Hadoop 1.x Architecture

Apache Hadoop 1.x or earlier versions are using the following Hadoop Architecture. It is a Hadoop 1.x High-level Architecture. We will discuss in-detailed Low-level Architecture in coming sections.

Hadoop 1.x Major Components

Hadoop 1.x Major Components components are: **HDFS and MapReduce**. They are also known as "Two Pillars" of Hadoop 1.x.

HDFS:

HDFS is a Hadoop Distributed FileSystem, where our BigData is stored using Commodity Hardware.

It is designed to work with Large DataSets with default block size is 128MB

HDFS component is again divided into two sub-components:

1. **Name Node**

   Name Node is placed in Master Node.

   It used to store Meta Data about Data Nodes like "How many blocks are stored in Data Nodes, Which Data Nodes have data, Slave Node Details, Data Nodes locations, timestamps etc" .

2. **Data Node**

   Data Nodes are places in Slave Nodes. It is used to store our Application Actual Data. It stores data in Data Slots of size 128MB by default.

MapReduce:

MapReduce is a Distributed Data Processing or Batch Processing Programming Model. Like HDFS, MapReduce component also uses Commodity Hardware to process "High Volume of Variety of Data at High Velocity Rate" in a reliable and fault-tolerant manner.

MapReduce component is again divided into two sub-components:

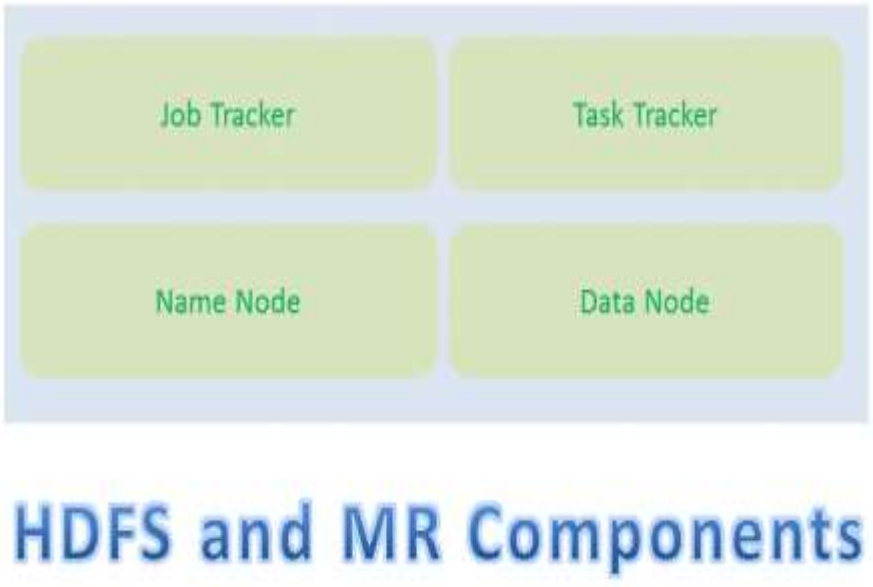
1. **Job Tracker**

   Job Tracker is used to assign MapReduce Tasks to Task Trackers in the Cluster of Nodes. Sometimes, it reassigns same tasks to other Task Trackers as previous Task Trackers are failed or shutdown scenarios.

   Job Tracker maintains all the Task Trackers status like Up/running, Failed, Recovered etc.

2. **Task Tracker**

   Task Tracker executes the Tasks which are assigned by Job Tracker and sends the status of those tasks to Job Tracker.



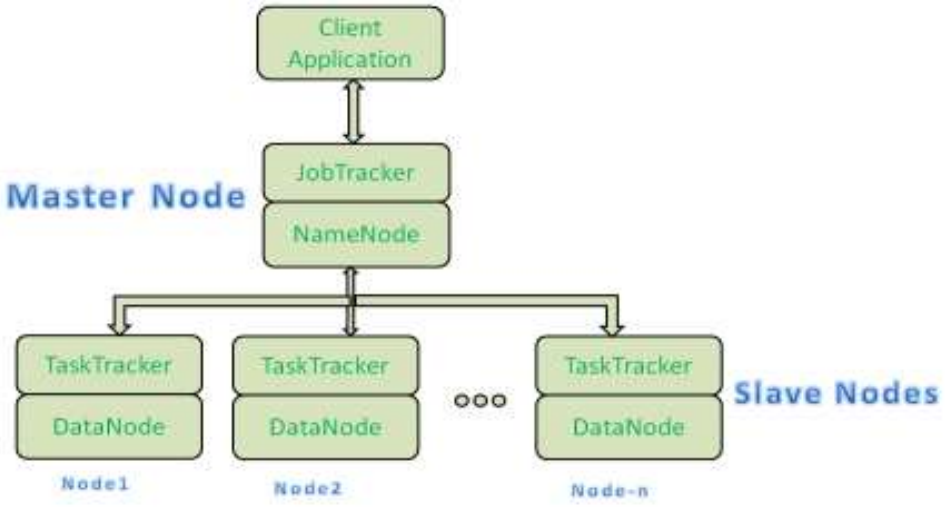
HDFS and MR Components

We will discuss these four sub-component's responsibilities and how they interact each other to perform a "Client Application Tasks" in detail in next section.

