“Weather Data Analysis using Hadoop and MapReduce”

Innovative Assignment Summary Report

2CS702 Big Data Analytics

By

Jenish Kothari (18BCE096)

Gaurang Kothiya (18BCE097)

Department of Computer Science Engineering,

Institute of Technology,
Nirma University,
Ahmedabad 382481,

November 2021.

I. Abstract

Nowadays analysis of data has become an important aspect. We already have a huge amount of raw data. But data without any meaning is of no use. So in order to understand what the data is, we need to do analysis. We can analyse small amounts of data with our laptops, but nowadays trillions of megabytes of data is being generated every passing minute. In order to do analysis of such a huge amount of data, we need to have extra processing power or some other frameworks to run the algorithms. Hadoop is such a framework which helps us in processing huge amounts of data. Mapreduce algorithm also helps in proper analysis of data. In this paper we will identify how big data can be analysed with the help of Hadoop and MapReduce.

Keywords - Big Data, Hadoop , MapReduce, data

II. Introduction

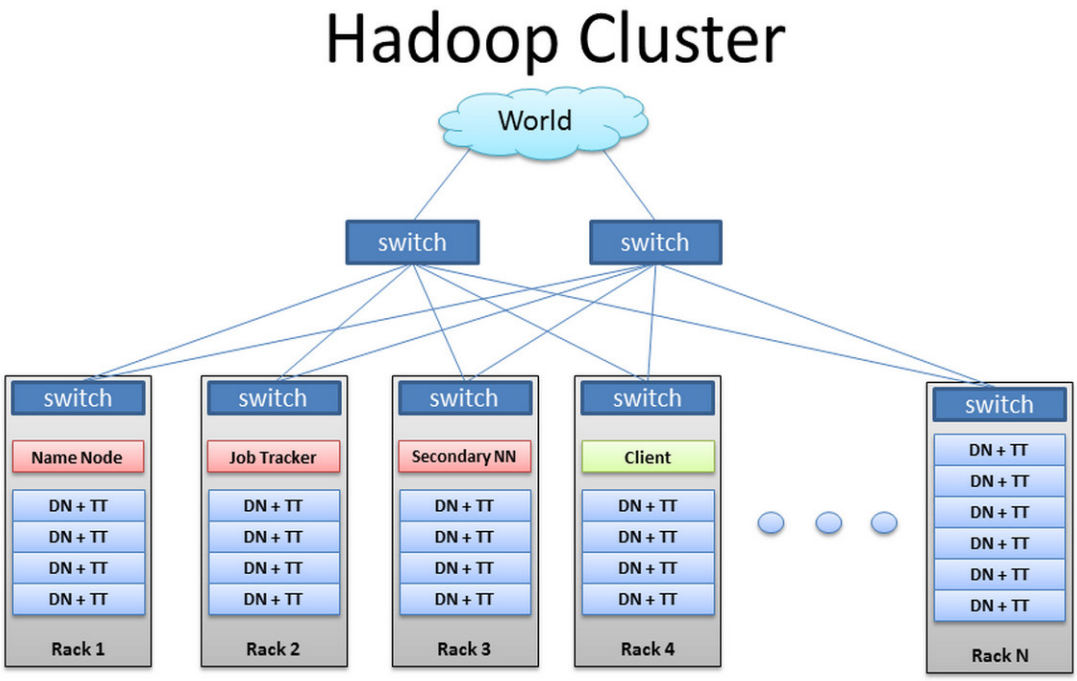
Forecasting is an essential requirement nowadays so as to prepare well in advance for any calamity or a catastrophe. Also there is a huge amount of weather data generated by different machines like Anemometer, Temperature sensors and many other devices. We can slice the data and process it according to the requirements i.e. we can take data between two dates or between hours also. Since the data is huge, normal machines will run out of RAMs and we need more processing power. Hence for processing

such huge data we will use Big Data Tools and Algorithms like Hadoop and Mapreduce.

Here, in this paper, first we find out what is hadoop and what is mapreduce and then in the second part we write an pseudocode or an algorithm to find out cold days in some location using MapReduce algorithm and Hadoop framework.

