Hadoop

In the previous module, you learnt about the concept of Big Data and its characteristics. The Four V's namely volume, velocity, variety and veracity of Big Data were discussed in detail. Due to the enormous size of Big Data, data management and processing requires different infrastructure as compared to the traditional ones. In this module, you learnt about one of the popular Big Data processing platforms - Hadoop. For an analyst working on Big Data, it is important to be acquainted with the architecture and functioning of Hadoop. Over time, Hadoop has been redeveloped - Hadoop 2.0 to overcome some of the limitations of its initial version - Hadoop 1.0. The architecture of Hadoop 2.0 has been discussed in detail in this module.

Origin and Characteristics of Hadoop

History of Hadoop

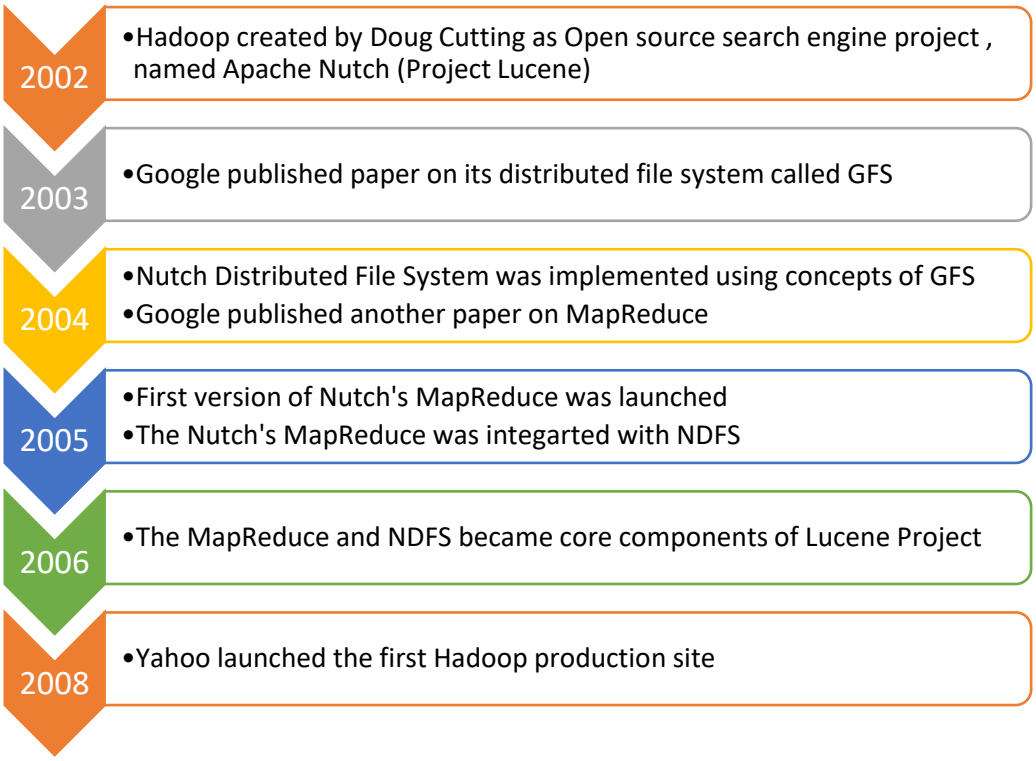

Figure 1: History of Development of Hadoop

Hadoop is a framework for processing big data. It is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It uses the fact that a large number of commodity machines can be stringed together to process Big Data. You should be familiar with the terms "commodity machine" and "cluster" to understand the concept of Hadoop in detail.

1. **Commodity Machine** is any server/machine/hardware with the standard hardware, which has not been customized for huge data processing. For example: the laptops, desktops etc.
2. A **cluster** is a collection of commodity machines which are interconnected to each other through a network. For example: The PCs in a cyber cafe are connected to each other and hence can be called a cluster.

