Big Data Lab

Hadoop MapReduce Programs

[Document subtitle]

1. Write a map reduce program to analyze the given weather report data and to generate a report with cities having maximum temperature for a particular year.

driver.java

package temp;

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

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

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

public class driver {

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

*if*(args.length!=2){

System.out.println("input valid arg");

System.exit(-1);

}

JobConf conf=*new* JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

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

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

JobClient.runJob(conf);

}

}

mapper.java

package temp;

import java.io.IOException;

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

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

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

public void map(LongWritable key, Text value,

OutputCollector<Text, IntWritable> values, Reporter r)

throws IOException {

String s=value.toString();

String s1=s.substring(15, 19);

int temperature;

*if*(s.charAt(87)=='+'){

temperature=Integer.parseInt(s.substring(88,92));

} *else*{

temperature=Integer.parseInt(s.substring(87,92));

}

values.collect(*new* Text(s1),*new* IntWritable(temperature));

}

}

reducer.java

package temp;

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

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

public void reduce(Text key, Iterator<IntWritable> value1,

OutputCollector<Text, IntWritable> values, Reporter r)

throws IOException {

int maxvalue=Integer.MIN\_VALUE;

*while*(value1.hasNext()){

maxvalue=Math.max(maxvalue, value1.next().get());

}

values.collect(key, *new* IntWritable(maxvalue));

}

}

Manifest.txt

Main-Class: temp.driver

2. Write a map reduce program to print the multiplication of two different matrices.

MatrixMultiplication.java

package Matrix;

import org.apache.hadoop.fs.Path;

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

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

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

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 MatrixMultiplication {

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

Configuration conf = *new* Configuration();

*// A is an m-by-n matrix; B is an n-by-p matrix.*

conf.set("m", "2");

conf.set("n", "5");

conf.set("p", "3");

Job job = *new* Job(conf, "MatrixMultiplication");

job.setJarByClass(MatrixMultiplication.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setMapperClass(MatrixMapper.class);

job.setReducerClass(MatrixReducer.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

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

job.waitForCompletion(true);

}

}

MatrixMapper.java

package Matrix;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

public class MatrixMapper extends Mapper<LongWritable, Text,

Text, Text> {

public void map(LongWritable key, Text value, Context

context) throws IOException, InterruptedException {

Configuration conf = context.getConfiguration();

int m = Integer.parseInt(conf.get("m"));

int p = Integer.parseInt(conf.get("p"));

String line = value.toString();

String[] indicesAndValue = line.split(",");

Text outputKey = *new* Text();

Text outputValue = *new* Text();

*if* (indicesAndValue[0].equals("A")) {

*for* (int k = 0; k < p; k++) {

outputKey.set(indicesAndValue[1] + "," + k);

outputValue.set("A," + indicesAndValue[2] + "," +

indicesAndValue[3]);

context.write(outputKey, outputValue);

}

} *else* {

*for* (int i = 0; i < m; i++) {

outputKey.set(i + "," + indicesAndValue[2]);

outputValue.set("B," + indicesAndValue[1] + "," +

indicesAndValue[3]);

context.write(outputKey, outputValue);

}

}

}

}

MatrixReducer.java

package Matrix;

import java.io.IOException;

import java.util.HashMap;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

public class MatrixReducer extends Reducer<Text, Text, Text, Text> {

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

context) throws IOException, InterruptedException {

String[] value;

HashMap<Integer, Float> hashA = *new* HashMap<Integer,

Float>();

HashMap<Integer, Float> hashB = *new* HashMap<Integer,

Float>();

*for* (Text val *:* values) {

value = val.toString().split(",");

*if* (value[0].equals("A")) {

hashA.put(Integer.parseInt(value[1]),

Float.parseFloat(value[2]));

} *else* {

hashB.put(Integer.parseInt(value[1]),

Float.parseFloat(value[2]));

}

}

int n =

Integer.parseInt(context.getConfiguration().get("n"));

float result = 0.0f;

float a\_ij;

float b\_jk;

*for* (int j = 0; j < n; j++) {

a\_ij = hashA.containsKey(j) *?* hashA.get(j) *:* 0.0f;

b\_jk = hashB.containsKey(j) *?* hashB.get(j) *:* 0.0f;

result += a\_ij \* b\_jk;

