARN (Yet Another Resource Negotiator)

* Apache Hadoop YARN (Yet Another Resource Negotiator) is a cluster management technology.
* YARN is one of the key features in the second-generation Hadoop 2 version of the Apache Software Foundation's open source distributed processing framework. Originally described by Apache as a redesigned resource manager, YARN is now characterized as a large-scale, distributed operating system for big data applications.

