实验三倒排索引——实验报告

一. 小组信息:

集群账号: 2017st15

141220138 臧风云 2714359840@qq.com

141220132 银琦 141220132@smail.nju.edu.cn

二. 实验环境

Ubuntu 14、Java 1.7.0_25、IntelliJ 17

三. 实验流程

- 1.在 Intellij 上生成 InvertedIndex.jar,拷贝到本机的 hadoop 用户下
- 2.在本机 Hadoop 用户下将 jar 包拷贝到集群下 scp -r InvertedIndex.jar 2017st15@114.212.190.91:~/
- 3.执行 jar 包

基础:hadoop jar InvertedIndex.jar InvertedIndex /data/wuxia_novels/. /user/2017st15/output 选作 1: hadoop jar InvertedIndex.jar FrequencySort /output/part-r-00000 /user/2017st15/sort-out

选作 2: hadoop jar InvertedIndex.jar TFIDF /output/part-r-00000 /user/2017st15/tf_out

4. 将执行后的文件保存到集群账户上

hadoop fs -get /user/2017st15/output hadoop fs -get /user/2017st15/sort-out hadoop fs -get /user/2017st15/tf_out

5. 将文件复制到本地查看

scp -r 2017st15@114.212.190.91:~/output ~/

scp -r 2017st15@114.212.190.91:~/sort-out ~/

scp -r 2017st15@114.212.190.91:~/tf_out ~/

四. Map 和 Reduce 的设计思路

(1) 倒排索引

Map: 将输入文本 value 信息按照空格进行拆分, 获取单词, 输入的 context 获取 mapper 分配的 split, 得到作家小说名, 将单词和作家小说名作为输出的 key, 输出的 value 为"1"。

Combiner:将 map 的输出作为输入,将一个 key 的所有 value 进行合并,得到一个单词在一个文件的总次数。对输入的 key 进行划分,将单词作为输出的 key,将作家小说名+':'+count 作为输出 value.

Reducer:对 Combiner的输出进行处理,输入的 key 为单词,输入的 values为所有的作家小说名+':'+count,计算 key 的总 count,除以该 key 出现的文件总数得到频率。对 values 进行合并作为输出的 value。

(2) 选作 1——频率排序

Map: 输入 key 无意义, value 为倒排索引输出文件的一行信息即一个单词的信息, 因为 MapReduce 会按照 key 自动排序, 对 value 进行拆分, 将频率作为输出的 key (Double Writable 类型), 将 value 的其他信息作为输出的 value。

Reduce:将 value 中的单词作为输出的 key, value 中的其他信息和频率(即输入的 key)作

为输出的 value。

(3) 选作 2——TF-IDF

Map: 输入的 key 无意义,输入的 value 为倒排索引的输出文件的一行信息。建立 ArrayList<String>Authors 存储作者信息,建立 ArrayList<Integer> times 存储词的数目,根据" 号来提取小说名,以及提取小说名中的作者名,并统计该词出现的数目。输出 key 为单词加作者,value 为 TF 和包含该次的文档数。

Reduce: 对 Map 输出进行处理,使用公式 Math.log((double)218 / (double)(count + 1)) / Math.log(2);计算 IDF。输出的 key 即为输入的 key,输出的 value 为 TF-IDF。(这里对文档总数的获取没有实现,所以直接使用了 218)。

