# 作业六

## 林恺越 181098163

## 设计思路：使用MapReduce对这个问题进行处理的时候可以通过两个MapReduce任务完成这个需求：

1、对原始数据进行反转解析，找出都有谁的好友里面有该用户；例如对于原始数据

100, 200 300 400 500
200, 100 300 400
300, 100 200 400 500
400, 100 200 300

500, 100 300
600, 100

通过在map task中解析成 < 200，100>< 300，100>< 400，100>< 500，100>...的形式，这一系列的键值对表示所有代号为key值的用户，他的好友里面都有value值代表的用户。然后对这些键值对在reduce task中对key值相同的键值对（<100，200>< 100，300>< 100，400>...）进行拼接，拼接成< 100　200，300，400 ...>的形式作为第一次MapReduce任务的输出。

```
100     600,500,300,400,200,
200     400,100,300,
300     400,200,100,500,
400     100,300,200,
500     300,100,
```

上图为第一次MapReduce的输出，以其中的第一行数据<100　600，500，300，400，200 >为例，这一行数据的含义是：600，500，400，300，200 这些用户的好友里面都有100。

2、根据第一次MapReduce的输出结果，只要把分隔符后面的用户两两任意组合，就可以得到这两个用户的一个共同好友。以第一行数据为例，拆分之后可以得到：< 600-500 , 100>, < 600-300 , 100>, < 600-400，100> ... 然后再在reduce task中按照相同的key值对键值对进行拼接，就得到了整个数据集中任意两个用户之间的共同好友的列表。

## 代码实现：

```java
public static class findfriendstep1Mapper extends Mapper<LongWritable, Text, Text, Text> {
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] userAndfriends = line.split(", ");
        String user = userAndfriends[0];
        String[] friends = userAndfriends[1].split(" ");
        for (String friend : friends) {
            context.write(new Text(friend), new Text(user));

        }
    }
}

public static class findfriendstep1Reducer extends Reducer<Text, Text, Text, Text> {
    public void reduce(Text friend, Iterable<Text> users, Context context) throws IOException, InterruptedException {
        StringBuffer sb = new StringBuffer();
        for (Text user : users) {
            sb.append(user).append(",");
        }
        context.write(friend, new Text(sb.toString()));

    }
}
```

第一步

```java
public static class findfriendstep2Mapper extends Mapper<LongWritable, Text, Text, Text> {
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] friendAndusers = line.split("\t");
        String friend = friendAndusers[0];
        String[] users = friendAndusers[1].split(",");
        Arrays.sort(users);
        for (int i = 0; i <= users.length - 2; i++) {
            for (int j = i + 1; j <= users.length - 1; j++) {
                context.write(new Text("[" + users[i] + "," + users[j] + "]:"), new Text(friend));
            }
        }
    }
}

public static class findfriendstep2Reducer extends Reducer<Text, Text, UserTestWritable, NullWritable> {
    private UserTestWritable out = new UserTestWritable();
    public void reduce(Text user, Iterable<Text> friends, Context context) throws IOException, InterruptedException {
        StringBuffer sb = new StringBuffer();
        for (Text friend : friends) {
            sb.append(friend).append(",");
        }
        out.setValue(sb);
        out.setKey(user);
        //String sbstr=sb.toString();
        //context.write(user, new Text("[" +(sbstr.substring(0,sbstr.length()-1))+"]"));

        context.write(out,NullWritable.get());
    }
}
```

