**Assignment 3.1**

**Major components of Hadoop 2.x are:**

1. **HDFS :** HDFS is designed using distributed File system, It can run on commodity hardware, It is reliable, fault tolerant and design using low cost hardware, It is suitable for distributed storage and Processing.It provides High performance data access.

It follows the Master slave architecture**:**

1. **Name Node:** It acts as master, it manages the file system namespace, regulate client access to file and executes file system operations such as renaming, closing, and opening files and directories
2. **Data Node**: It is a slave in HDFS Architecture. It perform read-write operations on the file systems, as per client request and perform operations such as block creation, deletion, and replication according to the instructions of the name node.

In HDFS data is splitted into blocks and stored across the multiple nodes of cluster, Block size in Hadoop 2.x is 128 bytes which is configurable. In HDFS blocks are replicated across cluster of nodes .The default replication factor is 3 which is configurable.

Read and write operations performed in HDFS as follows:

To Read a file from HDFS client needs to interact with name node as it is the single place where metadata of file is present, name node gives the address of data nodes where the block is present and data nodes are sorted according to their proximities to the client. Client will interact with respective data nodes.

To write a file into HDFS client needs to interact with name node. Name node will provide the address of data node where client will start writing a data. As soon as client finish writing the data, the data node starts copying the data into another data node which in turn copies the data into another data node .After required replicas complete it will send the acknowledgement to client

1. **Map Reduce** : It is a processing engine in hdoop. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. 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
2. **YARN(Yet Another Resource Negotiator)**: It is a tool that enable other data processing frameworks to run on Hadoop. A more substantive take on YARN would describe it as a general-purpose resource management tool that can enabled to schedule and assign CPU cycles and memory (and in the future, other resources, such as network bandwidth) from the Hadoop cluster to waiting applications. YARN raises exciting possibilities. Singlehandedly, YARN has converted Hadoop from simply a batch processing engine into a platform for several distinct modes of data processing, from traditional batch to interactive queries to streaming analysis.

**Anatomy of YARN**

To run an application on YARN, a client contacts the resource manager and asks it to

run an application master process. The resource manager then finds a node manager that can launch application master in container. Precisely what the application master does once it is running depends on the application.It could simply run a computation in the container it is running in and return the result to the client. Or it could request more containers from the resource managersand use them to run a distributed computation

.