Working Principle of Hadoop

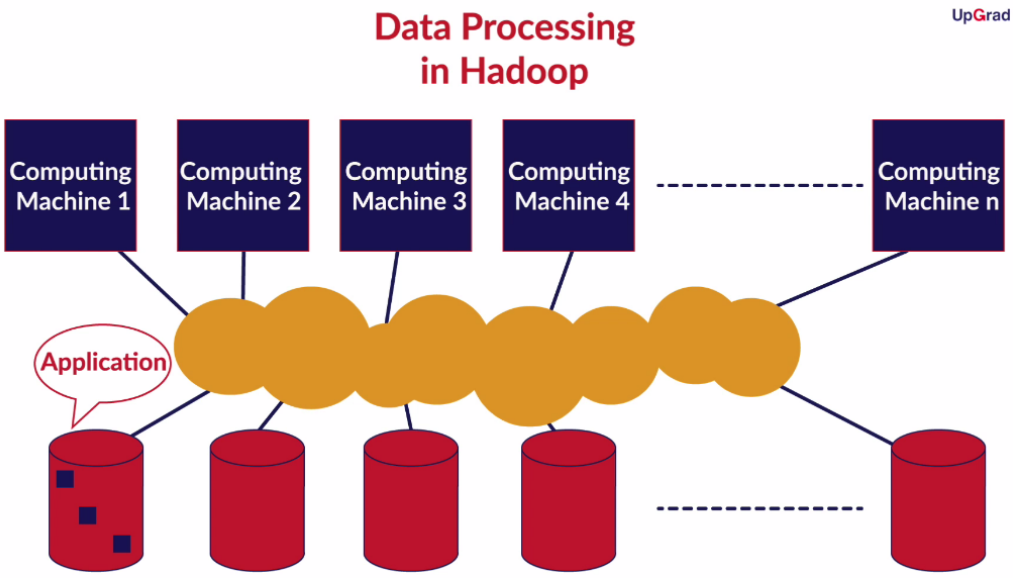

Figure 2: Data Processing in Hadoop

Hadoop is developed to process Big Data. Computation of enormous amount of data on one machine may be time-consuming. Hence, Hadoop works on the principle that the computations for data processing should be done close to the data. In other words, the processing of data should be done on respective cluster machines where the data resides. This is known as the principle of data locality.

Architecture of Hadoop – An Overview

A Hadoop cluster is primarily made up of two components (refer to the figure below):

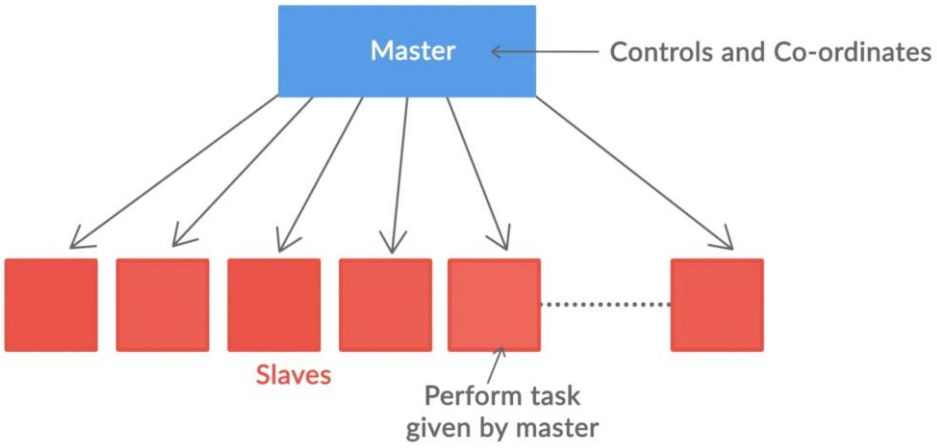

Figure 3: Components of Hadoop cluster

As the figure above shows, the Hadoop cluster is made up of master node(s) and multiple slave nodes.

- The **master node** book keeps the information about the files scattered over cluster. In addition, it also maintains information about the progress of the job done by the different slave nodes.

- Every **slave node** performs the task it is allocated by the master node.

Building blocks of Hadoop

The basic version of Hadoop is made up of three major building blocks:

- **HDFS**: Hadoop Distributed File System is responsible for storage of data using distributed environment. It is the storage system of Hadoop. HDFS stores data on different machines (also known as nodes) of the cluster.
- **YARN**: Yet Another Resource Negotiator (YARN) is the resource management and allocation layer of Hadoop. It ensures that each node has the sufficient resources to process the data.

- **MapReduce**: It is the processing engine of Hadoop. All the data processing happens as per the concepts of MapReduce.

Architecture of Hadoop 2.0

In Hadoop 2.0, the MapReduce is dedicated for data processing and YARN (Yet Another Resource Negotiator) is dedicated for the resource management.

Hadoop 2.0 mainly has three components –

- HDFS (Hadoop Distributed File System)

- YARN (Yet Another Resource Negotiator)
- MapReduce

HDFS

Hadoop Distributed File System has four types of nodes – name node, standby node, secondary node and data node.