How Hadoop 1.x Major Components Works

Hadoop 1.x components follow this architecture to interact each other and to work parallel in a reliable and fault-tolerant manner.

Hadoop 1.x Components High-Level Architecture



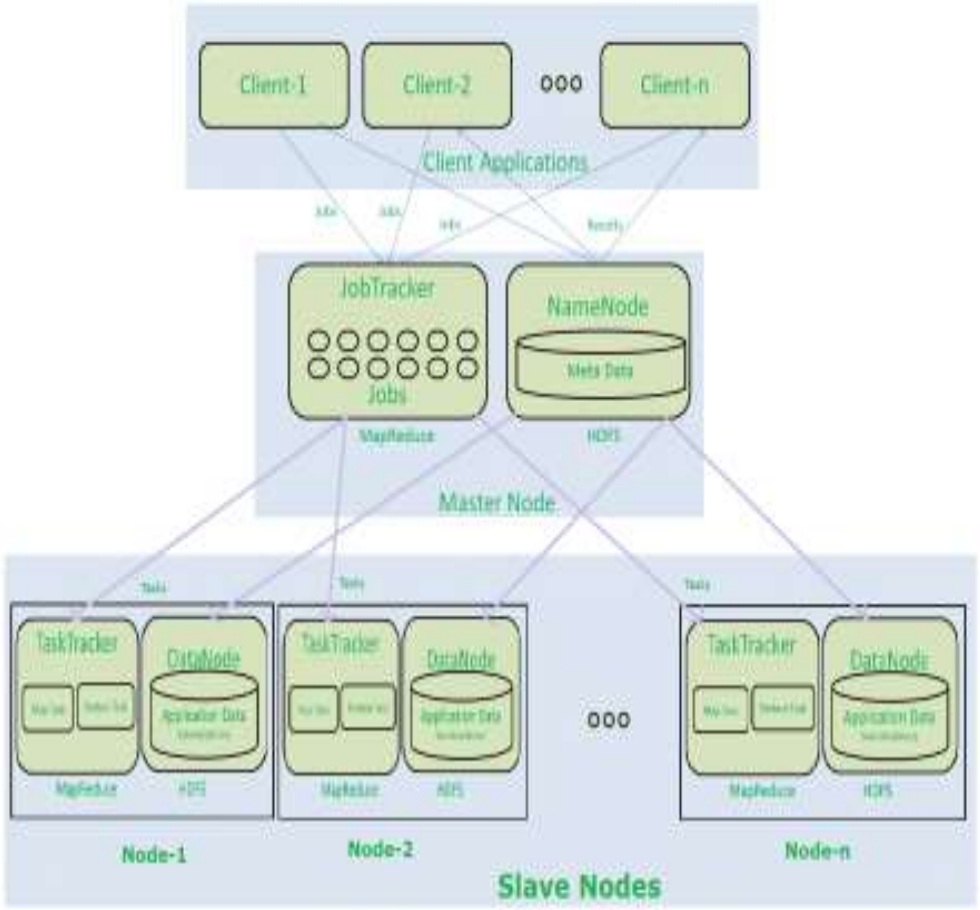
Hadoop 1.x Components Architecture

- Both Master Node and Slave Nodes contain two Hadoop Components:
  - HDFS Component
  - MapReduce Component
- Master Node's HDFS component is also known as "Name Node".
- Slave Node's HDFS component is also known as "Data Node".
- Master Node's "Name Node" component is used to store Meta Data.
- Slave Node's "Data Node" component is used to store actual our application Big Data.
- HDFS stores data by using 64MB size of "Data Slots" or "Data Blocks".
- Master Node's MapReduce component is also known as "Job Tracker".
- Slave Node's MapReduce component is also known as "Task Tracker".
- Master Node's "Job Tracker" will take care assigning tasks to "Task Tracker" and receiving results from them.
- Slave Node's MapReduce component "Task Tracker" contains two MapReduce Tasks:
  - Map Task
  - Reduce Task

  We will discuss in-detail about MapReduce tasks (Mapper and Reducer) in my coming post with some simple End-to-End Examples.

- Slave Node's "Task Tracker" actually performs Client's tasks by using MapReduce Batch Processing model.
- Master Node is a Primary Node to take care of all remaining Slave Nodes (Secondary Nodes).

Hadoop 1.x Components In-detail Architecture



Hadoop 1.x Architecture

Hadoop 1.x Architecture Description

- Clients (one or more) submit their work to Hadoop System.
- When Hadoop System receives a Client Request, first it is received by a Master Node.
- Master Node's MapReduce component "Job Tracker" is responsible for receiving Client Work and divides into manageable independent Tasks and assign them to Task Trackers.
- Slave Node's MapReduce component "Task Tracker" receives those Tasks from "Job Tracker" and perform those tasks by using MapReduce components.
- Once all Task Trackers finished their job, Job Tracker takes those results and combines them into final result.
- Finally Hadoop System will send that final result to the Client.