A review on HDFS Architecture in Hadoop and Hadoop Components

[1]Mrs Usha G.R., [2]Dr. Dharmanna L.

[1]Asst. Professor, [2]Professor & Head
Departement of Information Science
SDM Institute of Technology, Ujire – 574 240, INDIA

***Abstract*-As Big data is demanded, Hadoop has discovered as a popular tool. The big data management in various enterprises, Hadoop is playing an important role. Apache Hadoop is an open source distributed computing framework for storing and processing huge datasets distributed across many clusters. Apache Hadoop is having its specific components to store and analyze the variety of datasets. Principal part of this paper presents the Hadoop Distributed File System (HDFS) architecture in that how DataNodes and NameNode have communication to achieve fault-tolerance and about the MapReduce. MapReduce and Hadoop HDFS are the two components used by Hadoop to enable various types of operation on its platform.**

***Keywords*: Big data, HDFS, MapReduce components, Hadoop components**

I. INTRODUCTION

An open-source software framework written in Java, i.e., Apache Hadoop it consents for the distributed processing of large data sets across clusters of computers using simple programming models. Apache Hadoop is designed in such a way that to gauge up from single servers to thousands of computers, each contribution as local computation and storage. Bijesh Dhyani and Anurag Barthwal stated that Hadoop permits processing of huge data with less cost, Industry-standard servers that both store and process the data, and can scale without limits through distributed parallel processing. [1]. Ashwini A. Pandagale & Anil R. Surve explained that how Hadoop Framework has gained so much popularity as it is a remedial solution for large data sets [2]. Navya Francis et al. [3] proposed that Hadoop is a tool consists of two main concepts MapReduce and HDFS to handle big data to work faster than other traditional database operations. MapReduce concept easily programmed within a short period of time using Pig (parallel execution of data flows) and Hive (to retrieve big data from Hadoop clusters).

Veersetty Degade et al. [4] analysis the weather data using Apache Hadoop and Apache Spark on pseudo node and Hadoop Distributed Multi node cluster**,** compare the performance and gives the final decision that Apache Hadoop is the best technique to process the weather data with high performance. Altti Ilari Maarala et al. [5] presents for low latency traffic flow analysis based on real-time, high velocity data, a Big Data platform can be used effectively. With the experiments carried out, Apache Flume is an suitable technology for ingesting real-time data to Spark Streaming analysis. The author also introduced a Hadoop as Big Data cloud platform with ingestion, study, storage and data query APIs to provide programmable environment for analytics system development and assessment.However, Distributed File System and MapReduce programming are the two main components of Hadoop.

II. THE HDFS ARCHITECTURE

HDFS can be presented as the master and slave architecture. The HDFS master as NameNode and the slave as DataNode.

- **NameNode:** This is an HDFS master sever that manages the file system namespace with adjusts the access i.e., open, close, rename, and more to files by the client. NameNode keeps the metadata i. e. the size, type, namespace information and block information of data. It stores data in main memory as faster accessible and also stored on disk for persistence storage in the disk [3]. It splits the input data into blocks and declares which data block will be store in which DataNode. Negative of NameNode is the whole system will get down, if it gets crashed.

- **Secondary NameNode: The** performing periodic Check point of data is a responsible of the Secondary NameNode. Thus, if the NameNode flops at any time, it can be swapped with a snapshot image stored by the secondary NameNode checkpoints in the storage.

- **DataNode**: It is a slave machine that stores the copies of the partitioned dataset and serves the data as the request comes. DataNode can also perform block creation and deletion.

.

The internal mechanism of HDFS splits the file into one or more blocks; these blocks are kept in a set of data nodes. The default duplication factor is three, the HDFS strategy is to place the first copy on the local node, second copy on the local rack with a different node, and a third copy into different racks with different nodes to achieve fault tolerance. **64 MB** is the block size defined in HDFS. If we required, this can be increased as required.

