**BIGDATA HADOOP**

DATA :- COLLECTTION OF FACTS AND FIGURE WHICHJ ARE EITHER PROCESSED OR ENPROCESSED

**DATA TYPE**

**STRUCTURED**: RDBMS:ORACLE,MYSQL,SQL.DB2

**SEMI STRUCTURED** :XML(extensible markup language),PDFS,EMAIL,CSV(comma-separated values),

JSON(JavaScript Object Notation)

**UNSTRUCTURED** :AUDIO,VIDEO,IMAGES

**BIGDATA DEFINATION**

BEYOND THE STORAGE AND PROCESSING CAPACITY OF A SINGLE CENTRALISED SERVER AND WHICH

SATISFIE'S 3V'S

**VOLUME** :STORAGE

**VELOCITY** :SPEED

**VARIETY :**STRUCTURED DATA,UNSTRUCTURED DATA,SEMI STRUCTURED DATA

**HADOOP**(**High Availability Distributed Object Oriented Platform**)

GOOGLE

NETEEZA: (PROJECT NAME IN GOOGLE)

DOUGE CUTTING(WORK ON NETEEZA PROJECT)

GFS: **Google File System (STORE)**

MAPREDUCE**: (PROCESS)**

GFS\* + MAPREDUCE: PROJECT

HADOOP

HDFS+MAPREDUCE: HADOOP PROJECT

STORAGE+PROCESSING

APACHE HADOOP (It is a collection of [open-source](https://en.wikipedia.org/wiki/Open-source_software) software utilities that facilitates using a network of many computers to solve problems involving massive amounts of data and computation)

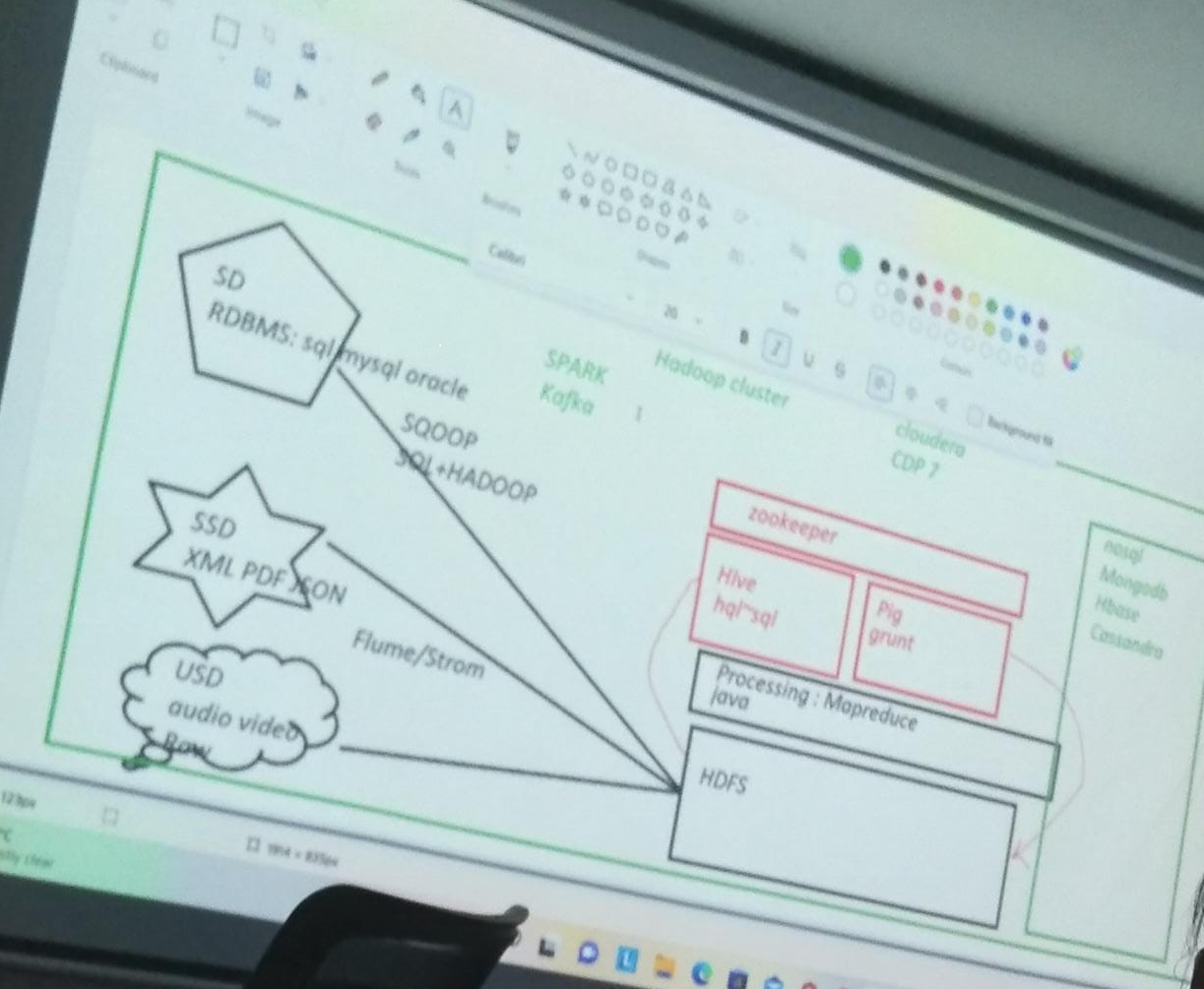
**OSS (OPEN-SOURCE SOFTWARE) JAVA TO HANDLE BIGDATA**

