**西南石油大学实验报告**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **课 程** | 大数据处理技术 | | **实验项目** | | | MapReduced程序设计 | | | | | **成绩** | | |  |
| **专业班级** | 网络工程1602 | | **组别** | | 无 | | **学号** | 201631063211 | | **指导教师** | | 赖俊良 | | |
| **姓 名** | 古皓 | **同组人姓名** | | 无 | | | | | **实验日期** | | | | 2019/6/21 | |

1. **实验目的**

掌握Map/Reduce相关原理和设计方法，设计相关的应用。

1. **实验内容**
2. 掌握Map/Reduce工作原理；
3. 设计Map/Reduce程序，测试计算性能
4. **实验步骤**

1、打开Eclipse创建Maven项目；

2、参考WordCount示例代码编写本实验代码，pom.xml文件拷贝至WordCount示例中的pom.xml文件即可；

3、代码逻辑：

MAP

输入：一行数据

处理：使用空格将字符串split成数组，提取国家(或品牌)和评分，分别作为键和值输出

输出：<国家, 评分> 或 <品牌, 评分>

Reduce：

输入：<国家，[该国家所有的评分数组]> 或 ：<品牌，[该品牌所有的评分数组]>

处理：计算平均分

输出：<国家，平均分> 或 <品牌，平均分>

**四、实验结果**

试验运行过程及结果

**public** **class** hadooptest3 **extends** Configured **implements** Tool {

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

**int** res = ToolRunner.*run*(**new** hadooptest3(), args);

System.*exit*(res);

}

// 执行

**public** **int** run(String[] args) **throws** Exception {

Configuration conf = **new** Configuration();

// 创建作业

Job job = Job.*getInstance*(conf, "hadooptest3");

// 指定作业的主类

job.setJarByClass(hadooptest3.**class**);

// 指定Map和Reduce类

job.setMapperClass(Map.**class**);

job.setReducerClass(Reduce.**class**);

// 指定输入格式为：文本格式文件

job.setInputFormatClass(TextInputFormat.**class**);

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

// 指定输出格式为：文本格式文件，键为文本、值为整型数

job.setOutputFormatClass(TextOutputFormat.**class**);

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

job.setOutputValueClass(FloatWritable.**class**);

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

// 执行MapReduce

**boolean** res = job.waitForCompletion(**true**);

**if**(res)

**return** 0;

**else**

**return** -1;

}

// Map过程

**public** **static** **class** Map **extends** Mapper<LongWritable, Text, Text, FloatWritable> {

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

//FloatWritable starsfianl;

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

@Override

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

**throws** IOException, InterruptedException {

String line = value.toString();

**int** i=0;

String stars="";

**float** a=0;

String[] words=line.split("\t");//对每行文本数据进行以制表符\t为分隔符的划分

**for**(i=0;i<words.length;i++) {

**if**(i==1)//当循环执行到第二列也就是品牌列时（这里也可以是4也就是国家列）

{

word.set(words[i]);//将该列的值输入到word中作为map的键

}

**if**(i==5)//当循环执行到最后一列也就是评分列

{

stars=words[i];//获取该列的字符串

**if**(stars.equals("Unrated"))//当该字符串为Unrated时表示此时该方便面未接受评定

**continue**;

a=Float.*parseFloat*(stars);//将该字符串转换位浮点数

}

**if**(word!=**null**&&a!=0)//当键值和value值都不为空时

{

//将键值对输入到中间目录中

context.write(word, **new** FloatWritable(a)); }

}

}

}

// Reduce过程

**public** **static** **class** Reduce **extends** Reducer<Text, FloatWritable, Text, FloatWritable> {

@Override

**protected** **void** reduce(Text key, Iterable<FloatWritable> val,

Reducer<Text, FloatWritable, Text, FloatWritable>.Context context)

**throws** IOException, InterruptedException {

// **TODO** Auto-generated method stub

**float** total = 0;

**float** num=0;

//得到该键值的数组的迭代器

Iterator<FloatWritable> values = val.iterator();

**while** (values.hasNext()) {

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

num++;

