**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



**LAB REPORT on**

**BIG DATA ANALYTICS (20CS6PEBDA)**

***Submitted by***

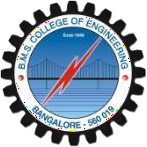
**Manoj H A(1BM19CS083)**

***in partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING BENGALURU-560019 May-2022 to July-2022**

**(Autonomous Institution under VTU)**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “**BIG DATA ANALYTICS**” carried out by **Manoj H A(1BM19CS073),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of Big data analytics -(20CS6PEBDA) work prescribed for the said degree.

Name of the Lab-In charge ANTARA ROY CHOUDHURY Designation Assistant Professor

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index Sheet**

|  |  |  |
| --- | --- | --- |
| **Sl.**  **No.** | **Experiment Title** | **Page No.** |
| **5** | **Hadoop Programs: Word Count** |  |
| **6** | **Hadoop Programs: Top N** |  |
| **7** | **Hadoop Programs: Average Temperature** |  |
| **8** | **Hadoop Programs: Join** |  |
| **9** | **Scala Programs: Word Count** |  |
| **10** | **Scala Programs: Word Count greater than 4** |  |

**Course Outcome**

|  |  |
| --- | --- |
| CO1 | Apply the concept of NoSQL, Hadoop or Spark for a given task |
| CO2 | Analyze the Big Data and obtain insight using data analytics mechanisms. |
| CO3 | Design and implement Big data applications by applying NoSQL, Hadoop or Spark |

**Hadoop Commands**

bmsce@bmsce-Precision-T1700:~$ sudo su hduser

[sudo] password for bmsce:

hduser@bmsce-Precision-T1700:/home/bmsce$ start-all.sh

This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh

Starting namenodes on [localhost]

hduser@localhost's password:

localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-bmsce-Precision-T1700.out

bmhduser@localhost's password:

localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-bmsce-Precision-T1700.out

Starting secondary namenodes [0.0.0.0]

hduser@0.0.0.0's password:

0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-hduser-secondarynamenode-bmsce-Precision-T1700.out

starting yarn daemons

starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-bmsce-Precision-T1700.out

hduser@localhost's password:

localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-bmsce-Precision-T1700.out

hduser@bmsce-Precision-T1700:/home/bmsce$ jps

5489 ResourceManager

5107 DataNode

5319 SecondaryNameNode

4935 NameNode

5944 Jps

5821 NodeManager

hduser@bmsce-Precision-T1700:/home/bmsce$ hdfs dfs -mkdir /max

hduser@bmsce-Precision-T1700:/home/bmsce$ hadoop fs -ls/

-ls/: Unknown command

hduser@bmsce-Precision-T1700:/home/bmsce$ hadoop fs -ls /

Found 19 items

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:10 /FFF

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:59 /LLL

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:04 /Welcome

drwxr-xr-x - hduser supergroup 0 2022-06-04 10:17 /abc

drwxr-xr-x - hduser supergroup 0 2022-06-04 10:18 /abc1

drwxr-xr-x - hduser supergroup 0 2022-06-01 09:44 /cs185

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:58 /cse

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:04 /dishagubald

drwxr-xr-x - hduser supergroup 0 2022-05-31 10:35 /duplicate

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:03 /file1

drwxr-xr-x - hduser supergroup 0 2022-06-06 14:23 /max

drwxr-xr-x - hduser supergroup 0 2022-06-01 14:56 /hello

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:40 /new

drwxr-xr-x - hduser supergroup 0 2022-05-31 10:28 /praveen138

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:33 /sajjan

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:37 /sajjan2

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:03 /test

drwxrwxr-x - hduser supergroup 0 2019-08-01 16:19 /tmp

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:19 /user

hduser@bmsce-Precision-T1700:/home/bmsce$ hdfs dfs -put /home/hduser/Desktop/Welcome.txt /max/WC.txt

hduser@bmsce-Precision-T1700:/home/bmsce$ hdfs dfs -cat /max/WC.txt

yo yo honey singh

hduser@bmsce-Precision-T1700:~/Desktop$ cat Welcome.txt

yo yo honey singh

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -put /home/hduser/Desktop/Welcome.txt /max/WC1.txt

put: `/max/WC1.txt': File exists

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -put /home/hduser/Desktop/Welcome.txt /max/WC21.txt

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -cat /max/WC21.txt

\yo yo honey singh

hduser@bmsce-Precision-T1700:~/Desktop$ hadoop fs -ls /max

Found 3 items

-rw-r--r-- 1 hduser supergroup 19 2022-06-06 14:28 /max/WC.txt

-rw-r--r-- 1 hduser supergroup 19 2022-06-06 14:44 /max/WC1.txt

-rw-r--r-- 1 hduser supergroup 19 2022-06-06 14:51 /max/WC21.txt

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -get /max/WC.txt /home/hduser/Downloads/WWC.txt

get: `/home/hduser/Downloads/WWC.txt': File exists

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -get /max/WC.txt /home/hduser/Downloads/WWE.txt

