ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

A Survey on Hadoop Mapreduce Framework

Jeet Samirkumar Shah

Bachelor of Engineering Student, Department of Computer Engineering, Vishwakarma Government Engineering College, Gujarat Technological University, Gujarat, India

ABSTRACT : In recent years, data computations and analysis are increasing rapidly with the development of web-based applications and mobile computation technology. With large-scale data, which is highly supported in decision-making, various fields around the globe face a major problem. We require efficient algorithms to process this big data. MapReduce has played an important role in meeting the growing demands on computing resources affected by voluminous data sets in the big data world. MapReduce is an emerging programming model designed to process extremely large data volumes in parallel mode by dividing the job into different independent tasks. MapReduce algorithm has two different phases: 1) Map and 2) Reduce. First phase takes set of data and convert it into another set of data in the form of key/value pairs. Reduce phase takes output of Map task as an input data and combine those into smaller number of tuples. As the name MapReduce implies, Map task is always performed before the Reduce task. Ease of scaling data processing over several computing machines is the major advantage of this algorithm. Moreover, MapReduce programming processes large volumes of data in a completely protected, fast and cost-effective way. This paper highlights and investigates Big data and its analytics using Hadoop MapReduce's recent models.

KEYWORDS : Big data, Map, Reduce, Algorithm, Large-scale data, Key-Value pair

I.INTRODUCTION

Hadoop MapReduce is a software algorithm which processes big volume of a data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner. This can be done by splitting the job into various tasks. A MapReduce algorithm is a combination of Map() and Reduce() function. Map() function takes an input data-set into consideration and splits it in independent chunks completely in a parallel manner. Then, the algorithm sorts the output of a Map() function and it becomes an input for the Reduce() function. And the Reduce() function performs aggregate operations like counting the number of same data that appears in a data-set. The input and output of both functions are stored in a file-system. [3] The "MapReduce Algorithm" well known as MapReduce "framework" or "architecture" demonstrates the processing with the distributed servers, running various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing redundant data and fault tolerance.[3]

MapReduce is a framework for processing vast amount of data which are divided and scattered across huge datasets which are stored in a large number of computers. The group of computers are treated as a cluster if all computers with same hardware features are working on a similar local server. Else it will be treated as a grid if they are distributed with different hardware configurations. But MapReduce has an advantage of data locality, to minimize the traversing for the data transfer.

II.LITERATURE REVIEW

Hadoop MapReduce framework briefly uses Big Data analytic methods. These methods is used to get the useful and needed information from the big amount of data and to reveal the patterns that are hidden in that big data. So, to fetch the information from the big data we need an algorithm, which will help us to find that information faster and

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

efficiently. MapReduce framework is developed by the Google. Apache Hadoop, which is an open source platform, must needed to run the MapReduce algorithm on a machine.

As we will have to work on a big data, we need high hardware configuration machine to run on. We can install Hadoop using the source code, but for that we need Apache maven, Windows SDK support in out IDE. Or else we can install it using the Hadoop binaries. However, after installing of Hadoop we can easily run our algorithm in the command prompt.

After applying the MapReduce algorithm, we can generate the data which has the same Key. And we can reveal the hidden structures or patterns from the big volume of data.

III.INPUTS AND OUTPUTS

The MapReduce architecture completely works on <Key, Value> pairs. Like, Map() function takes an input of the <Key, Value> pairs and produces an output of the different <Key, Value> pairs. Then, Reduce() function takes output of the Map() function as an input <Key, Value> pairs and produces a new <Key, Value> pair as an output.

For example, K = Key, V = Value

Input: <K1, V1>→ Map(K1,V1) → list <K2, V2>→ Combine →<K2, list(V2)>→ Reduce(K2, list(V2)) →<K3, list(V3)>→ Final Output.

