**HADOOP实验一**

1. **实验要求**
2. 学会使用基本的命令行编译运行mapreduce程序。
3. 对mapreduce代码编写有基本的熟悉，对mapreduce执行过程有更深层次的了解。
4. 在实验过程中学会自己查找解决问题的方法，培养处理独自处理问题的能力。
5. **实验过程**
6. 将我们将 Hadoop 的 classhpath 信息添加到 CLASSPATH 变量中，在~/.bashrc中添加最后两行命令。

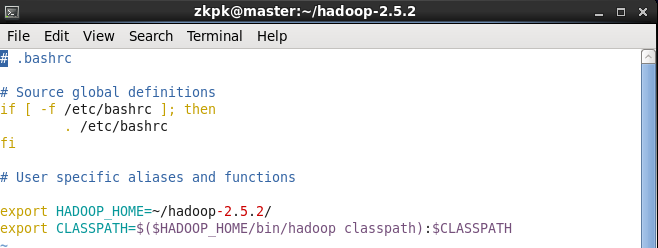


图1 配置CLASSPATH变量

1. 编译java文件

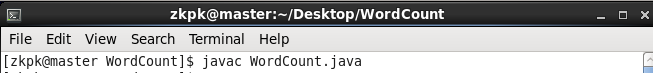


图2 编译java文件

1. 将class文件打包成jar包

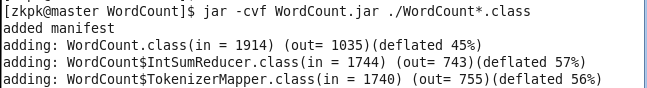


图3 将class文件打包

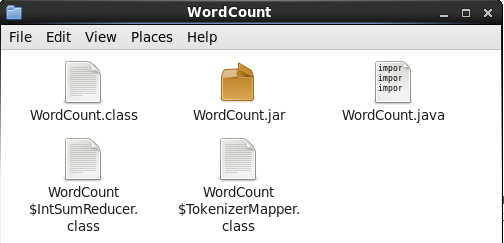


图4 生成的jar包

1. 新建input1文件夹：



图5 新建文件夹存储数据

1. 上传待统计文本到输入文件夹



图6 上传文件到HDFS

1. 运行WordCount.java编译生成的jar包。

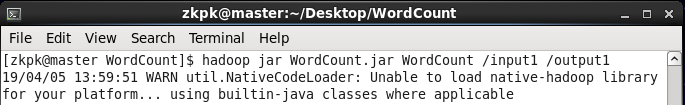


图7 使用Hadoop运行jar包

1. 查看统计结果（此处为了方便直接在eclipse中访问HDFS）



图7 待统计文件夹

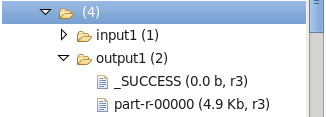


图8 生成结果文件

1. 查看WordCount统计结果

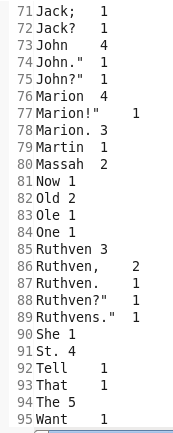


图9 词频统计结果。

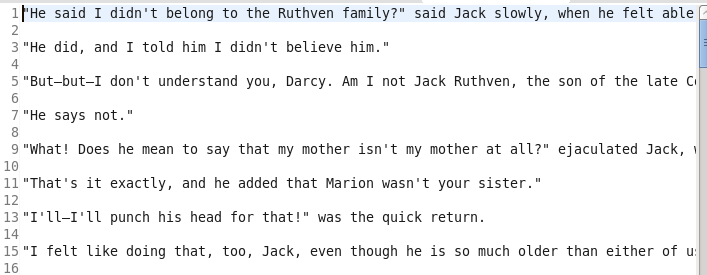


图10 待统计原文

在图10中是随意找的一篇英文小说，可以看到，在图9中由于使用的是空格作为分隔符，因此有些单词是连着标点符号的，这显然不是我们想要的，因此对其进一步进行处理。处理之后的结果如图11所示。

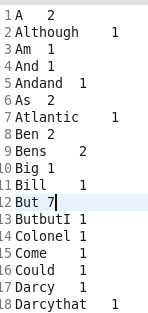


图11 进一步处理的统计结果

**三．部分代码解释**

**Map函数：**

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

private Text word = new Text();

public TokenizerMapper(){}

public void map(Object key, Text value, Mapper<Object, Text, Text,

IntWritable>.Context context) throws IOException, InterruptedException {

/\* 设置输入key、value对以及输出key、value对的格式 \*/

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

/\* 以空格为标识符进行分词 \*/

while(itr.hasMoreTokens()){

String stri2=itr.nextToken().replaceAll("\\pP","");

/\* 对标点符号进行处理，使用正则去掉标点符号 \*/

this.word.set(stri2);

context.write(this.word, one);

/\* 发送key-Value对 \*/

}

**Redeuce函数：**

private IntWritable result = new IntWritable();

public IntSumReducer(){}

public void reduce(Text key, Iterable<IntWritable> values, Reducer<Text, IntWritable,Text,IntWritable>.Context context) throws IOException,

InterruptedException{

int sum = 0;

IntWritable val;

for(Iterator itr = values.iterator(); itr.hasNext(); sum += val.get()){

val = (IntWritable)itr.next();

}

/\* 对相同key的value列表进行出现次数的相加 \*/

this.result.set(sum);

context.write(key, this.result);

/\* 发送key、value对 \*/

1. **实验总结**

通过本次实验，我对mapreduce代码的编写有了基本的掌握，虽然在上学期已经在云计算课程中编写过相关的代码，现在都相当于是重新熟悉一下mapreduce代码编写的过程。