### ****11.8. Ejemplo II****

Este ejemplo realiza la suma del número de líneas que contienen los ficheros que recibe como entrada.

**Pasos para ejecutar el ejemplo**

1. **Generar el fichero .java "LineCount.java"**

*sudo nano LineCount.java*

Dentro de este fichero deberemos pegar el código del ejemplo.

import java.io.IOException;

import java.util.\*;

import org.apache.hadoop.fs.Path;

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

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

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

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

public class LineCount {

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

private final static IntWritable uno= new IntWritable(1);

private Text word = new Text("Total Lineas");

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

output.collect(word, uno);

}

}

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

public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {

int sum = 0;

while (values.hasNext()) {

sum += values.next().get();

}

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

}

}

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

JobConf conf = new JobConf(LineCount.class);

conf.setJobName("LineCount");

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

conf.setMapperClass(Map.class);

conf.setCombinerClass(Reduce.class);

conf.setReducerClass(Reduce.class);

conf.setInputFormat(TextInputFormat.class);

conf.setOutputFormat(TextOutputFormat.class);

FileInputFormat.setInputPaths(conf, new Path(args[0]));

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

JobClient.runJob(conf);

}

}

1. **Crear un directorio llamdo “line-count”**

*sudo mkdir line-count-in*

*sudo chmod -R 777 line-count-in*

1. **Crear los ficheros .txt (a, b, c, d) y su contenido**

*sudo echo -e "uno\ndos\ntres" > line-count-in/a.txt*

*sudo echo -e "dos\ndos\ndos" > line-count-in/b.txt*

*sudo echo -e "tres\ntres\ntres" > line-count-in/c.txt*

*sudo echo -e "tres\ncuatro\ncinco\nseis" > line-count-in/d.txt*

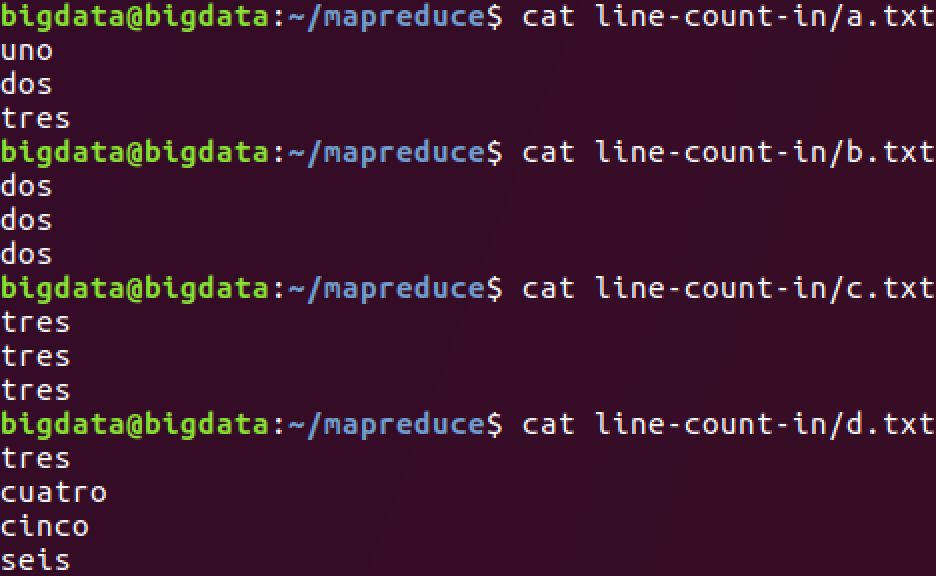
1. **Para comprobar que hemos creado correctamente los ficheros**

