ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X
Available online at www.ijarbest.com
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
Vol. 2, Issue 1, January 2016

A Systematic Big Data Study Using HDFS and Map Reduce

S.Lavanya[1], N.R.Vikram[2], M.Revathi[3]
Assistant Professor, CSE, Paavai Engineering College, Namakkal, India

***Abstract*— In modern beings, big data plays a vital role in processing/analyzing a large set of datasets. Similar to data mining, big data analytics provide an insight to uncover hidden patterns and useful information, in order to make better decisions. There are various techniques available to perform big data analytics. This paper provides a vision on big data, Hadoop, components of HDFS and working of MapReduce framework. It also offers creation and execution of MapReduce program in Java.**

***Index Terms*— Big data, Map reduce, Hadoop**

1. INTRODUCTION

What is Big Data?

Big Data is a data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the structures of your database architectures.

Big Data is a common buzzword in the world of IT nowadays and it describes the realization of greater business intelligence by storing, processing and analyzing data that was previously ignored due to the limitations of traditional data management technologies.

Big Data applies to information that can't be processed or analyzed using traditional processes or tools. Increasingly, organizations today are facing more and more Big Data challenges. They have access to a wealth of information, but they don't know how to get value out of it because it is sitting in its most raw form or in a semi structured or unstructured format; and as a result, they don't even know whether it's worth keeping (or even able to keep it for that matter).

Big Data era is in full force today because the world is changing.

1.1 Characteristics of Big Data

The characteristics of big data are often defined as the three Vs:

1. Volume
2. Variety
3. Velocity

1.2 The Volume of Data

The sheer volume of data being stored today is exploding. In the year 2000, 800,000 petabytes (PB) of data were stored in the world. Of course, a lot of the data that's being created today isn't analyzed at all and that's another problem we're trying to address with Big Insights. We expect this number to reach 35 zettabytes (ZB) by 2020. Twitter alone generates more than 7 terabytes (TB) of data every day, Facebook 10 TB, and some enterprises generate terabytes (and even Exabyte's) of data every hour of every day of the year.

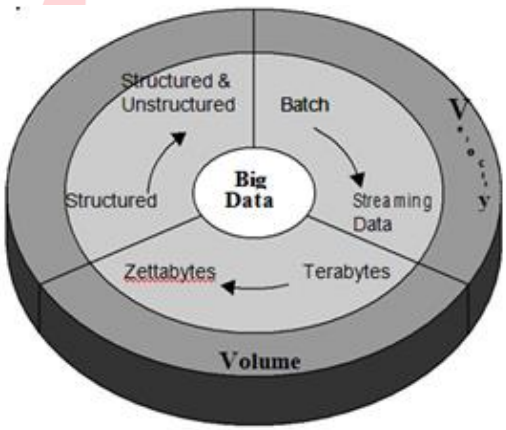

Fig 1 Volume of Big Data

1.3 The Velocity of Data

A conventional understanding of velocity typically considers how quickly the data is arriving and stored, and its associated rates of retrieval i.e. the speed at which the data is flowing. Big Data includes all types of data:

1. Structured: The data has a schema, or a schema can be easily assigned to it.
2. Semi-structured: Has some structure, but typically columns are often missing or rows have their own unique columns.
3. Unstructured: Data includes various structures like images, audio, video, etc.

Why is Big Data important?

Big Data is well suited for solving information challenges and they become even more vital when used in conjunction with Big Data platform.

Conventional database technologies are an important, and relevant, part of an overall analytic solution. In fact,

Big Data solutions are ideal for analyzing not only raw structured data, but semistructured and unstructured data from a wide variety of sources.

Big Data solutions are ideal when all, or most, of the data needs to be analyzed versus a sample of the data; or a sampling of data isn't nearly as effective as a larger set of data from which to derive analysis.

Big Data solutions are ideal for iterative and exploratory analysis when business measures on data are not predetermined.