}

*if* (result != 0.0f) {

context.write(null, *new* Text(key.toString() + "," +

Float.toString(result)));

}}}

Manifest.txt

Main-Class: Matrix.MatrixMultiplication

3. Write a Map Reduce program to analyze the given Earthquake data and generate statistics.

App.java

package earthquake;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

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

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

*/\*\**

*\* The main application class.*

*\**

*\* @author Umer Mansoor*

*\*/*

public class App

{

*/\*\**

*\* Application entry point.*

*\* @param args*

*\* @throws Exception - Bad idea but produces less cluttered code.*

*\*/*

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

*if* (args.length != 2) {

System.err.println("Usage: hadoopex <input path> <output path>");

System.exit(-1);

}

*// Create the job specification object*

Job job = *new* Job();

job.setJarByClass(App.class);

job.setJobName("Earthquake Measurment");

*// Setup input and output paths*

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

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

*// Set the Mapper and Reducer classes*

job.setMapperClass(EarthquakeMapper.class);

job.setReducerClass(EarthquakeReducer.class);

*// Specify the type of output keys and values*

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(DoubleWritable.class);

*// Wait for the job to finish before terminating*

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

}

}

EarthquakeMapper.java

package earthquake;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

*/\*\**

*\* This is the main Mapper class.*

*\**

*\* @author umermansoor*

*\*/*

public class EarthquakeMapper extends

Mapper<LongWritable, Text, Text, DoubleWritable>

{

*/\*\**

*\* The `Mapper` function. It receives a line of input from the file,*

*\* extracts `region name` and `earthquake magnitude` from it, which becomes*

*\* the output. The output key is `region name` and the output value is*

*\* `magnitude`.*

*\* @param key - Input key - The line offset in the file - ignored.*

*\* @param value - Input Value - This is the line itself.*

*\* @param context - Provides access to the OutputCollector and Reporter.*

*\* @throws IOException*

*\* @throws InterruptedException*

*\*/*

@Override

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

IOException, InterruptedException {

String[] line = value.toString().split(",", 12);

*// Ignore invalid lines*

*if* (line.length != 12) {

System.out.println("- " + line.length);

*return*;

}

*// The output `key` is the name of the region*

String outputKey = line[11];

*// The output `value` is the magnitude of the earthquake*

double outputValue = Double.parseDouble(line[8]);

*// Record the output in the Context object*

context.write(*new* Text(outputKey), *new* DoubleWritable(outputValue));

}

}

EarthquakeReducer.java

package earthquake;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

import org.apache.hadoop.io.Text;

public class EarthquakeReducer extends

Reducer<Text, DoubleWritable, Text, DoubleWritable>

{

*/\*\**

*\* The `Reducer` function. Iterates through all earthquake magnitudes for a*

*\* region to find the maximum value. The output key is the `region name` and*

*\* the value is the `maximum magnitude` for that region.*

*\* @param key - Input key - Name of the region*

*\* @param values - Input Value - Iterator over quake magnitudes for region*

*\* @param context - Used for collecting output*

*\* @throws IOException*

*\* @throws InterruptedException*

*\*/*

@Override

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

Context context) throws IOException, InterruptedException {

*// Standard algorithm for finding the max value*

double maxMagnitude = Double.MIN\_VALUE;

*for* (DoubleWritable value *:* values) {

maxMagnitude = Math.max(maxMagnitude, value.get());

}

context.write(key, *new* DoubleWritable(maxMagnitude));

}}

Manifest.txt

Main-Class: earthquake.App

4. Write a Map Reduce program to analyze the given natural numbers and generate statistics for the number as odd or even and print their sum.

driver.java

package sum;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat;

import org.apache.hadoop.mapred.FileOutputFormat;

import org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

public class driver extends Configured implements Tool {

@Override

public int run(String[] args) throws Exception

{

*if* (args.length < 2)

{

System.out.println("Please enter valid arguments");

*return* -1;

}

JobConf conf = *new* JobConf(driver.class);

