What is Hadoop Cluster

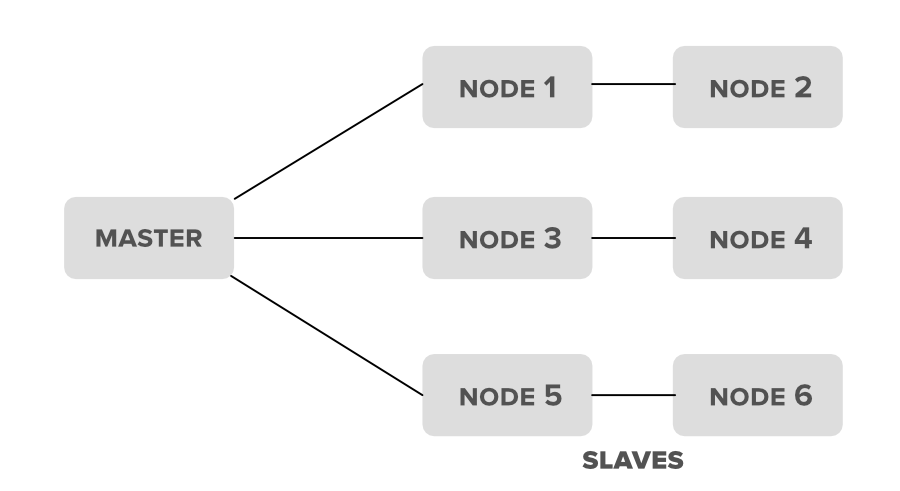
Cluster is a collection of something, a simple computer cluster is a group of various computers that are connected with each other through LAN(Local Area Network), the nodes in a cluster share the data, work on the same task and this nodes are good enough to work as a single unit means all of them to work together.

Similarly, a Hadoop cluster is also a collection of various commodity hardware(devices that are inexpensive and amply available). This Hardware components work together as a single unit. In the Hadoop cluster, there are lots of nodes (can be computer and servers) contains Master and Slaves, the Name node and Resource Manager works as Master and data node, and Node Manager works as a Slave. The purpose of Master nodes is to guide the slave nodes in a single Hadoop cluster.

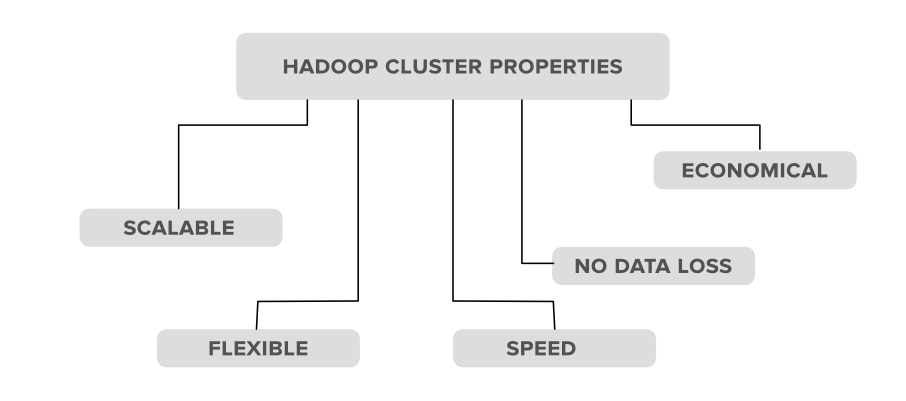
Hadoop clusters are used for storing, analyzing, understanding, and for finding the facts that are hidden behind the data or datasets which contain some crucial information. The Hadoop cluster stores different types of data and processes them.

* **Structured-Data:** The data which is well structured like Mysql.
* **Semi-Structured Data:** The data which has the structure but not the data type like XML, Json (Javascript object notation).
* **Unstructured Data:** The data that doesn’t have any structure like audio, video.