2. ABOUT HADOOP

Hadoop is a top-level Apache project in the Apache

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X
Available online at www.ijarbest.com
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
Vol. 2, Issue 1, January 2016

Software Foundation that's written in Java. Hadoop is an open source framework for distributed storage and processing of large sets of data on commodity (cheap) hardware. Hadoop enables businesses to quickly gain insight from massive amounts of structured and unstructured data.

Hadoop was first found by Google's work on its Google (distributed) File Sys-tem (GFS) and the MapReduce programming paradigm, in which work is broken down into mapper and reducer tasks to manipulate data that is stored across a cluster of servers for massive parallelism. The implementation of MapReduce remained as white paper. Later, Doug Cutting and Mike Cafarella who was working at Yahoo renamed GFS into HDFS i.e. Hadoop Distributed File System and Cutting implemented MapReduce program.

Hadoop is designed to scan through large data sets to produce its results through a highly scalable, distributed batch processing system. Hadoop comprises of two parts: a file system (the Hadoop Distributed File System) and a programming paradigm (MapReduce).

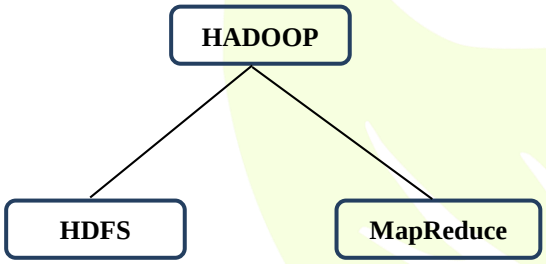

Fig2. Classification of Hadoop

2.1 Hadoop Distributed File System

Data in Hadoop is stored on a file system referred to as HDFS or the Hadoop Distributed File System. With HDFS, data is broken into chunks and distributed across a cluster of machines.

2.1.1 HDFS has the following characteristics:
Primary storage system for Hadoop: it stores large files as small blocks.
Reliability: Data is replicated so that disk failover is not only acceptable but expected and handled seamlessly.

A data file in HDFS is divided into blocks, and the default size of these blocks for Apache Hadoop is 64 MB. For example, typical file systems have an on-disk block size of 512 bytes, whereas relational databases typically store data blocks in sizes ranging from 4 KB to 32 KB. Remember that Hadoop was designed to scan through very large data sets, so it makes sense for it to use a very large block size so that each server can work on a larger chunk of data at the same time.

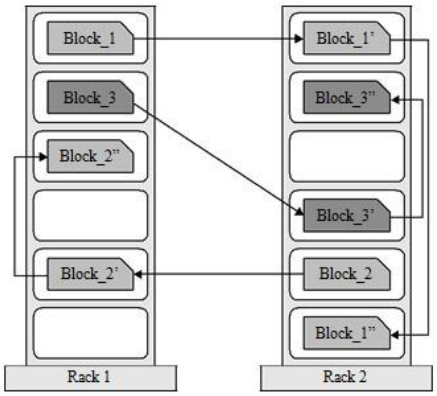

Fig. 3 Structure of file with data blocks

Figure 3 represents a file that is made up of three data blocks, where a data block (denoted as block_n) is replicated on two additional servers (denoted by block_n' and block_n"). The second and third replicas are stored on a separate physical rack, on separate nodes for additional protection.

2.2 Components of HDFS

A Hadoop instance of a cluster of HDFS machines often referred to as the Hadoop cluster or the HDFS cluster. There are two main components of an HDFS cluster:

1. NameNode: The "master: node of HDFS that manages the data(without actually storing it) by determining and maintaining how the chunks of data are distributed across the data nodes.
2. DataNode: the slaves which are deployed on each machine and provide the actual storage. They are responsible for serving read and write requests for the clients.
3. Secondary Name Node: It is responsible for performing periodic checkpoints. In the event of Name Node failure, you can restart the Name Node using the checkpoint.