第二步（注释掉的是使用基本数据类型时的代码）

## 自定义数据类型： UserTestWritable在第二次reduce中使用，作为key的输出类型

定义了输出结果out，类型为UserTestWritable，out的key是两个user，val是他们的共同好友。

最主要的是设置输出格式，我设置成直接输出out的key和val的值，所以最终在reduce中是context.write(out,NullWritable.get())，key和val的值都由out输出，这一步的reduce实际上不需要value值，所以设置为空。

```java
package ff;

import org.apache.hadoop.io.WritableComparable;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.Text;
public class UserTestWritable implements WritableComparable<UserTestWritable> {

    private StringBuffer val;
    private Text key;
    public UserTestWritable() {
    }
    public UserTestWritable(Text key, StringBuffer val) {
        this.key = key;
        this.val = val;
    }
    @Override
    public int compareTo(UserTestWritable o) {
        return 0;
    }
    @Override
    public void write(DataOutput out) throws IOException {
        out.writeUTF(new String(key.copyBytes()));
        out.writeUTF(val.toString());
    }
    @Override
    public void readFields(DataInput in) throws IOException {
    }
    public Text getKey() {
        return key;
    }
    public void setKey(Text key) {
        this.key = key;
    }
    public StringBuffer getValue() {
        return val;
    }
    public void setValue(StringBuffer val) {
        this.val = val;
    }
    public String toString() {
        return  "("+key.toString()+"[" +(val.toString().substring(0,val.toString().length()-1))+"])" ;
    }

}
```

**实验结果：**

```
([100,200]:[300,400])
([100,300]:[200,400,500])
([100,400]:[200,300])
([100,500]:[300])
([200,300]:[400,100])
([200,400]:[300,100])
([200,500]:[300,100])
([200,600]:[100])
([300,400]:[200,100])
([300,500]:[100])
([300,600]:[100])
([400,500]:[100,300])
([400,600]:[100])
([500,600]:[100])
```

**WEB截图：**

Logged in as: dr.who

# All Applications

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 0 | 2 | 0 | 0 B | 16 GB | 0 B | 0 | 16 | 0 | 2 | 0 | 0 | 0 | 0 |

Scheduler Metrics

| Scheduler Type | Scheduling Resource Type | Minimum Allocation | Maximum Allocation |
|---|---|---|---|
| Capacity Scheduler | [MEMORY] | <memory:1024, vCores:1> | <memory:8192, vCores:8> |

Show 20 entries Search:

| ID | User | Name | Application Type | Queue | StartTime | FinishTime | State | FinalStatus | Progress | Tracking UI | Blacklisted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|
| application_1604294175083_0002 | root | find friends step2 | MAPREDUCE | default | Mon Nov 2 13:25:43 +0800 2020 | Mon Nov 2 13:25:59 +0800 2020 | FINISHED | SUCCEEDED | | History | N/A |
| application_1604294175083_0001 | root | find friends step1 | MAPREDUCE | default | Mon Nov 2 13:25:26 +0800 2020 | Mon Nov 2 13:25:41 +0800 2020 | FINISHED | SUCCEEDED | | History | N/A |

Logged in as: dr.who

# Nodes of the cluster

- Cluster
  - About
  - Nodes
  - Node Labels
  - Applications
    - NEW
    - NEW_SAVING
    - SUBMITTED
    - ACCEPTED
    - RUNNING
    - FINISHED
    - FAILED
    - KILLED
  - Scheduler
- Tools

Cluster Metrics

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used | Memory Total | Memory Reserved | VCores Used | VCores Total | VCores Reserved | Active Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0 | 0 | 2 | 0 | 0 B | 16 GB | 0 B | 0 | 16 | 0 | 2 | 0 | 0 | 0 | 0 |

Scheduler Metrics

| Scheduler Type | Scheduling Resource Type | Minimum Allocation | Maximum Allocation |
|---|---|---|---|
| Capacity Scheduler | [MEMORY] | <memory:1024, vCores:1> | <memory:8192, vCores:8> |

Show 20 entries Search:

| Node Labels | Rack | Node State | Node Address | Node HTTP Address | Last health-update | Health-report | Containers | Mem Used | Mem Avail | VCores Used | VCores Avail | Version |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | /default-rack | RUNNING | lky181098163-slave-1:38530 | lky181098163-slave-1:8042 | Mon Nov 02 05:26:38 +0000 2020 | | 0 | 0 B | 8 GB | 0 | 8 | 2.7.2 |
| | /default-rack | RUNNING | lky181098163-slave-2:33091 | lky181098163-slave-2:8042 | Mon Nov 02 05:26:38 +0000 2020 | | 0 | 0 B | 8 GB | 0 | 8 | 2.7.2 |

Showing 1 to 2 of 2 entries First Previous 1 Next Last