**Hadoop Cluster Schema:**



**Hadoop Clusters Properties**



**1. Scalability:** Hadoop clusters are very much capable of scaling-up and scaling-down the number of nodes i.e. servers or commodity hardware. Let’s see with an example of what actually this scalable property means. Suppose an organization wants to analyze or maintain around 5PB of data for the upcoming 2 months so he used 10 nodes(servers) in his Hadoop cluster to maintain all of this data. But now what happens is, in between this month the organization has received extra data of 2PB, in that case, the organization has to set up or upgrade the number of servers in his Hadoop cluster system from 10 to 12(let’s consider) in order to maintain it. The process of scaling up or scaling down the number of servers in the Hadoop cluster is called scalability.

**2. Flexibility:** This is one of the important properties that a Hadoop cluster possesses. According to this property, the Hadoop cluster is very much Flexible means they can handle any type of data irrespective of its type and structure. With the help of this property, Hadoop can process any type of data from online web platforms.

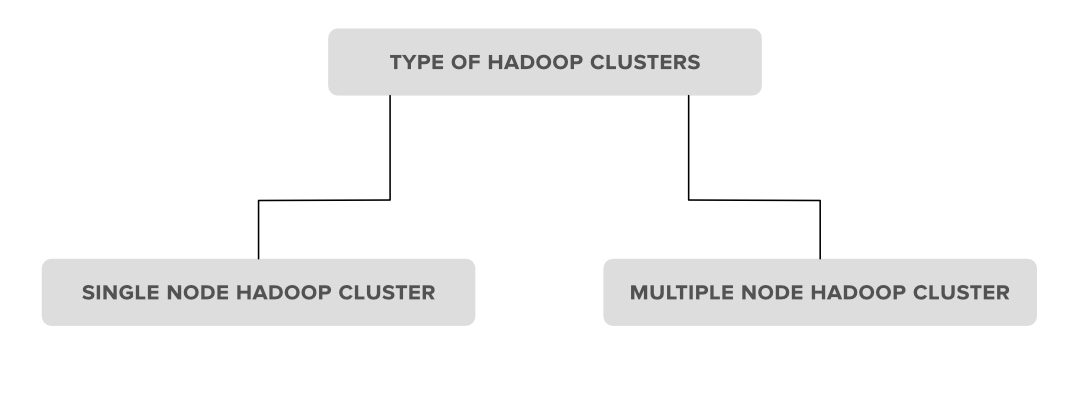
**3. Speed:**Hadoop clusters are very much efficient to work with a very fast speed because the data is distributed among the cluster and also because of its data mapping capability’s i.e. the MapReduce architecture which works on the Master-Slave phenomena.

**4. No Data-loss:**There is no chance of loss of data from any node in a Hadoop cluster because Hadoop clusters have the ability to replicate the data in some other node. So in case of failure of any node no data is lost as it keeps track of backup for that data.

**5. Economical:**The Hadoop clusters are very much cost-efficient as they possess the distributed storage technique in their clusters i.e. the data is distributed in a cluster among all the nodes. So in the case to increase the storage we only need to add one more another hardware storage which is not that much costliest.