**BASED ON 3V’S CONCEPT**

**VOLUME**

**VELOCITY**

**VARIETY**

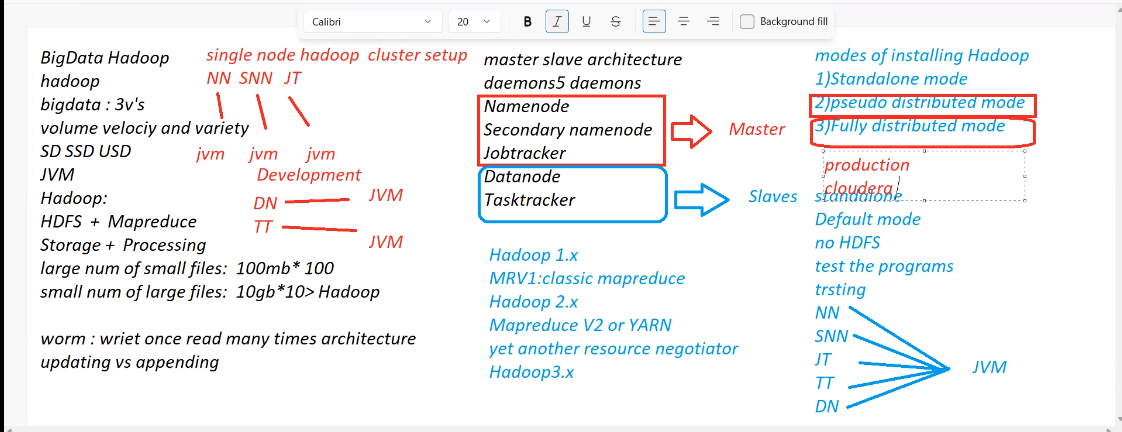
****

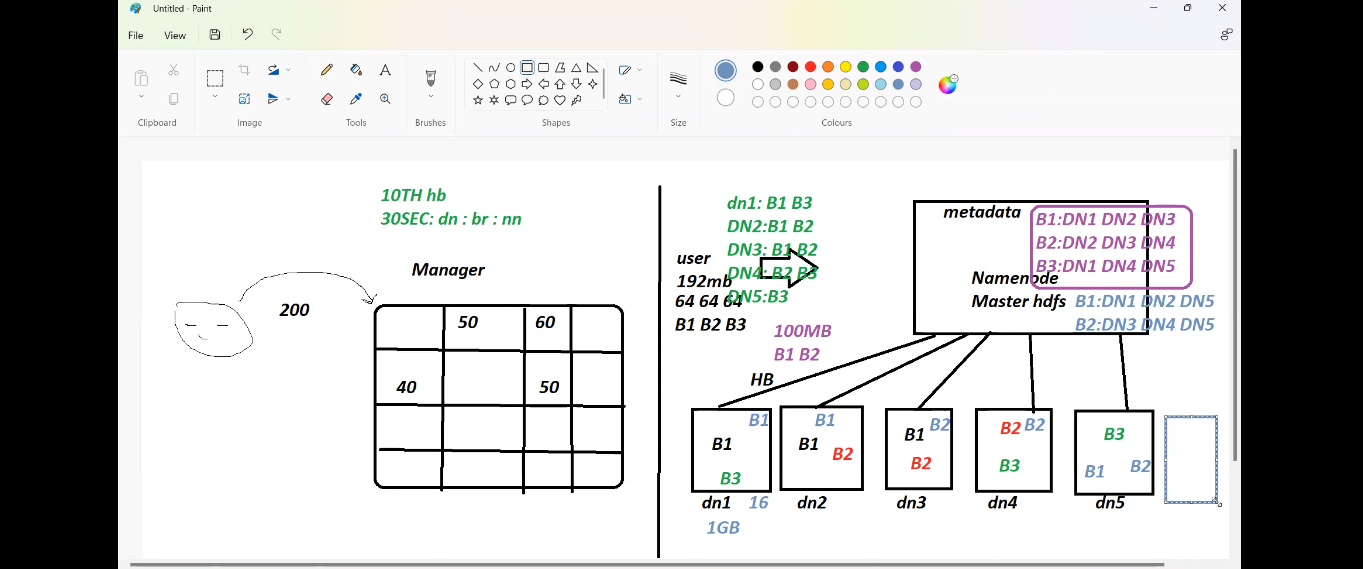
NODE = SERVER

Large number of small files: 100mb \*100

Small number of large files: 10gb\*10 we use this in Hadoop

