**Module 3: Hadoop MapReduce Framework**

**Assignment**: Calculate the size of a word and the number of words of that size in a text file.

1. **Java Program**

package in.edureka.mapreduce;

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.LongWritable;

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.Reducer.Context;

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

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

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

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import in.edureka.mapreduce.AlphaWordCount.Map;

import in.edureka.mapreduce.AlphaWordCount.Reduce;

public class AlphaWordCount {

public static class Map extends Mapper<LongWritable,Text,Text,IntWritable>{

//Defining a local variable count of type IntWritable

private static int count ;

//Defining a local variable word of type Text

private Text word = new Text();

public void map(LongWritable key, Text value,

Context context)

throws IOException,InterruptedException {

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

while (tokenizer.hasMoreTokens()) {

String thisWord = tokenizer.nextToken();

//finding the length of each token(word)

count= thisWord.length();

word.set(thisWord);

String word\_str = String.valueOf(count);

Text w\_count= new Text(word\_str);

/Sending to output collector which in turn passes the same to reducer

//So in this case the output from mapper will be the length of a word and that word

context.write(w\_count,new IntWritable(1));

}

}

}

public static class Reduce extends Reducer<Text,IntWritable,Text,IntWritable>{

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

Context context)

throws IOException,InterruptedException {

int sum=0;

// TODO Auto-generated method stub

for(IntWritable x: values)

{

sum+=x.get();

}

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

}

}

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

// TODO Auto-generated method stub

//JobConf conf = new JobConf(AlphaCount.class);

Configuration conf= new Configuration();

//conf.setJobName("mywc");

Job job = new Job(conf,"mywc");

job.setJarByClass(AlphaWordCount.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

//conf.setMapperClass(Map.class);

//conf.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

Path outputPath = new Path(args[1]);

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

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

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

outputPath.getFileSystem(conf).delete(outputPath);

//exiting the job only if the flag value becomes false

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

}

}

1. **Commands:**

124 23/02/2018 19:27:10 ls -ltr

125 23/02/2018 19:27:27 hadoop fs -ls

126 23/02/2018 19:29:55 hadoop jar AlphaWordCount.jar in.edureka.mapreduce.AlphaWordCount AlphaCountDir outputAlphaCountDir

127 23/02/2018 19:32:15 hadoop fs -ls outputAlphaCountDir

128 23/02/2018 19:32:42 hadoop fs -cat outputAlphaCountDir/part-r-00000

1. **Result**:

1 66

10 77

11 32

12 20

13 22

14 8

15 2

16 2

2 162

3 208

4 204

5 126

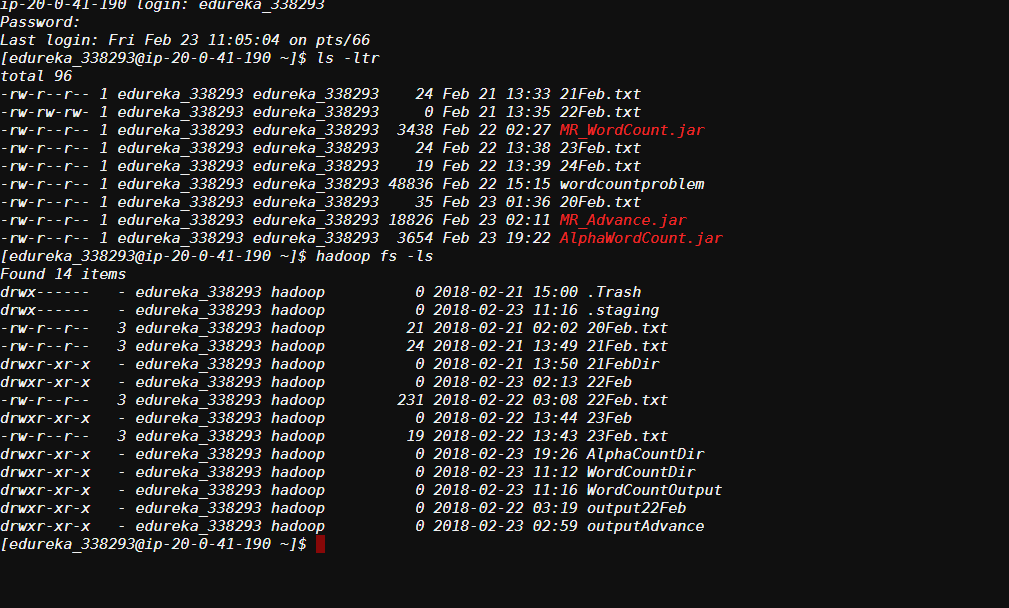
6 163

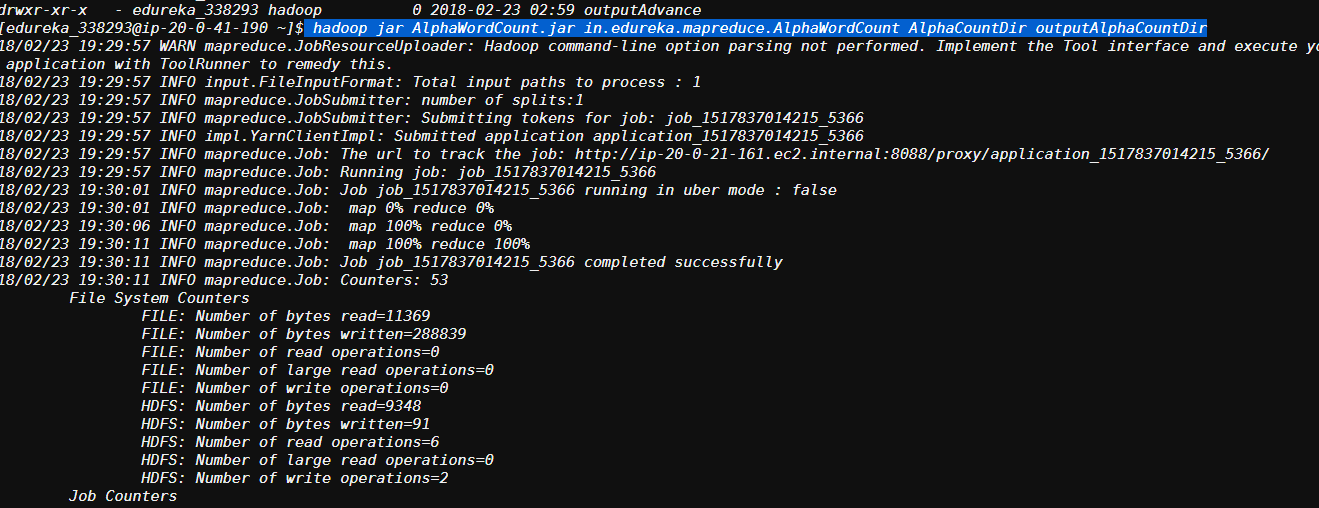
7 127

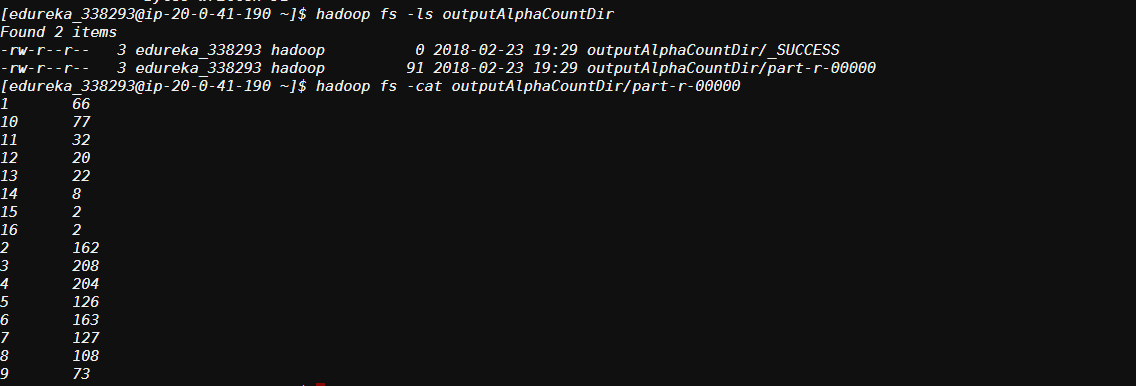
8 108

9 73

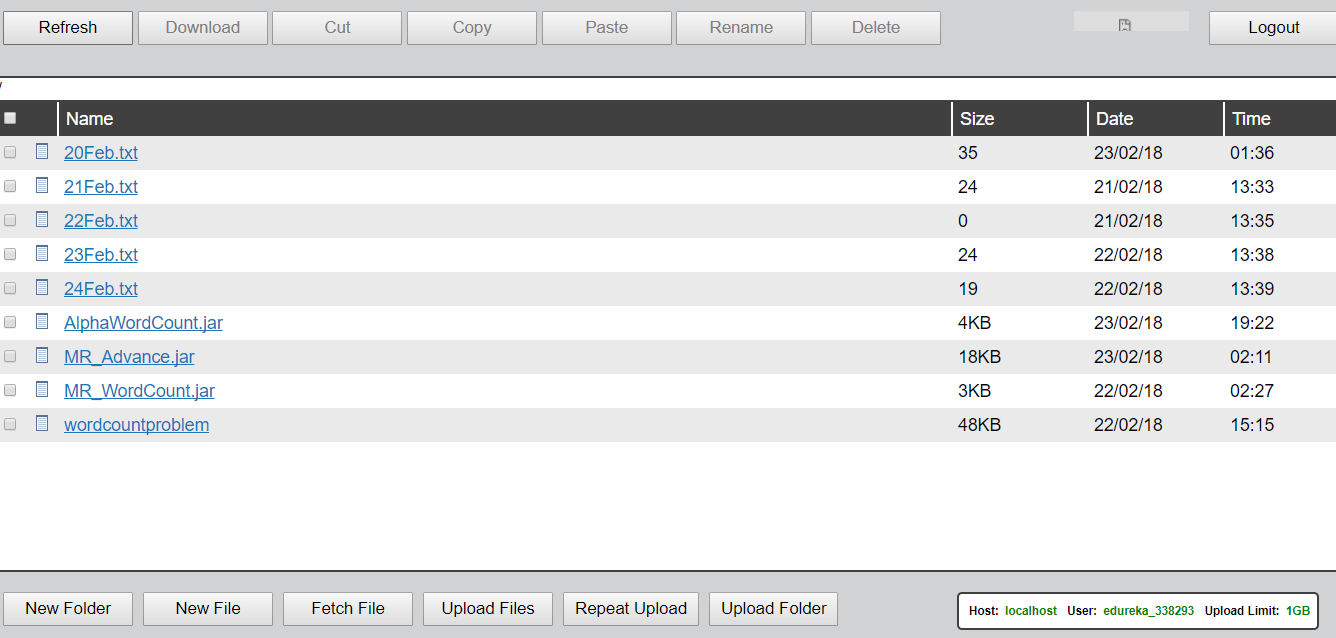
1. **Webconsole**:







1. **FTP:**



1. **HUE**

