****

**BENGALURU INDIA**

# Bigdata Analytics LAB

# B20EA0604

# 6th Semester

**B.Tech Computer Science & engineering**

**SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY**

|  |  |
| --- | --- |
| **Name** |  |
| **SRN** |  |
| **Branch** |  |
| **Semester** |  |
| **Section** |  |
| **Academic Year** |  |

|  |
| --- |
| **Learning Objectives of the Course:** |
| The objectives of this course are to:   1. Provide the knowledge of Map Reduce framework in solving problems related to big data. 2. Provide hands on experience on Hadoop environments. |

|  |
| --- |
| **Learning Outcomes of the Course:** |
| Upon successful completion of the course, students should be able to:   1. Execute MapReduce programs on Hadoop and analyze the results. 2. Conduct some experiments on MapReduce. |

|  |  |  |
| --- | --- | --- |
| **CONTENTS** | | |
| **Sl. No.** | **Lab Sessions** |  |
|  | Installation and Configuration of Hadoop. | 4 |
| 1 | Write a Map Reduce program to solve the problem of word count for different file size. | 5-8 |
| 2 | Write a Map Reduce program to solve the problem of character count. | 9-12 |
| 3 | Write Map Reduce program to sort data by student names based on values. | 13-14 |
| 4 | Write Map Reduce program to process weather dataset and calculate average temperature. | 15-17 |

INTRODUCTION:

Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

**MapReduce:** A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system.

**Install Hadoop**

**Prerequisites:**

**VMWare Workstation**: it is used for installing the operating system on it.

**OPERATING SYSTEM**: You can install Hadoop on Linux-based operating systems. Ubuntu and CentOS are very commonly used. In this tutorial, we are using CentOS.

**JAVA**: You need to install the Java 8 package on your system.

**HADOOP**: You require Hadoop package.

**1. Write a Map Reduce program to solve the problem of word count for**

**Different file size.**

**WC\_Mapper.java**

**import** java.io.IOException;

**import** java.util.StringTokenizer;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.Mapper;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reporter;

**public** **class** WC\_Mapper **extends** MapReduceBase **implements** Mapper<LongWritable,Text,Text,IntWritable>{

**private** **final** **static** IntWritable ***one*** = **new** IntWritable(1);

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

**public** **void** map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,

Reporter reporter) **throws** IOException{

String line = value.toString();

StringTokenizer tokenizer = **new** StringTokenizer(line);

**while** (tokenizer.hasMoreTokens()){

word.set(tokenizer.nextToken());

output.collect(word, ***one***);

}

}

}

**WC\_Reducer.java**

**import** java.io.IOException;

**import** java.util.Iterator;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reducer;

**import** org.apache.hadoop.mapred.Reporter;

**public** **class** WC\_Reducer **extends** MapReduceBase **implements**

Reducer<Text,IntWritable,Text,IntWritable> {

**public** **void** reduce(Text key, Iterator<IntWritable>

values,OutputCollector<Text,IntWritable> output,

Reporter reporter) **throws** IOException {

**int** sum=0;

**while** (values.hasNext()) {

sum+=values.next().get();

}

output.collect(key,**new** IntWritable(sum));

}

}

**WC\_Runner.java**

**import** java.io.IOException;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.FileInputFormat;

**import** org.apache.hadoop.mapred.FileOutputFormat;

**import** org.apache.hadoop.mapred.JobClient;

**import** org.apache.hadoop.mapred.JobConf;

**import** org.apache.hadoop.mapred.TextInputFormat;

**import** org.apache.hadoop.mapred.TextOutputFormat;

**public** **class** WC\_Runner {

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

JobConf conf = **new** JobConf(WC\_Runner.**class**);

conf.setJobName("WordCount");

conf.setOutputKeyClass(Text.**class**);

conf.setOutputValueClass(IntWritable.**class**);

conf.setMapperClass(WC\_Mapper.**class**);

conf.setCombinerClass(WC\_Reducer.**class**);

conf.setReducerClass(WC\_Reducer.**class**);

conf.setInputFormat(TextInputFormat.**class**);

conf.setOutputFormat(TextOutputFormat.**class**);

FileInputFormat.*setInputPaths*(conf,**new** Path("input.txt"));

FileOutputFormat.*setOutputPath*(conf,**new** Path("output"));

JobClient.*runJob*(conf);

}

}

**INPUT:**

Reva University is in Bangalore

Bangalore is in Karnataka

**OUTPUT:**

Bangalore,2

in, 2

is,2

Karnataka, 1

Reva, 1

University, 1

**2. Write a Map Reduce program to solve the problem of Character count.**

**WC\_Mapper.java**

**import** java.io.IOException;

**import** java.util.StringTokenizer;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.Mapper;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reporter;

**public** **class** WC\_Mapper **extends** MapReduceBase **implements** Mapper<LongWritable,Text,Text,IntWritable> {

**public** **void** map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,

Reporter reporter) **throws** IOException{

String line = value.toString();

String tokenizer[] = line.split("");

**for**(String SingleChar : tokenizer)

{

Text charKey = **new** Text(SingleChar);

IntWritable One = **new** IntWritable(1);

output.collect(charKey, One);

}

}

}

**WC\_Reducer.java**

**import** java.io.IOException;

**import** java.util.Iterator;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.MapReduceBase;

**import** org.apache.hadoop.mapred.OutputCollector;

**import** org.apache.hadoop.mapred.Reducer;

**import** org.apache.hadoop.mapred.Reporter;

**public** **class** WC\_Reducer **extends** MapReduceBase **implements**

Reducer<Text,IntWritable,Text,IntWritable> {

**public** **void** reduce(Text key, Iterator<IntWritable>

values,OutputCollector<Text,IntWritable> output,

Reporter reporter) **throws** IOException {

**int** sum=0;

**while** (values.hasNext()) {

sum+=values.next().get();