1.Store data and processing data line by line read and write it not read and write randomly.



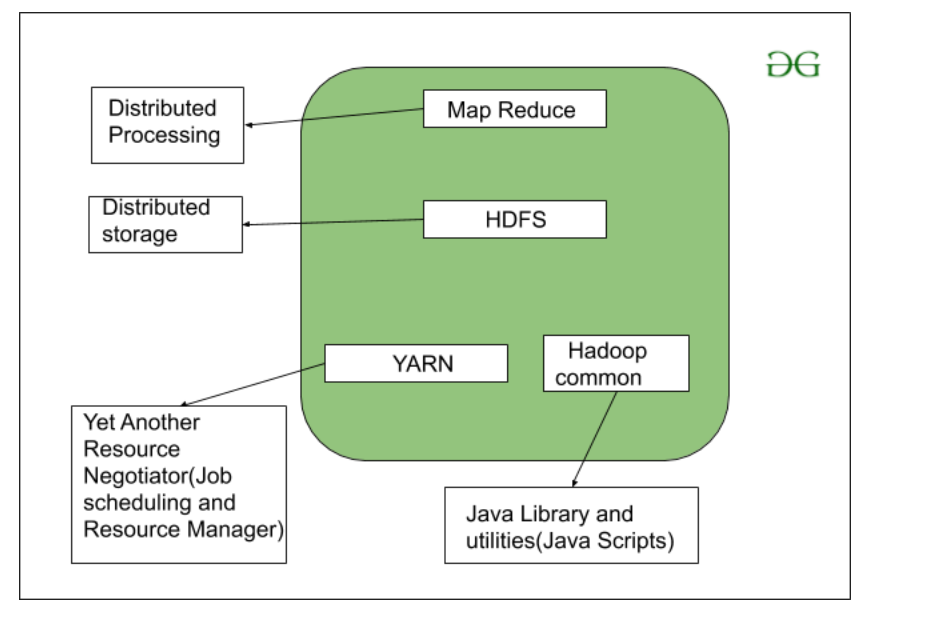


**Hadoop – Architecture**

* Difficulty Level : [Basic](https://www.geeksforgeeks.org/basic/)
* Last Updated : 22 Feb, 2022