Block replication

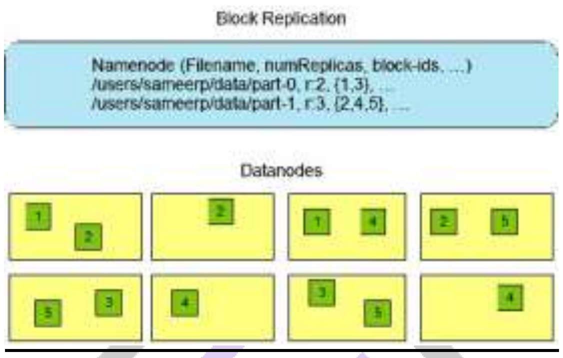
Block Replication
Namenode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1,3}, ...
/users/sameerp/data/part-1, r:3, {2,4,5}, ...
Datanodes
1
2
2
1
4
2
5
5
3
4
3
5
4

Figure 1.NameNode and data nodes appearance

III. COMMUNICATION BETWEEN NAMENODE AND DATANODE

Communication between NameNode and DataNode can be done by sensing Heartbeat signal from DataNodes to NameNode in every 10 minutes once (or periodically) based on settings [6]. NameNode, at regular intervals of time receives a Heartbeat signal and a Block echo from each of the DataNodes in the cluster.

- Reception of a Heartbeat indicates that the DataNode is functioning accurately.
- BlockReport comprises a list of all blocks on a DataNode.

When heartbeat signal sent by DataNode, if it is not received by NameNode after a stipulated time, the data node is pointed as not working (dead). Since blocks will be under simulated the system begins replicating the blocks that were stored on the dead DataNode. The replication of data blocks from one DataNode to another is performed by NameNode. The replication data transfer happens only between DataNodes and the data is never permits through the NameNode. A DataNode stores a block and forwards the request to next block.

DataNode Stores a block in HDFS and it acts as a stage for running a task. It also does block deletion, creation, replication upon the instruction from NameNode. DataNode checks for accurate name, space, id, if found then connect to NameNode, else loses connection and maintains current status of the block in its node meanwhile generate block report. Regular interval of hour DataNode sends block report to name node and henceforth always have updated statistics about DataNode.

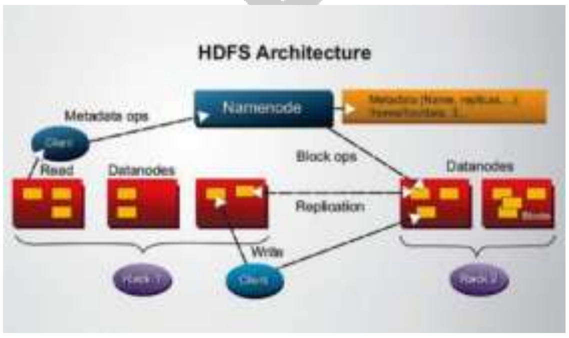

Figure 2: HDFS architecture

- Writing a file into HDFS Client asks NameNode as want to write a file. Name node tells which block is free, so client writes in free block of DataNode. Thus writing file to HDFS is performed.

- Reading a file from HDFS Client interacts with NameNode to get block location. NameNode sends address of block that holds the data and also address of replicated blocks after getting the information. After getting the information, client reads the data from data node [6].

IV UNDERSTANDING MAPREDUCE

A programming model for dealing out large datasets distributed on a large cluster is known as MapReduce. The heart of Hadoop is called MapReduce. Its programming pattern allows performing huge data processing across thousands of servers configured with Hadoop clusters. This has been derived from Google MapReduce.

Hadoop MapReduce is acts as software framework for writing applications effortlessly; with this framework we easily perform large amounts of data (multi-terabyte datasets) in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce paradigm is having two phases, namely, Map and Reduce that mainly deal with key and value pairs of data. The Map and Reduce task run consecutively in a cluster; the output of the Map phase becomes the input for the Reduce phase. These phases are described as follows:

• **Map phase: T**he datasets are assigned to the task tracker to perform the Map phase after the data is distributed across the cluster. The data functional operation will be performed over the data and release the output. The output of the Map phase is mapped key and value pairs.

• **Reduce phase**: The master node then gathers the answers to all the Sub problems and pools them in some way to form the output.

The job submission, job initialization, task assignment, task implementation, growth and status update, and job completion-related activities, which are mainly accomplished by the JobTracker node and executed by Task Tracker is consists of MapReduce. Client application submits a job to the JobTracker. Then input is divided across the cluster. The JobTracker then calculates the number of map and reducer to be processed. It commands the TaskTracker to start executing the job. Then, the TaskTracker copies the resources to a local machine and launches Java Virtual Machine to map and decrease program over the data. Along with this, the TaskTracker periodically sends update to the JobTracker, which can be considered as the heartbeat that helps to update Job ID, job status, and usage of resources.

