Hadoop 2.x has basically 3 components which are the basic pillars of its eco-system –

1. HDFS
2. YARN
3. Map Reduce.

Below diagram depicts high level architecture of hadoop -2.x-

HADOOP ECOSYSTEM(Pig,Hive,Mahout,HBASE,Sqoop,Flume etc)

MAP REDUCE(Data Processing)

YARN (Resource Management)

HDFS (Hadoop distributed filesystem)

All the components which run on top of hadoop eco-system are related or follow the path of above mentioned components in some way or other.

1. **HDFS-** HDFS is the primary storage system of Hadoop. It is a java based file system that provides scalable, fault tolerance, reliable and cost efficient data storage. It is designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware. It is built with the idea that the most efficient data processing system is a write-once and read-many times pattern.

In HDFS the data is stored in blocks with a block size of 128 MB by default. Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units. Also a file in HDFS that is smaller than a single block does not occupy a full block’s underlying storage which is not likely the case in a filesystem for a single disk, i.e a 1 MB file stored with a block size of 128 MB uses 1MB of disk space, not whole 128 MB.

A file can be larger than any single disk in the network. There is nothing that requires the blocks from a file to be stored on the same disk, so they can take advantage of any of the disks in the cluster.

One important factor of HDFS is its ability to replicate data to other data nodes. To insure against corrupted blocks and disk and machine failure, each block is replicated to other data nodes, normally we have 3 replications per block. If a block becomes unavailable, a copy can be read from another location. A block that is no longer available due to corruption or machine failure can be replicated from its alternative locations to other live machines to bring the replication factor back to normal level.

HDFS has 2 components-

1. **Name Node-** It basically stores metadata i.e number of blocks etc. Apart from that it also manages file system namespace and maintains filesystem tree.

All these information is stored in the form of 2 files- i. namespace image

ii. edit log

1. **Data Node-**They are working labors of the filesystem. They store and retrieve blocks when they are told to by clients or by namenode, and they report back to the namenode periodically with lists of blocks that they are storing.

In HADOOP 2 there is support for high availability (HA). In this system there are a pair of namenodes in active-standby configuration. In the event of failure of the active namenode, the standby takes over its duties to continue servicing client requests without interruption.

Below diagram is the brief what has been mentioned above-

B2 BLOCK

B1 BLOCK

B2 BLOCK

B1 BLOCK

B2 BLOCK

B1 BLOCK BLOCK

**NODE MANAGER**

**DATA NODE**

**NODE MANAGER**

**NODE MANAGER**

**ACTIVE NAMENODE**

APPLICATION MANAGER

SCHEDULER

**RESOURCE MANAGER**

**CLIENT APPLICATION**

**STANDBY NAMENODE**

**DATA NODE**

**DATA NODE**

1. **YARN-** It stands for YET ANOTHER RESOURCE NEGOTIATOR. It is introduced in hadoop 2.x to improve the map reduce implementation.

YARN run its services via 2 long running daemon-

1. **Resource Manager-** It runs on Master node. It is available 1 per cluster and is used to manage the use of resources across the cluster. It uses 2 other processes named Application Manager and Scheduler for MapReduce task and resource management.

The Resource manager has two major components-

1. **Scheduler-** The Scheduler is responsible for allocating resources to the various running applications. The Scheduler is pure scheduler in the sense that it performs no monitoring or tracking of status for the application. Also, it offers no guarantees about restarting failed tasks either due to application failure or hardware failures. The Scheduler performs its scheduling function based on the resource requirements of the applications; it does so based on the abstract notion of a resource Container which incorporates elements such as memory, cpu, disk, network etc.
2. **Application Manager -** The Applications Manager is responsible for accepting job-submissions, negotiating the first container for executing the application specific Application Master and provides the service for restarting the Application Master container on failure. The Application Master has the responsibility of negotiating appropriate resource containers from the Scheduler, tracking their status and monitoring for progress.
3. **Node Manager-** This daemon process run on slave nodes. Its main task is to launch and monitor containers on all the nodes in the cluster. A container executes an application specific process with a contained set of resources i.e memory, CPU and so on. It also reports resource utilization back to the resource manager.

To run an application on YARN, a client contacts the resource manager and asks it to

run an application master process. The resource manager then finds a node manager that can launch the application master in a container. Precisely what the application master does once it is running depends on the application. It could simply run a computation in the container it is running in and return the result to the client. Or it could request more containers from the resource managers and use them to run a distributed computation.

Below diagram depicts the same-

**NODE MANAGER**

**RESOURCE MANAGER**

**NODE MANAGER**

**NODE MANAGER**

1. **MAPREDUCE-** MAPREDUCE is a programming technique in HADOOP ecosystem which is used to analyze large datasets in parallel before reducing it to find the results.

Using this we can process any and all data, regardless of type or format — whether structured, semi-structured, or unstructured. Original data remains available even after batch processing for further analytics, all in the same platform.

It is designed to match the massive scale of HDFS and Hadoop, so that we can process unlimited amounts of data, fast, all within the same platform where it’s stored.

In the Hadoop ecosystem, Hadoop MapReduce is a framework based on YARN architecture. YARN based Hadoop architecture, supports parallel processing of huge data sets and MapReduce provides the framework for easily writing applications on thousands of nodes, considering fault and failure management.

The basic principle of operation behind MapReduce is that the “Map” job sends a query for processing to various nodes in a Hadoop cluster and the “Reduce” job collects all the results to output into a single value. Map Task in the Hadoop ecosystem takes input data and splits into independent chunks and output of this task will be the input for Reduce Task. In The same Hadoop ecosystem Reduce task combines Mapped data tuples into smaller set of tuples. Meanwhile, both input and output of tasks are stored in a file system. MapReduce takes care of scheduling jobs, monitoring jobs and re-executes the failed task.

The most basic logical flow of a mapreduce task is shown below-

**OUTPUT**

**REDUCE**

**SHUFFLE/SORT**

**INPUT**

**MAP**