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 Framework)
* Common Utilities or Hadoop Common

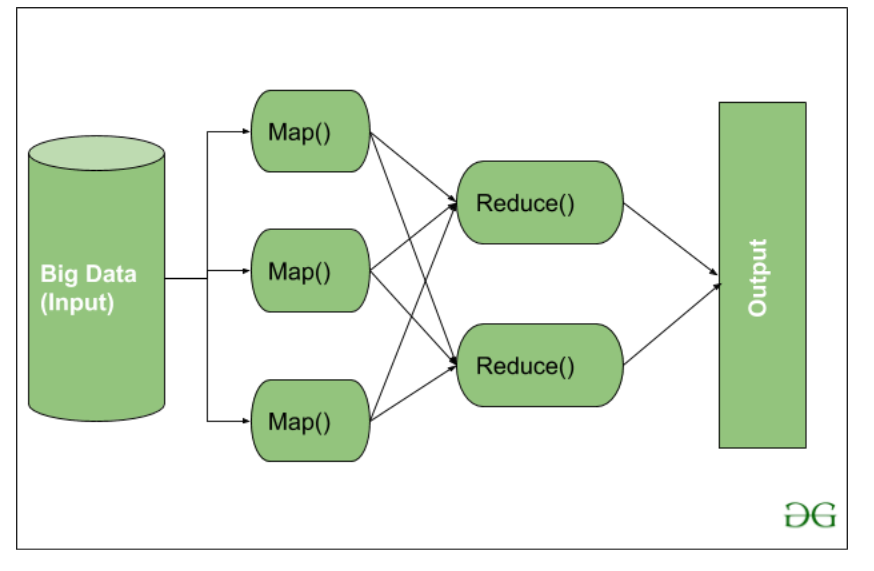


Let’s understand the role of each one of this component in detail.

**1. MapReduce**

MapReduce nothing but just like an Algorithm or a [data structure](https://www.geeksforgeeks.org/data-structures/) 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:

In first phase, **Map** is utilized and in next phase **Reduce** is utilized.



Here, we can see that the *Input* is provided to the Map() function then it’s *output* is used as an input to the Reduce function and after that, we receive our final output. Let’s understand What this Map() and Reduce() does.

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.

Let’s understand the *Map Task* and *Reduce Task* in detail.

**Map Task:**

* **RecordReader** The purpose of *recordreader* is to break the records. It is responsible for providing key-value pairs in a Map() function. The key is actually is its locational information and value is the data associated with it.
* **Map:**A map is nothing but a user-defined function whose work is to process the Tuples obtained from record reader. The Map() function either does not generate any key-value pair or generate multiple pairs of these tuples.
* **Combiner:**Combiner is used for grouping the data in the Map workflow. It is similar to a Local reducer. The intermediate key-value that are generated in the Map is combined with the help of this combiner. Using a combiner is not necessary as it is optional.
* **Partitionar:**Partitional is responsible for fetching key-value pairs generated in the Mapper Phases. The partitioner generates the shards corresponding to each reducer. Hashcode of each key is also fetched by this partition. Then partitioner performs it’s(Hashcode) modulus with the number of reducers(*key.hashcode()%(number of reducers)).*

**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.

**2. HDFS**

HDFS(Hadoop Distributed File System) is utilized for storage permission is a Hadoop cluster. It 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.