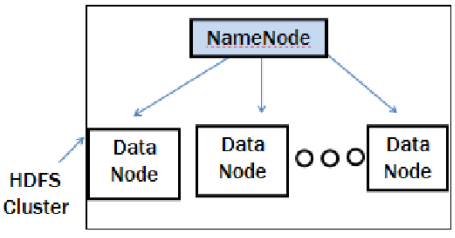

Fig. 4 Components of HDFS

2.3 Block Storage

Loading a file into HDFS involves the following steps:

A client application sends a request to the NameNode that specifies where they want to put the file in the filesystem

The Namenode determines how the data is broken down into blocks and which DataNodes will be used to store those blocks. This information is given to the client application.

The client application communicates directly with each DataNodes, writing the blocks onto the DataNodes.

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X
Available online at www.ijarbest.com
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
Vol. 2, Issue 1, January 2016

The DataNodes replicate the newly created blocks based on instructions from the NameNode.

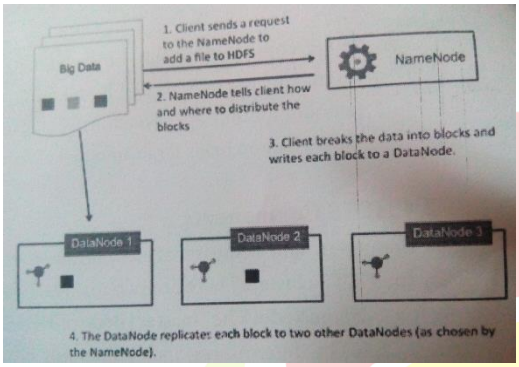

Fig. 5 Understand of Block Storage

2.4 Inputting Data into HDFS
Options for Data Input
The first task in using a Hadoop cluster is getting our big data into HDFS. There are several options to bring big data into HDFS.

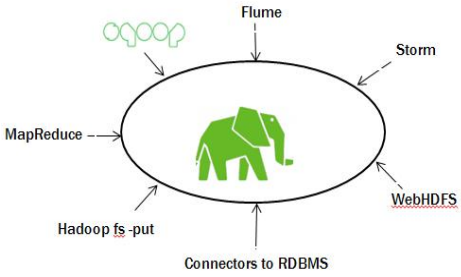

Fig. 6 Connecting big data and HDFS

The put command is used to read input file of nearly petabyte into HDFS.
Hadoop fs –put – myinput.txt

3. MAP REDUCE FRAMEWORK

Overview of MapReduce
MapReduce is the heart of Hadoop. It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster.
The term MapReduce actually refers to two separate and distinct tasks that Hadoop programs perform.

1. Map Job: Data is input into the mapper, where it is transformed into another set of data. Individual elements are broken down into tuples (key/value pairs) and are prepared for the reducer.
2. The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

3.1 Understanding MapReduce
In MapReduce, data are defined as (key, value) pairs. Mapper takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain:
Map (k1, v1) --> list of (k2, v2)

After that, the MapReduce framework collects all pairs with the same key (k2) from all lists and groups them together, creating one group for each key (k2). Reducer is then applied in parallel to each group, which in turn produces a collection of values in the same domain:

Reduce (k2, list of v2) --> list of (v3)

Thus, the MapReduce framework transforms a list of (k,v) pairs into a list of values.

A 5-step parallel and distributed computation:
1. Prepare the input for Mapper: the MapReduce framework designates Map processors, assigns the input key (k1) value each Mapper would work on, and provides that Mapper with all input data associated with that key value.
2. Run the user-provided Mapper code: Map() is run exactly once for each key (k1) value, generating output organized by key (k2) values.
3. Shuffle the Mapper output to Reducers: the MapReduce framework designates Reduce processors, assigns the key (k2) value each Reducer would work on, and provides that Reducer with all the Mapper-generated data associated with that key value.
4. Run the user-provided Reducer code: Reduce() is run exactly once for each key (k2) value, generating the output value (v3).
5. Produce the final output: the MapReduce framework collects all the Reducer output, and sorts it by key (k2) value to produce the final outcome.

