**Assignment 3.1**

**Problem Statement**

**List the Components of Hadoop 2.x and explain each component in detail.**

Hadoop 2.x has the following two Major Components:

1. HDFS (Hadoop Distributed File System)

2. YARN (Yet Another Resource Negotiator)

**1.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 blocks 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).

**HDFS 2.x Daemons**

* Hadoop 2.x allows Multiple Name Nodes for HDFS Federation
* New Architecture allows HDFS High Availability mode in which it can have Active and StandBy Name Nodes (No Need of Secondary Name Node in this case)
* Hadoop 2.x Non HA mode has same Name Node and Secondary Name Node working.

**Name Node**

* It is responsible for manage metadata about files distributed across the cluster. It is on a master node
* It manages information like location of file blocks across cluster and it’s permission
* This process reads all the metadata from a file named fsimage and keeps it in memory
* After this process is started, it updates metadata for newly added or removed files in RAM
* It periodically writes the changes in one file called edits as edit logs
* This process is a heart of HDFS, if it is down HDFS is not accessible any more

##### **Secondary Name Node**

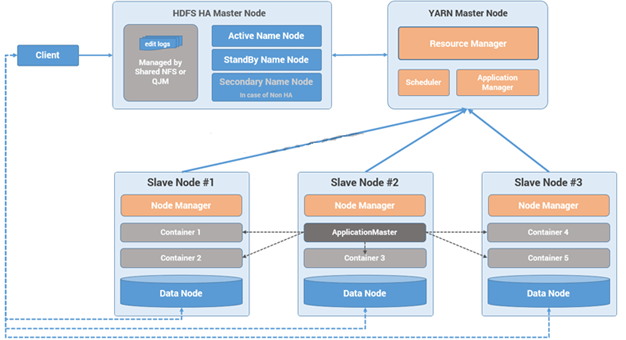
* For this also, only single instance of this process runs on a cluster
* This process can run on a master node (for smaller clusters) or can run on a separate node (in larger clusters) depends on the size of the cluster
* One misinterpretation from name is “This is a backup Name Node” but **IT IS NOT!!!!!**
* It manages the metadata for the Name Node. In the sense, it reads the information written in edit logs (by Name Node) and creates an updated file of current cluster metadata
* Than it transfers that file back to Name Node so that fsimage file can be updated
* So, whenever Name Node daemon is restarted it can always find updated information in fsimage file

##### **Data Node**

* There are many instances of this process running on various slave nodes(referred as Data nodes)
* It is responsible for storing the individual file blocks on the slave nodes in Hadoop cluster
* Based on the replication factor, a single block is replicated in multiple slave nodes(only if replication factor is > 1) to prevent the data loss
* Whenever required, this process handles the access to a data block by communicating with Name Node
* This process periodically sends heart bits to Name Node to make Name Node aware that slave process is running

**Standby Namenode**

In Hadoop 2.0, with the introduction of HA, the Standby Namenode came into picture. The standby namenode is the node that removes the problem of SPOF (Single Point Of Failure) that was there in Hadoop 1.x. The standby namenode provides automatic failover in case Active Namenode (can be simply called 'Namenode' if HA is not enabled) fails.   
  
Moreover, enabling HA is not mandatory. But, when it is enabled, you can't use Secondary Namenode. So, either Secondary Namenode is enabled OR Standby Namenode is enabled.



