**ASSIGNMENT 1.5**

**Components of hadoop 2.x and explanation**

Hadoop 2.x has the following three Major Components:

* HDFS
* YARN
* MapReduce

These three are also known as Three Pillars of Hadoop 2.x. Here major key component change is YARN. It is really game changing component in BigData Hadoop System.

1)HDFS

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware.

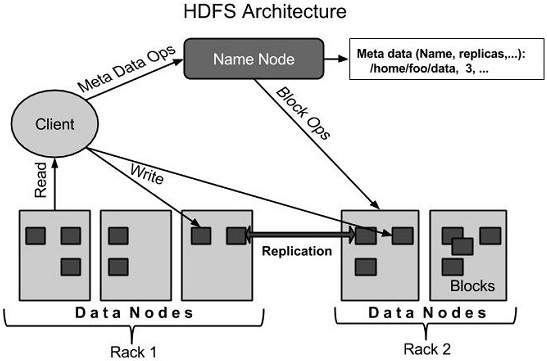
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

* It is suitable for the distributed storage and processing.
* Hadoop provides a command interface to interact with HDFS.
* The built-in servers of namenode and datanode help users to easily check the status of cluster.
* Streaming access to file system data.
* HDFS provides file permissions and authentication.

HDFS Architecture

Given below is the architecture of a Hadoop File System.



HDFS follows the master-slave architecture and it has the following elements.

**Namenode**

The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. The system having the namenode acts as the master server and it does the following tasks:

* Manages the file system namespace.
* Regulates client’s access to files.
* It also executes file system operations such as renaming, closing, and opening files and directories.

**Datanode**

The datanode is a commodity hardware having the GNU/Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

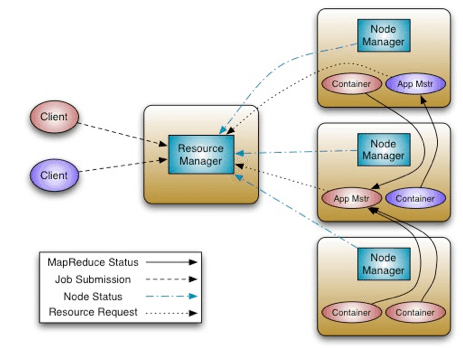
* Data nodes perform read-write operations on the file systems, as per client request.
* They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