*cat line-count-in/a.txt*

*cat line-count-in/b.txt*

*cat line-count-in/c.txt*

*cat line-count-in/d.txt*



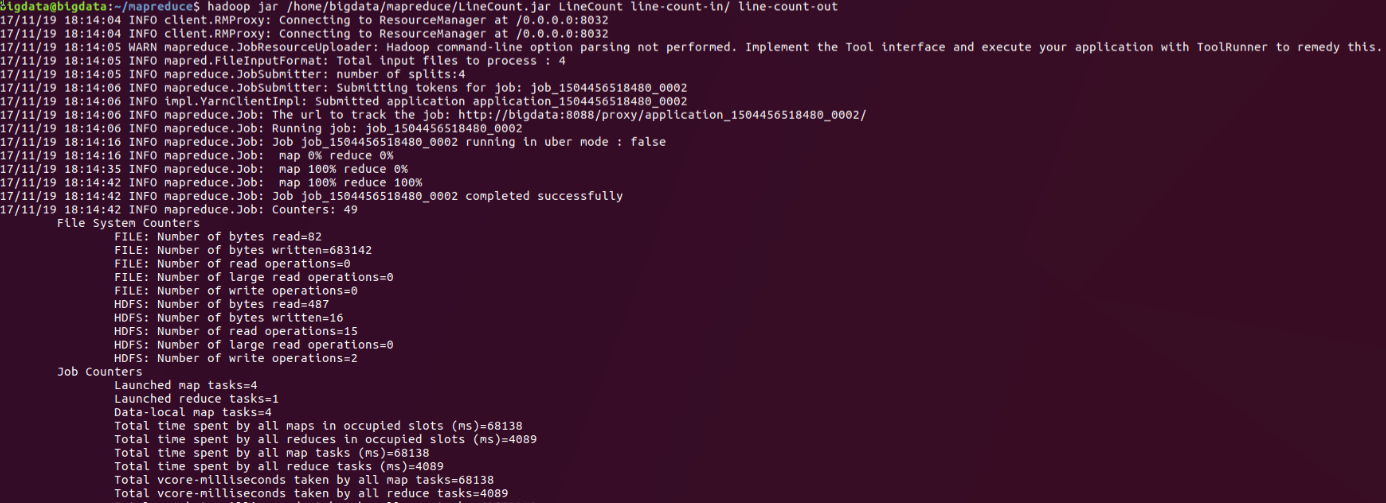
Una vez tengamos todos los ficheros correctamente creados, los subiremos a HDFS con el siguiente comando:

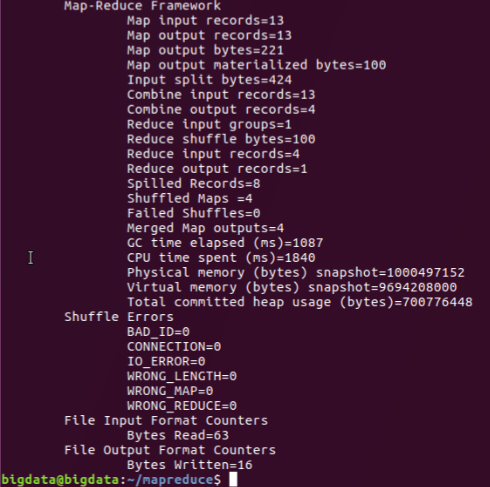
*hdfs dfs -put line-count-in/ /user/bigdata/*

1. **Executar exemplo**

Llegados a este punto, ya estamos listos para ejecutar nuestro ejemplo. Para ello lanzaremos el siguiente comando:

*hadoop jar /home/bigdata/mapreduce/LineCount.jar LineCount line-count-in/ line-count-out*

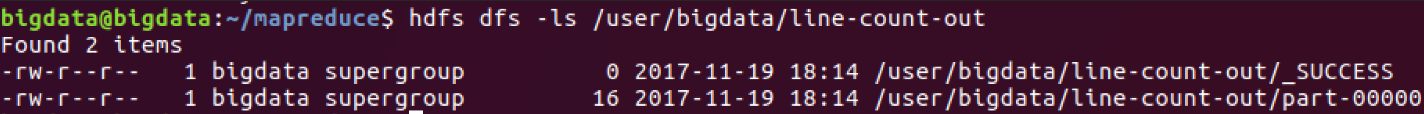




1. **Resultados**

Nuestro ejemplo generará dos ficheros, para comprobarlo ejecutaremos:

*hdfs dfs -ls /user/bigdata/line-count-out*



El fichero que contiene el resultado es "part-00000", para consultar su contenido ejecutaremos:

*hdfs dfs -cat /user/bigdata/line-count-out/part-00000*