}

output.collect(key,**new** IntWritable(sum));

}

}

**WC\_Runner.java**

**import** java.io.IOException;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.IntWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapred.FileInputFormat;

**import** org.apache.hadoop.mapred.FileOutputFormat;

**import** org.apache.hadoop.mapred.JobClient;

**import** org.apache.hadoop.mapred.JobConf;

**import** org.apache.hadoop.mapred.TextInputFormat;

**import** org.apache.hadoop.mapred.TextOutputFormat;

**public** **class** WC\_Runner {

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

JobConf conf = **new** JobConf(WC\_Runner.**class**);

conf.setJobName("CharCount");

conf.setOutputKeyClass(Text.**class**);

conf.setOutputValueClass(IntWritable.**class**);

conf.setMapperClass(WC\_Mapper.**class**);

conf.setCombinerClass(WC\_Reducer.**class**);

conf.setReducerClass(WC\_Reducer.**class**);

conf.setInputFormat(TextInputFormat.**class**);

conf.setOutputFormat(TextOutputFormat.**class**);

FileInputFormat.*setInputPaths*(conf,**new** Path("input.txt"));

FileOutputFormat.*setOutputPath*(conf,**new** Path("output"));

JobClient.*runJob*(conf);

}

}

**INPUT:**

REVA UNIVERSITY

**OUTPUT:**

,1 //Number of lines

,1 //Number of white spaces

A,1

E,2

I,2

N,1

R,2

S,1

T,1

U,1

V,2

Y,1

**3. Write a Map reduce program to sort data by student name based on values.**

**SortStudNames.java**

**Java project name: SortStudNames**

**Class name: SortStudNames.java**

**import** java.io.IOException;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.fs.Path;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.NullWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.Mapper;

**import** org.apache.hadoop.mapreduce.Reducer;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

**public** **class** SortStudNames {

**public** **static** **class** SortMapper **extends** Mapper <LongWritable, Text, Text, Text >

{

**protected** **void** map(LongWritable key,Text value, Context context) **throws** IOException, InterruptedException {

String[] token = value.toString().split(",");

context.write(**new** Text(token[1]), **new** Text(token[0]+ "-" +token[1]));

}

}

**public** **static** **class** SortReducer **extends** Reducer <Text, Text, NullWritable, Text>

{

**public** **void** reduce(Text key, Iterable <Text> values, Context context) **throws** IOException, InterruptedException {

**for** (Text details : values )

{

context.write(NullWritable.*get*(), details);

}

}

}

**public** **static** **void** main (String[] args) **throws** IOException, InterruptedException, ClassNotFoundException

{

Configuration conf = **new** Configuration();

Job job = **new** Job(conf);

job.setJarByClass(SortStudNames.**class**);

job.setMapperClass(SortMapper.**class**);

job.setReducerClass(SortReducer.**class**);

job.setOutputKeyClass(Text.**class**);

job.setOutputValueClass(Text.**class**);

FileInputFormat.*setInputPaths*(job,**new** Path("input.csv"));

FileOutputFormat.*setOutputPath*(job,**new** Path("output"));

System.*exit*(job.waitForCompletion(**true**)? 0:1);

}

}

**INPUT:**

1001,Sahana

1003,Banu

1005,Deepa

1007,Akshay

**OUTPUT:**

1007,Akshay

1003,Banu

1005,Deepa

1001,Sahana

**4. MapReduce program in Java for processing a weather dataset and calculate the average temperature of a particular year.**

**Project Name:WeatherMapReduce**

**Class Name:WeatherMapReduce**

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

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;

public class WeatherMapReduce {

public static class WeatherMapper

extends Mapper<LongWritable, Text, Text, DoubleWritable>{

private final static DoubleWritable temperature = new DoubleWritable();

private Text date = new Text();

public void map(LongWritable key, Text value, Context context

) throws IOException, InterruptedException {

String[] line = value.toString().split(",");

if (line.length == 3) {

date.set(line[0]);

temperature.set(Double.parseDouble(line[2]));

context.write(date, temperature);

}

}

}

public static class WeatherReducer

extends Reducer<Text,DoubleWritable,Text,DoubleWritable> {

private DoubleWritable result = new DoubleWritable();

public void reduce(Text key, Iterable<DoubleWritable> values,

Context context

) throws IOException, InterruptedException {

double sum = 0;

int count = 0;

for (DoubleWritable val : values) {

sum += val.get();

count++;

}

double avg = sum / count;

result.set(avg);

context.write(key, result);

}

}

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

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "weather analysis");

job.setJarByClass(WeatherMapReduce.class);

job.setMapperClass(WeatherMapper.class);

job.setReducerClass(WeatherReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(DoubleWritable.class);

FileInputFormat.setInputPaths(job, new Path("input.txt"));

FileOutputFormat.setOutputPath(job, new Path("output"));

System.exit(job.waitForCompletion(true)?0 : 1);

}

}

INPUT:

Input.txt

2022-01-01,New York,32

2022-01-01,Los Angeles,55

2022-01-02,New York,30

2022-01-02,Los Angeles,58

2022-01-03,New York,35

2022-01-03,Los Angeles,60

2022-01-04,New York,28

2022-01-04,Los Angeles,56

2022-01-05,New York,33

2022-01-05,Los Angeles,61

2022-01-06,New York,29

2022-01-06,Los Angeles,59

2022-01-07,New York,36

2022-01-07,Los Angeles,57

Output:

2022-01-01 43.5

2022-01-02 44.0

2022-01-03 47.5

2022-01-04 42.0

2022-01-05 47.0

2022-01-06 44.0

2022-01-07 46.5