A. MapReduce components

The master-slave architecture is accomplished by MapReduce, included with the following components:

• **JobTracker**: This is the master node of the MapReduce system, which manages the jobs and resources in the cluster (TaskTracker). The JobTracker tries to planning about each map as close to the actual data being processed on the TaskTracker, which is running on the same DataNode as the underlying block.

JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop platform. There is only One Job Tracker process run on any Hadoop cluster. Job Tracker runs on its own JVM process. In a typical production cluster its run on a discrete machine. Each slave node is configured with job tracker node location. The JobTracker is single point of failure for the Hadoop MapReduce service. All running jobs are froze, If it goes down. JobTracker in Hadoop performs following actions.

- ✓ Applications are submitted to the Job tracker by Client.
- ✓ The NameNode talks with JobTracker to determine the location of the data
- ✓ Location of the TaskTracker nodes with available slots at or near the data is done by The JobTracker.
- ✓ The JobTracker assigns the work to the selected TaskTracker nodes.
- ✓ The monitoring of TaskTracker nodes is frequent. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker.
- ✓ A TaskTracker will report the JobTracker when a task fails to perform. The JobTracker is decides then what to do: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the TaskTracker as undependable.
- ✓ After the work is completed, the status of the JobTracker updates.
- ✓ Client applications can do survey of the JobTracker for information.

• **TaskTracker:** on each machine, TaskTrackers are deployed. The main function of TaskTracker is for running the map and reducing tasks as instructed by the JobTracker. Only One TaskTracker processes run on any Hadoop slave node. TaskTracker runs on its own JVM process. Every TaskTracker is arranged with a set of slots; these indicate the number of tasks that it can accept. Heartbeat messages can also send by TaskTrackers to the JobTracker, usually every few minutes, to support the JobTracker that it is still alive. These messages also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be substitute.

A separate JVM processes starts by the TaskTracker to do the actual work (called as Task Instance) this is to guarantee that process failure does not take down the TaskTracker. The TaskTracker observers these task instances, capturing the output and exit codes. When the Task instances finish, successfully or not, the task tracker notifies the JobTracker.

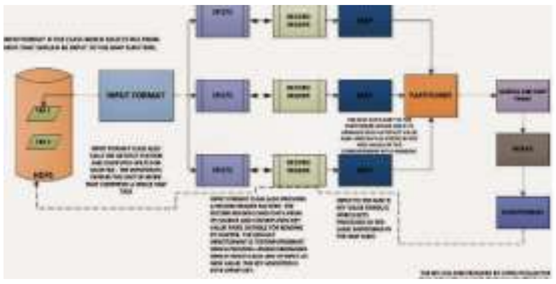

Figure 3: Simple MapReduce model

By using the class InputFormat, select the file which is an input to the map function. An InputFormat is also responsible for generating the **input splits** and splitting them into records. We using is text format as the default input format. In HDFS, the data is split into number of splits (normally 64/128MB)

A single map is processing of the logical representation of a block. i.e., an input split. . By default split=block InputFormat class demands the getSplits() function and computes splits for each file and then sends them to the JobTracker, which uses their storage locations to schedule map tasks to process them on the TaskTracker. To obtain a RecordReader for that split, On a TaskTracker, the map task passes the split to the createRecordReader() method on InputFormat

Loads of data from its source and converts into key-value pairs suitable for reading by mapper done by using the RecordReader. The map task uses one to create record key-value pairs, and then it passes to the map function.

As per the logic mentioned in the map code, an Input to the map function which is the key-value pair (K, V) gets processed.Then for the Partitioner, the output of the mapper is sent. It controls the subdividing of the keys of the intermediate map-outputs. Using Hash function, to derive the partition a key is used. The number of reduce tasks for the job is the same as the total number of partitions. Hence partition controls which of the MapReduce tasks the intermediate key (and hence the record) is sent for reduction. Partitioner governs, which reducer is responsible for a particular key. The use of Partitioner is optional.

Output of the mapper is first written on the local disk for sorting and shuffling process. It is also in the form of key-value pair. Shuffling is costly process, but we cannot ignore shuffling. This is the place where volume of data is travelling across network. This volume can be high. For example, if I have petabytes of data, then huge volume of data is travelling across network.

We can compress the data i.e., output of Partitioner, then compressed output will travel across network and then un-compress when shuffling is completed. There are some optimize available i.e., combiner data.Next component is sorting, which sorts all our data based on our key, so sorting is based on the key. MapReduce job output will in sorted order.