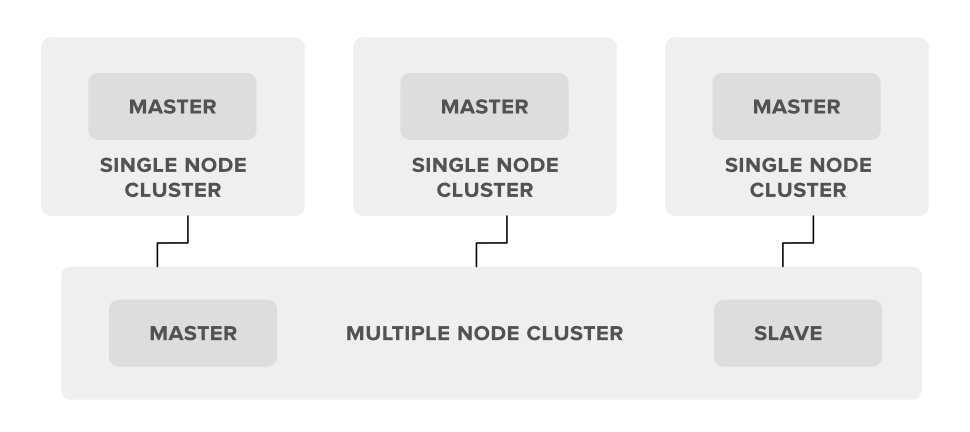
**Types of Hadoop clusters**

**1. Single Node Hadoop Cluster**  
**2. Multiple Node Hadoop Cluster**



**1. Single Node Hadoop Cluster:** In Single Node Hadoop Cluster as the name suggests the cluster is of an only single node which means all our Hadoop Daemons i.e. Name Node, Data Node, Secondary Name Node, Resource Manager, Node Manager will run on the same system or on the same machine. It also means that all of our processes will be handled by only single JVM(Java Virtual Machine) Process Instance.

**2. Multiple Node Hadoop Cluster:**In multiple node Hadoop clusters as the name suggests it contains multiple nodes. In this kind of cluster set up all of our Hadoop Daemons, will store in different-different nodes in the same cluster setup. In general, in multiple node Hadoop cluster setup we try to utilize our higher processing nodes for Master i.e. Name node and Resource Manager and we utilize the cheaper system for the slave Daemon’s i.e.Node Manager and Data Node.



What are Hadoop cluster Configuration files?

**Configuration Files**are the files which are located in the extracted tar.gz file in the etc/hadoop/ directory.  
All Configuration Files in [Hadoop](https://data-flair.training/blogs/hadoop-introduction-tutorial-quick-guide/)are listed below,

**1) HADOOP-ENV.sh**->>It specifies the environment variables that affect the JDK used by Hadoop Daemon (bin/hadoop). We know that Hadoop framework is wriiten in Java and uses JRE so one of the environment variable in Hadoop Daemons is $Java\_Home in Hadoop-env.sh.

**2) CORE-SITE.XML**->>It is one of the important configuration files which is required for runtime environment settings of a Hadoop cluster. It informs Hadoop daemons where the NAMENODE runs in the cluster. It also informs the Name Node as to which IP and ports it should bind.

**3) HDFS-SITE.XML**->>It is one of the important configuration files which is required for runtime environment settings of a Hadoop. It contains the configuration settings for NAMENODE, DATANODE, SECONDARYNODE. It is used to specify default block replication. The actual number of replications can also be specified when the file is created,

**4) MAPRED-SITE.XML**->>It is one of the important configuration files which is required for runtime environment settings of a Hadoop. It contains the configuration settings for [MapReduce](https://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/). In this file, we specify a framework name for MapReduce, by setting the MapReduce.framework.name.

**5) Masters**->>It is used to determine the master Nodes in [Hadoop cluster](https://data-flair.training/blogs/install-cloudera-hadoop-cdh5-ubuntu/). It will inform about the location of SECONDARY NAMENODE to Hadoop Daemon.  
The Mater File on Slave node is blank.

**6) Slave**->>It is used to determine the slave Nodes in Hadoop cluster.  
The Slave file at Master Node contains a list of hosts, one per line.  
The Slave file at Slave server contains IP address of Slave nodes.

## **What is MapReduce?**

A MapReduce is a data processing tool which is used to process the data parallelly in a distributed form. It was developed in 2004, on the basis of paper titled as "MapReduce: Simplified Data Processing on Large Clusters," published by Google.

The MapReduce is a paradigm which has two phases, the mapper phase, and the reducer phase. In the Mapper, the input is given in the form of a key-value pair. The output of the Mapper is fed to the reducer as input. The reducer runs only after the Mapper is over. The reducer too takes input in key-value format, and the output of reducer is the final output.