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -getmerge /max/WC1.txt /max/WC21.txt /home/hduser/Desktop/new.txt

hduser@bmsce-Precision-T1700:~/Desktop$ cat new.txt

yo yo honey singh

yo yo honey singh

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -copyToLocal /max/WC1.txt /home/hduser/Desktop

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -cat /max/WC1.txt

yo yo honey singh

hduser@bmsce-Precision-T1700:~/Desktop$ hadoop fs -mv /max /vj

hduser@bmsce-Precision-T1700:~/Desktop$ hadoop fs -ls /vj

Found 3 items

-rw-r--r-- 1 hduser supergroup 19 2022-06-06 14:28 /vj/WC.txt

-rw-r--r-- 1 hduser supergroup 19 2022-06-06 14:44 /vj/WC1.txt

-rw-r--r-- 1 hduser supergroup 19 2022-06-06 14:51 /vj/WC21.txt

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs -cp /CSE/ /Manoj

Error: Could not find or load main class .Manoj

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -ls/

-ls/: Unknown command

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs dfs -ls /

Found 19 items

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:10 /FFF

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:59 /LLL

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:04 /Welcome

drwxr-xr-x - hduser supergroup 0 2022-06-04 10:17 /abc

drwxr-xr-x - hduser supergroup 0 2022-06-04 10:18 /abc1

drwxr-xr-x - hduser supergroup 0 2022-06-01 09:44 /cs185

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:58 /cse

drwxr-xr-x - hduser supergroup 0 2022-06-03 15:04 /dishagubald

drwxr-xr-x - hduser supergroup 0 2022-05-31 10:35 /duplicate

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:03 /file1

drwxr-xr-x - hduser supergroup 0 2022-06-01 14:56 /hello

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:40 /new

drwxr-xr-x - hduser supergroup 0 2022-05-31 10:28 /praveen138

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:33 /sajjan

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:37 /sajjan2

drwxr-xr-x - hduser supergroup 0 2022-06-01 15:03 /test

drwxrwxr-x - hduser supergroup 0 2019-08-01 16:19 /tmp

drwxr-xr-x - hduser supergroup 0 2022-06-03 12:19 /user

drwxr-xr-x - hduser supergroup 0 2022-06-06 14:51 /vj

hduser@bmsce-Precision-T1700:~/Desktop$ hdfs -cp /CSE/ /LLL

Error: Could not find or load main class .LLL

hduser@bmsce-Precision-T1700:~/Desktop$ hadoop fs -cp /cse/ /LLL

hduser@bmsce-Precision-T1700:~/Desktop$ hadoop fs -ls /LLL

Found 3 items

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:59 /LLL/FFF

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:59 /LLL/LLL

drwxr-xr-x - hduser supergroup 0 2022-06-06 12:59 /LLL/cse

hduser@bmsce-Precision-T1700:~/Desktop$

Hadoop Programs

1. Word Count

WCMapper Java Class file.

// Importing libraries

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

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

public class WCMapper extends MapReduceBase implements Mapper<LongWritable,

Text, Text, IntWritable> {

// Map function

public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter rep) throws IOException

{

String line = value.toString();

// Splitting the line on spaces for (String word : line.split(" "))

{

if (word.length() > 0)

{

output.collect(new Text(word), new IntWritable(1));

} } } }

Reducer Code

// Importing libraries

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 WCReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

// Reduce function

public void reduce(Text key, Iterator<IntWritable> value, OutputCollector<Text, IntWritable> output,

Reporter rep) throws IOException

{

int count = 0;

// Counting the frequency of each words while (value.hasNext())

{

IntWritable i = value.next(); count += i.get();

}

output.collect(key, new IntWritable(count));

}

}

Driver Code:

// Importing libraries

import java.io.IOException;

import org.apache.hadoop.conf.Configured; 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.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class WCDriver extends Configured implements Tool { public int run(String args[]) throws IOException

{

if (args.length < 2)

{

System.out.println("Please give valid inputs"); return -1;

}

JobConf conf = new JobConf(WCDriver.class); FileInputFormat.setInputPaths(conf, new Path(args[0])); FileOutputFormat.setOutputPath(conf, new Path(args[1])); conf.setMapperClass(WCMapper.class); conf.setReducerClass(WCReducer.class); conf.setMapOutputKeyClass(Text.class); conf.setMapOutputValueClass(IntWritable.class); conf.setOutputKeyClass(Text.class); conf.setOutputValueClass(IntWritable.class); JobClient.runJob(conf);

return 0;