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

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

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setMapOutputKeyClass(Text.class);

conf.setMapOutputValueClass(IntWritable.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(IntWritable.class);

JobClient.runJob(conf);

*return* 0;

}

*// Main Method*

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

{

int exitcode = ToolRunner.run(*new* driver(), args);

System.out.println(exitcode);

}

}

mapper.java

package sum;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reporter;

public class mapper extends MapReduceBase implements Mapper<LongWritable,

Text, Text, IntWritable> {

@Override

*// Map function*

public void map(LongWritable key, Text value, OutputCollector<Text,

IntWritable> output, Reporter rep)

throws IOException

{

*// Splitting the line into spaces*

String data[] = value.toString().split(" ");

*for* (String num *:* data)

{

int number = Integer.parseInt(num);

*if* (number % 2 == 1)

{

*// For Odd Numbers*

output.collect(*new* Text("ODD"), *new* IntWritable(number));

}

*else*

{

*// For Even Numbers*

output.collect(*new* Text("EVEN"),

*new* IntWritable(number));

}

}

}

}

reducer.java

package sum;

import java.io.IOException;

import java.util.Iterator;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase;

import org.apache.hadoop.mapred.OutputCollector;

import org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class reducer extends MapReduceBase implements Reducer<Text,

IntWritable, Text, IntWritable> {

@Override

*// Reduce Function*

public void reduce(Text key, Iterator<IntWritable> value,

OutputCollector<Text, IntWritable> output, Reporter rep)

throws IOException

{

*// For finding sum and count of even and odd*

*// you don't have to take different variables*

int sum = 0, count = 0;

*if* (key.equals("ODD"))

{

*while* (value.hasNext())

{

IntWritable i = value.next();

*// Finding sum and count of ODD Numbers*

sum += i.get();

count++;

}

}

*else*

{

*while* (value.hasNext())

{

IntWritable i = value.next();

*// Finding sum and count of EVEN Numbers*

sum += i.get();

count++;

}

}

*// First sum then count is printed*

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

output.collect(key, *new* IntWritable(count));

}

}

Manifest.txt

Main-Class: sum.driver

5. Write a map-reduce program to analyze the given Insurance data and generate a statistics report.

InsuranceDriver.java

package Insurance;

import org.apache.hadoop.fs.Path;

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

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

public class InsuranceDriver {

public static void main(String[] args) {

JobClient my\_client = *new* JobClient();

*// Create a configuration object for the job*

JobConf job\_conf = *new* JobConf(InsuranceDriver.class);

*// Set a name of the Job*

job\_conf.setJobName("ConstructionType");

*// Specify data type of output key and value*

job\_conf.setOutputKeyClass(Text.class);

job\_conf.setOutputValueClass(IntWritable.class);

*// Specify names of Mapper and Reducer Class*

job\_conf.setMapperClass(Insurance.InsuranceMapper.class);

job\_conf.setReducerClass(Insurance.InsuranceReducer.class);

*// Specify formats of the data type of Input and output*

job\_conf.setInputFormat(TextInputFormat.class);

job\_conf.setOutputFormat(TextOutputFormat.class);

*// Set input and output directories using command line arguments,*

*//arg[0] = name of input directory on HDFS, and arg[1] = name of output directory to be created to store the output file.*

FileInputFormat.setInputPaths(job\_conf, *new* Path(args[0]));

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

my\_client.setConf(job\_conf);

*try* {

*// Run the job*

JobClient.runJob(job\_conf);

} *catch* (Exception e) {

e.printStackTrace();

}

}

}

InsuranceMapper.java

package Insurance;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

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

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

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

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

String valueString = value.toString();

String[] SingleCountryData = valueString.split(",");

output.collect(*new* Text(SingleCountryData[16]), one);

}

}

InsuranceReducer.java

package Insurance;

import java.io.IOException;

import java.util.*\**;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

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

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

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

Text key = t\_key;

int frequencyForCountry = 0;

*while* (values.hasNext()) {

*// replace type of value with the actual type of our value*

IntWritable value = (IntWritable) values.next();

frequencyForCountry += value.get();

}

output.collect(key, *new* IntWritable(frequencyForCountry));