五. 倒排索引代码分析

```
//mapper
class InvertedIndexMapper extends Mapper<Object,Text,Text,Text>{
    @Override
    //key 意义, value 为一行文本信息, context 包含文件名信息
    protected void map(Object key, Text value, Context context) throws IOException,
Interrupted Exception \\
    {
         //获取书名信息,根据文件名中的'.'进行拆分,去掉.txt.segmented
         FileSplit fileSplit = (FileSplit)context.getInputSplit();
         String[] split = String.valueOf(fileSplit.getPath().getName()).split("\\.");
         String filename;
         if(split.length==4)
             filename=split[0]+split[1];
         else
             filename=split[0];
         //对 value 进行划分获得单词组
         StringTokenizer itr = new StringTokenizer(value.toString());
         while(itr.hasMoreTokens()) {
             //获得 keyinfo ,格式为"单词+: +filename"
             Text keyinfo = new Text(itr.nextToken()+':'+filename);
             //对每个单词依次输出, value 为'1'
             context.write(keyinfo, new Text("1"));
        }
    }
}
//combiner
class InvertedIndexCombiner extends Reducer<Text,Text,Text,Text,Text>{
    //key 格式为"单词+: +filename", values 为含有 key 的所有键值对中的 value
    protected void reduce(Text key, Iterable<Text>values ,Context context)throws IOException,
InterruptedException
    {
         Text valueinfo =new Text();//输出 value
         Text keyinfo = new Text();//输出 key
```

```
//计算一个单词在一个文件出现的次数
         int sum=0;
         for(Text x:values){
             sum+= Integer.parseInt(x.toString());
        }
         //按照 ':' 位置对 key 划分
         int index = key.toString().indexOf(":");
         //单词作为输出 key
         keyinfo=new Text(key.toString().substring(0,index));
         //输出的 value 格式为"filename+: +count"
         valueinfo=new
Text(key.toString().substring(index+1,key.toString().length())+':'+Integer.toString(sum));
         context.write(keyinfo,valueinfo);
    }
}
//reducer
class InvertedIndexReducer extends Reducer<Text,Text,Text,Text> {
    @Override
    //key 为单词, values 为含有 key 所有文件中数量的集合,格式为"filename+: +count"
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException
    {
         ArrayList<String>novel = new ArrayList<String>();//用于记录 filename
         ArrayList<Integer>times = new ArrayList<Integer>();//记录 key 在每个文件出现次数
         for(Text x : values) {
             //根据':'拆分
             int index = x.toString().indexOf(':');
             String novelname = x.toString().substring(0,index);
             //count 计算单词出现作品数量
             int count = Integer.parseInt(x.toString().substring(index+1,x.toString().length()));
             if(novel.indexOf(novelname)==-1){
                  novel.add(novelname);
                 times.add(count);
             }
             else{
                 int temp = times.get(novel.indexOf(novelname));
                 temp+=count;
                 times.set(novel.indexOf(novelname),temp);
             }
         }
         //sum 计算单词在每个小说中出现次数之和
         double sum = 0;
```

```
for(int i: times){
    sum+=(double)i;
}

//计算频率,保留两位小数
double avg = sum / times.size();
String avgf = String.format("%.2f",avg);
//value,格式为 "频率+, +filename1+: +count1+; +filename2+: +count2+; +...."
StringBuilder all = new StringBuilder();
all.append(avgf+',');
for(int i=0; i < times.size();i++){
    all.append(novel.get(i)+':'+ Integer.toString(times.get(i))+';');
}

//输出
context.write(key, new Text(all.toString().substring(0,all.length()-1)));
}
```

六. 输出结果截图

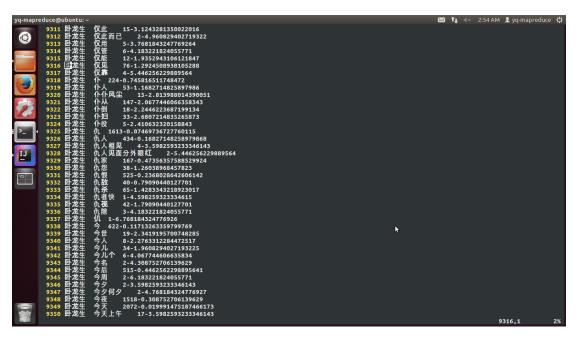
(1) 倒排索引

```
[2017st15@master01 ~]$ hadoop fs -ls /user/2017st15/output
17/04/30 17:22:45 WARN util.NativeCodeLoader: Unable to load native-hadoop libra
ry for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r-- 3 2017st15 hadoop_user 0 2017-04-30 16:58 /user/2017st15/o
utput/_SUCCESS
-rw-r--r-- 3 2017st15 hadoop_user 105642849 2017-04-30 16:58 /user/2017st15/o
utput/part-r-00000
```