III. Hadoop

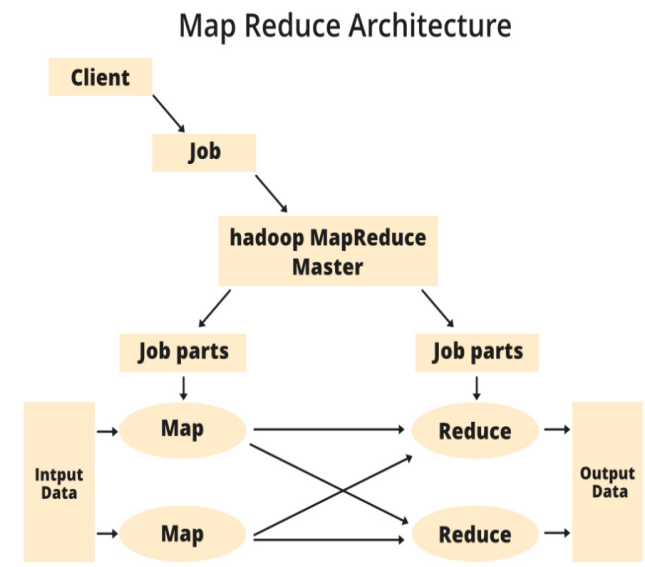
Hadoop and Mapreduce are the most used techniques for analysing Big Data. Hadoop is an open source, distributed big scale data processing framework and it supports distributed processing of data using simple programming models. The project of Apache Hadoop consists of the HDFS and Hadoop Map Reduce in addition to other modules. It is an open-source framework for processing a large amount of data across a group of computers using the high-level languages. These modules provide an easy way of using languages, graphical interfaces and also for managing data on thousands of computers. Hadoop cluster is a set of machines together networked in a location.



Apache Hadoop use cases are many, and show up in many industries, including: risk, fraud and portfolio analysis in financial services; behavior analysis and personalization in retail; social network, relationship and sentiment analysis for marketing; drug interaction modelling and genome data processing in healthcare and life sciences and so on to name a few. In addition, many companies provide Hadoop commercial execution and/or support, including Cloudera, IBM, MapReduce, EMC, and Oracle. According to Gartner Research, Big Data Analytics is a trending topic in 2014. Hadoop is an open framework mostly used for Big Data Analytics.

IV. MapReduce Algorithm

Mapreduce is one of the three components of Hadoop Architecture. HDFS i.e. Hadoop Distributed File System is responsible for storing the file, whereas MapReduce is responsible for processing the file.



Mapreduce task is divided into 2 phases , Map Phase and Reducer Phase :

1. **Map Phase :**
   As the name suggests, the task of a mapper is to map key-value pairs. The Map() function will be executed in the memory repository on each of these input key-value pairs and generates intermediate key-value pairs which works as an input for the Reducer Phase.

2. **Reducer Phase :**
   The intermediate key-value pairs generated from the mapper phase works as an input for the reducer phase. Reducer aggregates or groups the data as per the reducer algorithm specified by the user.

3. **Pseudocode :**

```
Class mapper
method  map(docid   a,  doc d)
                for all w   in  d   do
```

```
                        H   =   associative_array(string    à
integer);
                        for all u   in  neighbours(w)   do
                                H[u]++;
                        emit(w, H);
class   Reducer
        method reduce(term  w,  stripes [H1,    H2, …])
                Hf  =   assoiative_array(string à   integer);
                for all H   in  [H1,    H2, …]  do
                        sum(Hf, H);             //  sum
same-keyed  entries
                emit(w, Hf)
```

V. Implementation Details

Importing Libraries :

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
```

```
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
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;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
```

Mapper Class :

```
public static class MaxTemperatureMapper extends
            Mapper<LongWritable, Text, Text, Text> {

        /**
        * @method map
        * This method takes the input as a text data type.
        * Now leaving the first five tokens, it takes
        * 6th token is taken as temp_max and
        * 7th token is taken as temp_min. Now
        * temp_max > 30 and temp_min < 15 are
        * passed to the reducer.
        */

    // the data in our data set with
    // this value is inconsistent data
    public static final int MISSING = 9999;

    @Override
        public void map(LongWritable arg0, Text Value, Context
context)
                throws IOException, InterruptedException {

        // Convert the single row(Record) to
        // String and store it in String
```

```
        // variable name line

        String line = Value.toString();

            // Check for the empty line
            if (!(line.length() == 0)) {

                // from character 6 to 14 we have
                // the date in our dataset
                String date = line.substring(6, 14);

                // similarly we have taken the maximum
                // temperature from 39 to 45 characters
                float temp_Max = Float.parseFloat(line.substring(39,
45).trim());