}

}

Manifest.txt

Main-Class: Insurance.InsuranceDriver

6. Write a map-reduce program to analyze the given employee record data and generate a statistics report with the total number of female and male employees and their average salary.

driver.java

package avg;

import java.util.*\**;

import java.io.IOException;

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

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

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

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

public class driver {

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

*if*(args.length!=2){

System.out.println("input valid arg");

System.exit(-1);

}

JobConf conf=*new* JobConf(driver.class);

conf.setMapperClass(mapper.class);

conf.setReducerClass(reducer.class);

conf.setOutputKeyClass(Text.class);

conf.setOutputValueClass(DoubleWritable.class);

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

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

JobClient.runJob(conf);

}

}

mapper.java

package avg;

import java.io.IOException;

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

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

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

public class mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,DoubleWritable>{

public void map(LongWritable key, Text empRecord, OutputCollector<Text , DoubleWritable> values1, Reporter r) throws IOException {

String[] word = empRecord.toString().split("\\t");

String sex = word[3];

Double salary = Double.parseDouble(word[8]);

values1.collect(*new* Text(sex), *new* DoubleWritable(salary));

}

}

reducer.java

package avg;

import java.util.*\**;

import java.io.IOException;

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

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

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

public class reducer extends MapReduceBase implements Reducer<Text, DoubleWritable, Text, DoubleWritable>{

*//last arg is Double*

@Override

public void reduce(Text arg0, Iterator<DoubleWritable> arg1, OutputCollector<Text, DoubleWritable> arg2, Reporter arg3)

throws IOException {

*// TODO Auto-generated method stub*

*try* {

Double total = (Double) 0.0;

int count = 0;

*while* (arg1.hasNext()) {

total += arg1.next().get();

count++;

}

Double avg = (Double) total / count;

String out = "Total: " + total + " :: " + "Average: " + avg;

arg2.collect(arg0, *new* DoubleWritable(avg));

arg2.collect(arg0, *new* DoubleWritable(total));

} *catch* (Exception e) {

e.printStackTrace();

}}}

Manifest.txt

Main-Class: avg.driver

7. Write a map-reduce program to analyze the given sales records over a period of time.

driver.java

package sales;

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.Text;

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

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 driver {

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

Configuration conf = *new* Configuration();

Job job = Job.getInstance(conf, "driver");

*// job.setJarByClass(WordCount.class);*

job.setMapperClass(mapper.class);

*// job.setCombinerClass(IntSumReducer.class);*

job.setReducerClass(reducer.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);

}

}

mapper.java

package sales;

import java.io.IOException;

import java.util.StringTokenizer;

import java.util.regex.PatternSyntaxException;

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.io.*\**;

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 mapper extends Mapper<Object, Text, Text, IntWritable>{

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

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

public void map(Object key, Text value, Context context

) throws IOException, InterruptedException {

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

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

*// word.set(itr.nextToken());*

*// context.write(word, one);*

*// }*

String[] line = value.toString().split(",");

*if*(line[0].equals("Transaction\_date")){

*return*; *//header of csv*

}

*// for(String val: line){*

*// System.out.print(val + " | ");*

*// }*

*// System.out.println();*

String country = "\_country\_" + line[7];

String payment\_type = "\_payment\_type\_" + line[3];

int price = Integer.parseInt(line[2]);

context.write(*new* Text(country), *new* IntWritable(price));

context.write(*new* Text(payment\_type), *new* IntWritable(1));

}

}

reducer.java

package sales;

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.Text;

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

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 reducer extends Reducer<Text,IntWritable,Text,IntWritable> {

*// private IntWritable result = new IntWritable();*

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

Context context

) throws IOException, InterruptedException {

String temp = key.toString();

*if*(temp.substring(0, 9) == "\_country\_"){

int total\_sales = 0;

*for*(IntWritable val*:* values){

total\_sales+=val.get();

}

context.write(key, *new* IntWritable(total\_sales));

} *else*{

int payment\_freq = 0;

*for*(IntWritable val*:* values){

payment\_freq+=val.get();

}

context.write(key, *new* IntWritable(payment\_freq));

}

}

}

Manifest.txt

Main-Class: sales.driver