大数据处理综合实验
——MapReduce课程实验介绍

南京大学 计算机科学与技术系

实验三 MyJoin

- **实验内容与要求**
  - 将order.txt和product.txt使用MapReduce进行join操作，输出到hdfs上
  - 使用hive将上一步结果建表，并通过查询语句查询结果

oid	odata	pid	oamount
1001	20190731	4	2
1002	20190731	3	100
1003	20190731	2	40
1004	20190731	2	23
1005	20190801	4	55
1006	20190801	3	20
1007	20190801	2	3
1008	20190801	4	23
1009	20190802	2	10
1010	20190802	2	2
1011	20190802	3	14
1012	20190802	3	18

pid	pname	price
1	chuizi	3999
2	huawei	3999
3	xiaomi	2999
4	apple	5999

oid	odata	pid	pname	price	oamount
1001	20190731	4	apple	5999	2
1002	20190731	3	xiaomi	2999	100
1003	20190731	2	huawei	3999	40
1004	20190731	2	huawei	3999	23
1005	20190801	4	apple	5999	55
1006	20190801	3	xiaomi	2999	20
1007	20190801	2	huawei	3999	3
1008	20190801	4	apple	5999	23
1009	20190802	2	huawei	3999	10
1010	20190802	2	huawei	3999	2
1011	20190802	3	xiaomi	2999	14
1012	20190802	3	xiaomi	2999	18

```
SELECT * from order
        LEFT JOIN product
        WHERE order.pid = product.id
```

实验三 MyJoin

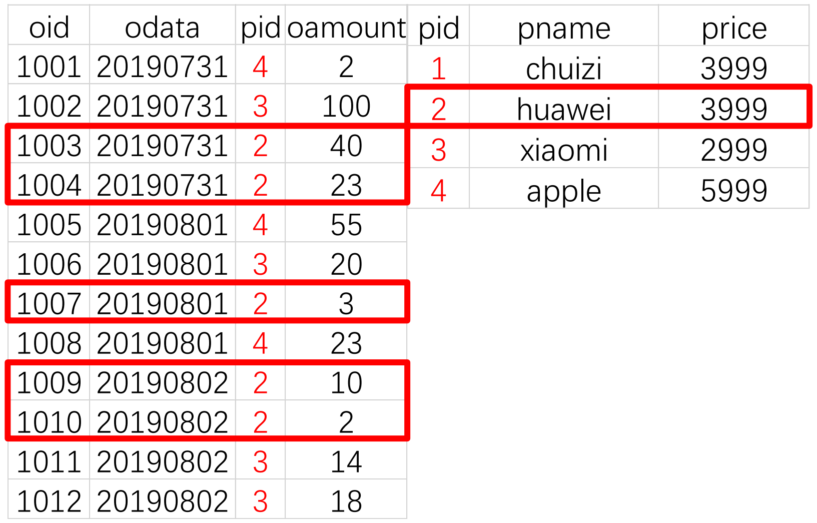
– 设计思路

– order.txt、product用mapper读进去

– 数据封装：OrderBean{oid, odata, pid, pname,price, oamount}

– 相同pid的数据同一组进入reducer

– 组内product放在order前面



oid	odata	pid	oamount
1001	20190731	4	2
1002	20190731	3	100
1003	20190731	2	40
1004	20190731	2	23
1005	20190801	4	55
1006	20190801	3	20
1007	20190801	2	3
1008	20190801	4	23
1009	20190802	2	10
1010	20190802	2	2
1011	20190802	3	14
1012	20190802	3	18

pid	pname	price
1	chuizi	3999
2	huawei	3999
3	xiaomi	2999
4	apple	5999

```
2 huawei 3999
1003 20190731 2 40
1004 20190731 2 23
1007 20190801 2 3
1010 20190802 2 2
```

实验三 MyJoin

– 数据封装

– 首先我们需要设计一个Java Bean用来表示product和order里的每一行数据，它包含以下字段：

- 订单ID: oid
- 订单日期: odata
- 商品ID: pid
- 商品名称: pname
- 商品单价: price
- 购买数量: oamount

```
public class OrderBean implements
WritableComparable<OrderBean> {

    private String oid;
    private String odata;
    private String pid;
    private String pname;
    private String price;
    private String oamount;
……
}
```

实验三 MyJoin

- Mapper
- 读取两个文件。

```
protected void setup(Context context) throws IOException,
InterruptedException {
    //读取文件
}

protected void map(LongWritable key, Text value, Context
context) throws IOException, InterruptedException {

}
```