## **Steps in Map Reduce**

* The map takes data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
* Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
* An output of sort and shuffle sent to the reducer phase. The reducer performs a defined function on a list of values for unique keys, and Final output <key, value> will be stored/displayed.

## **Sort and Shuffle**

The sort and shuffle occur on the output of Mapper and before the reducer. When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk. Using the input from each Mapper <k2,v2>, we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2, list(v2)> is sent as input to reducer phase.

## **Usage of MapReduce**

* It can be used in various application like document clustering, distributed sorting, and web link-graph reversal.
* It can be used for distributed pattern-based searching.
* We can also use MapReduce in machine learning.
* It was used by Google to regenerate Google's index of the World Wide Web.
* It can be used in multiple computing environments such as multi-cluster, multi-core, and mobile environment.

To understand the working of Map reduce let us take an example

Suppose there is a word file containing some text with name as sample.txt. The content of the file is as follows:

Hello I am GeeksforGeeks

How can I help you

How can I assist you

Are you an engineer

Are you looking for coding

Are you looking for interview questions

what are you doing these days

what are your strengths

Now while storing this file in Hadoop, HDFS broke this file into four parts and named each part as first.txt, second.txt, third.txt, and fourth.txt. So, file will have 2 lines.

Now, when user process this file Map-Reduce comes into the picture. Suppose this user wants to run a query on this sample.txt. So, instead of bringing sample.txt on the local computer, we will send this query on the data. To keep a track of our request, we use ***Job Tracker*** (a master service). Job Tracker traps our request and keeps a track of it.

For this the user will write a query like:

J$hadoop jar query.jar DriverCode sample.txt result.output

Here,

1. **query.jar** : query file that needs to be processed on the input file.
2. **sample.txt**: input file.
3. **result.output**: directory in which output of the processing will be received.

now the Job Tracker traps this request and asks Name Node to run this request on sample.txt. Name Node then provides the metadata to the Job Tracker. Job Tracker now knows that sample.txt is stored in first.txt, second.txt, third.txt, and fourth.txt. As all these four files have three copies stored in HDFS, so the Job Tracker communicates with the ***Task Tracker*** (a slave service) of each of these files but it communicates with only one copy of each file which is residing nearest to it.

\*\*Applying the desired code on local first.txt, second.txt, third.txt and fourth.txt is a process., This process is called **Map**.

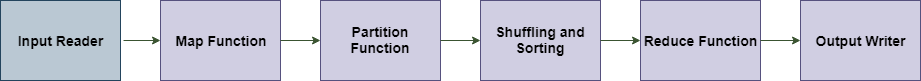
In Hadoop terminology, the main file sample.txt is called input file and its four subfiles are called input splits. So, ***in Hadoop the number of mappers for an input file are equal to number of input splits of this input file***. The responsibility of handling these mappers is of Job Tracker. the task trackers are slave services to the Job Tracker. So, in case any of the local machines breaks down then the processing over that part of the file will stop and it will halt the complete process. So, each task tracker sends heartbeat and its number of slots to Job Tracker in every 3 seconds. This is called the status of Task Trackers. In case any task tracker goes down, the Job Tracker then waits for 10 heartbeat times, that is, 30 seconds, and even after that if it does not get any status, then it assumes that either the task tracker is dead or is extremely busy.

Now if the system has generated output for individual first.txt, second.txt, third.txt, and fourth.txt. But this is not the user’s desired output. To produce the desired output, all these individual outputs have to be merged or reduced to a single output. This reduction of multiple outputs to a single one is also a process which is done by ***REDUCER***. **In Hadoop, as many reducers are there, those many number of output files are generated.** By default, there is always one reducer per cluster.

Hence, Map and Reduce are two different processes of the second component of Hadoop, that is, Map Reduce. These are also called phases of Map Reduce. Thus we can say that Map Reduce has two phases. Phase 1 is Map and Phase 2 is Reduce.

# **Data Flow In MapReduce**

MapReduce is used to compute the huge amount of data . To handle the upcoming data in a parallel and distributed form, the data has to flow from various phases.



