# 大数据处理综合实验

# 实验四 SecondSort 实验报告

组别：2020st39

组长：171860662 山越 1025331716@qq.com

目录



## 一． 实验设计

### （一） 设计思路

对于使用 MapReduce 完成对数据的二次排序，由于 MapReduce 框架无论是默认排序或者是自定义排序都只对 key 值进行排序，但是原本数据不是 key 值，所以我们将原始数据的 key 和 value 组合成为一个新的 key 值，即将原始数据(value1,value2)组合成(<value1,value2>,value2)的形式，这样就能够对新 key 值进行排序。

首先我们定义了新 key 值的类 MyKey，由于数据均为数字，所以我们将 MyKey 中的两个变量都定义为 int 类型。

在 map 函数中，我们就对原始数据进行了操作，将原始数据的 key 和 value 组合成新的 key 值 mykey，再将原始的 value 值设置为 myvalue，将重新处理好的数据发送给 Reducer。

同时，我们通过 setGroupingComparatorClass 来对发往 Reducer 的键值进行分组操作，在这里我们定义分组函数 Comparator，将(<value1,value2>,value2)中 value1 相减，如果结

果等于 0，即 value1 的值相同，则将数据分到同一组中。

在 reducer 函数中，我们将传递过来的分好组的数据重新处理成原始数据的形式，通过 Mykey 的 getFirst 函数读取出 value1，并利用 iterator 读取每组的 value2 值，最后输出排序的结果。

## （二） 源代码说明

### SecondSort.java

```java
//SecondSort.java
import org.apache.hadoop.conf.Configuration;

public class SecondSort {
    public SecondSort() {
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        if (args.length != 2) {
            System.err.println("Usage: InvertedIndex <in> <out>");
            System.exit(2);
        }

        Job job = Job.getInstance(conf, "SecondSort");
        job.setJarByClass(SecondSort.class);
        job.setMapperClass(SecondSortMapper.class);
        job.setGroupingComparatorClass(SecondSortComparator.class); //自定义分组比较器
        job.setReducerClass(SecondSortReducer.class);
        job.setMapOutputKeyClass(MyKey.class); //map输出key类型为自定义的MyKey类型
        job.setMapOutputValueClass(IntWritable.class); //map输出value类型为IntWritable
        job.setOutputKeyClass(Text.class); //reduce输出key类型为Text
        job.setOutputValueClass(IntWritable.class); //reduce输出value类型为IntWritable
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

## MyKey.java

```java
//MyKey.java
import java.io.DataInput;

public class MyKey implements WritableComparable<MyKey> {
    private int first = 0;
    private int second = 0;

    public MyKey() {
    }

    public void set(int first, int second) {

    public int getFirst() {

    public int getSecond() {

  //由于MyKey要作为map传出的key，所以需要实现Comparable接口中的compareTo方法
  //先比较first，若相等再比较second，这样保证在Reducer端中同一组内，已经根据second降序排好
    public int compareTo(MyKey o) {
        if (this.first != o.first) {
            return this.first - o.first;
        } else {
            return this.second != o.second ? o.second - this.second : 0;
        }
    }

  //由于MyKey要作为map传出的key，所以需要实现Writable接口中的write和readFields方法
    public void write(DataOutput dataOutput) throws IOException {
        dataOutput.writeInt(this.first);
        dataOutput.writeInt(this.second);
    }

    public void readFields(DataInput dataInput) throws IOException {
        this.first = dataInput.readInt();
        this.second = dataInput.readInt();
    }
}
```

## SecondSortMapper.java

```java
//SecondSortMapper.java
import java.io.IOException;

public class SecondSortMapper extends Mapper<LongWritable, Text, MyKey, IntWritable> {
    private final MyKey mykey = new MyKey(); //输出的key
    private final IntWritable myvalue = new IntWritable(); //输出的value

    public SecondSortMapper() {
    }

    public void map(LongWritable key, Text value, Mapper<LongWritable, Text, MyKey, IntWritable>.Context context)
            throws IOException, InterruptedException {
        String[] str = value.toString().split("\t"); //对每行内容进行分割
        this.mykey.set(Integer.parseInt(str[0]), Integer.parseInt(str[1])); //填充MyKey类型的变量
        this.myvalue.set(Integer.parseInt(str[1]));
        context.write(this.mykey, this.myvalue); //发射键值对
    }
}
```

## SecondSortReducer.java

```java
//SecondSortReducer.java
import java.io.IOException;

public class SecondSortReducer extends Reducer<MyKey, IntWritable, Text, IntWritable> {
    private final Text first = new Text();

    public SecondSortReducer() {
    }

    public void reduce(MyKey key, Iterable<IntWritable> values, Reducer<MyKey, IntWritable, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        //进入reduce时，每一组具有相同的first，且组内已经根据second降序排好；不同组间也根据first升序排好
        this.first.set(Integer.toString(key.getFirst())); //获取当前组的相同的first
        Iterator<IntWritable> var4 = values.iterator();

        while(var4.hasNext()) {
            IntWritable value = (IntWritable)var4.next(); //获取该组中的下一条记录
            context.write(this.first, value); //发射键值对
        }

