实验四

2020st23小队成员信息:

171860572	组长: 侯策
171860570	周吴成
171860597	陈文龙
171860588	史文泰

设计思路

• 数据结构思路:

- o 本次实验文件输入每一行为两个整型数,分别代表第一次与第二次排序的数字,故定义 IntPair 类型存储。由于 key 为 IntPair 类型,且输出也是 IntPair ,故最终将处理过的 key 输出即可,输出 value 类型为 NullWritable,代表可以被序列化但没有实际意义。由 自定义 IntPair 衍生出如下设计:
 - 重写 compareTo()函数,先按从小到大比较 first ,相同则按从大到小比较 second ,在 MapReduce 中自动进行排序时被调用,完成**"对第一列数字按照升序排列,在每组中按照第二列数组进行降序"**的功能。
 - 实现 writableComparable<IntPair>接口,并重写 write()和 readFields()函数, 完成序列化与反序列化操作。
 - 重写 toString() 函数,用于最终结果输出。
- 。 由于显式定义了带参的构造函数,需要提供无参构造函数。

• Mapper 思路:

o 按行读入每条记录,分隔出属性值并新建 IntPair 对象发送即可。代码如下:

```
// 将前后整数分隔
String[] values = value.toString().split("\t");
// 构造IntPair对象
IntPair intPair = new IntPair(Integer.parseInt(values[0]),
Integer.parseInt(values[1]));
// 发送<IntPair, NullWritable>
context.write(intPair, NullWritable.get());
```

• Partitioner 思路:

o Partitioner 需要将相同哈希值的键值对合并在同一个分区中发送给同一个 Reducer ,在实验中使用默认的1个 Reducer ,使得所有结果放在一个文件中,故**无需重写** Partitioner 。

• Reducer 思路:

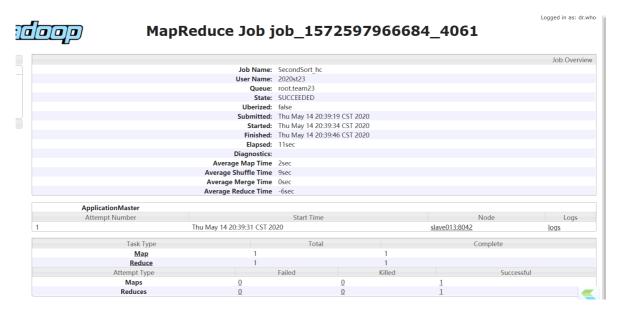
o Reducer 类只需将输入的键值对打印即可。代码如下:

```
for (NullWritable value : values) {
    // 直接输出结果即可
    context.write(key, NullWritable.get());
}
```

输出结果开头部分截图

```
File - /user/2020st23/lab4 hc 2/part-r-00...
                                                                   of 2 ◄ ◆ ▶ ▶  €
                                                           Page 1
0
    98
                                                                                98
0
0
   97
0
   97
   96
0
   95
0
   93
0
0
   91
0
   91
0
   90
0
   90
0
   87
0
   87
0
    86
0 86
```

WebUI执行报告



分工情况

由于本次实验整体代码量较少,且实验细节上与实验三有重叠之处,我们小组4人每个人独立完成了一版代码,并分别在集群上运行(集群上有运行记录)

提交的源代码只上交了其中一套。