Assignment 3.3: Components of Hadoop 2.x

Below are the three major components of Hadoop 2.x

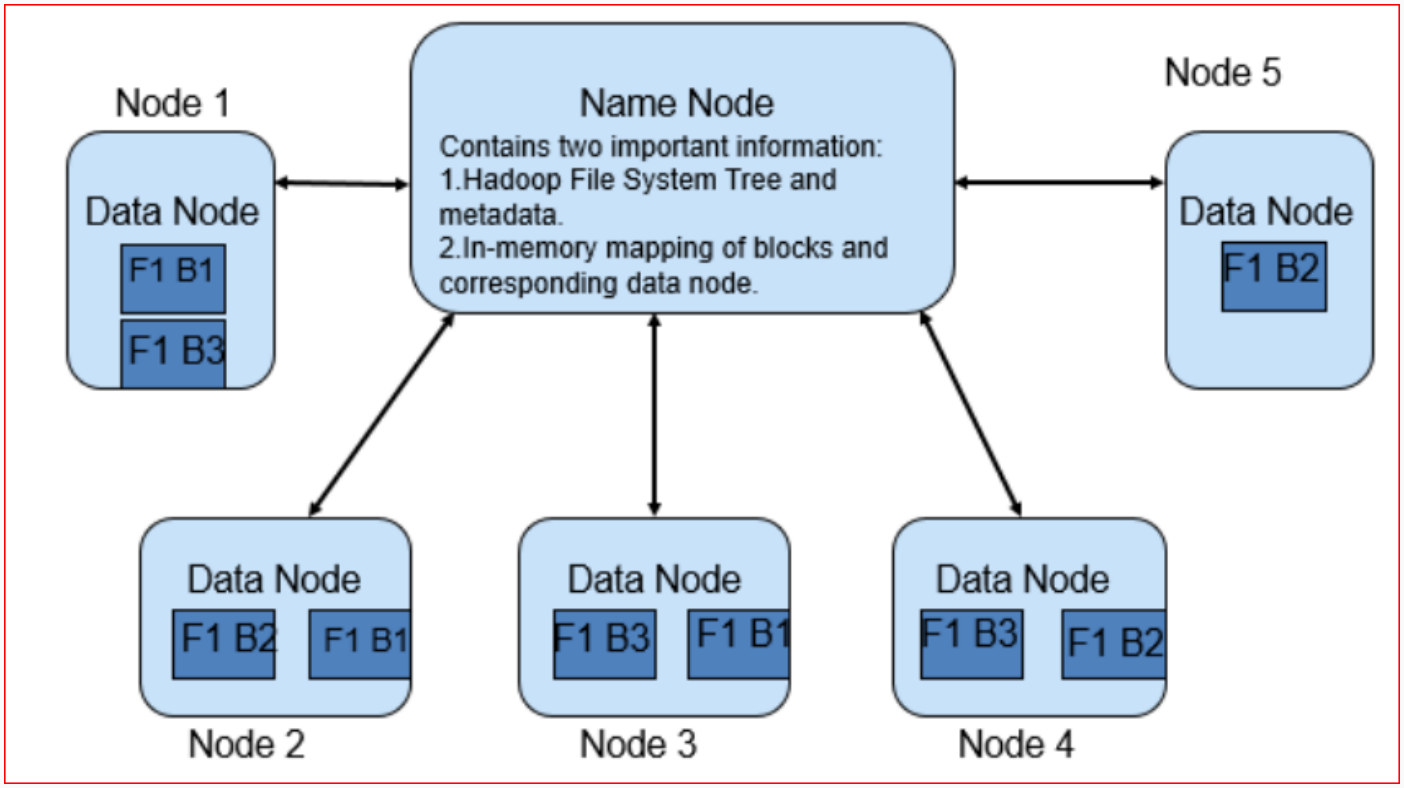
a. Hadoop Distributed File System

b. MapReduce

c. YARN

a. Hadoop Distributed File System

HDFS is the primary storage system of Hadoop. Hadoop distributed file system (HDFS) is java based file system that provides scalable, fault tolerance, reliable and cost efficient data storage for big data. HDFS is a distributed filesystem that runs on commodity hardware. HDFS is already configured with default configuration for many installations. Most of the time for large clusters configuration is needed. Hadoop interact directly with HDFS by shell-like commands.



Components of HDFS:

i. NameNode

It is also known as Master node. NameNode does not store actual data or dataset. NameNode stores Metadata i.e. number of blocks, their location, on which Rack, which Datanode the data is stored and other details. It consists of files and directories.

