

第四次实验报告

姓名	学号	邮箱	院系
姜帅	221220115	js13156223725@163.com	计算机学院

任务1: 社交网络中的互相关注好友

设计思路

- 输入阶段: 读取文件夹中的所有文本文件, 逐行解析用户及其关注列表。
- Map阶段: 为每对用户关系生成键值对, 键为排序后的用户对, 值表示关注发起者, 例如: key 为 A-B, value为 A
- Reduce阶段: 收集所有的值, 得到每个值对应的关注发起者的列表, 检查列表长度是否为2, 且两个值是否不同。若满足, 说明两个用户互相关注。输出互相关注的用户对, 值为空。

核心代码

以下是Mapper的核心代码片段, 附注释

```
1 // 输入格式: user_name: friend1 friend2 friend3 ...
2 String line = value.toString().trim();
3 if (line.isEmpty() || !line.contains(":")) {
4     return;
5 }
6 // 分割用户名和朋友列表
7 String[] parts = line.split(":");
8 String user = parts[0].trim();
9 String[] friends = parts.length > 1 ? parts[1].trim().split("\\s+") :
  new String[0];
10 // 为每个朋友生成键值对
11 for (String friend : friends) {
12     if (!friend.isEmpty()) {
13         // 创建排序后的用户对作为key, 确保user_A < user_B
14         String pairKey = user.compareTo(friend) < 0 ? user + "-" +
  friend : friend + "-" + user;
15         // value为当前用户, 表示这个用户关注了这个朋友
16         context.write(new Text(pairKey), new Text(user));
17     }
18 }
```

Reducer的核心代码片段:

```

1 // 收集所有用户
2 List<String> users = new ArrayList<>();
3 for (Text value : values) {
4     users.add(value.toString());
5 }
6 // 如果恰好有两个不同的用户，说明是互相关注
7 if (users.size() == 2 && !users.get(0).equals(users.get(1))) {
8     context.write(key, new Text(""));
9 }

```

执行过程及结果

在集群上输入以下命令执行作业:

```

1 yarn jar /home/221220115stu/task1-1.0-SNAPSHOT.jar
  com.example.MutualFriends /user/root/Exp4 /user/221220115stu/lab4/task1

```

输出结果文件位于 `/user/221220115stu/lab4/task1` 下, 输出结果的部分截图如下:

文件 - `/user/221220115stu/lab4/task1/pa...`

Page 1 of 4394

```

1-75
1-79
1-8
1-84
1-85
1-88
1-90
1-91
10-12
10-162
10-167
10-192
10-205
10-236
10-25

```

取消

下载

执行报告

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU V-Cores	Allocated Memory MB	Reserved CPU V-Cores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1745632591680_0380	221220115stu	Mutual Friends Finder	MAPREDUCE	2025class03	0	Tue May 6 03:26:01 +0800 2025	Tue May 6 03:26:01 +0800 2025	Tue May 6 03:26:20 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0		History	0

任务2: 社交网络寻找共同关注

设计思路

本任务旨在通过MapReduce框架查找社交平台中用户对的共同关注者。输入包括两部分：第一部分为互相关注用户对文件（格式为 `userA-userB`），第二部分为包含用户关注列表的文件夹（格式为 `user_name: friend1 friend2 ...`）。任务目标是计算每对互相关注用户的共同关注者，并根据共同关注者数量与阈值 `x` 的关系，将结果分为两类输出：`small`（共同关注者数量

$\leq x$) 和 `large` (共同关注者数量 $> x$)。整体流程如下:

该任务采用两阶段mapreduce处理

- 第一阶段mapper, 读入用户关注列表和互相关注对**两个文件**, 使用一个mapper统一处理, 处理过程为:
 - 如果输入来自用户关注列表, 例如 A: f1 f2 f3, 则输出的key为 A, value为 L: f1, f2, f3 这里的L起到一个标签的作用, 具体作用在下一条说明
 - 如果输入来自互相关注对, 例如 A-B, 则分别输出 (A, P:B) 和 (B, P:A), 这里的P同样是起到标签的作用, 目的是在接下来的reduce阶段中将用户关注列表信息和互相关注信息区分开
- 第一阶段的reducer处理过程为:
 - 对每个user, 收集其完整的关注列表 (L开头的 value)
 - 对每个user, 收集其完整的与之互相关注的用户 (P开头的value)
 - 对于每一对 user - P: other_user, 输出 (user-other_User, user 的关注列表), 用于下一阶段配对交集计算
- 第二阶段mapper:
 - 无实质性作用, 只是起到一个传递作用
- 第二阶段reducer:
 - 对每个互相关注对 user1-user2, 收集user1和user2的互相关注列表, 存放于两个HashSet中, 计算这两个集合的交集
 - 判断交集大小, 即user1和user2共同关注者的数量, 与阈值x的关系, 按照与x的大小关系判断是输出到large文件中还是small文件中