The name node is the master node of the Hadoop cluster that stores the location of blocks. So, whenever a client wants to access a block, it is the name node that is contacted. If the name node fails, the standby node takes over the charge instantaneously. It also has a secondary Node which periodically takes the snapshot (image) of the information stored in name node.

Hadoop stores huge amount of data, which requires an appropriate storage, without slowing down the processing. The HDFS takes care of this aspect and hence it is an important component in Hadoop architecture.

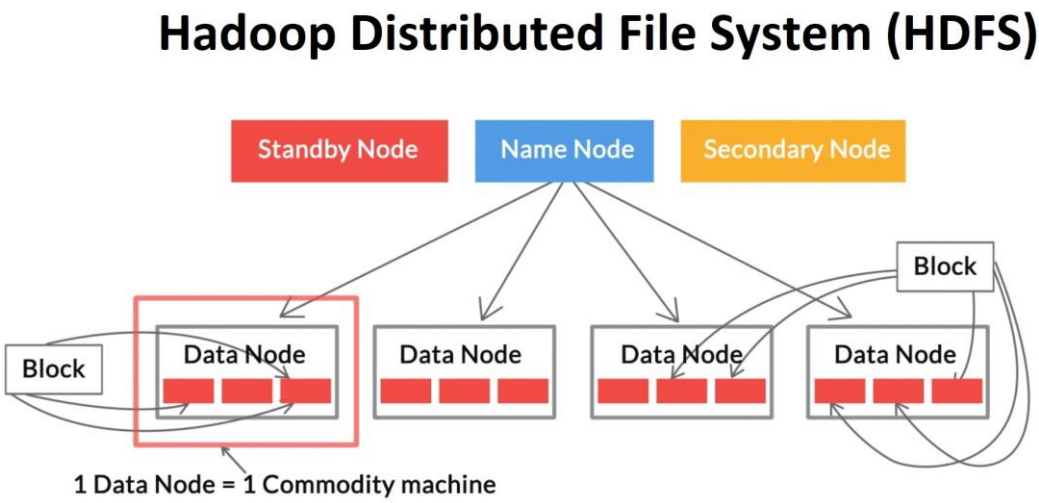

Figure 4: Data storage in HDFS

HDFS breaks down the data (to be stored) into chunks called "blocks". These blocks are stored as independent units on the disks (of the data nodes) and hence can be accessed individually. The size of the block is kept such that it is not too small to avoid too many block creations. On the other hand, the block size is not kept too large to avoid slowdown of data processing.

Advantages of HDFS Blocks

1. There is no requirement to store all the blocks of a file on the same disk. Therefore, the blocks can be stored on different commodity machines (also known as nodes) in the cluster. This enables the HDFS to execute multiple maps to process chunks of data parallelly, rather than the entire file being executed by a single host.

2. To insure against corrupted blocks and node failure, each block is replicated to a small number of physically separate machines. Generally, at-least one of the replicas is maintained in the same "rack" which ensures ease of data access in case the original node fails. Another replica is maintained in a different rack to ensure data availability in case of rack failure. Thus, blocks fit well with replication for providing fault tolerance and availability.

Features of HDFS

Hadoop file system comprises of several components. These components are connected to each other through links. It might be possible that one of the components or the link connecting two components fails during the operation. HDFS is designed in a way so that it can tolerate faults and failures as well as save the data.

Some of the faults that HDFS deals with are:

Data Node failure: If a data node fails, the replicas of that block stored somewhere else in the cluster will be accessed.

Rack failure: In Hadoop terminologies, you learnt that the data nodes in Hadoop cluster are arranged in the form of racks. If one of the racks fails, all the data nodes in the rack get isolated. In the case of rack failure, name node finds the data nodes in other racks with the same set of data and assigns the task to them.

Name Node failure: In case a name node fails, the secondary/standby name node immediately takes up the job of name node.

The advantages of HDFS are:

1. **Scalability:** This method of distributing data among different machines is easily scalable because as the data grows, you just have to add more nodes to the cluster to increase its storage and processing capacity.
2. **Fault tolerance:** If the name node fails, you have a backup node to take over and keep the cluster alive. In case a few data blocks get corrupted, the cluster will still function; unlike a single computer, where if a part of the file gets corrupted, the whole file becomes unavailable.
3. **Low-cost implementation:** Hadoop runs on commodity machines, which are low-cost computers, which leads to a lower implementation cost.
4. **Flexibility:** Hadoop can process any kind of data (structured, semi-structured and unstructured data), and is not restricted to only structured data — unlike the traditional databases — which makes Hadoop more flexible.

