BDA Lab

Assignment-2

Q Write a map reduce program to count the frequency of given word in a given file.

Step 1 Start Hadoop Services

```
vboxuser@Ubuntu-hadoop:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Ubuntu-hadoop]
vboxuser@Ubuntu-hadoop:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
vboxuser@Ubuntu-hadoop:~$
```

Step 2 create file

```
vboxuser@Ubuntu-hadoop:~$ nano WordFrequency.java
```

Step 3 Write word counting program

```
import java.io.IOException;

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.Reducer;
```

```
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordFrequency {

   public static class TokenizerMapper
         extends Mapper<Object, Text, Text, IntWritable> {

      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();
      private String searchWord;

      @Override
      protected void setup(Context context) {
         searchWord = context.getConfiguration().get("word");
      }
      public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException {

         String[] words = value.toString().split("\\s+");

         for (String w : words) {
            if (w.equalsIgnoreCase(searchWord)) {
               word.set(searchWord);
               context.write(word, one);
            }}}
   }

   public static class IntSumReducer
```

```
        extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context)
        throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      context.write(key, new IntWritable(sum));
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    conf.set("word", args[2]);
    Job job = Job.getInstance(conf, "Word Frequency");
    job.setJarByClass(WordFrequency.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Step 4 Compile Program

```
vboxuser@Ubuntu-hadoop:~$ export HADOOP_CLASSPATH=$(hadoop classpath)
javac -classpath ${HADOOP_CLASSPATH} -d WordFrequency WordFrequency.java
jar -cvf wordfreq.jar -C WordFrequency/ .
added manifest
adding: WordFrequency.class(in = 1573) (out= 846)(deflated 46%)
adding: WordFrequency$TokenizerMapper.class(in = 2428) (out= 980)(deflated 59%)
adding: WordFrequency$IntSumReducer.class(in = 1669) (out= 707)(deflated 57%)
```

Step 5 Create Input file and upload to hdfs

```
vboxuser@Ubuntu-hadoop:~$ echo "hadoop spark hadoop mapreduce hadoop spark" > in
put.txt
```

```
vboxuser@Ubuntu-hadoop:~$ hdfs dfs -mkdir /input
hdfs dfs -put input.txt /input
```

Step 6 Run MapReduce Job

```
vboxuser@Ubuntu-hadoop:~$ hadoop jar wordfreq.jar WordFrequency /input /output H
adoop
```

```
2026-02-15 18:41:49,876 INFO mapreduce.Job:  map 0% reduce 0%
2026-02-15 18:41:54,183 INFO mapreduce.Job:  map 100% reduce 0%
2026-02-15 18:42:01,305 INFO mapreduce.Job:  map 100% reduce 100%
2026-02-15 18:42:01,325 INFO mapreduce.Job: Job job_1771180586441_0002 completed
 successfully
```

Step 7 View Output

```
vboxuser@Ubuntu-hadoop:~$ hdfs dfs -cat /output/part-r-00000
Hadoop  3
```