After sort, data will be given as input to reducer. After reduce the data will be given as output format, final (k,v) pairs. Combiner can be viewed as mini reducers in the map phase. Example:

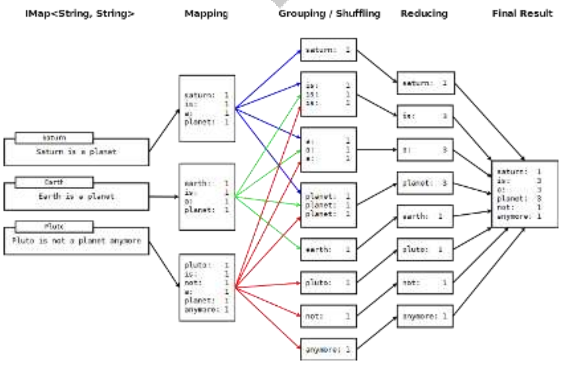

Figure 4: MapReduce example

V Hadoop components

The Big Data may be comprises of large and complex datasets that can be structured, semi-structured, or unstructured and will typically not fit into memory to be processed.
Ex: social network, cc camera, hospital data etc.

Big Data has its features in terms of three V's is

- Volume: Volume denotes to the size of the dataset. It may be in size of KB, MB, GB, TB, or PB
- Velocity: Velocity denotes to the low latency and real-time speed at which the analytics need to be applied.

Ex: perform analytics on a continuous stream of data originating from social networking sites may be YouTube, Facebook etc.

- Variety: Variety refers to the various types of the data that can exist like text, audio, video, and photos.

Data can be of structured, semi-structured and unstructured

- Structured-RDBMS data, MS excel enterprise data...etc.
- Semi structured-XML files, log files, html files...etc.
- Unstructured-E-mail, word document, pdf, images, audios, videos...etc.

An ecosystem of other products built above the core HDFS and MapReduce layer to enable various types of operations on the platform are included in the Hadoop. Hadoop has few popular components, they are as follows:

- **Mahout**: Mahout is an wide-ranging library of machine learning algorithms.Mahout provides the data science tools to automatically find meaningful patterns in those big data sets done once big data is stored on the Hadoop Distributed File System (HDFS). The aim of the Apache Mahout project is to make it faster and easier to turn big data into big information.

A. Pig:

As we knew the Hadoop has of its popular components known as Pig. Pig has its own syntax of language for expressing data analysis programs to examine large datasets.

Yahoo has developed a convenient tool called as Apache Pig for analyzing huge data sets efficiently and easily. Apache Pig can used to handle structured, unstructured, and semi-structured data. Apache Pig provides a high level data flow language Pig Latin that is optimized, extensible and easy to use. Programmers need to write scripts using Pig Latin language to analyze data using Apache Pig. After it all these scripts are internally converted to Map and Reduce tasks. A Pig Latin statement is termed as an operator that takes a relation as input and produces another relation as output. Apache Pig has a component known as Pig Engine that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs. Pig programs have their own structure to handle large data sets with considerable parallelization. Pig has two execution modes. Namely:

- Local Mode - Specify local mode using the -x flag (pig -x local), to run Pig in local mode.
- MapReduce Mode - in MapReduce mode to run Pig, you need access to a Hadoop cluster and HDFS installation. The default mode is the MapReduce mode; it specify by using the -x flag (pig OR pig -x MapReduce).

The following image illustrations the Hadoop technology stack.

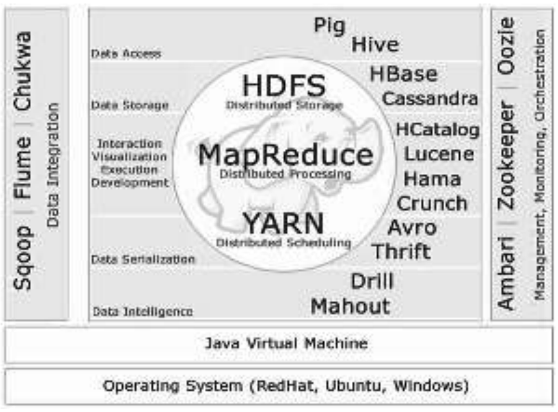

Figure 5 Hadoop technology stack

B. **Hive**:

A data warehouse system for Hadoop that facilitates easy data summarization, Ad-hoc queries, and the analysis of large datasets stored in HDFS is known as Hive. Facebook has developed a tool in Hadoop is Hive. It is a data warehouse built on top of Hadoop and provides a simple language known as HiveQL (Hive Query Language). It is similar to SQL for querying, data summarization and analysis. Hive makes querying faster through indexing Hive provides tools to enable easy data extract/transform/load (ETL). It provides the structure on a variety of data formats. We can access files stored in Hadoop Distributed File System (HDFS is used to querying and managing large datasets residing in) or in other data storage systems such as Apache HBase by using Hive.