YARN – Yet Another Resource Negotiator

YARN is the resource management layer of Hadoop 2.0.

YARN has three important components:

1. **Resource Manager**: As slaves run multiple tasks, they need different resources for the execution. The necessary resources are allocated by resource manager. In addition, it keeps track of the nodes and the applications running on a cluster.

   The resource manager has two main components:

   Scheduler: The scheduler is responsible for allocating resources to the various running applications.

   Applications Manager: The applications manager is responsible for accepting job-submissions, negotiating the first container (resources) for executing the application specific Application Master.

2. **Node Manager**: The node manager keeps track of all the applications running on a slave machine, the resources allocated for each of the applications, and communicates the status of the node to the Resource Manager.

3. **Application Master**: The application master executes and monitors the resources of a container (the place/memory space where the actual execution of work occurs) for a task assigned to it on a slave machine. It is also responsible for negotiating resources from the resource manager.

Following steps are involved:

1. The Client submits the job to the applications manager which is a part of the resource manager.
2. The applications manager then contacts the name node to get the location of the data nodes which have the data relevant to the job.
3. The resource manager with the help of one of its component (called scheduler), schedules the job on these nodes.
4. The application master on nodes starts negotiating for the necessary resources (like memory, data blocks, CPU etc) with the resource manager.
5. The application master starts the execution upon getting the required resources.
6. The node manager looks after performance of node on which it is running (Specifically CPU and RAM usage etc.)
7. The application master reports the results after execution to the resource manager.

MapReduce

MapReduce is the programming framework which is used to process the data.

MapReduce has two components, a map phase and a reduce phase. The function of **map phase** is to convert the incoming data (stored in blocks) into **key-value** pairs. These pairs are then fed into the **reduce phase**. The function of reduce phase is to aggregate these values on the basis of keys across ALL the blocks in the cluster.

Hadoop Ecosystem

The Hadoop ecosystem comprises of the tools and services which enable Hadoop to blend its working with the existing systems.

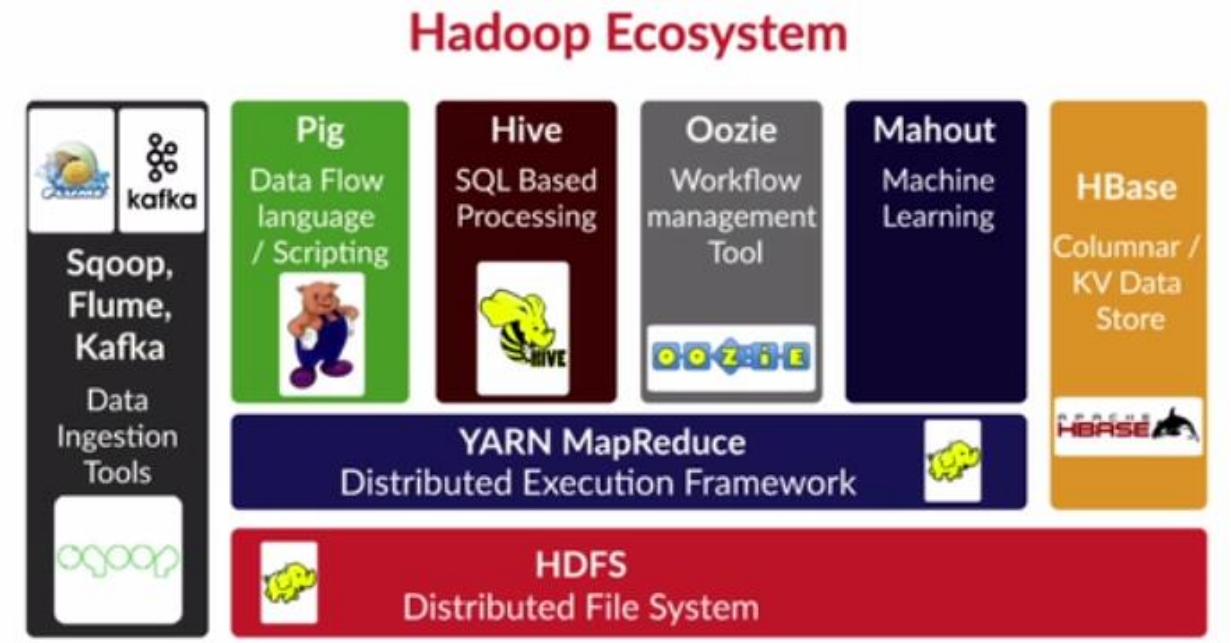

Figure 6: Hadoop Ecosystem

Sqoop, Flume and Kafka: These tools are designed to efficiently transfer bulk data between Hadoop and structured and unstructured data stores.

