Session 03: Exploring MapReduce

Assignment 01 Question

**Problem Statement**

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

**Solution**

Hadoop 2.x is featured with Name Node HA which is referred as HDFS High Availability (HA).

* Hadoop 2.x supports two Name Nodes at a time one node is active and another is standby node
* Active Name Node handles the client operations in the cluster
* StandBy Name Node manages metadata same as Secondary Name Node in Hadoop 1.x
* When Active Name Node is down, Standby Name Node takes over and will handle the client operations then after
* HDFS HA can be configured by two ways
  + Using Shared NFS Directory
  + Using Quorum Journal Manager

Hadoop2.x has mainly 2 set of daemons

* **HDFS 2.x Daemons:**  Name Node, Secondary Name Node (not required in HA) and Data Nodes
* **MapReduce 2.x Daemons (YARN):**  Resource Manager, Node Manager

**HDFS 2.x Daemons**

The working methodology of HDFS 2.x daemons is same as it was in Hadoop 1.x Architecture with following differences.

* 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 same as in Hadoop 1.x architecture

##### 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 2.

**Yarn in detail**

In Hadoop 1.x Architecture JobTracker daemon was carrying the responsibility of Job scheduling and Monitoring as well as was managing resource across the cluster. And TaskTracker daemon was executing map reduce tasks on the slave nodes. YARN has divided the responsibilities of JobTracker to two processes ResourceManager and ApplicationMaster and instead of TaskTracker is using NodeManager daemon for map reduce task execution.

YARN has total three major components

* ResourceManager
* NodeManager
* 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

### What is 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

## Brief overview of YARN Architecture

You can see how above components are arranged in a typical YARN Cluster in following figure. Now let’s discuss about step by step Job Execution process in YARN Cluster.

**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 for 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