## Lab – 6

## Implement WordCount Program on Hadoop framework

Mapper Code:

```
 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> {  public void map(LongWritable key, Text
value, OutputCollector<Text,
 IntWritable> output, Reporter rep) throws IOException
 {
 String line = value.toString();
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: You have to copy paste this program into the WCDriver Java Class file.

```
 // 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);
}
}
```

**From the following link extract the weather data https://github.com/tomwhite/hadoopbook/tree/master/input/ncdc/all**

**Create a Map Reduce program to**

**a) find average temperature for each year from NCDC data set.**

AverageDriver package temp; import org.apache.hadoop.fs.Path;

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

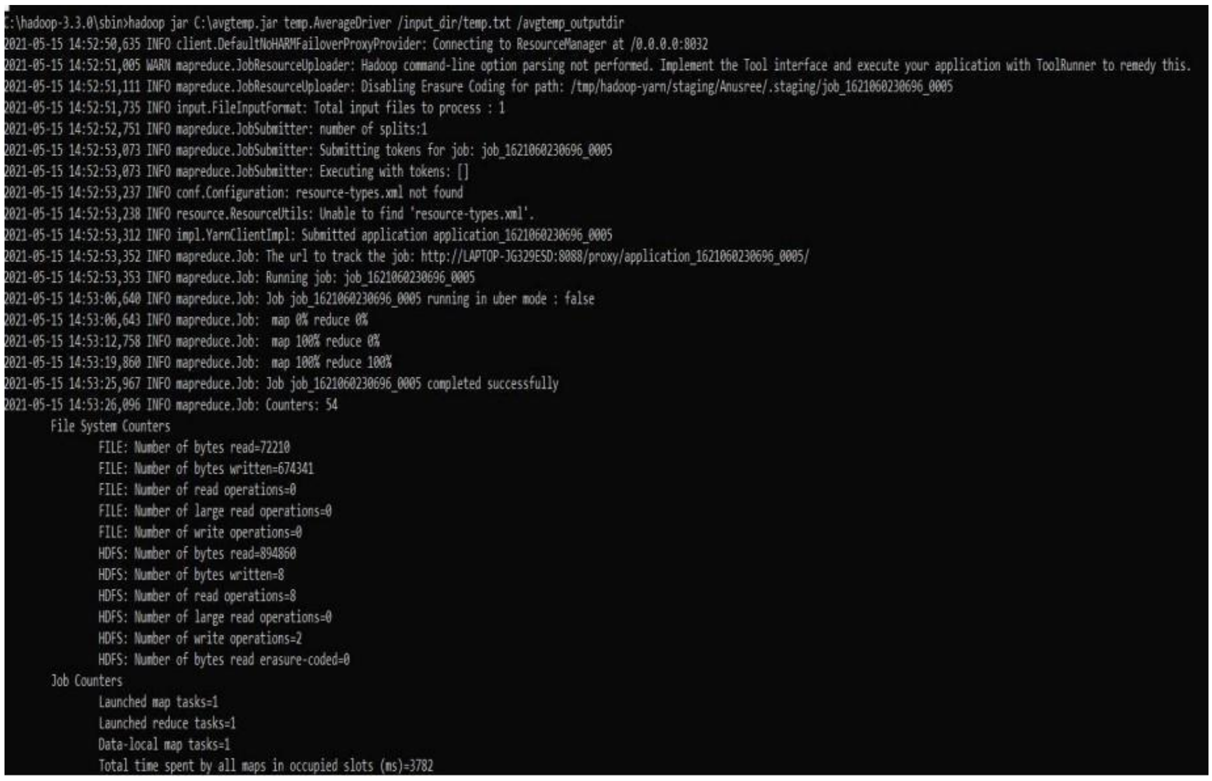
org.apache.hadoop.io.Text; import

org.apache.hadoop.mapreduce.Job; import

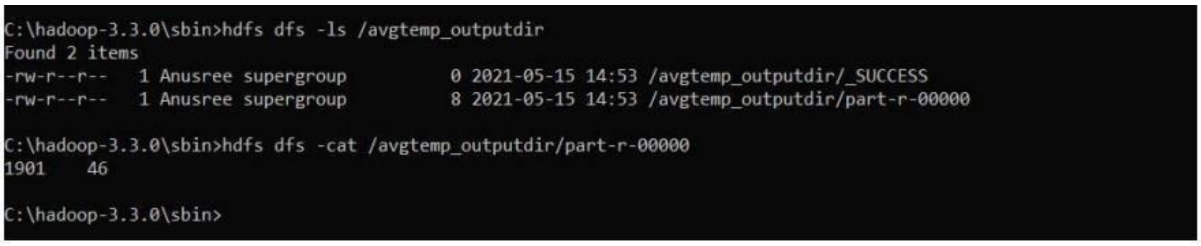
org.apache.hadoop.mapreduce.lib.input.FileInputFormat; import

```
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));
 }}
```



```
:\hadoop-3.3.0\sbin>hadoop jar C:\avgtemp.jar temp.AverageDriver /input_dir/temp.txt /avgtemp_outputdir
2021-05-15 14:52:50,635 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2021-05-15 14:52:51,005 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2021-05-15 14:52:51,111 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/Anusree/.staging/job_1621060230696_0005
2021-05-15 14:52:51,735 INFO input.FileInputFormat: Total input files to process : 1
2021-05-15 14:52:52,751 INFO mapreduce.JobSubmitter: number of splits:1
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1621060230696_0005
2021-05-15 14:52:53,073 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-05-15 14:52:53,237 INFO conf.Configuration: resource-types.xml not found
2021-05-15 14:52:53,238 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-05-15 14:52:53,312 INFO impl.YarnClientImpl: Submitted application application_1621060230696_0005
2021-05-15 14:52:53,352 INFO mapreduce.Job: The url to track the job: http://LAPTOP-JG329ESD:8088/proxy/application_1621060230696_0005/
2021-05-15 14:52:53,353 INFO mapreduce.Job: Running job: job_1621060230696_0005
2021-05-15 14:53:06,640 INFO mapreduce.Job: Job job_1621060230696_0005 running in uber mode : false
2021-05-15 14:53:06,643 INFO mapreduce.Job:  map 0% reduce 0%
2021-05-15 14:53:12,758 INFO mapreduce.Job:  map 100% reduce 0%
2021-05-15 14:53:19,860 INFO mapreduce.Job:  map 100% reduce 100%
2021-05-15 14:53:25,967 INFO mapreduce.Job: Job job_1621060230696_0005 completed successfully
2021-05-15 14:53:26,096 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=72210
                FILE: Number of bytes written=674341
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=894860
                HDFS: Number of bytes written=8
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3782
```



```
C:\hadoop-3.3.0\sbin>hdfs dfs -ls /avgtemp_outputdir
Found 2 items
-rw-r--r--   1 Anusree supergroup          0 2021-05-15 14:53 /avgtemp_outputdir/_SUCCESS
-rw-r--r--   1 Anusree supergroup          8 2021-05-15 14:53 /avgtemp_outputdir/part-r-00000

C:\hadoop-3.3.0\sbin>hdfs dfs -cat /avgtemp_outputdir/part-r-00000
1901    46

C:\hadoop-3.3.0\sbin>
```