IV.MAPREDUCE WORKFLOW

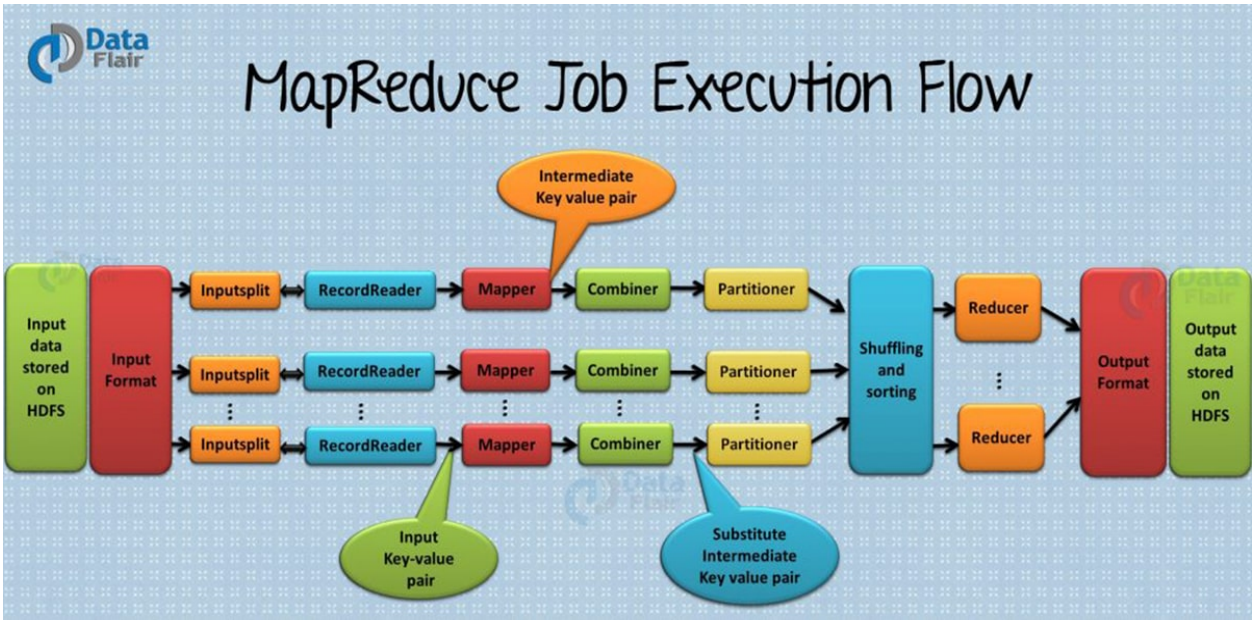

Figure – 1Job Execution Flow of MapReduce [2]

1. Input Files

The data which we want to apply MapReduce algorithm is stored in Input files. It is mostly available in HDFS.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

2. Input Format

Input Format generally defines the basic implementation which we will have to apply on data. For example: - how we will have to split the data and how we are going to read it.

3. Input Splits

It is generated by Input Format in a logical way that represent the data which will be processed by a separate Mapper. For each split there will be one map task and each split is divided into records and it will be processed by the mapper.

4. Record Reader

RecordReader fetch the input data from the InputSplits and converts it into Key-Value pairs. This will continue until the RecordReader does not fetch all the data from InputSplits. After that, these Key-Value pairs will be sent to the Mapper for further processing.

5. Mapper

Mapper processes each data which came from RecordReader and generates new Key-Value pair for each input set. After processing this data Mapper will produce an output data which will be an input to the Combiner. Since this is a temporary output data which may change in the Combiner or Reducer, it will be stored in a local disk and not in HDFS.

6. Combiner

Combiner performs an aggregation on local block data which came from the mapper, which helps Reducer to minimize the traversing of the data transfer to find the same value for the different key. Thus, it will reduce the sorting task of the data which will be remaining to be processed in the Reducer.

7. Partitioner

When we have more than one Reducer, we will use each combiner output to partition data. As a result of this, it is possible to bring same key value in the same partition. In fact, partition will be done on the basis of Key value. Reducer will then fetch each partition for the further processing.

8. Shuffling and Sorting

Shuffling and Sorting is an important phase in MapReduce algorithm because it will do merge and sort on the map output data. The data will be merged together by the key, divide among reducers and sort based on key. So that every reducer has all the values and has a pair with same key. If we will do this shuffling before execution of task by the map function, we can save lot more time and even we can finish the tasks in small amount of time.