.

- **HBase**: HBase(Hadoop Database) is a distributed, column-oriented database. It uses HDFS for the underlying storage and is horizontally scalable. HBase supports both batch style computations using MapReduce and atomic queries (random reads). One of the largest users of HBase is Facebook with its messaging platform built on top of HBase in 2010 and also Facebook uses HBase for streaming data analysis, internal monitoring system, Nearby Friends Feature, Search Indexing and scraping data for their internal data warehouses.

- **Sqoop:** A tool designed for efficiently transferring bulk data between Hadoop and Structured Relational Databases is known as Apache Sqoop. Sqoop is an abbreviation for (Sq)L to Had(oop). For importing data from external sources into related Hadoop components like HDFS, HBase or Hive, is done by using a Sqoop. For exporting data from Hadoop to other external structured data stores, we can even use Sqoop. It has parallelized data transfer, mitigates excessive loads, allows data imports, copies data quickly. and efficient data analysis.

- **ZooKeper:** It is a centralized service to maintain configuration information, naming, providing distributed synchronization, and group services, which are very useful for a variety of distributed systems. Common objects needed in large cluster environments are maintained by ZooKeeper. A ZooKeeper server is a machine that keeps a copy of the state of the entire system and persists this information in local log files. A very large Hadoop cluster can be supported by multiple ZooKeeper servers (in this case, a master server synchronizes the top-level servers). Each client machine communicates with one of the ZooKeeper servers to retrieve and update its synchronization. Within ZooKeeper, an application can create what is called a znode.The znode can be updated by any node in the cluster, and any node in the cluster can register to be informed of changes to that znode information.

- **Ambari**: A web-based tool for provisioning, dealing, and monitoring Apache Hadoop clusters, which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig, and Sqoop, is known as Ambari.It provides a console for viewing cluster health such as heat maps and ability to view MapReduce, Pig and Hive applications visually along with features to diagnose their performance characteristics in a user-friendly manner.

- **Flume:** To gather and aggregate large amounts of data, we use a component known as Flume. For collecting data from its origin and sending it back to the resting location (HDFS), a Flame is used. Flume achieves this by outlining data flows. that consist of 3 primary structures source, channel and sink. Flume to bulk transfer data from MySQL database table to HDFS the processes that run the dataflow with flume are known as agents and event is known as the bits of data that flow via flume.

- **Oozie:**Oozie is a known as workflow scheduler where the workflows are expressed as Directed Acyclic Graphs. In Oozie, the workflows are executed based on data and time dependencies. It runs in a Java servlet container Tomcat and makes use of a database to store all the running workflow instances, their states ad variables along with the workflow definitions to manage Hadoop jobs (MapReduce, Sqoop, Pig and Hive).

VI CONCLUSION

Hadoop Framework has gained so much popularity as it is a remedial solution for large data sets. As a traditional analyzing method was storing restricted and time consuming. In this paper, a brief idea about Hadoop Framework and its components are explained. Storage of data and processing stored data gives brief idea about how both phases run in reality.

REFERENCES

[1] Dhyani, Bijesh, and Anurag Barthwal. "Big Data Analytics using Hadoop." *International Journal of Computer Applications* 108.12 (2014).
[2] Pandagale, Ashwini A., and Anil R. Surve. "Big Data Analysis Using Hadoop Framework."
[3] Navya Francis and Sheena Kurian K "Data Processing for Big Data Applications using Hadoop Framework" in International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 3, March 2015.
[4] Veershetty Dagade Mahesh Lagali Supriya Avadhani Priya Kalekar " Big Data Weather Analytics Using Hadoop" International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) ISSN: 0976-1353 Volume 14 Issue 2 –APRIL 2015
[5] Maarala, Altti Ilari, et al. "Low latency analytics for streaming traffic data with Apache Spark." Big Data (Big Data), 2015 IEEE International Conference on.IEEE, 2015.
[6] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[7]https://www.google.co.in/search?q=Simple+MapReduce+model