Subodh Journal of Recent Trends in Information Technology
Volume 10, Issue 01, June 2019 | ISSN No. 0975-9875

Study of HDFS Architecture and Services

Khushboo Verma
Khushbooverma250186@gmail.com

ABSTRACT - In this world, different type of information is being produced every second. We need to store this huge information so that we can use it whenever needed. HDFS is designed to store this big amount of data in number of servers. The data is replicated and stored in number of servers. In this paper I am going to describe the architecture of HDFS and important terms related with that.
KEYWORDS - big data; hadoop; map reduce; HDFS; namenode; datanode; secondary namenode.

INTRODUCTION

Now a days, along with structured data, an unstructured and semi-structured data is also produced by different applications in very less time like images, videos, text etc. in large amount. So, this type of large data is termed as BIG DATA[1]. The size of this big data is in terabytes[2], petabytes and even exabytes. Our RDBMS is not sufficient to handle this big amount of data because RDBMS can only handle structured data. To handle or we can say to store and process this type of data HADOOP came into the picture[1]. Hadoop is an open source framework by Apache Software Foundation written in java language for storing and processing big data sets. Hadoop is a combination of HDFS (Hadoop distributed file system) and mapreduce[3].

1. WHY HADOOP NOT FOR SMALL DATA?

Small files are much smaller than HDFS data block. HDFS is made to handle less files of large size, but it cannot work efficiently with large number of small size files as it has to maintain metadata of these files in namenode. So, maintaining metadata of large number of small size file makes namenode size big, which will slow down the process[12].

2. HDFS ARCHITECHTRE

HDFS is a specially designed file system as its memory blocks are 64MB of size[11] by default for storing huge data sets with the use of cluster of commodities hardware(cheap hardware)[8] like our PCs because hadoop needs large number of hardware to store huge data and it can support commodities hardware with streaming access pattern (write ones read anywhere).Whereas mapreduce is used for processing that stored data[8].

Five services of HDFS and their interaction with each other

Master services

- Namenode
- Jobtracker

Secondary master service

- Secondarynamenode

Slave services

- Datanode
- Tasktracker[8]

Every master service can interact with each other and every slave service also can interact with each other.Every master service can interact with its corresponding slave service like name node can interact with data node and job tracker can interact with task tracker but any master node cannot interact with slave node of other master node.

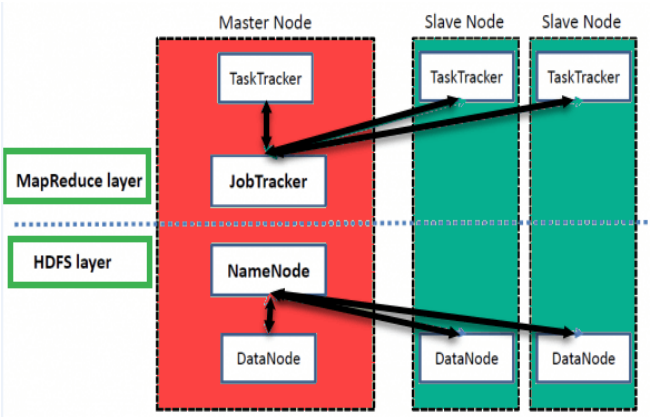

Fig 1: Master / Slave architecture[6]

3. WORKING

Assumption

A client has data of 200 MB of size in his local machine (we are taking small data size for our convenience). The file name is file.txt.

In HDFS we have number of commodities hardware also known as data nodes of memory block size of 64 MB by default (can change it into 128 MB also according to our data size) to store this 200MB data file. So the client has to break this file into three input splits of 64MB size and one input splits of 8MB size. These four input splits are going to store in different data nodes.

The four input splits will be-

- w.txt of64MB
- x.txt of64MB
- y.txt of64MB
- z.txt of8MB

This HDFS architecture is master-slave architecture [8].

a. NameNode

Namenode is a master node which keeps the metadata of stored files[4] like file name (including path), input splits, name of data node where input splits are stored, size, owner, group, permission, block size etc.

b. Data node

Data node is a slave node which actually stores the data[8].Client send request to name node for storing his data in data nodes and in response name node will send data nodes details in which client can store his data input splits[4].

Let us assume the four input splits will be stored in data node number 1, 3, 5, 7. So, this information of data node number and its stored input split name will be send to the name node from client and with this information name node create the metadata and store it in its RAM.

c. Metadata

- file name –file.txt
- File size – 200MB
- Four inputsplits
  1. w.txt in data node1
  2. x.txt in data node3
  3. y.txt in data node5
  4. z.txt in data node7

  (I have not written all the metadata here)

d. Namenode failure

If namenode get failed due to any reason our metadata get lost and data nodes will become inaccessible because information of data node and its stored input splits is stored in metadata only.