9. Reducer

Reducer takes the output of the Mapper function after applying shuffling and sorting on it. It will aggregate the similar key-value pairs and after performing aggregates on all key-value pairs, it will combine them all into the one output and send it to the HDFS file system which will store the output permanently. All Reducers are independent of each other, so they run in a parallel manner.

V. PERFORMANCE

MapReduce performance is totally dependent on the function which we are using. i.e. Combiner, Partitioner or Shuffle and Sort function. With the use of this functions we can decrease the time of the data processing because we can apply shuffling on the data even if map function is still processing on the data. And other thing that programmer needs to write is only the map and reduce function on the programming model. Moreover, the data which will be generated by Combiner, Partitioner and Mapper function does not require more space to store in the HDFS or local disk. It will help to reduce the space complexity of the program. And the input data does not need to traverse the whole network (does

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

not matter if it is a local network or global network.) So, using these functions we can reduce the time and space complexity of the programming model.

VI. SOURCE CODE EXAMPLE

Drive.java

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
public class drive
{
@SuppressWarnings("deprecation")
public static void main(String[] args) throws Exception
{
Job job = new Job();
job.setJarByClass(drive.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

job.setMapperClass(Map.class);
job.setReducerClass(Red.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

String inputPath = args[0];
String outputPath = args[1];
FileInputFormat.addInputPath(job, new Path(inputPath));
FileOutputFormat.setOutputPath(job, new Path(outputPath));
System.exit(job.waitForCompletion(true) ? 0 : 1);
}
}
```

Block1 : Main Driver Class for the MapReduce Algorithm in Java[8]

Map.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class Map extends Mapper<LongWritable, Text, Text, IntWritable>
{
private final static IntWritable one =new IntWritable(1);
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

```
private Text word =new Text();

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException
{
String words[] = value.toString().split(" ");
for (int i=0; i<words.length; i++) {
word.set(words[i]);
context.write(word, one);
}
}
}
```

Block2 : MAPfunction logic in Map Class for the MapReduce Algorithm in Java[8]

Red.java

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class Red extends Reducer<Text, IntWritable, Text, IntWritable>
{
public void reduce(Text key, Iterable<IntWritable>values,Context context) throws IOException, InterruptedException
{
int sum = 0;
while(values.iterator().hasNext())
{
sum += values.iterator().next().get();
}
context.write(key, new IntWritable(sum));
}
}
```

Block2 :REDUCERfunction logic in Red Class for the MapReduce Algorithm in Java[8]

VII. JOB CONFIGURATION

User can specify an execution of a MapReduce job to Hadoop via job. So, job acts primarily as an interface. A job gives information about the implementations of Mapper, Combiner, Partitioner, Reducer, InputFormat, OutputFormat. Some of the job parameters include Job.setNumReduceTask(int),Job.setJobName(), Job.setOutputhPath(), Job.setMapperClass() etc. Additionally, job also gives details of the Comparator to be used, whether intermediate job outputs should be compressed or not, maximum number of attempts per task etc.

VIII. TASK EXECUTION & ENVIRONMENT

The **MRAppMaster** works as a parent and Mapper/Reducer task behave as child process. So, the environment of parent process (MRAppMaster) is inherited by the Mapper/Reducer task. The **mapreduce.{map|reduce}.java.opts** is an option/parameter available for user to specify additional options to the child jvm. If mapreduce.{map|reduce}.java.opts has the symbol of @taskid@, then it is incorporated with the value of the MapReduce task's taskid.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

1. Memory Management

User must specify the size of memory in megabytes (MB). The maximum virtual memory of any child process can be set by using **mapreduce.map|reduce}.memory.mb.** This size of virtual memory is a per process limit. The memory assigned to some child process is also configurable.

2. Map Parameters

A record released from a map are serialized into a buffer and information will be hold on into accounting buffers. The following are the options available for Map parameters: -

1.) mapreduce.task.io.sort.mb(int type) – The maximum size in MegaBytes of the buffer storing records released from the map. When the value exceeds this limit, the buffer contents will be sorted and written to disk simultaneously while the map keeps outputting records.
2.) mapreduce.map.sort.spill.percent(float type) – The soft limit of the buffer. A thread will begin to spill the data of disk if this limit is crossed.

