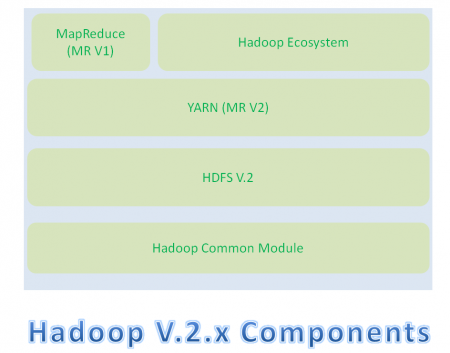
# Hadoop 2.x Architecture and its components

Apache Hadoop 2.x or later versions are using the following Hadoop Architecture. It is a Hadoop 2.x High-level Architecture.



* Hadoop Common Module is a Hadoop Base API (A Jar file) for all Hadoop Components. All other components works on top of this module.
* HDFS stands for Hadoop Distributed File System. It is also know as HDFS V2 as it is part of Hadoop 2.x with some enhanced features. It is used as a Distributed Storage System in Hadoop Architecture.
* YARN stands for Yet Another Resource Negotiator. It is new Component in Hadoop 2.x Architecture. It is also know as “MR V2”.
* Map Reduce is a Batch Processing or Distributed Data Processing Module. It is also know as “MR V1” as it is part of Hadoop 1.x with some updated features.

### Hadoop 2.x Major Components

Hadoop 2.x has the following three Major Components:

* HDFS
* YARN
* MapReduce

## HDFS – Hadoop Distributed File System (Storage Component)

HDFS is a distributed file system which stores the data in distributed manner. Rather than storing a complete file it divides a file into small blocks (of 64 or 128 MB size) and distributes them across the cluster. Each block is replicated (3 times as per default configuration) multiple times and is stored on different nodes to ensure data availability. Normally HDFS can be installed on native file systems like xfs, ext3 or ext4 (Similar to Unix/Linux file systems).

You can write file and read file from HDFS. You cannot update any file on HDFS. Recently Hadoop has added the support of appending content to the file which was not there in previous releases.

HDFS 2.X has two components HDFS federation and HDFS High Availability.

## HDFS Federation

In HDFS federation there are multiple name nodes each storing Meta data and block mapping of files and directories contained in a particular subdirectory. The list of subdirectories managed by a name node is name space volume. The blocks of files belonging to a name space is called Block pool.

For example , we can have two subdirectories /usr and shar. One name node for meta data and block mapping for name space volume and another name node for /share.



Let files and subdirectories under /usr constitute the name space volume1 for Namenode1.

Let files and subdirectories under /share constitute the name space volume2 for Namenode2.One data node can contain blocks of different namespace volumes. So, one data node can contain blocks of different name space volumes.

## 2.)High Availability



There is a pair of name does in active-standby configuration. In case of active-node failure, standby node takes over.

The name nodes must use highly -available shared storage to update edit logs. Edit logs are read by the standby Name node when it takes the responsibility of active Name node. Data nodes must send the block reports to both the name nodes. The check pointing is done by standby Name node.

## Map Reduce (Data Processing Component)

Map Reduce is the algorithm of executing any task on distributed system. Using Map Reduce one can process a large file in parallel manner. Map Reduce framework executes any task on different nodes as full file is distributed across the cluster in a form of various blocks.

It has two phases, Map(Mapper Task) and Reduce (Reducer Task)

* Each of these tasks would run on individual blocks of the data
* First mapper task would take each line of elements as an input and generates intermediate key value pairs
* Each mapper task is executed on a single block of data
* Than reducer task will take list of key value pairs for same keys, process the data and generates the final output
* A phase called shuffle and sort will take place between mapper and reducer task will send the data to proper reducer tasks
* Shuffle process maps the mapper output with the same key to the collection of values as a value
  + For example (key1, val1) and (key1, val2) will be converted to (key1, [val1, val2])
* The mapper and reducer tasks would in parallel
* The reducer tasks can start their word as soon as mapper tasks are completed

Lets understand Map Reduce algorithm using a simple example of word count,

We have a file with below content,

So Input to mapper task would be below (key, value) pair,

(2103, "Map Reduce is easy to learn")

(2130, "Map Reduce is simple")

***Note:*** Here key is file offset, most of the time it is not used

Now our mapper task will extract individual words from the line and will generate list of (key , value) pairs,

So Output of mapper tasks will be,

(Map, 1),

(Reduce, 1),

(is, 1),

(easy, 1),

(to, 1),

(learn, 1),

(Map, 1),

(Reduce, 1),

(is, 1),

(simple, 1)

Now this data is shuffled and sorted by key. So output after shuffle and sort process will be,

(Map, [1,1]),

(Reduce, [1,1]),

(easy, [1]),

(is, [1,1]),

(learn, [1]),

(simple, [1]),

(to, [1])

Above output will be the input of reducer tasks which will aggregate the values with the same keys and generate the below output,

(Map, 2),

(Reduce, 2),

(easy, 1),

(is, 2),

(learn, 1),

(simple, 1),

(to, 1)

Finally these voutput will be written into a file.

## YARN (MapReduceV2)

YARN has total three major components

* Resource Manager
* Node Manager
* Application Master

## 1) Resource Manager

* This daemon process resides on the Master Node (not necessarily on Name Node of Hadoop)
* Responsible for,
  + Managing resources scheduling for different compute applications in an optimum way
  + Coordinating with two process on master node, **Scheduler** and **ApplicationManager**

#### Scheduler

* This daemon process resides on the Master Node (runs along with Resource Manager daemon )
* Responsible for,
  + Scheduling the job execution as per submission request received by Resource Manager
  + Allocating resources to applications submitted to the cluster
  + Coordinating with ApplicationManager daemon and keeping track of resources of running applications

#### ApplicationManager

* This daemon process resides on the Master Node (runs along with Resource Manager daemon )
* Responsible for,
  + Helping Scheduler daemon to keeps track of running application by coordination
  + Accepting job submissions from client
  + Negotiating first container for executing application specific task with suitable Application Master on slave node

## 2) Node Manager

* This daemon process resides on the slave nodes (runs along with Data Node daemon)
* Responsible for,
  + Managing and executing containers
  + Monitoring resource usage (i.e. usage of memory, cpu, network etc..) and reporting it back to Resource Manager daemon
  + Periodically sending heart-bits to Resource Manager for its health status update

## 3) Application Master

* This daemon process runs on the slave node (along with the Node Manager daemon)
* It is per application specific library works with Node Manager to execute the task
* The instance of this daemon is per application, which means in case of multiple jobs submitted on cluster, it may have more than one instances of Application Master on slave nodes
* Responsible for,
  + Negotiating suitable resource containers on slave node from Resource Manager
  + Working with one or multiple Node Managers to monitor task execution on slave nodes
* Container is considered to be a small unit of resources (like cpu, memory, disk) belong to the Slave Node
* Scheduler process running along with Resource Manager daemon allocates the resources as a container
* At the beginning of a job execution with YARN, container allows Application Master process to make a use of some resources on any slave node on the cluster
* Then Application Master manages the application execution across other containers on slave nodes of a YARN cluster