Tasks of NameNode

- Manage file system namespace.
- Regulates client's access to files.
- Executes file system execution such as naming, closing, opening files and directories.

ii. DataNode

It is also known as Slave. HDFS Datanode is responsible for storing actual data in HDFS. Datanode performs read and write operation as per the request of the clients. Replica block of Datanode consists of 2 files on the file system. The first file is for data and second file is for recording the block's metadata. HDFS Metadata includes checksums for data. At start-up, each Datanode connects to its corresponding Namenode and does handshaking. Verification of namespace ID and software version of DataNode take place by handshaking. At the time of mismatch found, DataNode goes down automatically.

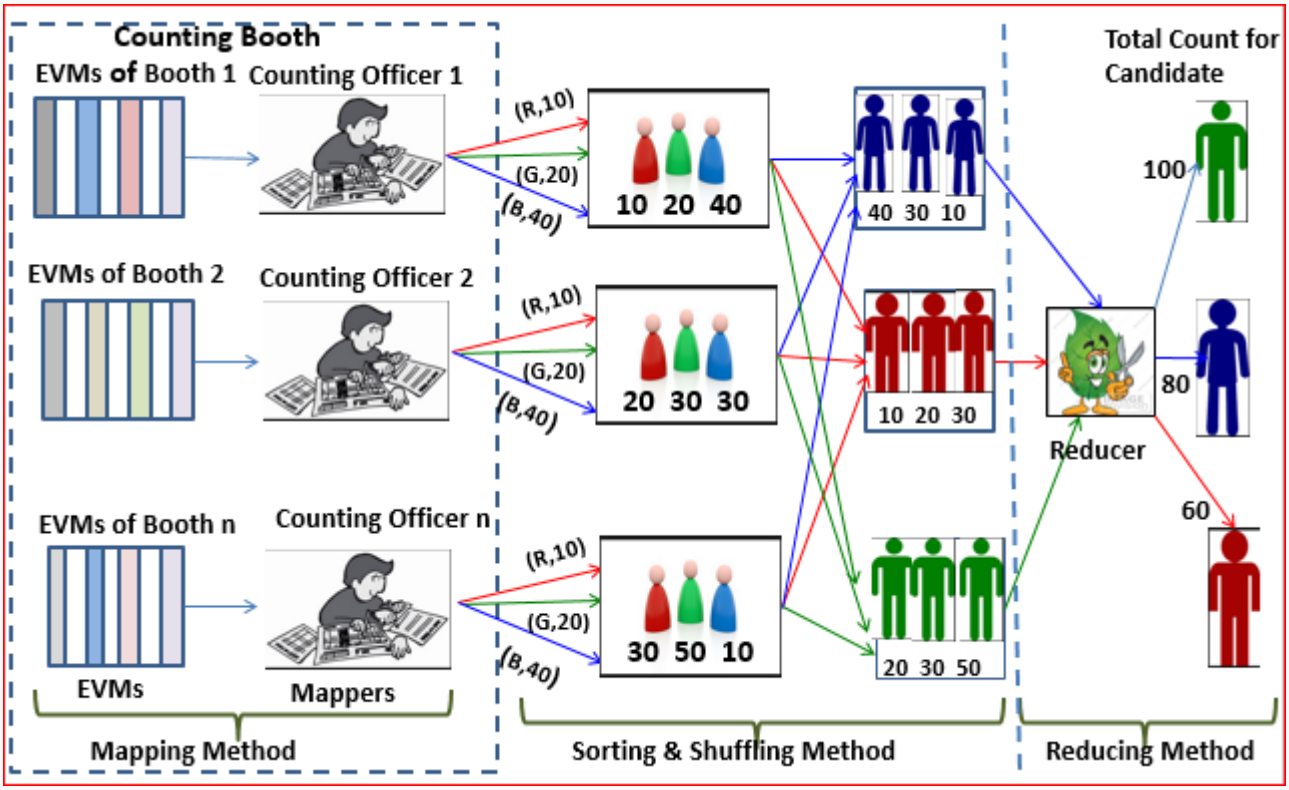
Tasks of DataNode

- DataNode performs operations like block replica creation, deletion and replication according to the instruction of NameNode.
- DataNode manages data storage of the system.

b. MapReduce

Hadoop MapReduce is the core component of hadoop which provides data processing. MapReduce is a software framework for easily writing applications that process the vast amount of structured and unstructured data stored in the Hadoop Distributed File system.

Hadoop MapReduce programs are parallel in nature, thus are very useful for performing large-scale data analysis using multiple machines in the cluster. By this parallel processing, speed and reliability of cluster is improved.



Working of MapReduce

MapReduce works by breaking the processing into two phases:

- Map phase

- Reduce phase

Each phase has key-value pairs as input and output. In addition, programmer also specifies two functions: **map function** and **reduce function**

Map function takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

Reduce function takes the output from the Map as an input and combines those data tuples based on the key and accordingly modifies the value of the key.

