BIG DATA ANALYTICS

ASSIGNMENT 1: Study of Hadoop

(Amitabh Saini, TY-E-4, 141541)

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

* Setup:

CLUSTER

WORKER1

MASTER

WORKER4

WORKER3

WORKER2

WORKER5

Master: Performs Data Distribution

Worker: Performs Data Processing

The interconnected network of machines together is called a **Cluster**. A Cluster can be of two types: **Single node Cluster** (Worker and Master on a single machine) and **Multi node Cluster** (Worker and Master on various machines).

* Hadoop provides 2 actions :
* Data Processing
* Data Distribution
* Data Processing:
* Hadoop processes data by the method of Map Reducing. Map pertaining to the division of work into small number of tasks and Reducing relating to the output collection of each worker and combining them together.
* It is the way of processing data in parallel.
* Data Distribution:
* A file is divided into blocks which are small chunks of memory. It is cut as per records by master.
* Master puts each block in worker nodes independently. He keeps record as to where he has put the block of file in a node.
* This distribution is called Hadoop’s distribution of data and the file system is called Hadoop’s Distribution File System (HDFS).
* It is the way Hadoop puts the data by divided file into blocks and placing them into the nodes.
* Block size may vary in different Hadoop services, for eg. Apache Hadoop has block size of 64MB and CloudEra’s Hadoop has block size of 256MB.
* Workers are the **Data nodes**. They are limited to data. Master is known as **Name node**. It will never process data but will make it processed by data nodes, i.e. machines.

* Issues
* Data loss
* Worker/s is slow
* Lack of monitoring
* Uneven distribution of data
* Inter dependency
* Synchronization issue
* Master over occupied
* Lack of resources on worker
* Worker goes down
* Lack of processing ability of worker
* Master fails
* Ordering issue
* To avoid **Data loss**, data is replicated. It is converted to various blocks and stored in different workers. The number of times data is replicated is called Replication factor. By default, Replication factor is 3. It is configurable, i.e. we can change it anytime we want.
* To avoid **Uneven distribution of data**, the distribution of data, in which a file can be divided to blocks is limited to some number. It is 64 MB for Hadoop (Apache) and Cloud Era; 128 MB for HottonWorks; 256 MB for MapR. It is also configurable. Hence, there is even distribution of data and **Inter dependency** will never occur.
* **Master fails:** To solve this issue, we have a backup name node, called as a Secondary Name node. Its role is to take a backup from Primary Name node and keep itself ready when any master fails.
* Name nodes (Master) have a personal assistant known as the **Job tracker**, and the Data nodes (Worker) have a personal assistant known as the **Task tracker**.
* **Job tracker** plays an important role in data processing. It finds out what data is to be processed and interacts with the Name node and then, finds out all the information regarding the location and size. Thus, it creates small tasks of the big task.
* The user opens the file containing the data to be processed, then that file goes to the Job tracker, it interacts with the Name node and after finding out all the information regarding location, size, etc. it divides the big task into small number of tasks and gives it to Task tracker. Then, the Task tracker checks the availability of data node and puts tasks into queue, which were assigned by the Job tracker. And hence, the output of data node is propagated back to master and then to the end user.
* **Task tracker** gives incremental updates of processing to the Job tracker, for each task the workers are executing. Job tracker combines the updates of the tasks and conveys to the user, as to how much the job is completed.
* **Lack of Resources (on Workers):** To avoid this issue, the Job trackers ensures that if a task is taking a lot of time, it kills the task and retasks it on the node. It checks three times whether the job is successful or failed. Thus, as Job tracker is involved, the problem of **Master being over occupied** is resolved.
* Worker (Data nodes) sends heartbeats (signals) every 3 seconds to the Master node. The signal contains checksums as to how many are corrupted or failed. Thus, the issue of **Worker goes down** is fixed because the signals are being sent every 3 seconds to keep a check whether worker is alive or not. If any data node fails to send heart beat 3 times, the Job tracker wipes all the data in that node and considers it as dead. Hence, **Synchronization issue** is also solved. Since, Job tracker is available to keep a check at all times, there is no chance of **Lack of monitoring**.
* **Lack of processing of worker: Mutual exclusion** allows this issue to be solved. It works on the principle that if a work needs to be done fast, you make a competition out of it. Hence, creating a race condition between various processors for the same task makes the job to be done fast. Thus, this issue can be solved but it has both pros and cons. As data is divided into blocks and work is done in synchronization, **Ordering issue** is also solved.
* **Demons (Working for Hadoop):**

Name node,

Secondary Name node,

Data node,

Job tracker; and

Task tracker

* **Data Locality:** Hadoop follows Data locality, it means that code is moved to data, data is not moved. The Job tracker takes code to data, across systems and block is not moved. Hence, code is not local.