}

// Main Method

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

{

int exitCode = ToolRunner.run(new WCDriver(), args); System.out.println(exitCode);

}

}

Output :

hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /input\_manoj

Found 2 items

drwxr-xr-x - hduser supergroup 0 2022-06-20 15:16 /input\_manoj/output\_manoj

-rw-r--r-- 1 hduser supergroup 52 2022-06-20 15:15 /input\_manoj/sample.txt

hduser@bmsce-Precision-T1700:~$ hdfs dfs -ls /input\_manoj/output\_manoj

Found 2 items

-rw-r--r-- 1 hduser supergroup 0 2022-06-20 15:16 /input\_manoj/output\_manoj/\_SUCCESS

-rw-r--r-- 1 hduser supergroup 63 2022-06-20 15:16 /input\_manoj/output\_manoj/part-00000

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /input\_manoj/output\_manoj/part-0000

cat: `/input\_khushil/output\_khushil/part-0000': No such file or directory

hduser@bmsce-Precision-T1700:~$ hdfs dfs -cat /input\_manoj/output\_manoj/part-00000

am 1

awesome 1

hadoop 2

hi 1

i 1

im 1

is 1

manoj 1

learing 1

1. Top N

Driver-TopN.class

**package** samples.topn;

**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.IntWritable;

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

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

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

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat; **import** org.apache.hadoop.mapreduce.lib.output.FileOutputFormat; **import** org.apache.hadoop.util.GenericOptionsParser;

**public class** TopN {

**public static void** main(String[] args) **throws** Exception { Configuration conf = **new** Configuration();

String[] otherArgs = (**new** GenericOptionsParser(conf, args)).getRemainingArgs();

**if** (otherArgs.length != 2) { System.err.println("Usage: TopN <in> <out>"); System.exit(2);

}

Job job = Job.getInstance(conf); job.setJobName("Top N"); job.setJarByClass(TopN.**class**); job.setMapperClass(TopNMapper.**class**); job.setReducerClass(TopNReducer.**class**); job.setOutputKeyClass(Text.**class**); job.setOutputValueClass(IntWritable.**class**);

FileInputFormat.addInputPath(job, **new** Path(otherArgs[0])); FileOutputFormat.setOutputPath(job, **new**

Path(otherArgs[1])); System.exit(job.waitForCompletion(**true**) ? 0 : 1);

}

**public static class** TopNMapper **extends** Mapper<Object, Text,

Text, IntWritable> {

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

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

**private** String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-

:()?!\"']";

**public void** map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) **throws** IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(**this**.tokens, " ");

StringTokenizer itr = **new** StringTokenizer(cleanLine);

**while** (itr.hasMoreTokens()) { **this**.word.set(itr.nextToken().trim()); context.write(**this**.word, one);

}

}

}

}

TopNCombiner.class

**package** samples.topn;

**import** java.io.IOException;

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

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

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

**public class** TopNCombiner **extends** Reducer<Text, IntWritable, Text, IntWritable> {

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

**int** sum = 0;

**for** (IntWritable val : values) sum += val.get();

context.write(key, **new** IntWritable(sum));

}

}

TopNMapper.class

**package** samples.topn;

**import** java.io.IOException;

**import** java.util.StringTokenizer;

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

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

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

**public class** TopNMapper **extends** Mapper<Object, Text, Text, IntWritable> {

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

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

**private** String tokens = "[\_|$#<>\\^=\\[\\]\\\*/\\\\,;,.\\-

:()?!\"']";

**public vo```\\id** map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context) **throws** IOException, InterruptedException {

String cleanLine = value.toString().toLowerCase().replaceAll(**this**.tokens, " ");

StringTokenizer itr = **new** StringTokenizer(cleanLine);

**while** (itr.hasMoreTokens()) { **this**.word.set(itr.nextToken().trim()); context.write(**this**.word, one);

}

}

}

TopNReducer.class

**package** samples.topn;

**import** java.io.IOException;

**import** java.util.HashMap;

**import** java.util.Map;

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

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

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

**import** utils.MiscUtils;

**public class** TopNReducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {

**private** Map<Text, IntWritable> countMap = **new** HashMap<>();

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

**int** sum = 0;

**for** (IntWritable val : values) sum += val.get();

**this**.countMap.put(**new** Text(key), **new** IntWritable(sum));

}

**protected void** cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context) **throws** IOException, InterruptedException {

Map<Text, IntWritable> sortedMap = MiscUtils.sortByValues(**this**.countMap);

