Assignment1：实现基于MapReduce框架的算法

作业要求

1. 搭建MapReduce基本框架
2. 实现上课介绍的基于MapReduce的算法，本次作业选择实现自然连接的算法

作业过程

一、环境搭建

在Windows系统下搭建Hadoop的伪分布式集群，使用IDEA编写Java代码进行实验测试。Hadoop版本为2.10.1，jdk版本为21.0.2。

- 搭建Hadoop集群的过程参考https://blog.csdn.net/Geeksongs/article/details/111940739，这里不做过多介绍，启动Hadoop集群方法是打开命令行窗口，输入`%HADOOP_HOME%/sbin/start-all.cmd`，会出现多个窗口，在浏览器输入`localhost:50070`可正常访问即成功。
- IDEA创建Maven项目，配置`pom.xml`，导入依赖。

二、代码实现

MapReduce直接继承Hadoop的Mapper类和Reduce类，结构参考如下

```
class WordCount{
    class TokenizerMapper extends Mapper{
        void map(Object key, Text value, Context context){
            // 将输入文本转化为键值对
        }
    }
    class IntSumReduce extends Reduce{
        void reduce(Text key, Iterable<IntWritable> values,Context context){
            // 将划分排序后的同一键值的键值对进行操作
        }
    }
    void main(String[] args){
        // 设置参数、路径，提交任务给hadoop集群处理
    }
}
```

根据MapReduce框架实现上课所讲的自然连接算法，将表Table_1(a,b)与表Table_2(b,c)进行自然连接：

- Map：输入两张表将Table_1(a,b)转为键值对(b,(Table_1,a))，将Table_2(b,c)转为键值对(b,(Table_2,c))发射
- Reduce：收到键值同为b的所有键值对(b,(Table_1,a))与(b,(Table_2,c))的列表，将a、c组合与b一同输出

下面通过代码注释对操作过程进行说明：

Map

```
public static class TupleMapper extends Mapper<Object, Text, Text, Text> {
    public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
    // 获取当前的表名:Table_1.txt / Table_2.txt
    org.apache.hadoop.mapreduce.InputSplit inputSplit = context.getInputSplit();
    String fileName = ((org.apache.hadoop.mapreduce.lib.input.FileSplit)
inputSplit).getPath().getName();
    // 将输入Text转为Sting类型,并根据"\n"分行
    String line = new String(value.getBytes(), StandardCharsets.UTF_8);
    StringTokenizer tokenArticle = new StringTokenizer(line,"\n");
    // 遍历每行,根据表名判断b的位置
    // 记录record="Table_1.txt a值"或"Table_2.txt c值"
    // 记录name = "b值"
    // 发射(key = name, value = record)
    while(tokenArticle.hasMoreElements()) {
        String record = fileName;
        String[] tokenLine = tokenArticle.nextToken().split(" ");
        String name;
        if(Objects.equals(fileName, "Table_1.txt")){
            record = record + " "+ tokenLine[0];
            name = tokenLine[1];
        }else {
            record = record + " "+ tokenLine[1];
            name = tokenLine[0];
        }
            context.write(new Text(name), new Text(record));
        }
    }
}
```

reduce

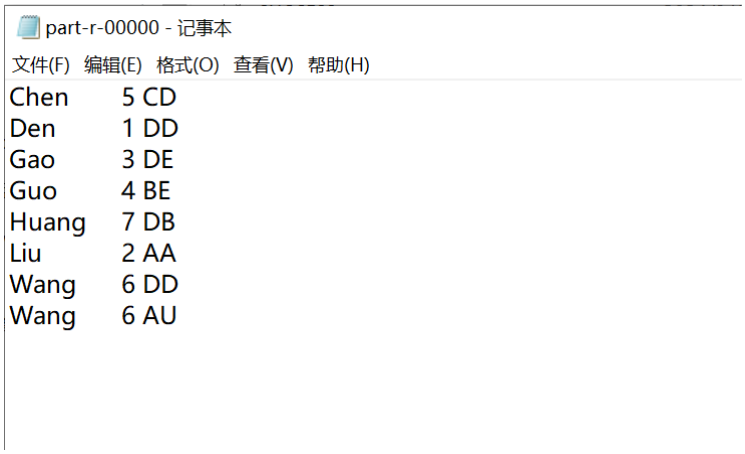
```
public static class MatchReduce extends Reducer<Text, Text, Text, Text> {
        public void reduce(Text key, Iterable<Text> values, Context context)
throws IOException, InterruptedException {
            // 对于一个键值b,left记录对应a值,right记录对应c值
            StringBuilder left = new StringBuilder();
            List<String> right = new ArrayList<>();
            for (Text value : values) {
                StringTokenizer itr = new StringTokenizer(value.toString());
                if (Objects.equals(itr.nextToken(), "Table_1.txt")) {
                    left.append(itr.nextToken());
                } else {
                    right.add(itr.nextToken());
                }
            }
            // 对a、c值组合，其中a值唯一,遍历c值即可
            // 发射(key = b, value = "a c")
            for (String tmp : right) {
                context.write(key, new Text(left+ " "+tmp));
            }
        }
    }
```

测试用例

在根目录下创建`input\Table_1.txt`、`input\Table_2.txt`，内容如下：

其中`Table_1.txt`各行表示(a,b)值，`Table_2.txt`各行表示(b,c)值

使用IDEA直接测试会在根目录下生成`output`目录，进入打开`part-r-00000`文件，查看结果符合预期：



打包运行

将项目打包成jar文件`hdfs1205.jar`，通过`NaturalJoin.bat`快速执行，已放置在项目根目录处，点击`NaturalJoin.bat`即可通过同目录下`input`内文件生成`output`目录，`part-r-00000`查看自然连接结果。

作业总结

本次作业主要做了两项工作，搭建Hapood环境以及实现MapReduce代码，其中对Hapood初次了解还不够深入，只是搭建了一个伪分布式环境，希望之后能够了解更多内容。在MapReduce代码实现自然连接算法中，整体代码量不大，主要是通过学习搭建一个MapReduce框架，只需要实现Map和Reduce阶段的代码，以及熟悉MapReduce常用的Text、IntWritable等数据类型，整体做下来对MapReduce的过程有了更深刻的了解。