**DSBDA LAB**

**GROUP A , ASS 3**

**Max ip count using Hadoop Mapreduce**

//**Driver**

import java.lang.InterruptedException;

import java.lang.ClassNotFoundException;

public class main\_A {

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

Configuration conf=new Configuration();

Job job=Job.getInstance(conf,"Word\_count");

job.setJarByClass(main\_A.class);

job.setMapperClass(mapper1.class);

job.setReducerClass(reducer1.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

job.waitForCompletion(true);

}

}

// **Mapper**

package wordcount2;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class MyMapper extends Mapper<Object, Text, Text, IntWritable> {

// Define the IntWritable variable for the output value

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

private Text ipAddress = new Text();

public void map(Object offset, Text value, Context context) throws IOException, InterruptedException {

// Split the line into tokens by spaces

StringTokenizer tokenizer = new StringTokenizer(value.toString(), " ");

// The first token is the IP address, so extract it

if (tokenizer.hasMoreTokens()) {

ipAddress.set(tokenizer.nextToken()); // Set the IP address as key

context.write(ipAddress, one); // Write the key (IP address) and value (1)

}

}

}

// **Reducer**

/\*package wcount;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class My\_Reducer 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));

}

}

\*/

package wcount;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class My\_Reducer extends Reducer<Text, IntWritable, Text, IntWritable> {

private Map<String, Integer> ipCountMap = new HashMap<>();

@Override

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

throws IOException, InterruptedException {

int sum = 0;

for (IntWritable val : values) {

sum += val.get();

}

ipCountMap.put(key.toString(), sum); // Store in memory

}

@Override

protected void cleanup(Context context) throws IOException, InterruptedException {

// Find max

String maxIp = null;

int maxCount = 0;

for (Map.Entry<String, Integer> entry : ipCountMap.entrySet()) {

if (entry.getValue() > maxCount) {

maxCount = entry.getValue();

maxIp = entry.getKey();

}

}

if (maxIp != null) {

context.write(new Text(maxIp), new IntWritable(maxCount));

}

}

}