Features of MapReduce

i. Simplicity

MapReduce jobs are easy to run. Applications can be written in any language such as java, C++, and python.

ii. Scalability

MapReduce can process petabytes of data.

iii. Speed

By means of parallel processing problems that take days to solve, it is solved in hours and minutes by MapReduce.

iv. Fault tolerance

MapReduce takes care of failures. If one copy of data is unavailable, another machine has a copy of the same key pair which can be used for solving the same subtask.

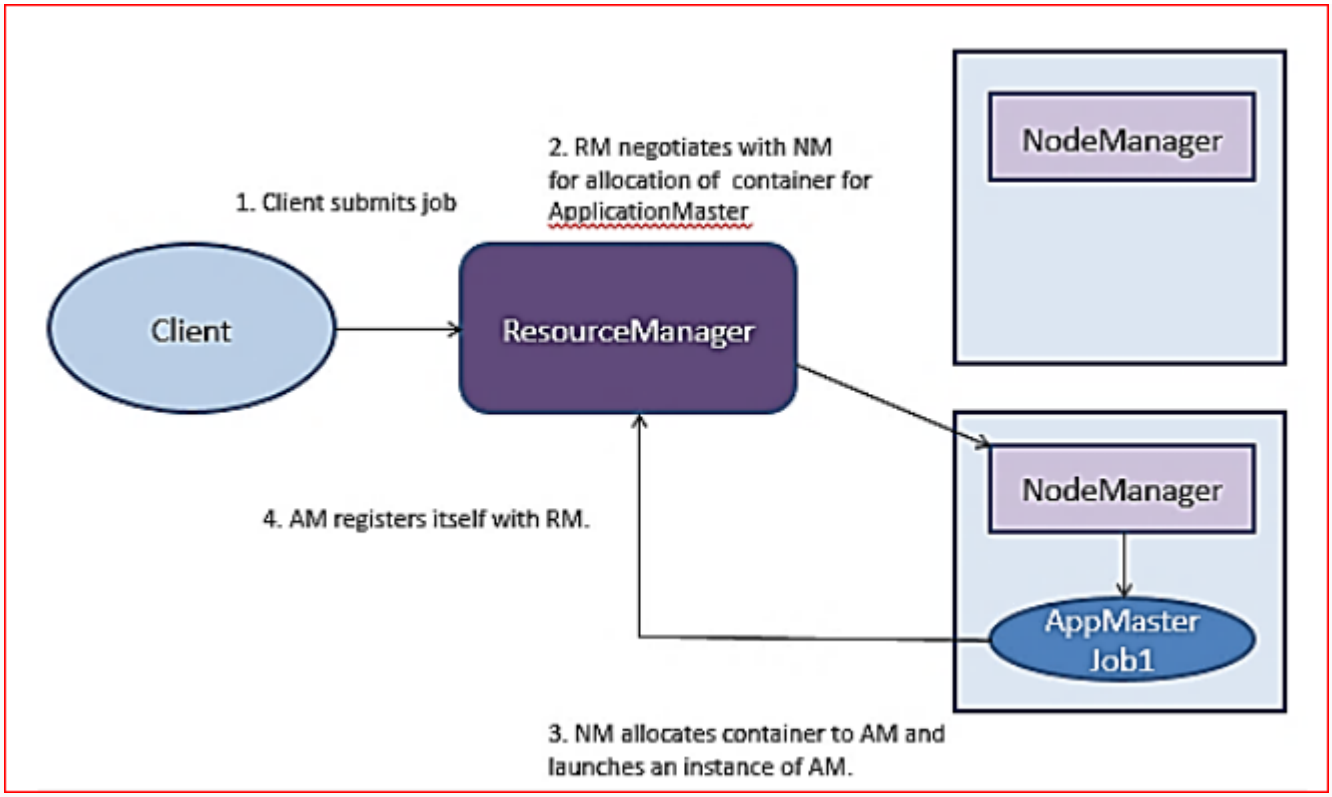
c. YARN

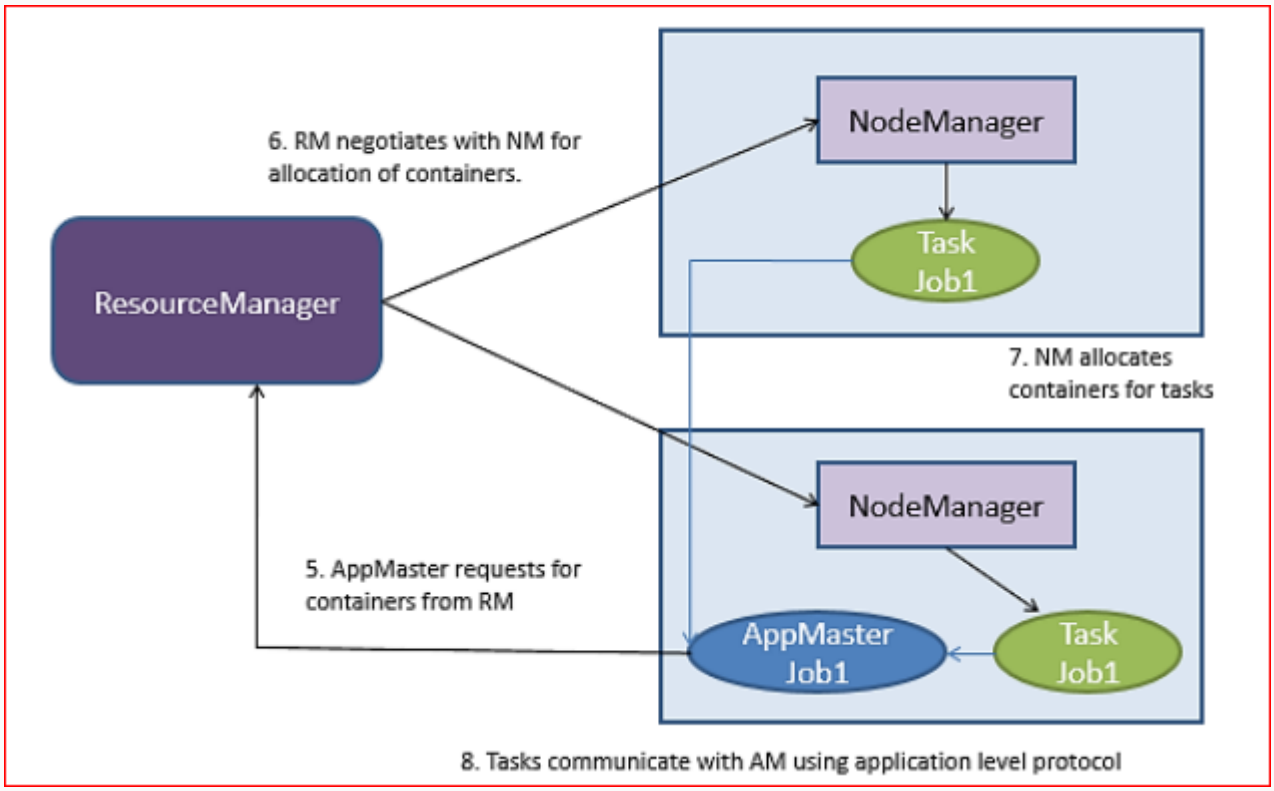
YARN provides the resource management. YARN is called as the operating system of hadoop as it is responsible for managing and monitoring workloads. It allows multiple data processing engines such as real-time streaming and batch processing to handle data stored on a single platform.

YARN has been projected as a data operating system for Hadoop2. Main features of YARN are:

- **Flexibility:** Enables other purpose-built data processing models beyond MapReduce (batch), such as interactive and streaming. Due to this feature of YARN, other applications can also be run along with Map Reduce programs in hadoop2.
- **Efficiency –** As many applications can be run on same cluster, efficiency of Hadoop increases without much effect on quality of service.
- **Shared –** Provides a stable, reliable, secure foundation and shared operational services across multiple workloads. Additional programming models such as graph processing and iterative modeling are now possible for data processing.

Apache Yarn Framework consists of a master daemon known as “Resource Manager”, slave daemon called node manager (one per slave node) and Application Master (one per application).