## **Phases of MapReduce data flow**

### Input reader

The input reader reads the upcoming data and splits it into the data blocks of the appropriate size (64 MB to 128 MB). Each data block is associated with a Map function.

Once input reads the data, it generates the corresponding key-value pairs. The input files reside in HDFS.

### Map function

The map function process the upcoming key-value pairs and generated the corresponding output key-value pairs. The map input and output type may be different from each other.

### Partition function

The partition function assigns the output of each Map function to the appropriate reducer. The available key and value provide this function. It returns the index of reducers.

### Shuffling and Sorting

The data are shuffled between/within nodes so that it moves out from the map and get ready to process for reduce function. Sometimes, the shuffling of data can take much computation time.

The sorting operation is performed on input data for Reduce function. Here, the data is compared using comparison function and arranged in a sorted form.

### Reduce function

The Reduce function is assigned to each unique key. These keys are already arranged in sorted order. The values associated with the keys can iterate the Reduce and generates the corresponding output.

### Output writer

Once the data flow from all the above phases, Output writer executes. The role of Output writer is to write the Reduce output to the stable storage.

Map Reduce API

## **MapReduce Mapper Class**

In MapReduce, the role of the Mapper class is to map the input key-value pairs to a set of intermediate key-value pairs. It transforms the input records into intermediate records.

These intermediate records associated with a given output key and passed to Reducer for the final output.

### Methods of Mapper Class

|  |  |
| --- | --- |
| void cleanup(Context context) | This method called only once at the end of the task. |
| void map(KEYIN key, VALUEIN value, Context context) | This method can be called only once for each key-value in the input split. |
| void run(Context context) | This method can be override to control the execution of the Mapper. |
| void setup(Context context) | This method called only once at the beginning of the task. |

## **MapReduce Reducer Class**

In MapReduce, the role of the Reducer class is to reduce the set of intermediate values. Its implementations can access the Configuration for the job via the JobContext.getConfiguration() method.

### Methods of Reducer Class

|  |  |
| --- | --- |
| void cleanup(Context context) | This method called only once at the end of the task. |
| void map(KEYIN key, Iterable<VALUEIN> values, Context context) | This method called only once for each key. |
| void run(Context context) | This method can be used to control the tasks of the Reducer. |
| void setup(Context context) | This method called only once at the beginning of the task. |

## **MapReduce Job Class**

The Job class is used to configure the job and submits it. It also controls the execution and query the state. Once the job is submitted, the set method throws IllegalStateException.

### Methods of Job Class

|  |  |
| --- | --- |
| **Methods** | **Description** |
| Counters getCounters() | This method is used to get the counters for the job. |
| long getFinishTime() | This method is used to get the finish time for the job. |
| Job getInstance() | This method is used to generate a new Job without any cluster. |
| Job getInstance(Configuration conf) | This method is used to generate a new Job without any cluster and provided configuration. |
| Job getInstance(Configuration conf, String jobName) | This method is used to generate a new Job without any cluster and provided configuration and job name. |
| String getJobFile() | This method is used to get the path of the submitted job configuration. |
| String getJobName() | This method is used to get the user-specified job name. |
| JobPriority getPriority() | This method is used to get the scheduling function of the job. |
| void setJarByClass(Class<?> c) | This method is used to set the jar by providing the class name with .class extension. |
| void setJobName(String name) | This method is used to set the user-specified job name. |
| void setMapOutputKeyClass(Class<?> class) | This method is used to set the key class for the map output data. |
| void setMapOutputValueClass(Class<?> class) | This method is used to set the value class for the map output data. |
| void setMapperClass(Class<? extends Mapper> class) | This method is used to set the Mapper for the job. |
| void setNumReduceTasks(int tasks) | This method is used to set the number of reduce tasks for the job |
| void setReducerClass(Class<? extends Reducer> class) | This method is used to set the Reducer for the job. |