**int** counter = 0;

**for** (Text key : sortedMap.keySet()) {

**if** (counter++ == 20)

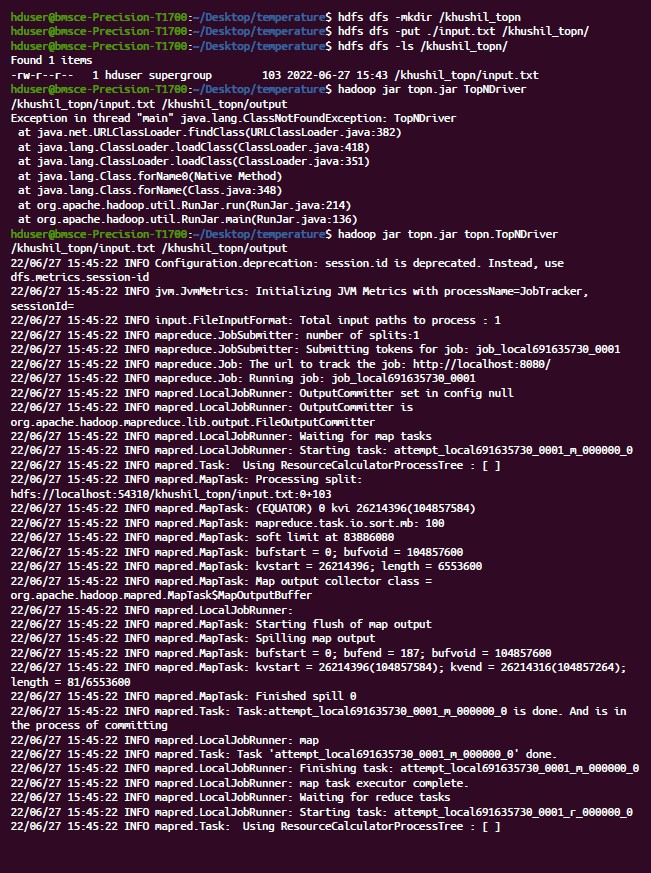
**break**;

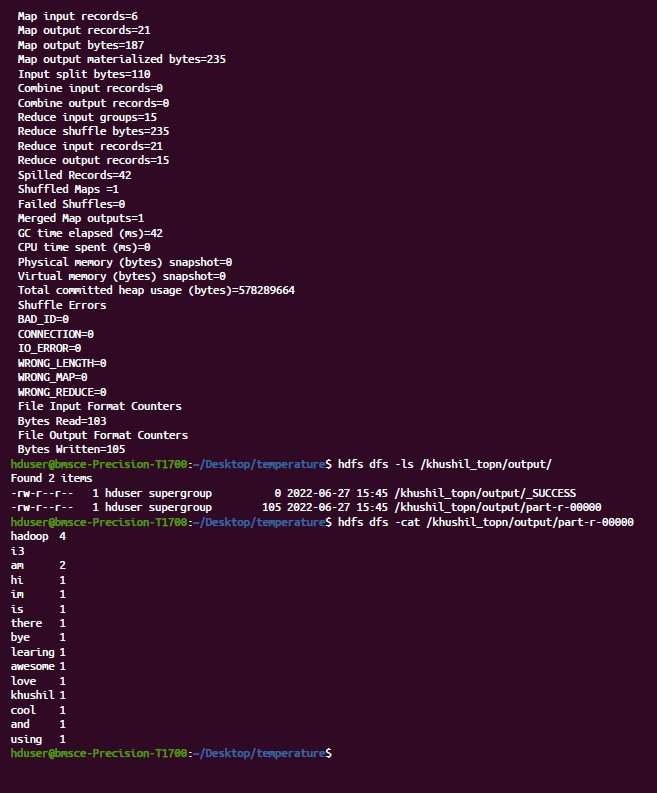
context.write(key, sortedMap.get(key));

}

}

}

Output:



1. Average Temperature

AverageDriver

**package** temp;

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

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

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

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

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

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

**public class** AverageDriver {

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

**if** (args.length != 2) {

System.err.println("Please Enter the input and output parameters");

System.exit(-1);

}

Job job = **new** Job(); job.setJarByClass(AverageDriver.**class**); job.setJobName("Max temperature"); FileInputFormat.addInputPath(job, **new** Path(args[0])); FileOutputFormat.setOutputPath(job, **new** Path(args[1])); job.setMapperClass(AverageMapper.**class**); job.setReducerClass(AverageReducer.**class**); job.setOutputKeyClass(Text.**class**); job.setOutputValueClass(IntWritable.**class**); System.exit(job.waitForCompletion(**true**) ? 0 : 1);

}

}

**AverageMapper**

**package** temp;

**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** AverageMapper **extends** Mapper<LongWritable, Text, Text, IntWritable> {

**public static final int** MISSING = 9999;

**public void** map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context) **throws** IOException, InterruptedException {

**int** temperature;

String line = value.toString(); String year = line.substring(15, 19); **if** (line.charAt(87) == '+') {

temperature = Integer.parseInt(line.substring(88, 92));

