云计算实践实验二报告：Hadoop实践

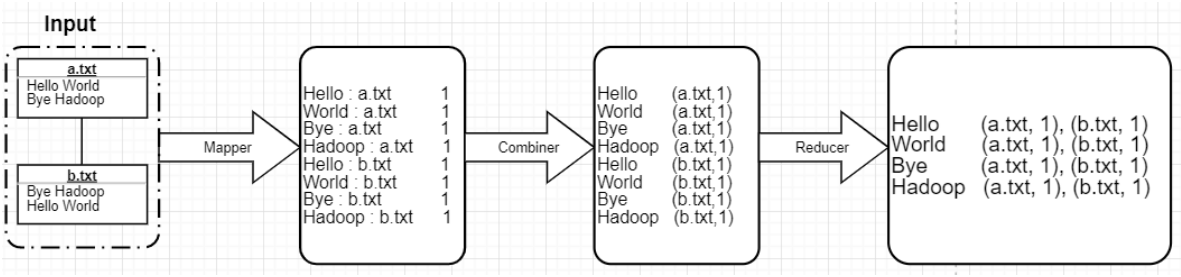
InvertedIndex程序

算法思路

在WordCount程序的基础上，该程序在算法上新增了Combiner类，即总共分为以下三个部分：

- Mapper类：以input文件夹中的每个文件作为输入，首先提取出每行的单词，同时保存其所在的文件名，对每一个单词都输出以 $word:filename \rightarrow "1"$ 格式的一个映射；
- Combiner类：以Mapper传来的映射为输入，用WordCount的方法累加每个key对应的value值，得到每个文件中每个单词的词频。由于最终需要得到一个 $word \rightarrow (filename1, count), (filename2, count)\cdots$格式的映射，即需要对每个单词进行汇总，因此对每个传来的映射，对key进行切片分离出word和filename，输出以$word \rightarrow (filename, count)$格式的映射；
- Reducer类：以Combiner传来的映射为输入，对相同key（即同一个单词）对应的value，将其用字符串fileList拼接起来，最后以其作为value输出；

流程图如下：



关键部分代码

- Mapper类

```
public static class InvertedIndexMapper extends Mapper<LongWritable, Text, Text,
IntWritable> {
        //定义key文本和FileSplit
        private Text keyInfo = new Text();
        private FileSplit split;
        // 统计词频时，需要去掉标点符号等符号，此处定义表达式
        private String pattern = "[^a-zA-Z0-9-]";

        @Override
        protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            //简单起见，只获取文件名
```

```
            split = (FileSplit)context.getInputSplit();
            int index = split.getPath().toString().indexOf("file");
            String fileName = split.getPath().toString().substring(index);
            // 将每一行转化为一个String
            String line = value.toString();
            // 将标点符号等字符用空格替换，这样仅剩单词
            line = line.replaceAll(pattern, " ");
            // 将String划分为一个个的单词
            String[] words = line.split("\\s+");
            // 将每一个单词初始化为词频为1，如果word相同，会传递给Reducer做进一步的操作
            for (String word : words) {
                if (word.length() > 0) {
                    //用单词:文件名的格式作为key
                    keyInfo.set(new Text(word)+":"+fileName);
                    context.write(new Text(keyInfo), new IntWritable(1));
                }
            }
        }
```

- Combiner类

```
//Combiner:将map传入的 单词:文件名-> 1 统计为 单词-> (文件名,词频)
    public static class InvertedIndexCombiner extends Reducer<Text, IntWritable,
Text, Text> {
        @Override
        protected void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
            // 初始化词频总数为0
            Integer count = 0;
            // 对key是一样的单词，执行词频汇总操作，也就是同样的单词:文件名，若是再出现则词频累
加
            for (IntWritable value : values) {
                count += value.get();
            }

            int index = key.toString().indexOf(":");
            // 重新设置key值为单词
            String word = key.toString().substring(0, index);
            Text outKey = new Text(word);

            // 重新设置value值由文件名和词频组成
            String fileName = key.toString().substring(index+1);
            Text outValue = new Text("("+fileName+", "+count+")");
            //输出
            context.write(outKey, outValue);
        }
    }
```

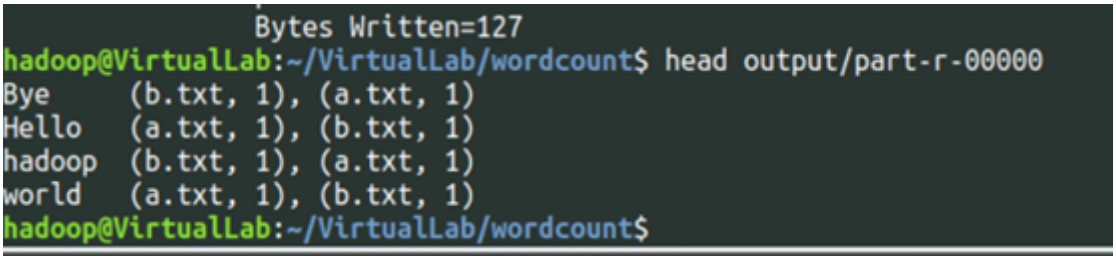
- Reducer类

```
//Reducer:将Combiner传入的 单词-> 文件名:词频 统计为 单词 -> (文件名1, 词频), (文件名2, 词
频)
    public static class InvertedIndexReducer extends Reducer<Text, Text, Text,
Text> {
        @Override
        protected void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
            String fileList = new String();
```

