云计算实验 MapReduce编程

原创

wow_awsl_qwq 于 2021-11-27 17:29:29 发布 2645 收藏 2

文章标签： 心理学 android android studio

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/qq_42641977/article/details/121578883

版权

一、实验题目

MapReduce编程

二、实验内容

本实验利用 Hadoop 提供的 Java API 进行编程进行 MapReduce 编程。

三、实验目标

掌握MapReduce编程。

理解MapReduce原理

【实验作业】简单流量统计

有如下这样的日志文件：

13726230503 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
13726230513 00-FD-07-A4-72-B8:CMCC 120.196.40.8 i02.c.aliimg.com 248 0 200
13826230523 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
13726230533 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
13726230543 00-FD-07-A4-72-B8:CMCC 120.196.100.82 Video website 1527 2106 200
13926230553 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
13826230563 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
13926230573 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
18912688533 00-FD-07-A4-72-B8:CMCC 220.196.100.82 Integrated portal 1938 2910 200
18912688533 00-FD-07-A4-72-B8:CMCC 220.196.100.82 i02.c.aliimg.com 3333 21321 200
13726230503 00-FD-07-A4-72-B8:CMCC 120.196.100.82 Search Engines 9531 9531 200
13826230523 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200
13726230503 00-FD-07-A4-72-B8:CMCC 120.196.100.82 i02.c.aliimg.com 2481 24681 200

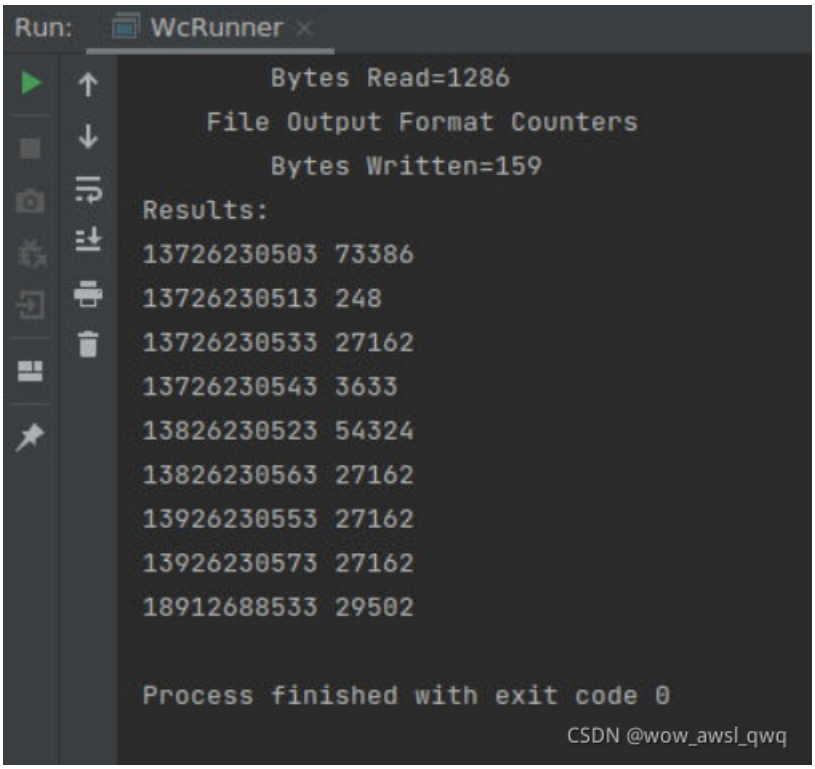
该日志文件记录了每个手机用户在一段时间内的网络流量信息，具体字段含义为：

手机号码 MAC地址 IP地址 域名 上行流量（字节数） 下行流量（字节数） 套餐类型

根据以上日志，统计出每个手机用户在该时间段内的总流量（上行流量+下行流量），统计结果的格式为：

手机号码 字节数量

实验结果



实验代码

WcMap.java

```
import java.io.IOException;
import org.apache.commons.lang.StringUtils;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

        public class WcMap extends Mapper<LongWritable, Text, Text, LongWritable>{
        @Override
        protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
                    String str = value.toString();
                    String[] words = StringUtils.split(str," ",10);
                    int i=0;
                    for(String word : words){
                        if(i==words.length-2||i==words.length-3)
                        context.write(new Text(words[0]), new LongWritable(Integer.parseInt(word)));
                        i++;
                    }
            }
        }
```