} **else** {

temperature = Integer.parseInt(line.substring(87, 92));

}

String quality = line.substring(92, 93);

**if** (temperature != 9999 && quality.matches("[01459]")) context.write(**new** Text(year), **new**

IntWritable(temperature));

}

}

AverageReducer

**package** temp;

**import** java.io.IOException;

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

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

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

**public class** AverageReducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {

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

**int** max\_temp = 0;

**int** count = 0;

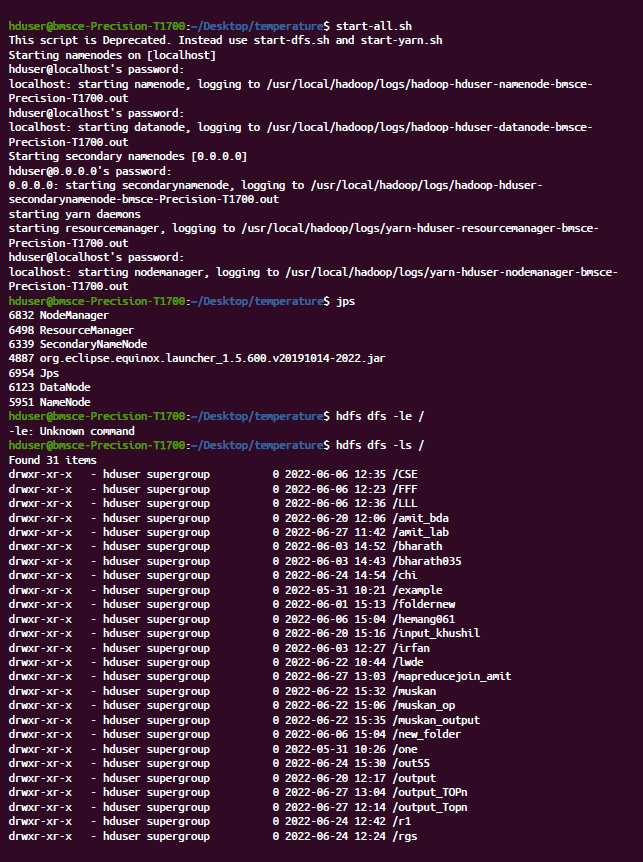
**for** (IntWritable value : values) { max\_temp += value.get(); count++;