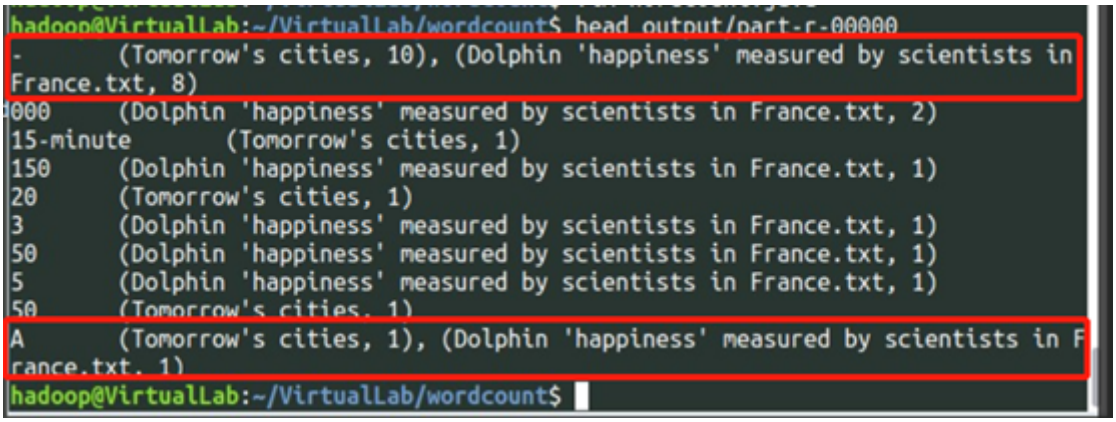
```
            // 对key是一样的单词，执行词频汇总操作，也就是同样的单词，若是再出现则使其value值拼接在一起
            for (Text file : values) {
                fileList += file.toString() + ", ";
            }
            //除去末尾的", "
            fileList = fileList.substring(0,fileList.length()-2);
            // 最后输出汇总后的结果，注意输出时，每个单词只会输出一次，紧跟着该单词的词频
            context.write(key, new Text(fileList));
        }
    }
```

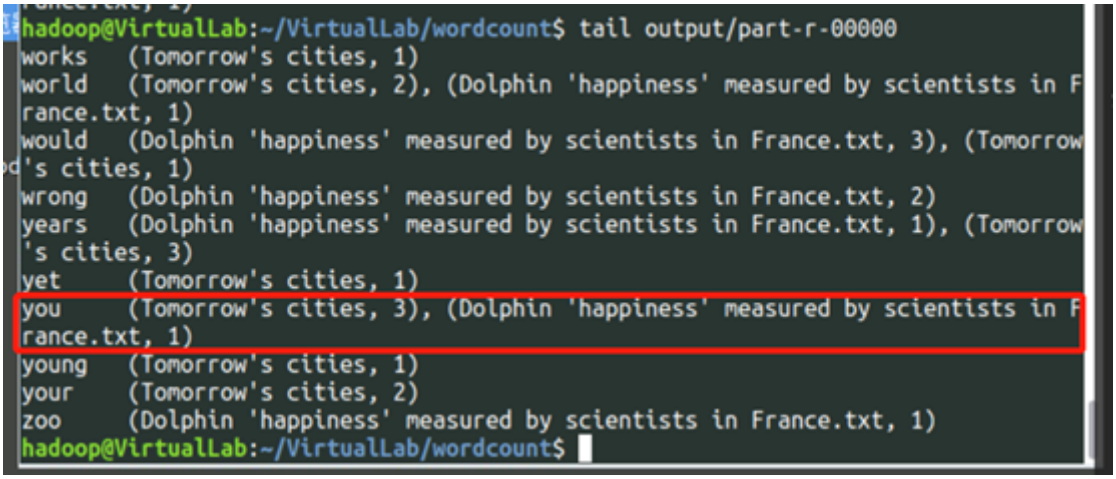
实验结果截图

- 对样例文本文件



- 对新闻文本文件





遇到的问题以及解决方法

- **对java语法以及hadoop使用尚不熟悉**

  在所给的WordCount程序代码的基础上加以修改，减少了不必要Debug的耗时

- **对vim编辑器的使用尚不熟悉**

  在对vim编辑器使用时，连将代码从一个文件复制粘贴到另外一个文件都成问题，百度许久也没解决，通过手打代码解决

- **FileSplit的使用，Text、String、Int、IntWritable类型之间的转换不熟悉**

  百度查看语法和示例

- **添加了Combiner类后，combiner类和reducer类的@Override处报错：方法不会覆盖或实现超类型的方法**

  本来觉得是combiner类和reducer类的传参类型必须相同，全都改为(Text,Iterable< Text >, Text, Text)后仍然报错，百度后发现可以删除不会报错，删除后成功运行

实验感想

本次实验总共花了一个下午解决，虽然其中走了许多弯路但总体不算太难，感谢TA对实验过程和MapReduce框架的详细说明，让算法的思路非常明了。

但原先通过提示没有办法找到算法思路，通过百度查看了InvertedIndex程序的一些思路，发现加入Combiner类后就非常容易完成，于是参考了该思路。但加入之后又出现了一些莫名其妙的报错，其中也不断地百度了一下java和hadoop语法的细节和用法，获益良多。

不过最后却在vim编辑器上的使用花了很多时间，不知道为什么用百度所说的复制粘贴方法无法将代码转移到另外一个文件，最后通过发邮件的方法解决，还是需要掌握一下vim编辑器的使用细节。

原本想在自己的虚拟机上完成，但发现vm虚拟机的ubuntu不知道为什么安装第一次可以使用，第二次就没反应完全开不了，百度也找不到有效的解决方法，非常感人的经历，最后还是在课堂已搭好的环境进行实验。

总体来说，在本次实验中了解了Hadoop中MapReducer框架的基本使用，并且加入了Combiner模块，还在实验过程中初步了解了java和hadoop语法的一些细节，收益匪浅。