**Code:**

**package** com.venkat.mapreduce;

**import** java.io.IOException;

**import** java.util.List;

**import** java.util.StringTokenizer;

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

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

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

**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.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;

**public** **class** AlphabetCount {

**public** **static** **class** AlphabetMapper **extends** Mapper<LongWritable, Text, ByteWritable, IntWritable> {

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

{

//write dp logic

StringTokenizer itr = **new** StringTokenizer(value.toString());

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

**while**(itr.hasMoreTokens()) {

String token = itr.nextToken();

word.set(token);

context.write(**new** ByteWritable((**byte**)token.length()), **new** IntWritable(1));

}

}

}

**public** **static** **class** AlphabetReducer **extends** Reducer<ByteWritable, IntWritable, ByteWritable, IntWritable> {

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

{

//business logic

**int** wordCount = 0;

**for**(IntWritable value: values)

{

wordCount += value.get();

}

//write ok ov to context. it will write to hdfs

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

}

}

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

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

Job job = **new** ~~Job~~(conf, "Alphabet Count");

//specify mapper, reducer and driver class

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

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

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

//set map output key, output value, reducer output key reducer output value

job.setMapOutputKeyClass(ByteWritable.**class**);

job.setMapOutputValueClass(IntWritable.**class**);

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

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

//set input and output format

job.setInputFormatClass(TextInputFormat.**class**);

job.setOutputFormatClass(TextOutputFormat.**class**);

//set input and output format

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

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

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

}

}

**Output File:**



**Output File Content:**

1 66

2 162

3 208

4 204

5 126

6 163

7 127

8 108

9 73

10 77

11 32

12 20

13 22

14 8

15 2

16 2