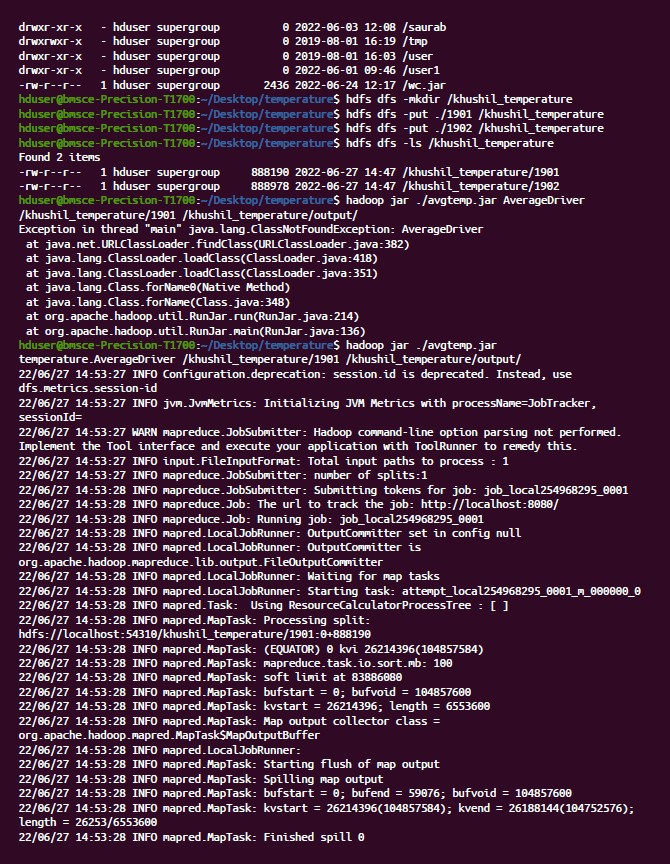
}

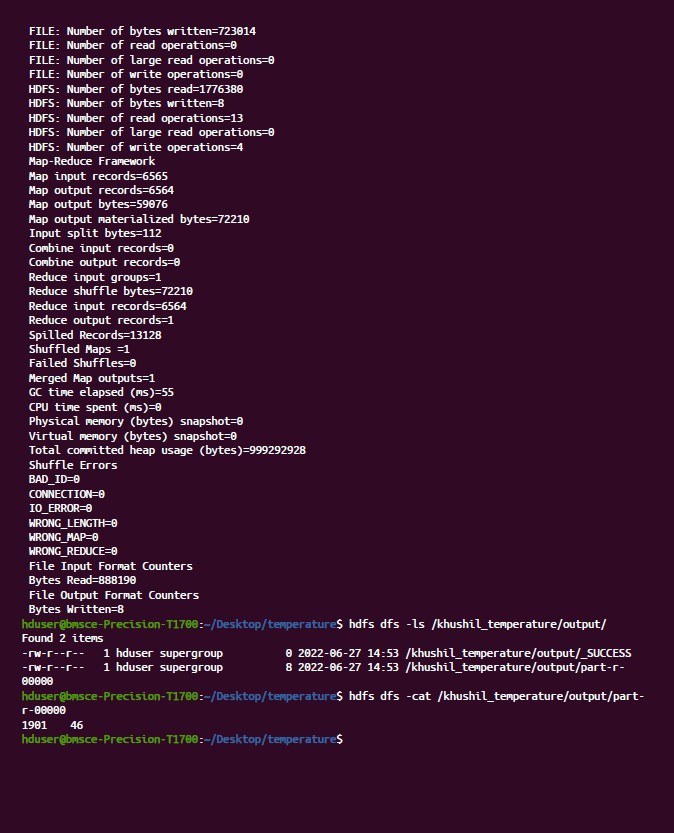
context.write(key, **new** IntWritable(max\_temp / count));

}

}

Output:





1. Join

// JoinDriver.java

import org.apache.hadoop.conf.Configured; import org.apache.hadoop.fs.Path;

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

import org.apache.hadoop.mapred.lib.MultipleInputs; import org.apache.hadoop.util.\*;

public class JoinDriver extends Configured implements Tool {

public static class KeyPartitioner implements Partitioner<TextPair, Text> { @Override

public void configure(JobConf job) {

}

@Override

public int getPartition(TextPair key, Text value, int numPartitions) { return (key.getFirst().hashCode() & Integer.MAX\_VALUE) %

numPartitions;

}

}

@Override

public int run(String[] args) throws Exception { if (args.length != 3) {

System.out.println("Usage: <Department Emp Strength input>

<Department Name input> <output>"); return -1;

}

JobConf conf = new JobConf(getConf(), getClass());

conf.setJobName("Join 'Department Emp Strength input' with 'Department Name input'");

Path AInputPath = new Path(args[0]); Path BInputPath = new Path(args[1]); Path outputPath = new Path(args[2]);

MultipleInputs.addInputPath(conf, AInputPath, TextInputFormat.class,

Posts.class);

MultipleInputs.addInputPath(conf, BInputPath, TextInputFormat.class, User.class);

FileOutputFormat.setOutputPath(conf, outputPath); conf.setPartitionerClass(KeyPartitioner.class); conf.setOutputValueGroupingComparator(TextPair.FirstComparator.class); conf.setMapOutputKeyClass(TextPair.class); conf.setReducerClass(JoinReducer.class); conf.setOutputKeyClass(Text.class);

JobClient.runJob(conf);

return 0;

}

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

int exitCode = ToolRunner.run(new JoinDriver(), args); System.exit(exitCode);

}

}

// JoinReducer.java

import java.io.IOException; import java.util.Iterator;

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

public class JoinReducer extends MapReduceBase implements Reducer<TextPair, Text, Text, Text> {

@Override

public void reduce (TextPair key, Iterator<Text> values, OutputCollector<Text, Text> output, Reporter reporter)

throws IOException

{

Text nodeId = new Text(values.next()); while (values.hasNext()) {

Text node = values.next();

Text outValue = new Text(nodeId.toString() + "\t\t" + node.toString()); output.collect(key.getFirst(), outValue);

}

}

}

// User.java

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FSDataInputStream; import org.apache.hadoop.fs.FSDataOutputStream; import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

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

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

public class User extends MapReduceBase implements Mapper<LongWritable, Text, TextPair, Text> {

@Override

public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output, Reporter reporter)

throws IOException

{

String valueString = value.toString();

String[] SingleNodeData = valueString.split("\t"); output.collect(new TextPair(SingleNodeData[0], "1"), new

Text(SingleNodeData[1]));

}

}

// Posts.java

import java.io.IOException;

import org.apache.hadoop.io.\*; import org.apache.hadoop.mapred.\*;

public class Posts extends MapReduceBase implements Mapper<LongWritable, Text, TextPair, Text> {

@Override

public void map(LongWritable key, Text value, OutputCollector<TextPair, Text> output, Reporter reporter)

throws IOException

{

String valueString = value.toString();

String[] SingleNodeData = valueString.split("\t"); output.collect(new TextPair(SingleNodeData[3], "0"), new

Text(SingleNodeData[9]));