i. Resource Manager (RM)

It is the master daemon of Yarn. It manages the global assignments of resources (CPU and memory) among all the applications. It arbitrates system resources between competing applications.

Resource Manager has two Main components

- Scheduler
- Application manager

a. Scheduler

The scheduler is responsible for allocating the resources to the running application. The scheduler is pure scheduler it means that it performs no monitoring no tracking for the application and even doesn't guarantees about restarting failed tasks either due to application failure or hardware failures.

b. Application Manager

It manages running Application Masters in the cluster, i.e., it is responsible for starting application masters and for monitoring and restarting them on different nodes in case of failures.

ii. Node Manager (NM)

It is the slave daemon of Yarn. NM is responsible for containers monitoring their resource usage and reporting the same to the ResourceManager. Manage the user process on that machine. Yarn NodeManager also tracks the health of the node on which it is running. The design also allows plugging long-running auxiliary services to the NM; these are application-specific services, specified as part of the configurations and loaded by the NM during startup. For MapReduce applications on YARN, a shuffle is a typical auxiliary service loaded by the NMs.

iii. Application Master (AM)

One application master runs per application. It negotiates resources from the resource manager and works with the node manager. It Manages the application life cycle.

The AM acquires containers from the RM's Scheduler before contacting the corresponding NMs to start the application's individual tasks.