实验三 MyJoin

– 排序

- Mapper<LongWritable, Text, OrderBean, NullWritable>
- 排序则根据OrderBean排序，放在key的位置上，需要自定义排序。
- 首先按照pid排序，相同pid则按照pname排序。

oid	odata	pid	oamount
1001	20190731	4	2
1002	20190731	3	100
1003	20190731	2	40
1004	20190731	2	23
1005	20190801	4	55
1006	20190801	3	20
1007	20190801	2	3
1008	20190801	4	23
1009	20190802	2	10
1010	20190802	2	2
1011	20190802	3	14
1012	20190802	3	18

pid	pname	price
1	chuizi	3999
2	huawei	3999
3	xiaomi	2999
4	apple	5999

```
2 huawei 3999
1003 20190731 2 40
1004 20190731 2 23
1007 20190801 2 3
1010 20190802 2 2
```

实验三 MyJoin

- 排序
- Mapper<LongWritable, Text, OrderBean, NullWritable>
- 排序则根据OrderBean排序，放在key的位置上，需要自定义排序。
- 首先按照pid排序，相同pid则按照pname排序。

oid	odata	pid	oamount
1001	20190731	4	2
1002	20190731	3	100
1003	20190731	2	40
1004	20190731	2	23
1005	20190801	4	55
1006	20190801	3	20
1007	20190801	2	3
1008	20190801	4	23
1009	20190802	2	10
1010	20190802	2	2
1011	20190802	3	14
1012	20190802	3	18

pid	pname	price
1	chuizi	3999
2	huawei	3999
3	xiaomi	2999
4	apple	5999

```
public int compareTo(OrderBean o) {
 //自定义比较方法
}
```

实验三 MyJoin

– 排序

– Mapper<LongWritable, Text, OrderBean, NullWritable>

– 排序则根据OrderBean排序，放在key的位置上，需要自定义排序。

– 首先按照pid排序，相同pid则按照pname排序。

oid	odata	pid	oamount	pid	pname	price
1001	20190731	4	2	1	chuizi	3999
1002	20190731	3	100	2	huawei	3999
1003	20190731	2	40	3	xiaomi	2999
1004	20190731	2	23	4	apple	5999
1005	20190801	4	55			
1006	20190801	3	20			
1007	20190801	2	3			
1008	20190801	4	23			
1009	20190802	2	10			
1010	20190802	2	2			
1011	20190802	3	14			
1012	20190802	3	18			

```
public int compareTo(OrderBean o) {
    //自定义比较方法
}
```

oid	odata	pid	pname	price	oamount
		2	huawei	3999	
1003	20190731	2			40
1004	20190731	2			23
1007	20190801	2			3
1009	20190802	2			10
1010	20190802	2			2
		3	xiaomi	2999	
…	…	…	…	…	…

实验三 MyJoin

– 自定义分组

oid	odata	pid	pname	price	oamount
		2	huawei	3999	
1003	20190731	2			40
1004	20190731	2			23
1007	20190801	2			3
1009	20190802	2			10
1010	20190802	2			2
		3	xiaomi	2999	
...	...	...	...	...	...

实验三 MyJoin

– 自定义分组

```
public class JoinComparator extends WritableComparator {
    public int compare(WritableComparable a, WritableComparable b)
{
        //自定义分组
    }
}
```

oid	odata	pid	pname	price	oamount
		2	huawei	3999	
1003	20190731	2			40
1004	20190731	2			23
1007	20190801	2			3
1009	20190802	2			10
1010	20190802	2			2
		3	xiaomi	2999	
…	…	…	…	…	…

实验三 MyJoin

– **Reducer**
– Reducer<OrderBean, NullWritable, OrderBean, NullWritable>

oid	odata	pid	pname	price	oamount
		2	huawei	3999	
1003	20190731	2			40
1004	20190731	2			23
1007	20190801	2			3
1009	20190802	2			10
1010	20190802	2			2
		3	xiaomi	2999	
…	…	…	…	…	…

实验三 MyJoin

– **Reducer**

– Reducer<OrderBean, NullWritable, OrderBean, NullWritable>

oid	odata	pid	pname	price	oamount
		2	huawei	3999	
1003	20190731	2	huawei	3999	40
1004	20190731	2	huawei	3999	23
1007	20190801	2	huawei	3999	3
1009	20190802	2	huawei	3999	10
1010	20190802	2	huawei	3999	2
		3	xiaomi	2999	
…	…	…	…	…	…

实验三 MyJoin

– Reducer

```
@Override
protected void reduce(OrderBean key, Iterable<NullWritable>
values, Context context) throws IOException,
InterruptedException {

    //取第一个product，然后依次填充到order内容当中
}
```

谢谢