}

}

// TextPair.java import java.io.\*;

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

public class TextPair implements WritableComparable<TextPair> { private Text first;

private Text second;

public TextPair() {

set(new Text(), new Text());

}

public TextPair(String first, String second) { set(new Text(first), new Text(second));

}

public TextPair(Text first, Text second) { set(first, second);

}

public void set(Text first, Text second) { this.first = first;

this.second = second;

}

public Text getFirst() { return first;

}

public Text getSecond() { return second;

}

@Override

public void write(DataOutput out) throws IOException { first.write(out);

second.write(out);

}

@Override

public void readFields(DataInput in) throws IOException { first.readFields(in);

second.readFields(in);

}

@Override

public int hashCode() {

return first.hashCode() \* 163 + second.hashCode();

}

@Override

public boolean equals(Object o) { if (o instanceof TextPair) {

TextPair tp = (TextPair) o;

return first.equals(tp.first) && second.equals(tp.second);

}

return false;

}

@Override

public String toString() { return first + "\t" + second;

}

@Override

public int compareTo(TextPair tp) { int cmp = first.compareTo(tp.first); if (cmp != 0) {

return cmp;

}

return second.compareTo(tp.second);

}

// ^^ TextPair

// vv TextPairComparator

public static class Comparator extends WritableComparator {

private static final Text.Comparator TEXT\_COMPARATOR = new Text.Comparator(); public Comparator() {

super(TextPair.class);

}

@Override

public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {

try {

int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1); int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2);

int cmp = TEXT\_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2); if (cmp != 0) {

return cmp;

}

return TEXT\_COMPARATOR.compare(b1, s1 + firstL1, l1 - firstL1,

b2, s2 + firstL2, l2 - firstL2);

} catch (IOException e) {

throw new IllegalArgumentException(e);

}

}

}

static {

WritableComparator.define(TextPair.class, new Comparator());

}

public static class FirstComparator extends WritableComparator {

private static final Text.Comparator TEXT\_COMPARATOR = new Text.Comparator(); public FirstComparator() {

super(TextPair.class);

}

@Override

public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2) {

try {

int firstL1 = WritableUtils.decodeVIntSize(b1[s1]) + readVInt(b1, s1); int firstL2 = WritableUtils.decodeVIntSize(b2[s2]) + readVInt(b2, s2); return TEXT\_COMPARATOR.compare(b1, s1, firstL1, b2, s2, firstL2);

} catch (IOException e) {

throw new IllegalArgumentException(e);

}

}

@Override

public int compare(WritableComparable a, WritableComparable b) { if (a instanceof TextPair && b instanceof TextPair) {

return ((TextPair) a).first.compareTo(((TextPair) b).first);

}

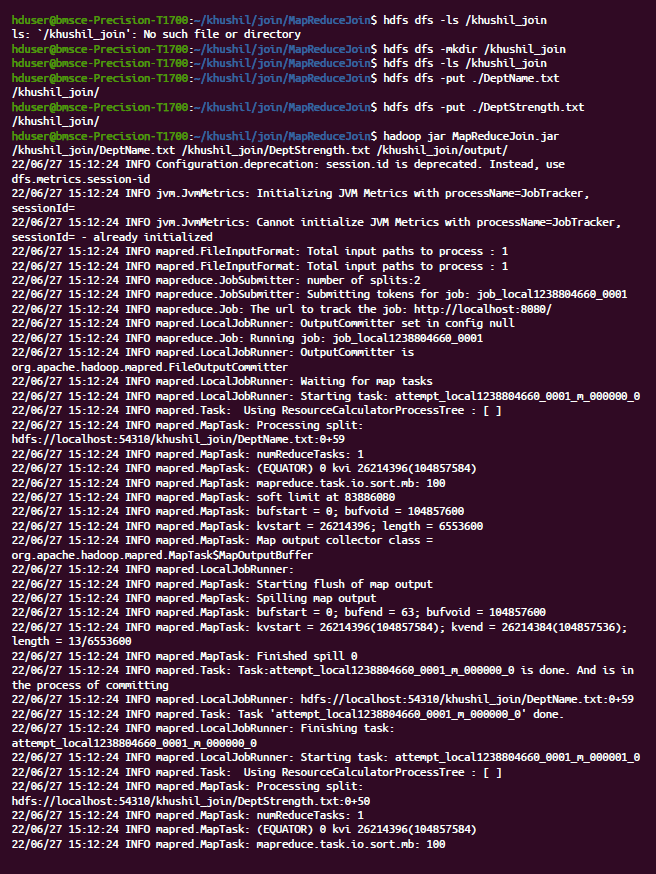
return super.compare(a, b);