    }
}
```

## SecondSortComparator.java

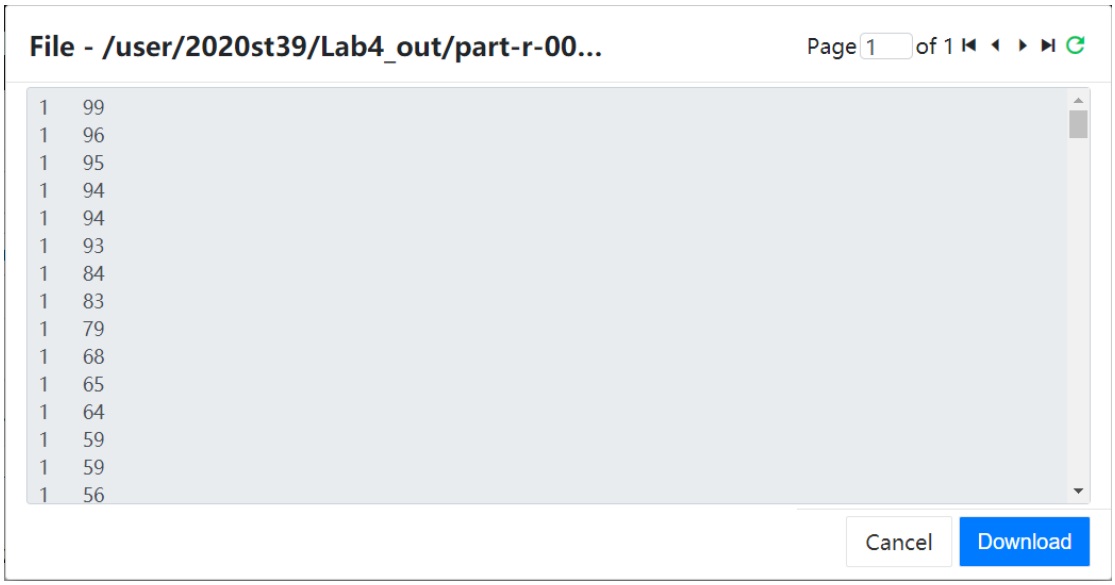
```java
//SecondSortComparator.java
import org.apache.hadoop.io.WritableComparable;

//利用reduce端的GroupingComparator来实现将MyKey.first相同的记录看成相同的key
public class SecondSortComparator extends WritableComparator {
    public SecondSortComparator() {
        super(MyKey.class, true);
    }

    public int compare(WritableComparable a, WritableComparable b) {
        MyKey mykey1 = (MyKey)a;
        MyKey mykey2 = (MyKey)b;
        return mykey1.getFirst() - mykey2.getFirst(); //比较两个key时，只比较first
      //若first相等，则会分到一组中，交由一个Reducer的一遍reduce函数处理
    }
}
```

# 二．程序运行和实验结果说明

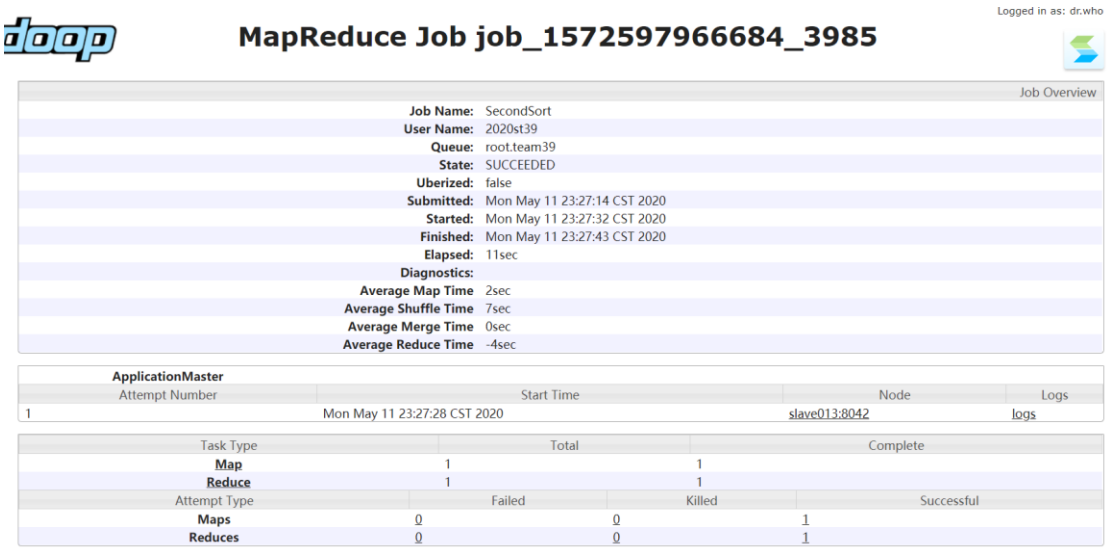
## （一） 输出结果

### 1. 输出文件

输出文件在 HDFS 上的路径：/user/2020st39/Lab4_out




## （二） WebUI 执行报告




# 三．小组成员分工

| 学号 | 姓名 | 分工 |
|---|---|---|
| 171860662 | 山越 | 提交集群和实验报告撰写 |
| 171860663 | 马少聪 | 代码完善和测试 |
| 171860664 | 谢鹏飞 | 代码实现 |
| 171860681 | 冯旭晨 | 实验报告撰写 |