Hive: It helps in accessing and analysing data in Hadoop with a query language like SQL. HIVE-QL is another query language introduced for this purpose.

There are many other tools such as oozie, mahout hbase etc., which are not being covered in the course.

Hadoop Vendors

Hadoop, being an open source tool, can be pulled into any company's production setup. However, the support and maintenance of the cluster are non-trivial and hence there are vendors who commercially ship these distributions for people to use, such that the maintenance overhead are adequately taken care of. Some of the Hadoop Vendors are:

cloudera	CDH
Hortonworks	Sandbox
MAPR	MapR Hadoop
IBM	BigInsights
hp	HP Hadoop
ORACLE	Oracle Hadoop

Figure 7: Hadoop Vendors

Industry applications of Hadoop

Spearheaded by companies like Yahoo!, there is a technology Boom in the industry. Every company, irrespective of their size has been considering the integration of Hadoop with their data analysis. Nearly all sectors are using Hadoop for improvement of their respective businesses. Here are a few areas where Hadoop is used for data analysis:

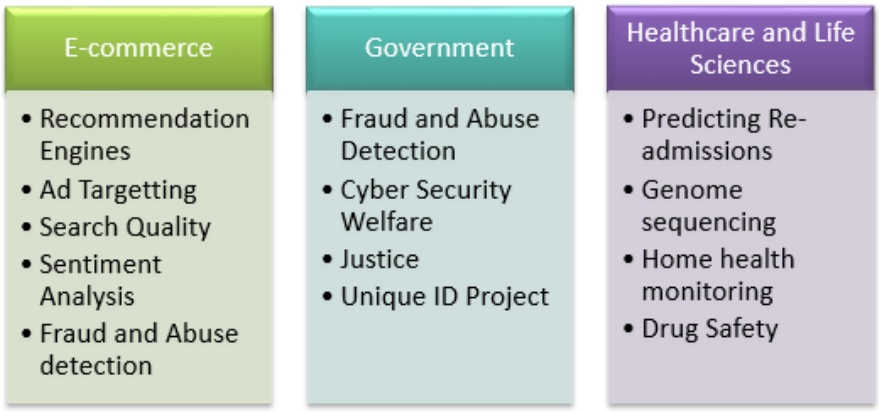

Figure 8: Industry Applications of Hadoop

Architecture of Hadoop 1.0

In case you are curious, let's take a look at the architecture of Hadoop 1.0.

Hadoop 1.0 has two building blocks namely **HDFS** and **MapReduce**.

HDFS

The HDFS works with 3 types of nodes (refer to the image below):

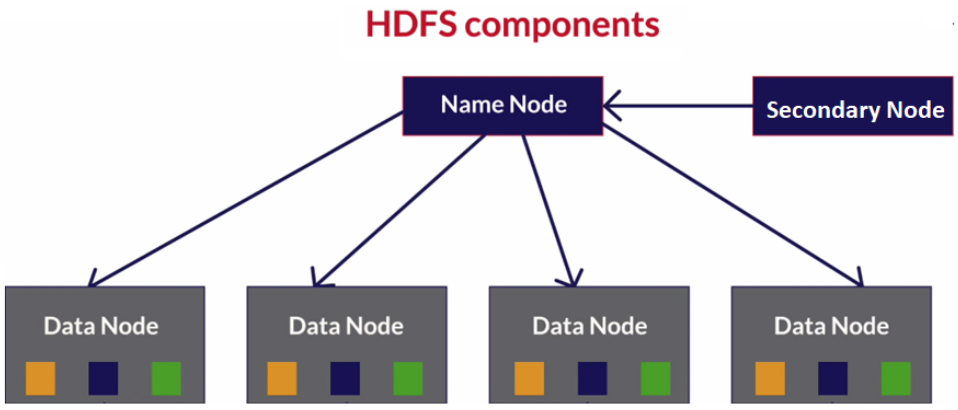

Figure 9: Different Nodes in HDFS

Name Node: The name node is hosted on the master node. Some of its important activities include:

- It keeps track of the nodes executing a job and the storage space available on each slave node.
- It also maintains the file system image - an organisation of files and directories hosted in Hadoop cluster, i.e., it stores the location of blocks stored in the Hadoop cluster.