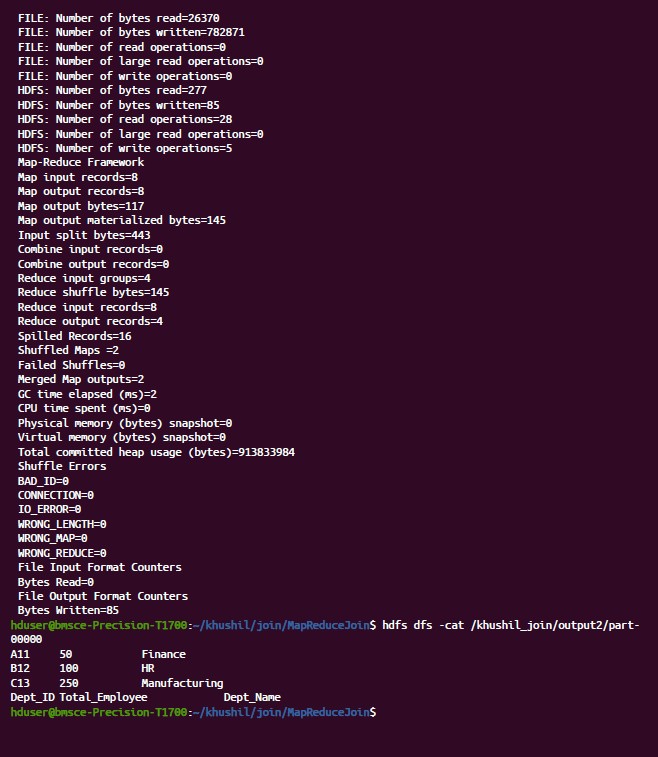
}

}

}

Output:





Scala Programming:

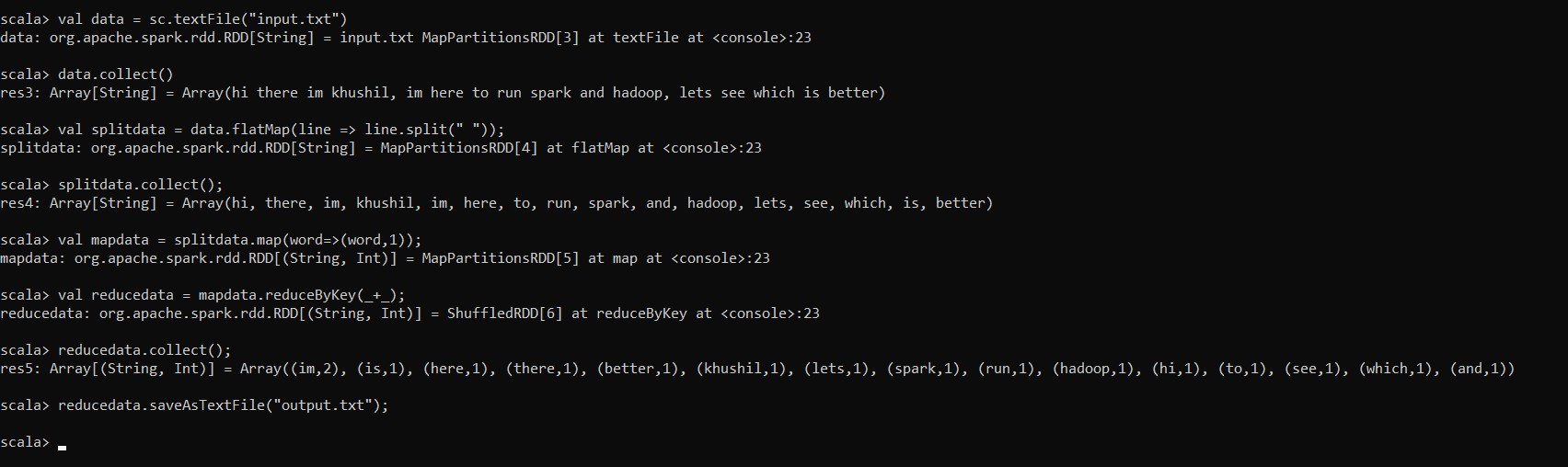
Lab 9:

val data=sc.textFile("sparkdata.txt") data.collect;

val splitdata = data.flatMap(line =**>** line.split(" ")); splitdata.collect;

val mapdata = splitdata.map(word =**>** (word,1)); mapdata.collect;

val reducedata = mapdata.reduceByKey(\_+\_); reducedata.collect;



Lab 10:

val textFile = sc.textFile("/home/bhoom/Desktop/wc.txt")

val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(\_ + \_)

import scala.collection.immutable.ListMap

val sorted=ListMap(counts.collect.sortWith(\_.\_2 > \_.\_2):\_\*)// sort in descending order based on values

println(sorted) for((k,v)<-sorted)

{

if(v>4)

{

print(k+",") print(v) println()

}}