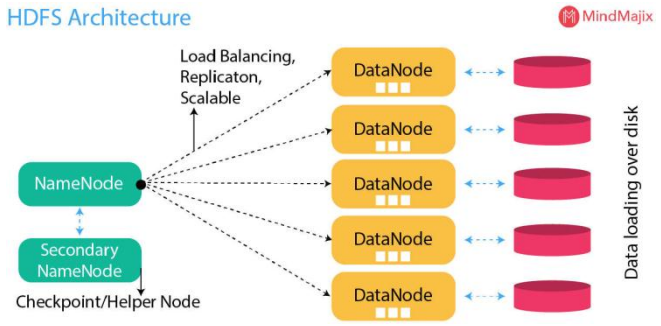

Fig 2: HDFS architecture [7]

4. FAILURE AND THE WAY TO HANDLE THEM

a. Data node failure

As we are using cheap hardware for storing our data, failure can take place due to any reason and our data get lost.

b. Solution

Name node maintain two more copies of every input split in different data nodes and also maintain this information in its metadata. For example – w.txt in datanode 1, 5, and 9 etc.So in total name node maintain three copies of everyinput split in different data nodes and maintain information about that to overcome failureofdata node. In HDFS, every datanode sends heartbeat to node. In HDFS, every data node sends heartbeat to show that it is alive and block report to show all of the blocks it is handling in every 10 minutes. If namenode does not receives heartbeat from any datanode, then it shows that this datanode is dead. Now namenode copy all of the blocks that node is handling to some other datanode and three copies of data is maintained[5].

Namenode metadata maintain two files :-

- fsimage
- editlog

fsimage is a screenshot of edited metadata at that point of time. It is an image file. Whereas editlog is a log file which contain currently edited metadata (adding, deleting, updating etc.). HDFS does not made changes directly to the fsimage but it make changes to the editlog and take snapshot of edited data as fsimage.

There is a secondary namenode in another master machine. Secondary namenode connect itself tothe name node at regular interval of time[9]. Secondary name node take current editlog from namenode and using this log file it create fsimage and make editlog empty. Now it saves this image file in its memory and also give copy of fsimage to namenode. Again secondary namenode reconnects with namenode after sometime and again takes new editlog and merge it with last fsimage to create new fsimage anderase everything from editlog to make it short in size. This process goes on and seconday namenode has collection of old fsimages and newly created also.

These images are known as checkpoints[10].Now if namenode crashed, then namenode restarted and rebuilt the metadata using fsimages from secondaynamenode.

6. CONCLUSION

In this review paper, an overview of big data is provided along with the process of how the data is stored in HDFS. This paper is also describing why hadoop is not suitable for small datasets. Five services of HDFS are also described here. Failure of name node and data node with it solution is also described.

7. REFERENCES

[1] Roshani K. Chaudhari, Prof. D. M. Dakhane, "contribution of hadoop to big data problems", *International Journal of Advanced Research in Computer Science and Software Engineering*, vol 5 issue 4 (2015).

[2] D. P. Acharjya, Kauser Ahmed P, "A Survey on Big Data Analytics: Challenges", Open Research Issues and Tools, *International Journal ofAdvanced Computer Science and Applications*, Vol. 7 No. 2 (2016).

[3] Himani Saraswat, Neeta Sharma, Abhishek Rai, "Enhancing the Traditional File System to HDFS: A Big Data Solution", *International Journal ofComputer Applications* (0975 – 8887), Volume 167 – No.9, (2017).

[4] Bhawana Sahare , Ankit Naik , Kavita Patel, "Study of HADOOP*", International Journal of Computer Science Trends and Technology*, Vol 2 Issue 6,(2014).

[5] T. Cowsalya, S.R. Mugunthan, "Hadoop architecture and fault tolerance based hadoop clusters in geographically distributed data center", *ARPN Journal of Engineering and Applied Sciences*, VOL. 10, NO. 7,(2015).

[6]https://www.guru99.com/learn-hadoop-in-10-minutes.html

[7] https://mindmajix.com/hadoop-tutorial

[8] Devateja G , Kashyap P V B , Suraj C, Harshavardhan C , Impana Appaji, "A Study: Hadoop Framework", *International Journal ofAdvance Engineering and Research Development*, Vol 3 Issue 2, (2016).

[9] V. S. Karwande , Dr. S. S.Lomte, Prof. R. A. Auti, "The Data Recovery File System for Hadoop Cluster -Review Paper", *International Journal of Computer Science and Information Technologies*, Vol. 6 (1) , (2015).

[10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler, "The Hadoop Distributed File System", *IEEE*,(2010).

[11] Bharti Gupta, Rajender Nath, Girdhar Gopal, Kartik,"An Efficient Approach for Storing and Accessing Small Files with Big Data Technology", *International Journal of Computer Applications*(0975– 8887), Vol 146 – No.1, (2016).

[12] Sachin Bendea , Rajashree Shedge, "Dealingwith Small Files Problem in Hadoop Distributed File System", *ScienceDirect/Procedia computer science*79, (2016)