Data Node: These are the nodes which store the data on their storage devices. In a Hadoop cluster, many data nodes are available. All the data nodes receive the instructions from the Name Node.

Secondary Node: The secondary node maintains a File System Image (FSI) of the name node and the edit log by updating it periodically. Then it merges the edit log with FSImage. In other words, the secondary node asks for updates in edit log to the name node and updates the changes for every block. Thus, a consistent copy of complete file system is maintained by the secondary node. This copy is then used to start a new name node.

Note:

- HDFS in Hadoop 1.0 is a Single Point of Failure i.e. SPOF as the name node is the only machine which has complete information about everything. So, if the name node fails, the entire cluster goes down till a new name node is brought live.

MapReduce

The second component of Hadoop 1.0 architecture is the MapReduce. The MapReduce is the execution engine of Hadoop. Its duty is to get the jobs executed. There are two main components of MapReduce:

Job Tracker: It is hosted inside the master (i.e. the name node) and receives the job execution request from the client. Its main duties are to break down the received job i.e. big computations into small parts, allocate the partial computations i.e. tasks to the slave nodes, monitor the progress and the report of task execution from the slave.

Task Tracker: It is present on the slave machine (i.e. data node). As there are multiple slave machines, many task trackers are available in a cluster. Its duty is to perform computation given by Job Tracker on the data available on the slave machine. The task tracker communicates the progress and reports the results to the Job Tracker.

When a client submits a job request, the following steps take place in Hadoop to execute the task:

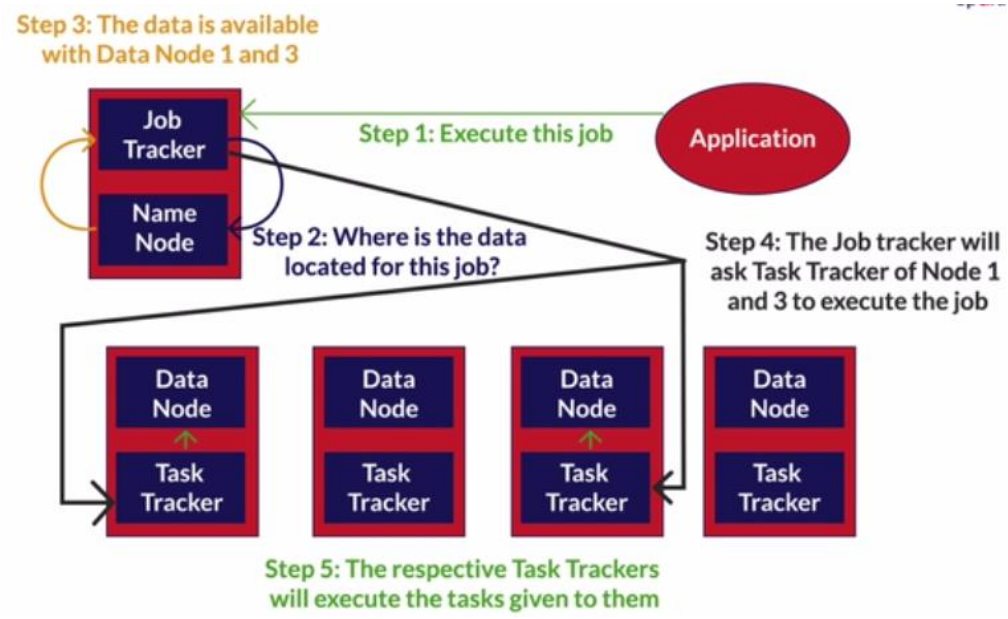

Figure 10: Steps of job execution in Hadoop 1.0

1. The client submits the job to Job Tracker
2. Job Tracker asks Name node the location of data
3. As per the reply from name node, the Job Tracker asks respective task trackers to execute the task on their data
4. All the results are stored on some Data Node and the Name Node is informed about the same.
5. The task Trackers inform the job completion and progress to Job Tracker.
6. The Job Tracker inform the completion to client
7. Client contacts the Name Node and retrieve the results

Note: Whenever the task tracker is assigned a new job by the job tracker, it launches a new process in the machine to keep track of the progress of the job. This is done to maintain isolation between two different jobs running on the same slave.

Limitations of Hadoop 1.0

- With only name node (and no standby node), Hadoop 1.0 has a Single Point of Failure, implying that if the master node fails, the entire system fails.

- The MapReduce Component is responsible for both data processing as well as resource management which puts a lot of pressure on the slave machines.