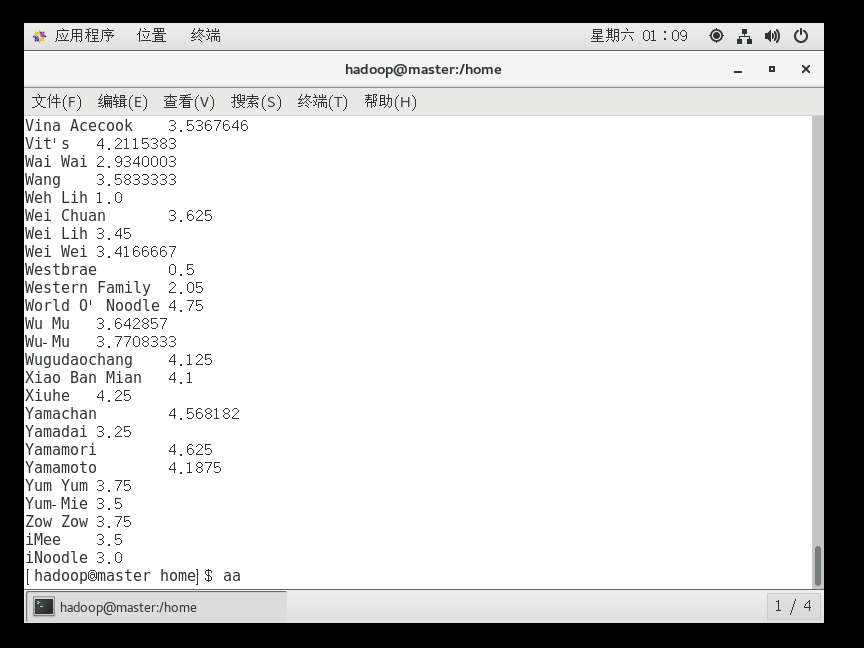
}

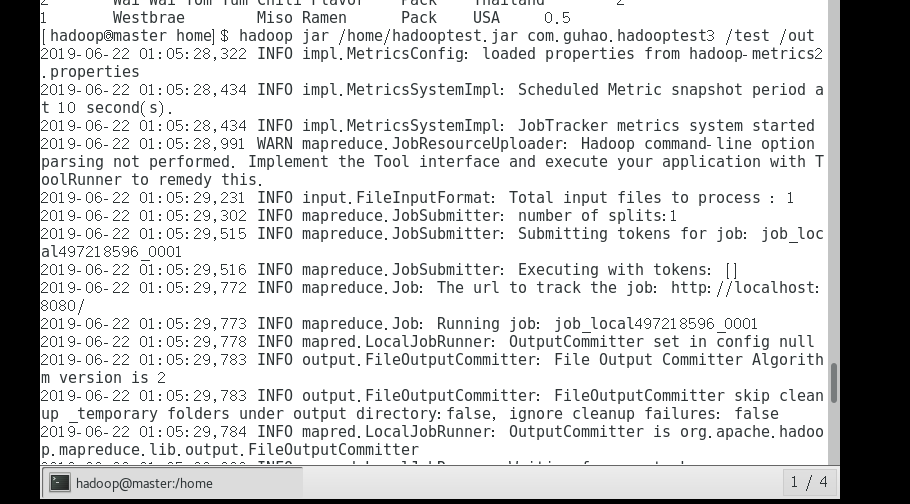
context.write(key, **new** FloatWritable(total/num));

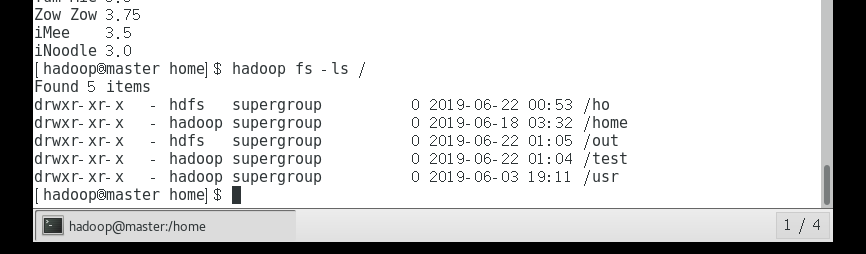
}

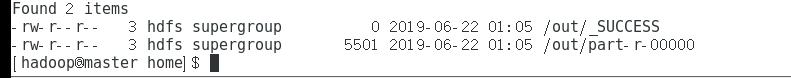
}

}









**五、实验体会**

此次试验内容不算复杂，遇到的问题主要还是文件权限的问题，通过百度查找解决了问题成功将代码跑出来了，通过自己去查资料解决问题也是一种提升自己知识认知的一个过程，对实验有了更广泛深入的理解，有了更全面的认识，总之这次实验受益良多获益匪浅！