## 实验5：MapReduce编程

### 一、实验目的

* 掌握MapReduce工作流程；
* 掌握MapReduce编程。

### 二、实验平台

* 操作系统：Linux（建议Ubuntu16.04）；
* Hadoop版本：3.3.1；
* JDK版本：1.8
* Java IDE：Eclipse。

### 三、实验步骤

（一）参照http://dblab.xmu.edu.cn/blog/631-2/（Hadoop2.0以上版本）或http://dblab.xmu.edu.cn/blog/2481-2/（hadoop3.0以上版本）中的介绍，在eclipse中编程实现WordCount程序，并分别在eclipse中编译运行和打包成jar包运行。

（二）编写MapReduce程序，实现对文本文件中的26个英文字母的出现次数进行统计，输出如：a(A):106

        b(B):78

        ......

（三）编写MapReduce程序，实现数据去重

**（四）**编写MapReduce程序，实现数据排序

**（五）**编写MapReduce程序，计算学生平均成绩

**（六）**到http://www.nber.org/patents/下载美国专利数据，编写MapReduce程序实现以下功能：

（1）专利被引次数统计

（2）国家专利数统计，统计一个国家占全球的专利申请比例

### 实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验报告题目：HBase操作及编程 | | | | | |
| 学号（班级）： | 2019212212251 | 姓名： | 洪灿麟 | 日期： | 2021/10/21 |
| 实验环境：ubuntu | | | | | |
| 实验内容与完成情况：   1. **参照http://dblab.xmu.edu.cn/blog/631-2/（Hadoop2.0以上版本）或http://dblab.xmu.edu.cn/blog/2481-2/（hadoop3.0以上版本）中的介绍，在eclipse中编程实现WordCount程序，并分别在eclipse中编译运行和打包成jar包运行。**   **部分代码：**    **Map方法**  **Reduce方法**    **Eclipse导入分布式配置文件**      **运行前检查hdfs是否存在output文件**    **打包jar并进行编译**    **查看是否存在**    **查看统计结果**     1. **编写MapReduce程序，实现对文本文件中的26个英文字母的出现次数进行统计**   **Map代码**    **Reduce**    **导出jar**    **运行**    **结果**     1. 编写MapReduce程序，实现数据去重   Map代码  只需要获取value，传输value即可  Reduce代码    只需要把输出类型更改为Nullwritabale即可，同时job任务设置null输出格式  运行结果       1. 编写MapReduce程序，计算学生平均成绩   创建排序文件并上传到hdfs上    每一行的属性有名字，科目，以及成绩。  新建一个Student类继承writablecompare，属性有名字，科目，成绩以及平均分    改写tostring以及序列化和反序列化    Map代码    输出的key为学生的名字，输出的value为Student类，方便后续的reduce以学生姓名作为分组  Reduce阶段    把同一个学生的成绩进行迭代计算，最后求出平均数，输出的key为学生名字，value为Student类，由于改写了student的输出为平均数取小数两位，因此最后会输出平均成绩。  设置job的输出配置等等    打包并导出为wordcount\_ave    运行并查看结果       1. 编写MapReduce程序， 实现数据排序   对于上一步的学生成绩进行排序    首先将Student类的compareTo方法改写，这边改成按平均成绩高低排序输出。  Map代码    样例用上一步算出来的学生的平均成绩来进行从高到低的排序。  因此字符串分割用制表符分割，第一个是名字第二个就是成绩。  Reduce代码    将迭代器里的名字直接取出来，因为reduce会进行分组排序操作，而student类规定了分数高的在前面，所以最后会形成很多个单一的分组，刚好可以实现排序功能。  Job代码    导出包    原文件    运行后    **（六）**到http://www.nber.org/patents/下载美国专利数据，编写MapReduce程序实现以下功能：  （1）专利被引次数统计  原文件    把第一行的备注给删了，第一列数字是专利号，第二列数字是引用次数。  