IX. JOB SUBMISSION AND MONITORING

A MapReduce job typically divides the input data into independent data-set. These data sets are then processed by the map tasks parallelly. Job performs various functions such as submitting jobs, tracing their progress, reading component-tasks reports and logs etc. Job submission involves several processes such as verifying input and output conditions of job, calculating the InputSplit value for the job, copying the jar and configurations of the job to the MapReduce system directory located on the Filesystem, submitting and monitoring of the jobs.

1. Job Authorization

Job Authorization is performed on the cluster. When it is enabled, access control checks are done by the JobTracker before it allows the submission of jobs to queues. TaskTracker and before client submit the jobs to the JobTracker. And it also checks the TaskTracker before it will complete the process and submit the output to the user.

2. Job Control

To accomplish complicated tasks, users may have to chain several MapReduce jobs. So various job-control options have to be considered: -

1.) Job.submit(): - Returning instantly while submitting a job.
2.) Job.waitForCompletion(Boolean): - Waiting for the job to finish while submitting it.

X. JOB INPUT

InputFormat is an abstract class that specifies and validates the format of input for a MapReduce job. It divides the input files into logical InputSplit instances. These instances are then allocated to specific Mapper. The default InputFormat is the TextInputFormat.

1. InputSplit

InputSplit is a public interface which is inherited from the superinterface “Writable”. InputSplit has the data which is the output of the Mapper. InputSplit has the data in the byte representation.

2. Record Reader

RecordReader retrieves <key, value> pairs from an InputSplit. The RecordReader converts the byte-oriented view of the input to a record-oriented view to the Mapper implementations for processing. Apart from presenting the tasks with keys and values, RecordReader has the responsibility of processing record boundaries.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

XI. JOB OUTPUT

OutputFormat is an abstract class that extends superclass Object. TextOutputFormat is the default OutputFormat. OutputFormat specifies and validates the output format of a MapReduce job by checking whether output directory does not exist. It offers RecordWriter implementation.

1. OutputCommiter

This is the public class which is used to commit the output that is generated by the MapReduce Algorithm. This class has many methods like commitJob, abortJob, abortTask, needsTaskCommit etc.

commitJob is used to commit the job if the job is completed successfully.

abortJob and abortTask is used to delete the job and task which is running right now.

needsTaskCommit is a Boolean method which is used to check whether task needs a commit or not.

2. RecordWriter

RecordWriter writes the output in form of <key,value> to the output file located in FileSystem.

3. Submitting Jobs to Queues

Queues is an abstract data structure. The jobs performed are submitted to Queues. So, queues can be said to be collection of jobs. They provide the feature to system of specifying functionality. Hadoop Schedulers are responsible for submitting jobs to queues.

XII. EXPERIMENTS

If we consider a sample input file as:

Saturn is a planet

Earth is a planet

Pluto is not a planet anymore

The MapReduce algorithm will be processed internally as follows:

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

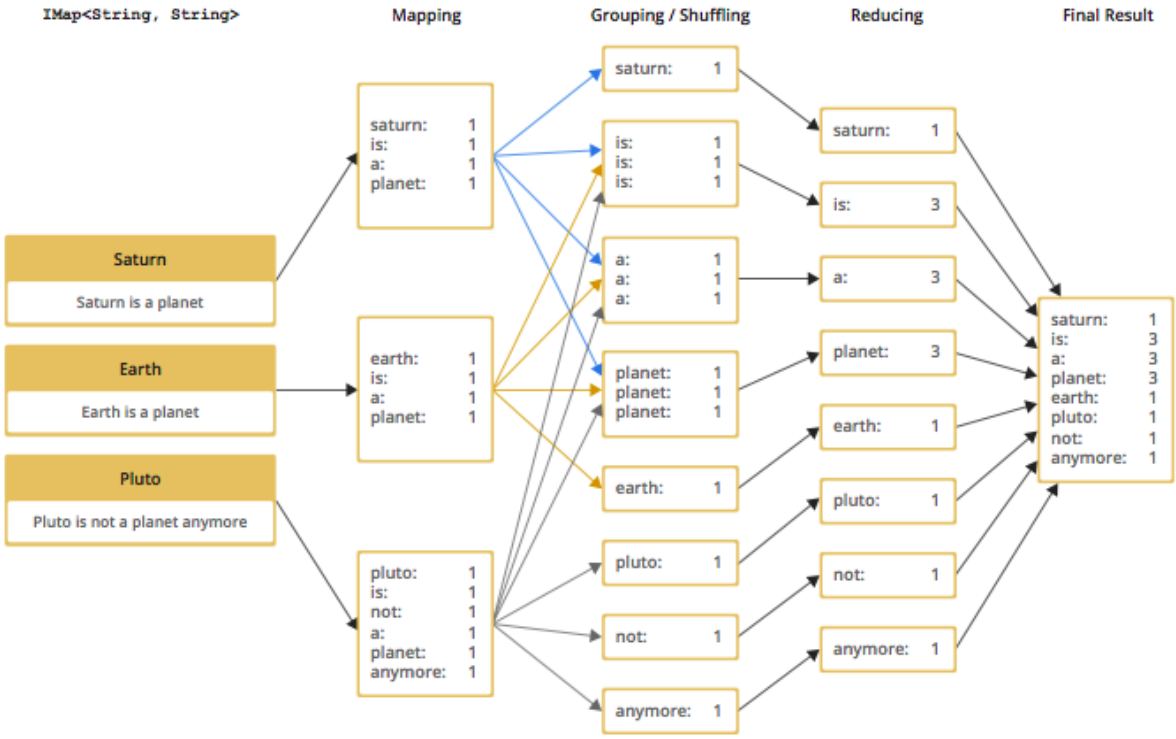

Figure – 2Understanding of the given Example [4]

The Figure - 2 gives us the basic idea about How the input will be, How Map function will work, Shuffle and Reduce function will work and How the Output will generate and in which format.

XIII. CONCLUSION

This is an era of big data. So, it is of utmost importance that we process and preserve that big data. Hadoop MapReduce is an open source software framework that processes big data in fault tolerant and reliable manner. With this framework, it is possible to divide the whole problem into independent parts parallelly. Each part is then mapped to intermediate value using Mapper. The Reducer will perform all the aggregates operation and the final result will be one single output. Thus, Hadoop MapReduce follows Master-Slave architecture which has capacity of processing voluminous data in fast, efficient and consistent manner

REFERENCES

[1] https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[2] https://data-flair.training/blogs/how-hadoop-mapreduce-works/
[3] https://en.wikipedia.org/wiki/MapReduce
[4] https://docs.hazelcast.org/docs/3.5-EA/manual/html/mr-essentials.html
[5] Ch. Shobha Rani et al, "MapReduce with Hadoop for Simplified Analysis of Big Data"International Journal of Advanced Research in Computer Science, 8 (5), May-June 2017,853-856.
[6] Russom, P "Big Data Analytics", The Data Warehousing Institute (TDWI) Best Practices Report, pp.1-40, 2011.
[7] https://www.edureka.co/blog/mapreduce-tutorial/
[8] https://acadgild.com/blog/compiling-and-running-mapreduce-job-from-command-line

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 7, Issue 4, April 2019

[9] Anjan K Koundinya, "Map/Reduce Design and Implementation of AprioriAlgorithm For Handling Voluminous Data-Sets" Advanced Computing: An International Journal (ACIJ), Vol.3, No.6, November 2012.
[10] Huang Lu and Chen Hai-Shan and Hu Ting-Ting, "Research on Hadoop Cloud Computing Model and its Applications" IEEE 2012 Third International Conference on Networking and Distributed Computing, 10.1109/ICNDC.2012.22,Dec 2012.
[11] Tripti Mehta and Neha Mangla "A Survey Paper on Big Data Analytics using Map Reduce and Hive on Hadoop Framework" Special Issue on International Journal of Recent Advances in Engineering & Technology (IJRAET) V-4 I-2 For National Conference on Recent Innovations in Science, Technology & Management (NCRISTM) ISSN (Online): 2347-2812, Gurgaon Institute of Technology and Management, Gurgaon 26th to 27th February 2016.
[12] Bernard Marr (2016).Big Data in Practice: How 45 Successful Companies Used Big Data Analytics to Deliver Extraordinary Results: Wiley.