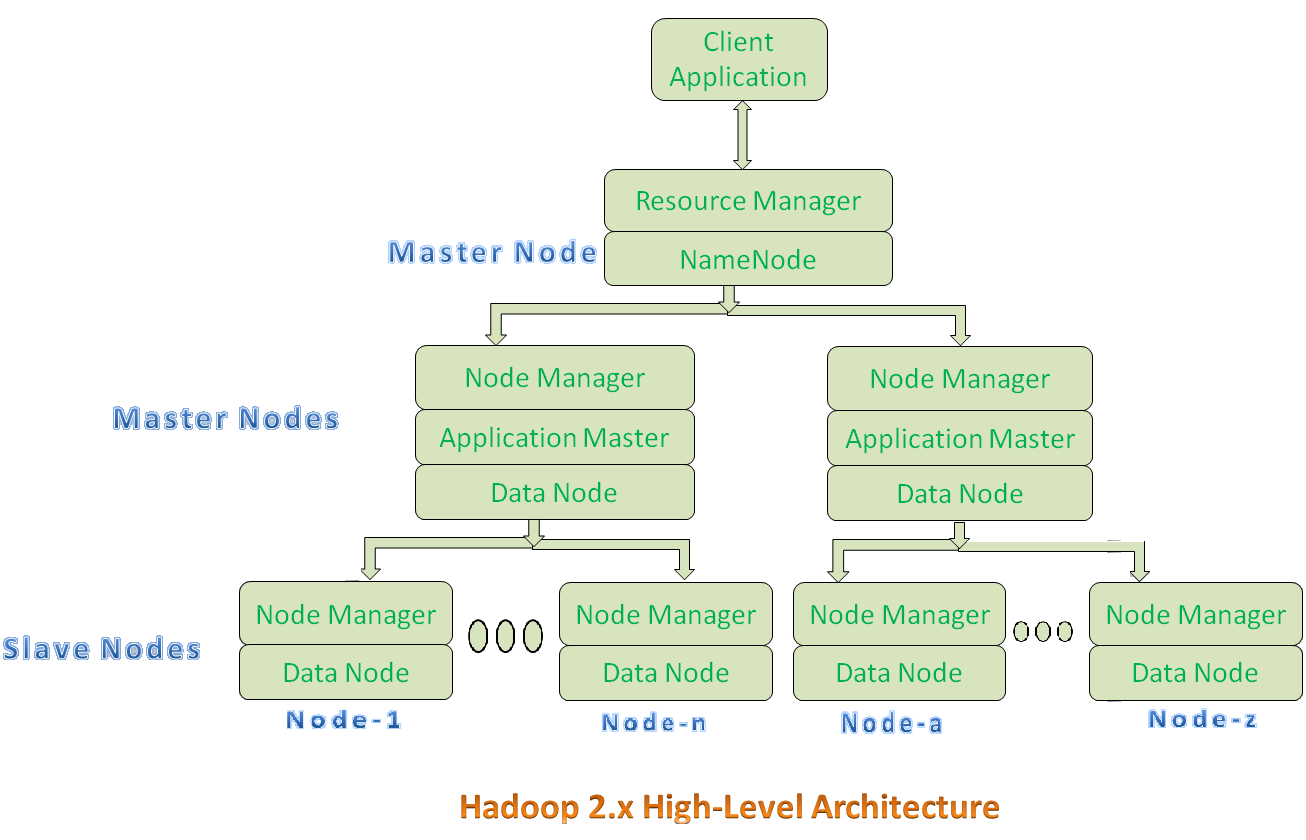
**Block**

Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

2)YARN

The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker i.e. resource management and job scheduling/monitoring, into separate daemons: a global ResourceManager and per-application ApplicationMaster (AM).

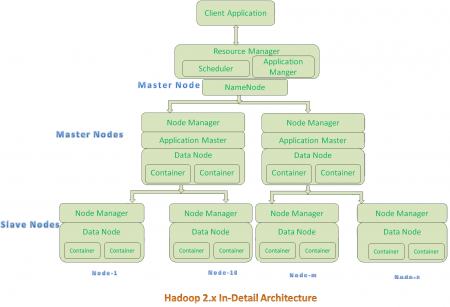




* All Master Nodes and Slave Nodes contains both MapReduce and HDFS Components.
* One Master Node has two components:
  1. Resource Manager(YARN or MapReduce v2)
  2. HDFS
* It’s HDFS component is also knows as NameNode. It’s NameNode is used to store Meta Data.
* In Hadoop 2.x, some more Nodes acts as Master Nodes as shown in the above diagram. Each this 2nd level Master Node has 3 components:
  1. Node Manager
  2. Application Master
  3. Data Node
* Each this 2nd level Master Node again contains one or more Slave Nodes as shown in the above diagram.
* These Slave Nodes have two components:
  1. Node Manager
  2. HDFS

It’s HDFS component is also knows as Data Node. It’s Data Node component is used to store actual our application Big Data. These nodes does not contain Application Master component.

**Hadoop 2.x Components In-detail Architecture**

[](https://cdn.journaldev.com/wp-content/uploads/2015/08/hadoop2.x-indetail-architecture.png)

**Hadoop 2.x Architecture Description**

**Resource Manager:**

* Resource Manager is a Per-Cluster Level Component.
* Resource Manager is again divided into two components:
  1. Scheduler
  2. Application Manager
* Resource Manager’s Scheduler is :
  1. Responsible to schedule required resources to Applications (that is Per-Application Master).
  2. It does only scheduling.
  3. It does care about monitoring or tracking of those Applications.

**Application Master:**

* Application Master is a per-application level component. It is responsible for:
  1. Managing assigned Application Life cycle.
  2. It interacts with both Resource Manager’s Scheduler and Node Manager
  3. It interacts with Scheduler to acquire required resources.
  4. It interacts with Node Manager to execute assigned tasks and monitor those task’s status.

**Node Manager:**

* Node Manager is a Per-Node Level component.
* It is responsible for:
  1. Managing the life-cycle of the Container.
  2. Monitoring each Container’s Resources utilization.

**Container:**

* Each Master Node or Slave Node contains set of Containers. In this diagram, Main Node’s Name Node is not showing the Containers. However, it also contains a set of Containers.
* Container is a portion of Memory in HDFS (Either Name Node or Data Node).
* In Hadoop 2.x, Container is similar to Data Slots in Hadoop 1.x. We will see the major differences between these two Components: Slots Vs Containers in my coming posts.

**3)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.

The Algorithm

* 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 **Shuffle** stage 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.