Map阶段    输出key格式为Text,保存专利号，value为Longwritable，保存引用次数。  Reduce阶段    输出直接把专利号输出，而引用次数迭代相加。  Driver端    输入输出按map和reduce阶段的设置，设置50个ReduceTasks来将文件输出成多个小文件，太大的文件会打不开。  导出jar并运行    查看是否生成文件    运行成功，下载到本地    随便打开一个子文件查看利用计算器验证一下    成功运行  （2）国家专利数统计，统计一个国家占全球的专利申请比例  分为两个mapreduce来进行处理，这样可以提高效率。  首先第一个maprudece做统计操作，统计每个国家的专利数。  Map阶段    输出key为国家名字，输出value为1，便于reduce统计  Reduce阶段    把相同国家加起来，最后输出国家和国家的专利数。  导出包并进行运算    可以看出一共有2923922行，处理完后输出162行，也就是有162个国家。 | | | | | |
| 此时对这个文件进行第二个mapreduce  自定义一个类来储存名字，以及专利数和输出最后所占的比例，后面如果有排序需求的话也可以用于后续操作。      Map阶段  输出的key为Text，设置相同内容来将所有国家分成同一组。  Reduce阶段    设置一个链表来存储myCont，第一次迭代器迭代把值给传递给链表，同时计算总文件数。  第二次遍历列表，计算每个国家的占比，最后输出。  此时的key为国家名字，value为自定义类型，输出为专利数和占比。  Driver阶段    设置输出格式  导包并运行    可以看出输入输出均没问题，成功运行。  查看文件成功输出专利总数以及所占比例    出现的问题：   * + - 1. 新建map类和reduce类来处理，编译没报错但是利用hadoop进行jar包的使用时报错。     输出结果时格式有问题，字符串会多打印一份   * + - 1. 设置输出格式的时候会报错，无论只能用特定的格式，更改不了kv格式。       2. 自定义排序的时候报错， Unable to initialize MapOutputCollector org.apache.hadoop.mapred.MapTask$MapOutputBuffer       3. 排序输出后老是倒序，更改了reduce输出格式后仍然没有解决。      * + - 1. 输出信息的时候，reduce接受到了map的五条信息，但是输出却一条都出不来。       2. 利用列表来保存迭代器里面的参数， 但是输出的时候只保存了最后一个参数，前面保存的全是最后的信息。 | | | | | |
| 解决方案（列出遇到的问题和解决办法，列出没有解决的问题）：  问题通过反复琢磨都解决完了。  第一个问题是因为map设计的时候第一个参数传递的是longwritable的形式，而使用的时候是用object的形式，所以报错了。而且用了super就不能运行，不知道为什么，这一点到现在也还不清楚。  第二个问题可能是传递参数的时候因为经过了一次combine规约操作，所以导致中间用多进行了一次reduce，因此最后输出就会多一份。  第三个问题也是因为没有删除掉job设置的combine操作，因为在本地会先进行一次reduce操作，而这个时候所用的不能是相同的reduce，不然格式会不一致，所以就报错，这个真的是不懂代码的意思完全发现不到。  第四个问题也困扰了很久，因为新建了自定义类来进行排序，编译挺顺利但是最后一直报错。一直找原因，一开始以为是没有默认构造函数，结果不是这个原因，后面更改了接口也不行，重写了gethash以及equal都不行。试了很多种构造方式，自定义各种类还是不行，只有系统默认的类型才可以。到后面发现是因为主class那边多导了个包，可能就是这个包影响了，删除了就解决了。    第五个问题就是自己蠢了，因为前面用了很多的方法来验证mapreduce的原理，因此创了很多个类，然后mapper的构造参数写错了一个，导致前面传参的时候接受不到，因此无论我怎么改变输出的都会是这个，太可恶了！  第六个问题是因为迭代器的问题，由于reduce接受到的迭代器用了两次，所以第二次的迭代失效了，可能是因为用了就会丢掉的关系吧。更改了代码测试了一下果然是这个问题。查询了一下百度发现迭代器原来是用完即抛，这牵扯到了java知识。      第七个问题    原因在于mt对象在一开始就创建实例了，导致最后加进去的对象其实是mt的实例，也就是地址，因此最后输出的都会是最新的那个。又是Java基础知识不过关了。 | | | | | |