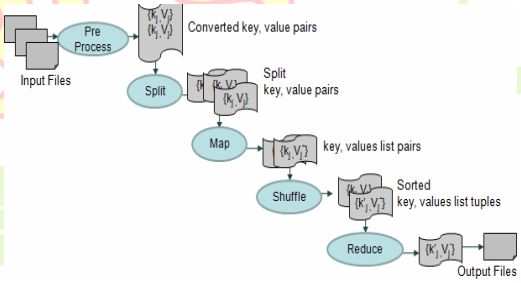

Fig. 7 Pipeline of MapReduce

The major steps are: map -> shuffle (or group) -> reduce (see figure below). Just like Prof. Harvey said in his class, it should be called "MapGroupReduce".

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X
Available online at www.ijarbest.com
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
Vol. 2, Issue 1, January 2016

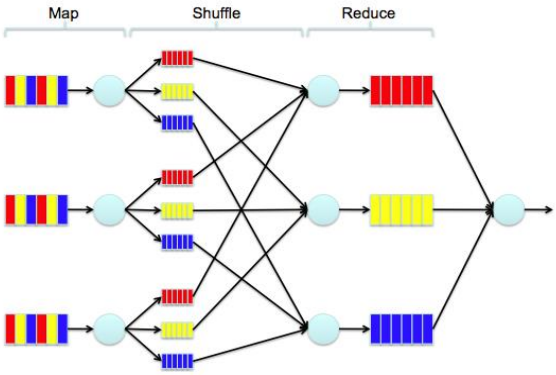

Fig. 8 Steps in MapReduce

3.2 Map Phase

The map phase involves running map tasks on Node Managers. The main purpose of the map phase is to read all of the input data. The goal (in order to gain the best performance) is to achieve data locality, where a map task runs on a Data Node where its Input split is stored. A block of data rarely maps exactly to an input split, but it is often close, especially when processing text data. Records that spill over to a subsequent block have to be pulled over the network so the map task can process the entire record, but this is normally an acceptable overhead.

The number of map tasks in a MapReduce job is based on the number of Input Splits. If no node manager is available where a specific block resides, then data will be lost locally and the block has to be pulled across the network.

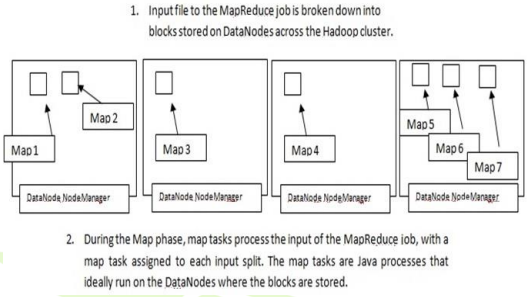

Fig. 9 Tasks in MapReduce

Map tasks output <key, value> pairs, which are written to a temporary file on the local File system. When a map task finishes, its output becomes immediately available to the reduce tasks. Each reducer asks each mapper for the <key, value> pairs designated for that reducer. This designating of records is called partitioning. As a reducer reads-in its <key, value> pairs, the values are aggregated into a collection and the entire input to the reducer is sorted by keys. This is referred to as the shuffle/sort phase.

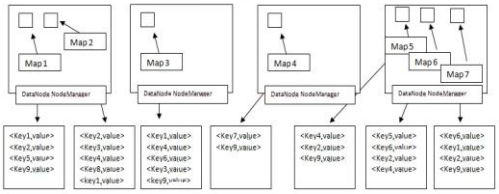

Fig. 10 Process of Mapping

The main purpose of the reduce phase is typically business logic i.e. going through the data output by the mappers and answering a question or solving a problem. The <key, value> pairs coming into the reducer are combined by key, meaning each key is presented once to the reducer along with all of the values that belong to that key. Reducers also output <key, value> pairs.