核心代码

一阶段Mapper的核心代码:

```
1 String line = value.toString().trim();
2 if (line.contains(":")) { // 处理关注列表: user: friend1 friend2 ...
3     String[] parts = line.split(":");
4     if (parts.length != 2) return;
5     String user = parts[0].trim();
6     String[] friends = parts[1].trim().split("\\s+");
7     context.write(new Text(user), new Text("L:" + String.join(", ",
8         friends)));
9 } else if (line.contains("-")) { // 处理互相关注对: userA-userB
10     String[] users = line.split("-");
11     if (users.length != 2) return;
12     String userA = users[0].trim();
13     String userB = users[1].trim();
14     context.write(new Text(userA), new Text("P:" + userB));
15     context.write(new Text(userB), new Text("P:" + userA));
16 }
```

一阶段Reducer的核心代码:

```

1 List<String> partners = new ArrayList<>();
2 String friendList = "";
3 // 收集互相关注信息 P:... 和关注列表信息 L:...
4 for (Text val : values) {
5     String value = val.toString();
6     if (value.startsWith("P:")) {
7         partners.add(value.substring(2));
8     } else if (value.startsWith("L:")) {
9         friendList = value.substring(2);
10    }
11 }
12 // 对于每一对user-P:other_user构建输出
13 for (String partner : partners) {
14     String pair = key.toString().compareTo(partner) < 0 ? key + "-" +
partner : partner + "-" + key;
15     context.write(new Text(pair), new Text(friendList));
16 }

```

二阶段Mapper的核心代码

```

1 String[] parts = value.toString().split("\\t");
2 if (parts.length != 2) return;
3 context.write(new Text(parts[0]), new Text(parts[1]));

```

二阶段Reducer的核心代码

```

1 List<String> lists = new ArrayList<>();
2 // 对于user1-user2, 收集user1和user2的关注列表信息
3 for (Text val : values) {
4     lists.add(val.toString());
5 }
6 if (lists.size() != 2) return;
7 // 将user1和user2的关注列表转化为集合, 取交集
8 Set<String> set1 = new HashSet<>
(Arrays.asList(lists.get(0).split(",")));
9 Set<String> set2 = new HashSet<>
(Arrays.asList(lists.get(1).split(",")));
10 set1.retainAll(set2);
11
12 if (set1.isEmpty()) return;
13 // 构建目标格式的输出生
14 String commonFriends = String.join(" ", set1);
15 Text formattedLine = new Text(key.toString() + ": " + commonFriends);
16 // 按照阈值,将输出结果分流
17 if (set1.size() <= threshold) {
18     mos.write("small", NullWritable.get(), formattedLine,
"small/part");
19 } else {
20     mos.write("large", NullWritable.get(), formattedLine,
"large/part");

```

```
21 }
```

执行过程及结果

在集群上输入以下命令执行作业: (x以10为例)

```
1 yarn jar /home/221220115stu/task1-1.0-SNAPSHOT.jar
  com.example.CommonFollowers /user/221220115stu/lab4/task1/part-r-00000
  /user/root/Exp4 10 /user/221220115stu/lab4/task2/s1
  /user/221220115stu/lab4/task2/s2
```

输出结果文件位于 `/user/221220115stu/lab4/task2/s2` 下 分
为 `/user/221220115stu/lab4/task2/s2/small/part-r-00000`
和 `/user/221220115stu/lab4/task2/s2/large/part-r-00000`

输出结果的部分截图如下:

- small:

文件 - `/user/221220115stu/lab4/task2/s2...`

Page 1 of 3143

```
1-189: 45 140 8 85 21
1-238: 88 45 48 15 161 21 140 105 84
10-12: 167 205 25
10-162: 25 236 167
10-167: 12 162 205 25
10-192: 236 205
10-205: 12 167 25 236 192
10-236: 25 205 192 162
10-25: 12 162 167 236 205
1000-1008: 996 857 1092
1000-1092: 1008 996 857
1000-1093: 952 996 984
1000-857: 1092 1008 996
1000-858: 952 996 941
1000-941: 952 996 858
```

取消

下载

- large:

文件 - `/user/221220115stu/lab4/task2/s2...`

Page 1 of 73385

```
1-100: 88 47 48 231 155 156 158 90 91 51 54 12 58 15 160 161 165 2 123 167 201 204 6 128 207 60 62 20 21 65 170
134 179 215 139 218 75 79 39 182 184 140 221 146 147 148 105 227 229 108 109 84 85
1-105: 88 45 47 48 231 155 156 238 90 91 54 12 15 161 165 166 2 123 167 201 204 6 127 128 8 207 60 62 20 21 65
170 134 137 215 139 218 35 79 39 184 140 221 100 146 147 148 227 229 108 109 84 41 85
1-108: 88 45 48 231 155 156 158 90 91 51 54 58 15 160 161 165 2 167 201 204 6 127 128 207 60 62 20 21 65 170
179 137 215 139 218 79 39 182 184 140 221 100 146 105 229 109 84 85
1-109: 88 45 47 48 231 155 156 158 90 91 51 54 58 15 160 161 166 2 167 201 6 127 128 8 207 60 21 170 134 179
137 215 139 218 79 39 182 184 140 100 146 148 105 227 229 108 84 85
1-12: 88 45 47 48 231 155 156 158 90 91 51 58 15 160 161 165 166 2 123 167 201 204 6 127 128 8 60 62 20 65 28
170 134 137 215 218 75 35 79 39 182 184 140 221 100 146 147 148 105 227 84 85
1-123: 88 45 47 48 231 155 158 90 91 51 12 58 15 160 161 165 166 167 201 204 127 128 8 60
```

取消

下载

执行报告

show 20 ▾ entries																		Search 115		
ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Reserved CPU Vcores	Reserved Memory MB	% of Queue	% of Cluster	Progress ▾	Tracking UI	Buttons
application_1745632591680_1009	221220115stu	Job2: Find common friends	MAPREDUCE	2025class03	0	Tue May 13 06:21:13 +0800 2025	Tue May 13 06:21:13 +0800 2025	Tue May 13 06:21:43 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0
application_1745632591680_1007	221220115stu	Job1: Emit pair and list	MAPREDUCE	2025class03	0	Tue May 13 06:20:51 +0800 2025	Tue May 13 06:20:51 +0800 2025	Tue May 13 06:21:11 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0

任务3: 好友推荐

设计思路

本任务通过MapReduce框架为社交平台用户推荐好友. 输入包括互相关注用户对文件（格式为 userA-userB）和用户关注列表文件夹（格式为 user_name: friend1 friend2 ...）。任务目标是为每个用户推荐最多5个未关注的好友候选，利用两阶段MapReduce作业完成。总体流程如下：

- 第一阶段 (Job1): 从互相关注用户对生成“朋友的朋友”候选对。
 - Map阶段设计: 输入为互相关注用户对 userA-userB，输出为互相关注关系生成双向键值对，即 (userA, userB) 和 (userB, userA)，表示双向好友关系。
 - Reduce阶段设计: 收集用户(例如Alice)的所有直接好友, 对好友列表两两配对, 例如收集 (Alice, Bob) 和 (Alice, Changxue) 为 (Alice, [Bob, Changxue])，输出 (Bob, Changxue)，(Changxue, Bob)
- 第二阶段 (Job2): 结合关注列表过滤已关注用户，保留每个用户最多5个推荐候选。
 - Map阶段设计: 输入为**Job1的输出**(候选对)以及**关注列表文件**, 分别解析候选对和关注列表, 进行输出, 例如:
 - 输入候选对为 (Bob, Changxue) 输出为 (Bob, C:Changxue)
 - 输入关注列表为 Bob: Alice David; 输出为 (Bob, F:Alice,David)
 - Reduce阶段设计: 输入为分组后的键值对，键为用户，值为候选列表 (C:) 和关注列表 (F:)
 - 分离已关注用户 (F:) 和候选用户 (C:)，从候选用户中移除已关注用户
 - 保留最多5个候选用户，格式化为逗号分隔的字符串
- 输出: 生成推荐结果，格式为 user: candidate1,candidate2,...

核心代码

Job1

Mapper阶段的核心代码:

```
1 String line = value.toString().trim();
2 if (line.isEmpty() || !line.contains("-")) return;
3 String[] parts = line.split("-", 2);
4 if (parts.length != 2) return;
5 // A-B 互相关注，双向输出
```

```

6 String a = parts[0].trim();
7 String b = parts[1].trim();
8 if (a.isEmpty() || b.isEmpty()) return;
9
10 user.set(a); friend.set(b);
11 ctx.write(user, friend);
12
13 user.set(b); friend.set(a);
14 ctx.write(user, friend);

```

Reducer阶段的核心代码:

```

1 // 收集所有直接好友
2 List<String> friends = new ArrayList<>();
3 for (Text t : vals) {
4     friends.add(t.toString());
5 }
6 // 两两配对，生成候选推荐对：如果 A 与 B 和 C 互粉，则输出 B->C 和 C->B
7 int n = friends.size();
8 for (int i = 0; i < n; i++) {
9     for (int j = 0; j < n; j++) {
10         if (i == j) continue;
11         String b = friends.get(i);
12         String c = friends.get(j);
13         outKey.set(b);
14         outVal.set(c);
15         ctx.write(outKey, outVal);
16     }
17 }