WcReduce.java

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WcReduce extends Reducer<Text, LongWritable, Text, LongWritable>{
    @Override
    protected void reduce(Text key, Iterable<LongWritable> values,Context context)
            throws IOException, InterruptedException {
        long count = 0;
        for(LongWritable value : values){
            count += value.get();
        }
        context.write(key, new LongWritable(count));
    }
}
```

WcRunner.java

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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;
import java.util.Scanner;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import java.net.URI;

public class WcRunner{
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);

        job.setJarByClass(WcRunner.class);

        job.setMapperClass(WcMap.class);
        job.setReducerClass(WcReduce.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(LongWritable.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(LongWritable.class);

        Scanner sc = new Scanner(System.in);
        System.out.print("inputPath:");
        String inputPath = sc.next();
        System.out.print("outputPath:");
        String outputPath = sc.next();

        try {
            FileSystem fs0 = FileSystem.get(new URI("hdfs://master:9000"), new Configuration());
            Path hdfsPath = new Path(outputPath);
            fs0.copyFromLocalFile(new Path("/headless/Desktop/workspace/mapreduce/WordCount/data/1.txt"),new Pat
h("/mapreduce/WordCount/input/1.txt"));
            if(fs0.delete(hdfsPath,true)){
```

```
            if(fs0.delete(hdfsPath,true)){
                System.out.println("Directory "+ outputPath +" has been deleted successfully!");
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
        FileInputFormat.setInputPaths(job, new Path("hdfs://master:9000"+inputPath));
        FileOutputFormat.setOutputPath(job, new Path("hdfs://master:9000"+outputPath));
        job.waitForCompletion(true);
        try {
            FileSystem fs = FileSystem.get(new URI("hdfs://master:9000"), new Configuration());
            Path srcPath = new Path(outputPath+"/part-r-00000");

            FSDataInputStream is = fs.open(srcPath);
            System.out.println("Results:");
            while(true) {
                String line = is.readLine();
                if(line == null) {
                    break;
                }
                System.out.println(line);
            }
            is.close();
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

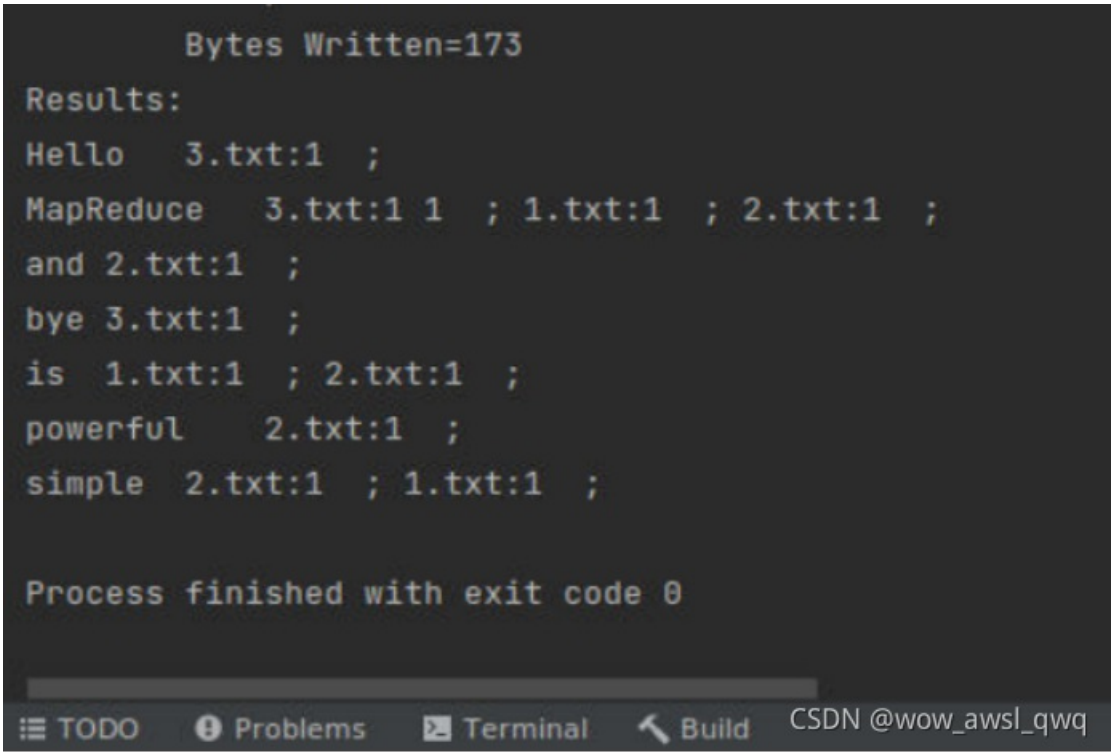
what's up,这个云计算实验也太多了吧，不写了

我又写了一题

(二)【实验作业】索引倒排输出行号
在索引倒排实验中，我们可以得到每个单词分布在哪些文件中，以及在每个文件中出现的次数，修改以上实现，在输出的倒排索引结果中可以得到每个单词在每个文件中的具体行号信息。输出结果的格式如下：
单词 文件名：行号，文件名：行号，文件名：行号

实验结果：

MapReduce在3.txt的第一行出现了两次所以有两个1




