MapReduce每日一练2 成绩统计

题目

编程要求

使用`MapReduce`计算班级每个学生的**最好成绩**，输入文件路径为`/user/test/input`，请将计算后的结果输出到`/user/test/output/`目录下。

测试说明

输入文件在你每次点击评测的时候，平台会为你创建，无需你自己创建，只需要启动`HDFS`，编写`java`代码即可。

输入文件的数据格式如下：

`张三 12`

`李四 13`

`张三 89`

`李四 92`

`...`

依照如上格式你应该输出：

```
张三 89
李四 92
```

具体本关的预期输出请查看右侧测试集。

代码

```
import java.io.IOException;
import java.util.StringTokenizer;

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

public class WordCount {
```

```
    /********** Begin **********/
    public static class TokenizerMapper extends Mapper<LongWritable, Text, Text,
IntWritable>{

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

        public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException{
            StringTokenizer itr = new StringTokenizer(value.toString(),"\n");
            while (itr.hasMoreTokens()) {
                String[] str = itr.nextToken().split(" ");
                String name = str[0];
                one.set(Integer.parseInt(str[1]));
                word.set(name);
                context.write(word,one);
            }
            // while(itr.hasMoreTokens()){
            //     word.set(itr.nextToken());
            //     context.write(word, value);
            // }
        }
    }

    public static class IntSumReducer extends Reducer<Text, IntWritable, Text,
IntWritable>{
        private IntWritable result  = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException{
            int sum = 0;
            for (IntWritable val : values){
                sum = Math.max(sum, val.get());
            }
            result.set(sum);
            context.write(key, result);
            }
        }

    public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();

        Job job = Job.getInstance(conf, "word count");

        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setReducerClass(IntSumReducer.class);

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

        String inputfile = "/user/test/input";
        String outputfile = "/user/test/output/";

        FileInputFormat.addInputPath(job, new Path(inputfile));
        FileOutputFormat.setOutputPath(job, new Path(outputfile));
```

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

    /********** End **********/
}
```

题解

如何使用MapReduce进行运算

我们通过一个示例，来体验`Map/Reduce`的使用。

我们从一个问题入手：目前我们想统计两个文本文件中，每个单词出现的次数。

首先我们在当前目录下创建两个文件：

创建`file01`输入内容：

```
Hello World Bye World
```

创建`file02`输入内容：

```
Hello Hadoop Goodbye Hadoop
```

将文件上传到`HDFS`的`/usr/input/`目录下：

不要忘了启动`DFS`： `start-dfs.sh`

然后创建文件夹并上传：

```
root@evassh-88003:~# hadoop fs -mkdir /usr/
root@evassh-88003:~# hadoop fs -mkdir /usr/input
root@evassh-88003:~# hadoop fs -put file01 /usr/input
root@evassh-88003:~# hadoop fs -put file02 /usr/input
```

在右侧代码区域编写，文件`WordCount.java`，添加如下内容：

```
public class WordCount {
//Mapper类
/*因为文件默认带有行数，LongWritable是用来接受文件中的行数，
第一个Text是用来接受文件中的内容，
第二个Text是用来输出给Reduce类的key，
IntWritable是用来输出给Reduce类的value*/
 public static class TokenizerMapper
       extends Mapper<LongWritable, Text, Text, IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable 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);
```

```
      }
    }
  }
  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }
  public static void main(String[] args) throws Exception {
    //创建配置对象
    Configuration conf = new Configuration();
    //创建job对象
    Job job = new Job(conf, "word count");
    //设置运行job的类
    job.setJarByClass(WordCount.class);
    //设置Mapper的类
    job.setMapperClass(TokenizerMapper.class);
    //设置Reduce的类
    job.setReducerClass(IntSumReducer.class);
    //设置输出的key value格式
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    //设置输入路径
    String inputfile = "/usr/input";
    //设置输出路径
    String outputFile = "/usr/output";
    //执行输入
    FileInputFormat.addInputPath(job, new Path(inputfile));
    //执行输出
    FileOutputFormat.setOutputPath(job, new Path(outputFile));
    //是否运行成功，true输出0，false输出1
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

点击评测，运行代码，可以看到`/usr/output`目录下已经生成了文件。

```
root@evassh-88003:~# hadoop fs -ls /usr/output
Found 2 items
-rw-r--r--   1 root supergroup          0 2018-08-01 02:00 /usr/output/_SUCCESS
-rw-r--r--   1 root supergroup         41 2018-08-01 02:00 /usr/output/part-r-00
000
```

我们来查看`part--r-00000`文件的内容：

```
root@evassh-88003:~# hadoop fs -cat /usr/output/part-r-00000
Bye     1
Goodbye 1
Hadoop  2
Hello   2
World   2
```