##### **2.MapReduce 2.x Daemons (YARN)**

MapReduce2 has replace old daemon process Job Tracker and Task Tracker with YARN components Resource Manager and Node Manager respectively. These two components are responsible for executing distributed data computation jobs in Hadoop

YARN has total three major components

1. ResourceManager
2. NodeManager
3. ApplicationMaster

**1) ResourceManager**

* This daemon process resides on the Master Node (not necessarily on NameNode 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 ResourceManager daemon )
* Responsible for,
  + Scheduling the job execution as per submission request received by ResourceManager
  + 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 ResourceManager 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 ApplicationMaster on slave node

**2) NodeManager**

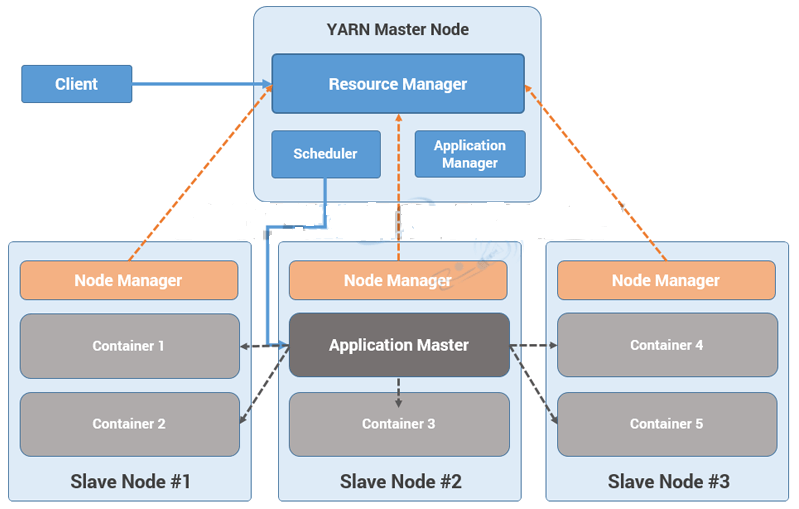
* This daemon process resides on the slave nodes (runs along with DataNode daemon)
* Responsible for,
  + Managing and executing containers
  + Monitoring resource usage (i.e. usage of memory, cpu, network etc..) and reporting it back to ResourceManager daemon
  + Periodically sending heart-bits to ResourceManager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

**3) ApplicationMaster**

* This daemon process runs on the slave node (along with the NodeManager daemon)
* It is per application specific library works with NodeManager 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 ApplicationMaster on slave nodes
* Responsible for,
  + Negotiating suitable resource containers on slave node from ResourceManager
  + Working with one or multiple NodeManagers to monitor task execution on slave nodes

**Container**

* It is considered to be a small unit of resources (like cpu, memory, disk) belong to the SlaveNode
* Scheduler process running along with ResourceManager daemon allocates the resources as a container
* At the beginning of a job execution with YARN, container allows ApplicationMaster process to make a use of some resources on any slave node on the cluster
* Then ApplicationMaster manages the application execution across other containers on slave nodes of a YARN cluster



**Flow of any job request**

**Step 1:**  Job/Application(which can be MapReduce, Java/Scala Application, DAG jobs like Apache Spark etc..) is submitted by the YARN client application to the ResourceManager daemon along with the command to start the ApplicationMaster on any container at NodeManager

**Step 2:**  ApplicationManager process on Master Node validates the job submission request and hand it over to Scheduler process for resource allocation

**Step 3:**  Scheduler process assigns a container or ApplicationMaster on one slave node

**Step 4:**  NodeManager daemon starts the ApplicationMaster service within one of its container using the command mentioned in Step 1, hence ApplicationMaster is considered to be the first container of any application

**Step 5:**  ApplicationMaster negotiates the other containers from ResourceManager by providing the details like location of data on slave nodes, required cpu, memory, cores etc..

**Step 6:**  ReourceManager allocates the best suitable resources on slave nodes and responds to ApplicationMaster with node details and other details

**Step 7:**  Then, ApplicationMaster send requests to NodeManagers on suggested slave nodes to start the containers

**Step 8:**  ApplicationMaster than manages the resources of requested containers while job execution and notifies the ResourceManager when execution is completed

**Step 9:**  NodeManagers periodically notify the ResourceManager with the current status of available resources on the node which information can be used by scheduler to schedule new application on the clusters

**Step 10:**  In case of any failure of slave node ResourceManager will try to allocate new container on other best suitable node so that

 ApplicationMaster can complete the process using new container