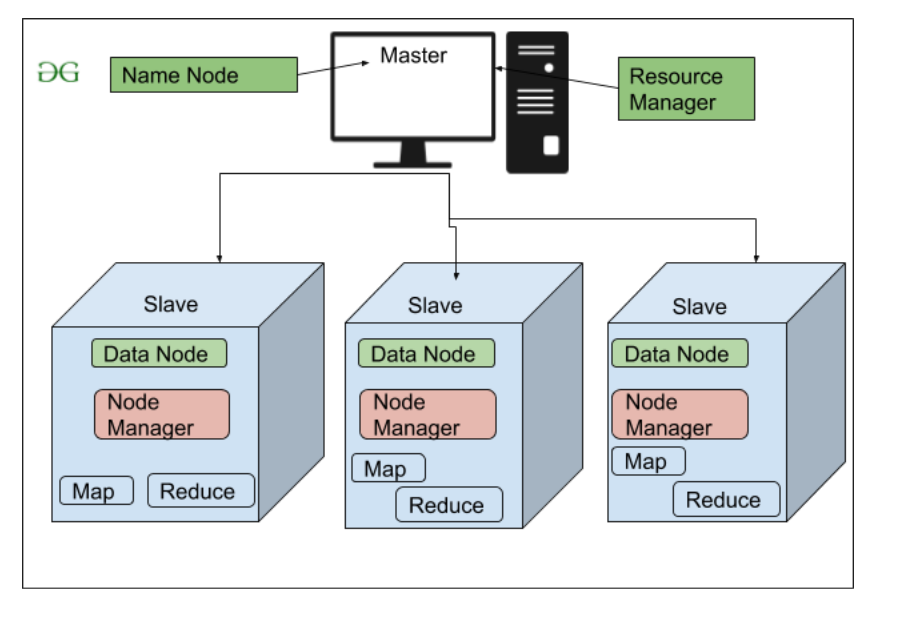
* NameNode(Master)
* DataNode(Slave)

**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 be the transaction logs that keep track of the user’s activity in a Hadoop cluster.

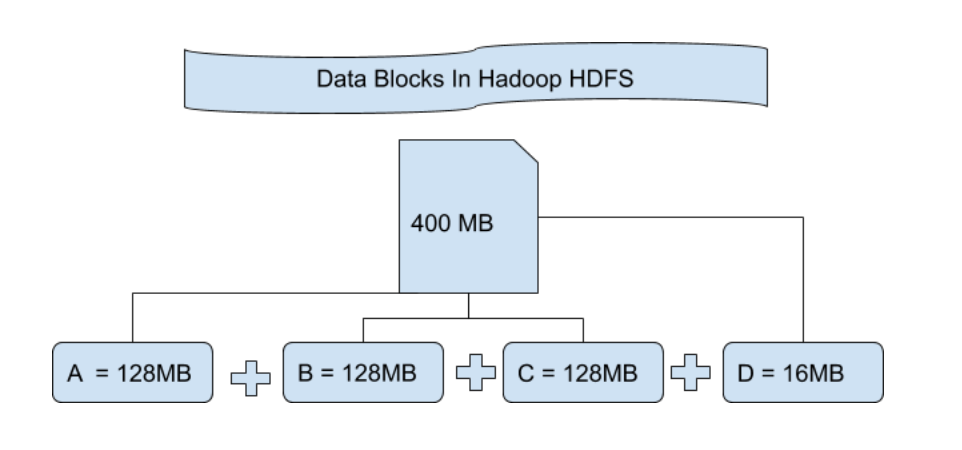
Meta Data can also be the name of the file, size, and the information about the location(Block number, Block ids) of Datanode that Namenode stores to find the closest DataNode for Faster Communication. Namenode instructs the DataNodes with the operation like delete, create, Replicate, etc.