                // similarly we have taken the minimum
                // temperature from 47 to 53 characters

                float temp_Min = Float.parseFloat(line.substring(47,
53).trim());

                // if maximum temperature is
                // greater than 30, it is a hot day
                if (temp_Max > 25.0) {

                    // Hot day
                    context.write(new Text("The Day is Hot Day :" +
date),
                                            new
Text(String.valueOf(temp_Max)));
                }

                // if the minimum temperature is
                // less than 15, it is a cold day
                if (temp_Min < 10) {

                    // Cold day
                    context.write(new Text("The Day is Cold Day :" +
date),
```

```
                            new Text(String.valueOf(temp_Min)));
                }
            }
        }

    }
```

Reducer Class :

```
public static class MaxTemperatureReducer extends
            Reducer<Text, Text, Text, Text> {

        /**
        * @method reduce
        * This method takes the input as key and
        * list of values pair from the mapper,
        * it does aggregation based on keys and
        * produces the final context.
        */

        public void reduce(Text Key, Iterator<Text> Values, Context
context)
                throws IOException, InterruptedException {


            // putting all the values in
            // temperature variable of type String
            String temperature = Values.next().toString();
            context.write(Key, new Text(temperature));
        }

    }



    /**
    * @method main
```

```
    * This method is used for setting
    * all the configuration properties.
    * It acts as a driver for map-reduce
    * code.
    */

    public static void main(String[] args) throws Exception {

        // reads the default configuration of the
        // cluster from the configuration XML files
        Configuration conf = new Configuration();

        // Initializing the job with the
        // default configuration of the cluster
        Job job = new Job(conf, "weather example");

        // Assigning the driver class name
        job.setJarByClass(MyMaxMin.class);

        // Key type coming out of mapper
        job.setMapOutputKeyClass(Text.class);

        // value type coming out of mapper
        job.setMapOutputValueClass(Text.class);

        // Defining the mapper class name
        job.setMapperClass(MaxTemperatureMapper.class);

        // Defining the reducer class name
        job.setReducerClass(MaxTemperatureReducer.class);

        // Defining input Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setInputFormatClass(TextInputFormat.class);

        // Defining output Format class which is
        // responsible to parse the dataset
        // into a key value pair
        job.setOutputFormatClass(TextOutputFormat.class);
```

```
        // setting the second argument
        // as a path in a path variable
//      Path OutputPath = new Path(args[1]);

        // Configuring the input path
        // from the filesystem into the job


        FileInputFormat.addInputPath(job, new Path("input"));


        // Configuring the output path from
        // the filesystem into the job
        FileOutputFormat.setOutputPath(job, new Path("output"));

        // deleting the context path automatically
        // from hdfs so that we don't have
        // to delete it explicitly


        // exiting the job only if the
        // flag value becomes false
        System.exit(job.waitForCompletion(true) ? 0 : 1);
```

System Requirements :

RAM : 8 GB
Hadoop Cluster : Single Node
Processor : Intel I5

VI. Conclusion and Future Work

We can also forecast using ML algorithms runned on Hadoop Architecture. As a future work we can also develop algorithms like mapreduce which simplifies the task. Using traditional systems to process millions of records is a lengthy task. The merit of MapReduce with Hadoop will speed up the processing of data,where the volume of data is increasing every day.

VII. References

[1] D. Jayanthi and G. Sumathi, "Weather data analysis using spark — An in-memory computing framework," 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), 2017, pp. 1-5, doi: 10.1109/IPACT.2017.8245142.

[2] V. Suryanarayana, B. S. Sathish, A. Ranganayakulu and P. Ganesan, "Novel Weather Data Analysis Using Hadoop and MapReduce – A Case Study," 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp. 204-207, doi: 10.1109/ICACCS.2019.8728444.

[3] A. K. Pandey, C. P. Agrawal and M. Agrawal, "A hadoop based weather prediction model for classification of weather data," 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2017, pp. 1-5, doi: 10.1109/ICECCT.2017.8117862.

[4] K. A. Ismail, M. Abdul Majid, J. Mohamed Zain and N. A. Abu Bakar, "Big Data prediction framework for weather Temperature based on MapReduce algorithm," 2016 IEEE Conference on Open Systems (ICOS), 2016, pp. 13-17, doi: 10.1109/ICOS.2016.7881981.