```

Job2

Mapper阶段的核心代码:

```

1 /*** === Job2 MapperA: 读取 Job1 的候选对 (userB \t userC) === */
2 String line = value.toString().trim();
3 if (line.isEmpty()) return;
4 String[] parts = line.split("\\t");
5 if (parts.length != 2) return;
6 String b = parts[0].trim();
7 String c = parts[1].trim();
8 if (b.isEmpty() || c.isEmpty()) return;
9
10 user.set(b);
11 tagAndVal.set("C:" + c);
12 ctx.write(user, tagAndVal);
13
14 /*** === Job2 MapperB: 读取关注列表 (user: f1 f2 ...) === */
15 String line = value.toString().trim();
16 if (!line.contains(":")) return;
17 String[] parts = line.split(":", 2);

```

```

18 String u = parts[0].trim();
19 String[] fs = parts[1].trim().split("\\s+");
20 if (u.isEmpty()) return;
21
22 // 将整个列表当成一个值
23 user.set(u);
24 tagAndVal.set("F:" + String.join(",", fs));
25 ctx.write(user, tagAndVal);

```

Reducer阶段的核心代码:

```

1 // 收集已关注列表
2 Set<String> followed = new HashSet<>();
3 // 收集所有候选
4 Set<String> candidates = new LinkedHashSet<>();
5 // 解析输入
6 for (Text t : vals) {
7     String s = t.toString();
8     if (s.startsWith("F:")) {
9         String[] fs = s.substring(2).split(",");
10        for (String f : fs) if (!f.isEmpty())
11            followed.add(f);
12    } else if (s.startsWith("C:")) {
13        String c = s.substring(2);
14        if (!c.isEmpty()) {
15            candidates.add(c);
16        }
17    }
18 }
19 // 过滤掉已关注的
20 candidates.removeAll(followed);
21 // 取前5
22 Iterator<String> it = candidates.iterator();
23 List<String> out = new ArrayList<>();
24 int cnt = 0;
25 while (it.hasNext() && cnt < 5) {
26     out.add(it.next());
27     cnt++;
28 }
29
30 if (!out.isEmpty()) {
31     result.set(String.join(",", out));
32     ctx.write(user, result);
33 }

```

执行过程及结果

在集群上输入以下命令执行任务:


```
1 yarn jar /home/221220115stu/task1-1.0-SNAPSHOT.jar \  
2 com.example.FriendRecommendation \  
3 /user/221220115stu/lab4/task1/part-r-00000 \  
4 /user/root/Exp4 \  
5 /user/221220115stu/lab4/tmp \  
6 /user/221220115stu/lab4/task3
```

输出结果文件位于 `/user/221220115stu/lab4/task3`

输出结果的部分截图如下:

文件 - `/user/221220115stu/lab4/task3/pa...`

Page 129 of 834

13591 13686,13522,13583,13681,13666
13592 13472,13685,13667,13594,13559
13594 13650,13647,13552,13488,13640
13595 13500,13669,13618,13529,13561
13596 13631,13607,13602,13571,13606
13599 13484,13522,13529,13494,13537
136 206,159,67
1360 1574,1537,1556,1754,1751
13600 13529,13530,13626,13603,13604
13601 13475,13660,13607,13631,13515
13602 13542,13484,13540,13571,13661
13603 13542,13678,13685,13641,13571
13604 13671,13571,13603,13606,13588
13605 13675,13636,13643,13659,13532
1360

取消

下载

执行报告

▼ entries																		Search: 115		
ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
zn_1745632591680_0444	221220115stu	FilterAndRecommend	MAPREDUCE	2025class03	0	Tue May 6 09:59:58 +0800 2025	Tue May 6 09:59:58 +0800 2025	Tue May 6 10:01:17 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0
zn_1745632591680_0443	221220115stu	GenerateCandidates	MAPREDUCE	2025class03	0	Tue May 6 09:59:09 +0800 2025	Tue May 6 09:59:09 +0800 2025	Tue May 6 09:59:56 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0
zn_1745632591680_0442	221220115stu	Mutual Common Friends	MAPREDUCE	2025class03	0	Tue May 6 09:50:39 +0800 2025	Tue May 6 09:50:39 +0800 2025	Tue May 6 09:51:05 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0
zn_1745632591680_0380	221220115stu	Mutual Friends Finder	MAPREDUCE	2025class03	0	Tue May 6 03:26:01 +0800 2025	Tue May 6 03:26:01 +0800 2025	Tue May 6 03:26:20 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A	N/A	N/A	0.0	0.0	<div></div>	History	0