**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.

**High Level Architecture Of Hadoop**



**File Block In HDFS:**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 you can also change it manually.



Let’s understand this concept of breaking down of file in blocks with an example. Suppose you have uploaded a file of 400MB to your HDFS then what happens is this file got divided into blocks of 128MB+128MB+128MB+16MB = 400MB size. Means 4 blocks are created each of 128MB except the last one. Hadoop doesn’t know or it doesn’t care about what data is stored in these blocks so it considers the final file blocks as a partial record as it does not have any idea regarding it. In the Linux file system, the size of a file block is about 4KB which is very much less than the default size of file blocks in the Hadoop file system. As we all know Hadoop is mainly configured for storing the large size data which is in petabyte, this is what makes Hadoop file system different from other file systems as it can be scaled, nowadays file blocks of 128MB to 256MB are considered in Hadoop.

**Replication In HDFS** Replication ensures the availability of the data. Replication is making a copy of something and the number of times you 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.

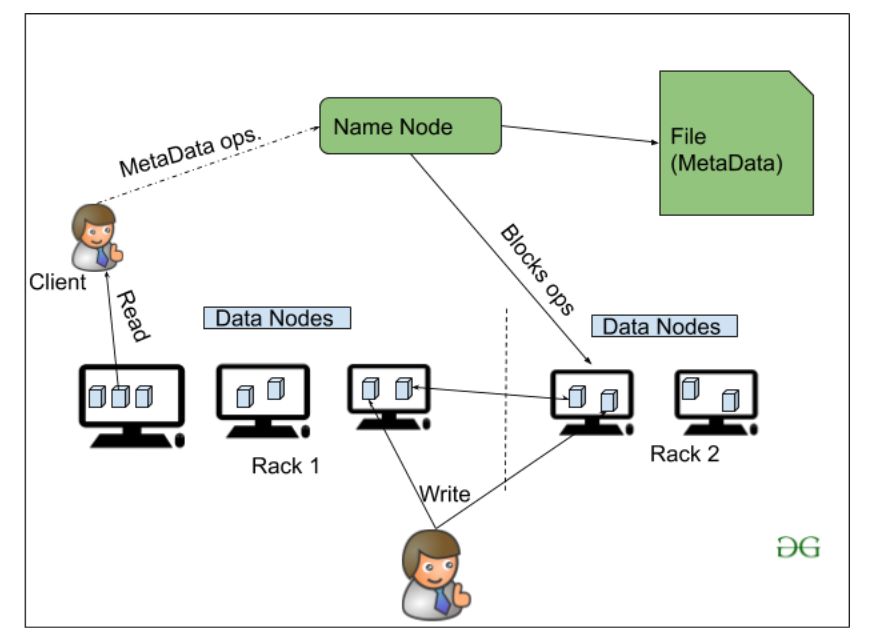
By default, the Replication Factor for Hadoop is set to 3 which can be configured means you can change it manually as per your requirement like in above example we have made 4 file blocks which means that 3 Replica or copy of each file block is made means total of 4×3 = 12 blocks are made for the backup purpose.

This is because for running Hadoop we are using commodity hardware (inexpensive system hardware) which can be crashed at any time. We are not using the supercomputer for our Hadoop setup. That is why we need such a feature in HDFS which can make copies of that file blocks for backup purposes, this is known as fault tolerance.

Now one thing we also need to notice that after making so many replica’s of our file blocks we are wasting so much of our storage but for the big brand organization the data is very much important than the storage so nobody cares for this extra storage. You can configure the Replication factor in your *hdfs-site.xml* file.

**Rack Awareness** The rack is nothing but just the physical collection of nodes in our Hadoop cluster (maybe 30 to 40). A large Hadoop cluster is consists of so many Racks . with the help of this Racks information Namenode chooses the closest Datanode to achieve the maximum performance while performing the read/write information which reduces the Network Traffic.

**HDFS Architecture**



**3. 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

**4. Hadoop common or Common Utilities**

Hadoop common or Common utilities are nothing but our java library and java files or we can say the java scripts that we need for all the other components present in a Hadoop cluster. these utilities are used by HDFS, YARN, and MapReduce for running the cluster. Hadoop Common verify that Hardware failure in a Hadoop cluster is common so it needs to be solved automatically in software by Hadoop Framework.