```java
import java.io.*;
import java.util.StringTokenizer;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileSplit;

public class MyMapper extends Mapper<Object,Text,Text,Text>{
    private Text keyInfo = new Text();
    private Text valueInfo = new Text();
    private FileSplit split;
    int num=0;

    public void map(Object key,Text value,Context context)
            throws IOException,InterruptedException{
        num++;
        split = (FileSplit)context.getInputSplit();
        StringTokenizer itr = new StringTokenizer(value.toString());
        while(itr.hasMoreTokens()){
            keyInfo.set(itr.nextToken()+" "+split.getPath().getName().toString());
            valueInfo.set(num+"");
            context.write(keyInfo,valueInfo);
        }
    }
}
```

```java
import java.io.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

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

    private Text info = new Text();

    public void reduce(Text key,Iterable<Text>values,Context context)
            throws IOException, InterruptedException{
        String  sum = "";
        for(Text value:values){
            sum += value.toString()+" ";
        }

                String record = key.toString();
        String[] str = record.split(" ");

        key.set(str[0]);
        info.set(str[1]+":"+sum);
        context.write(key,info);
    }
}
```

```java
import java.io.IOException;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MyReducer extends Reducer<Text,Text,Text,Text>{
    private Text result = new Text();
    public void reduce(Text key,Iterable<Text>values,Context context) throws

            IOException, InterruptedException{
        String value =new String();
        for(Text value1:values){
            value += value1.toString()+" ; ";
        }
        result.set(value);
        context.write(key,result);
    }
}
```

```java
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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;
import java.util.Scanner;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FileSystem;
import java.net.URI;

public class MyRunner {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf);
```

```
        job.setJarByClass(MyRunner.class);

        job.setMapperClass(MyMapper.class);
        job.setReducerClass(MyReducer.class);
        job.setCombinerClass(MyCombiner.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);


        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(Text.class);

        Scanner sc = new Scanner(System.in);
        System.out.print("inputPath:");
        String inputPath = sc.next();
        System.out.print("outputPath:");
        String outputPath = sc.next();

        try {
            FileSystem fs0 = FileSystem.get(new URI("hdfs://master:9000"), new Configuration());
            Path hdfsPath = new Path(outputPath);
            if(fs0.delete(hdfsPath,true)){
                System.out.println("Directory "+ outputPath +" has been deleted successfully!");
            }
        }catch(Exception e) {
            e.printStackTrace();
        }

        FileInputFormat.setInputPaths(job, new Path("hdfs://master:9000"+inputPath));

        FileOutputFormat.setOutputPath(job, new Path("hdfs://master:9000"+outputPath));

        job.waitForCompletion(true);

        try {
            FileSystem fs = FileSystem.get(new URI("hdfs://master:9000"), new Configuration());
            Path srcPath = new Path(outputPath+"/part-r-00000");

            FSDataInputStream is = fs.open(srcPath);
            System.out.println("Results:");
            while(true) {
                String line = is.readLine();
                if(line == null) {
                    break;
                }
                System.out.println(line);
            }
            is.close();
        }catch(Exception e) {
            e.printStackTrace();
        }

    }
}
```

后面的没写了