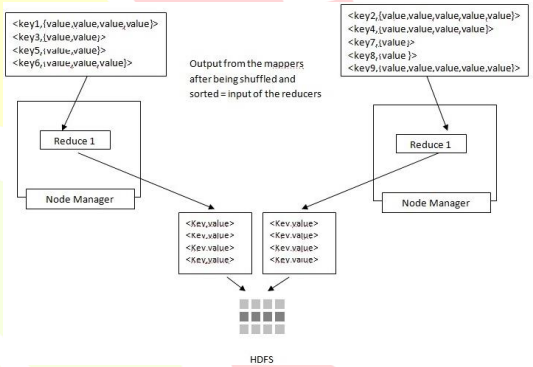

Fig. 11 Output after Mapping and Shuffling

3.3 Reduce Phase

The reducer fetches the records from the mapper and uses them to generate and output another set of <key, value> pairs that are output to HDFS.The reduce phase can actually be broken down in three phases:

Shuffle: Also referred to as the fetch phase, this is when reducers retrieve the output of the Mappers. All records with the same key are combined and sent to the same reducer.

Sort: This phase happens simultaneously with the shuffle phase. As the records are fetched and merged, they are sorted by key.

Reduce: The reduce method is invoked for each key, with the records combined into an iterable collection.

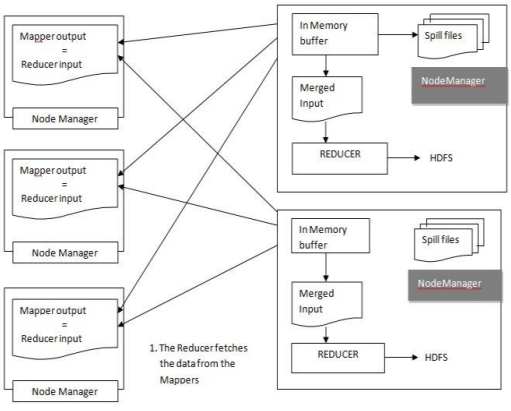

Fig. 12 Reduce phase of MapReduce

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X
Available online at www.ijarbest.com
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
Vol. 2, Issue 1, January 2016

3.4 Creation of Map Reduce Program

To perform Map Reduce function, three Java classes has to be created and executed. They are

1. Driver Class
2. Mapper Class
3. Reducer Class

a. Driver Class

```
Public class WordCount extends Configured implements
Tool
{
 Public int run(string args[])
 {
   If(args.length < 2)
   {
      System.Out.Println(" Specify the input and output
directory count");
      Return -1;
    }
   Jobconf conf = new Jobconf(WordCount.Class);
FileInputFormat.SetInputPath(WordMapper.Class);
FileOutputFormat.SetInputPath(WordReducer.Class);
Conf.SetMapInputKeyClass(longwriteable.class);
   Conf.SetMapInputValueClass(Text.class);
   Conf.SetOutputKeyClass(Text.class);
   Conf.SetOutputValuesClass(IntWriteable.class);
   Jobclient.runJob(conf);
 }
}
```

Main Method

```
Public static void main(string args[])
{
 int exitcode = ToolRunner.run(new WordCount(),args);
 system.exit(exitcode);
}
```

b. Mapper Class Code

```
Public class WordMapper extends MapReduceBase
implements Mapper<longwriteable,text,text,intwriteable>
{
  Public void map(intwriteable key, text values,
OutputCollected Output, Reporter report)
  {
   String S = values.tostring();
   for(string Word: S.Split("  "))
   {
    If(word.length() > 0)
    {
     Output.Collect(new text(word)), new intwriteable(1));
     }
   }
  }
}
```

c. Reducer class code

```
Public class Word Reducer extend MapReduce Base
implements Reducer<intwriteable,text,text,intwriteable>
{
 Public void reducer(text key,iterator value,outputcollector
output, Reporter report)
 {
  Int count = 0;
  While(values.hasnext())
  {
   Intwriteable I = values.next();
   Count += i.get();
  }
 Output.Collect(key, new intwriteable(count));
  }
}
```

3.5 Execution of Map Reduce Program

```
Hadoop jar
/usr/hdp/current/hadoop_mapreduce_historyserver/hadoop_
mapreduce_examples_2.6.0.2.2.0.0_2041.jar WordCount
test/constitution.txt count.txt
```

Here, constitution.txt is the input file and count.txt is the output file.

To view the output file

```
#hadoop fs –ls count.txt
#hadoop fs –cat count.txt/partr-r-00000
```

4. CONCLUSION

The final aspect of this paper is how well it integrates into our existing enterprises.

Big Data is not a replacement for traditional systems; it's just coordinating our traditional system with new age. HDFS and MapReduce framework can be used in almost all the fields like Financial Services Sector, Health and Life Sciences, Telecommunications, Defense, Surveillance, and Cyber Security.

REFERENCES

1. J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," vol. 51, no. 1. New York, NY, USA: ACM, Jan. 2008, pp. 107–113. [Online]. Available: http://doi.acm.org/10.1145/1327452.
2. J. Manyika et al., "Big Data: The Next Frontier for Innovation, Competition, and Productivity," 2011.
3. C. Lynch, "Big Data: How Do Your Data Grow?" Nature, vol. 455, no. 7209, pp. 28-29, 2008.
4. T. White, Hadoop: The Definitive Guide, 1st ed. O'Reilly Media, Inc.,2009.
5. Burghard C: Big Data and Analytics Key to Accountable Care Success. IDC Health Insights; 2012.
6. Dembosky A: "Data Prescription for Better Healthcare." Financial Times, December 12, 2012, p. 19; 2012. Available from: http://www.ft.com/intl/cms/ s/2/55cbca5a-4333-11e2-aa8f
7. Feldman B, Martin EM, Skotnes T: "Big Data in Healthcare Hype and Hope." October 2012. Dr. Bonnie 360;2012.http://www.west-info.eu/files/big-data-inhealthcare.pdf.

ISSN (ONLINE): 2395-695X
ISSN (PRINT): 2395-695X
Available online at www.ijarbest.com
International Journal of Advanced Research in Biology, Engineering, Science and Technology (IJARBEST)
Vol. 2, Issue 1, January 2016

8. Fernandes L, O'Connor M, Weaver V: Big data, bigger outcomes. J AHIMA 2012:38–42

BIOGRAPHY

S.Lavanya received the Master Degree from Anna University, Chennai in 2012. She is currently working as Assistant Professor in Paavai Engineering College, Tamil Nadu. Her research area includes Data Warehousing and Mining, Big Data and Image Processing.

He has received his Master's Degree in Information Technology from Madras Institute of Technology, Anna University Chennai in 2012.He is currently working as an assistant professor in Paavai Engineering College, Pachal, Namakkal. He has published 5International Journal Papers, 7 National and International Conferences. He has been the committee member for various international conferences. He is the Life member of ISTE. He served as a volunteer in ICATS conferences held in Paavai Institutions. His research interests include Image Processing, Big Data and Networking

She received her Master's Degree in Mainframe Technology from Anna University of Technology, Coimbatore in 2011.She is currently working as an assistant professor in Paavai Engineering College, Pachal, and Namakkal. She has Received 2 Prestigious award
1. Best student award in the year 2008.2. Best paper award in the international conference ISCO 2011.She has published 5 International Journal Papers, 15 International and National Conferences. She has been the reviewer for various international conferences. She is the LIFE member of ISTE. She is an organizer of IIT-B Spoken Tutorial and IITB Workshops in Paavai Institutions. She served as an organizer in ICATS conferences held in Paavai Institutions.
Her research interests include Image Processing, Big Data and Networking