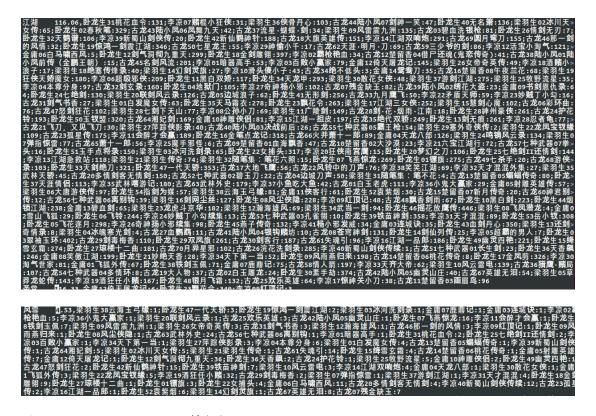
(2) 选作1——频数排序

(3) 选作 2——TF-IDF

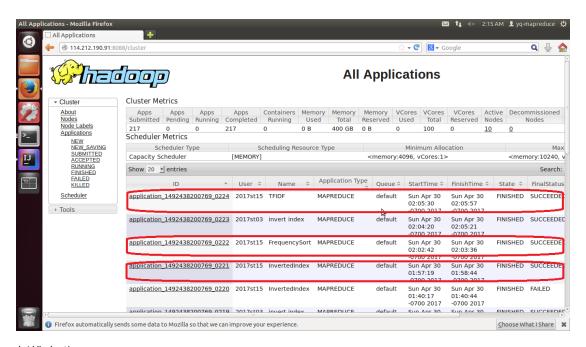
```
[2017st15@master01 -]$ hadoop fs -ls /user/2017st15/tf_out
17/04/30 17:26:21 MARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
ble
Found 2 items
-rw-r--r-- 3 2017st15 hadoop_user 0 2017-04-30 17:05 /user/2017st15/tf_out/_SUCCESS
-rw-r--r-- 3 2017st15_hadoop_user 16013861 2017-04-30 17:05 /user/2017st15/tf_out/part-r-00000
```



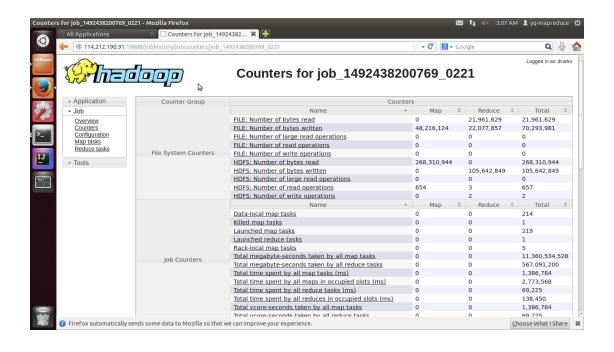
七. "江湖", "风雪"输出结果



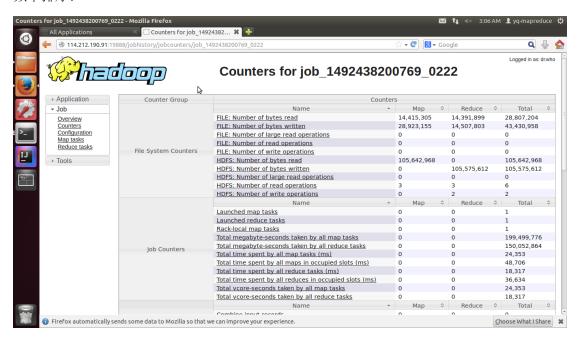
八. MapReduce Job 截图



倒排索引:



频率排序:



TF-IDF:

