# 日志分析

## Dailyarea

目标：统计每日访客地区[省份]分布指标job1：按省份统计当前日期IP、PV、UV；按provice+usid统计每个用户浏览网站的时长。并最终将结果数据以行的形式写入mysql，每行包括，省份，城市，ip数，每个用户的平均访问时间长度，PV.

思路：首先要按照城市，分类得到每城市的IP总量[按城市的PV,UID,UIP统计，中间结果，后面可以复用的]，PV；city+uid的访问时间【第一个MR】。

其实要计算平均访问时间。

分2个mr.

Ctime为开始访问时间

PV是每次页面点击都算依次(采用long变量保存)

UV是只记录uid的个数（采用hashSet<String>保存即可，最后调用size就是UV）

还可以计算独立IP，UIP,也和uid的计算方式类似

|  |  |
| --- | --- |
| Map1 input | 行级别的日志 [rowed, row] |
| Map1 out | 【1】<city,ip|1|uid|province>  【2】<city|uid, ctime> |
| Reduce1 input | 【1】<city,ip|1|uid|province>  【2】<city|uid, ctime> |
| Reduce1 output  [可存入mongo或者mysql，hdfs等] | 【1】  <null,provice:city:ip:pv:uv:uip：type=1> :这是中间结果可复用，pv,uv,uip  【2】  <null,provice:city:staylen:invalidPV:usefulUV:type=2>：这是每个用户的访问时间。这里为了要得到每个用户的访问时间长度，只能进行MR才能获取到访问路径并计算出来。这就意味着一次MR无法搞定，因为我们要获取的是同一个城市的平均访问时间，所以要得到所有用户的访问时间的和并除去无效的，除以总的人数。另外我们要注意目标是按省份统计，所以还要聚合一次 |
| Map2 input[r1的out]  需要判断type=1,2 |  |
| Map2 out  [主要是把city再拿出来] | 【1】<province,type=1|ip:pv:uid:province>  【2】<province,type=2|staylen:invalidPV:usefulUV> |
| Reduce2 in | 【2】【1】完成了合并 |
| Reduce2 out | <province:pv:uid:uip:date:aveStayLen,null>  得到的是按省份，是目标数据 |

如果我们按省份直接统计,pv，uid,uip容易，但要获得平均的访问时间，必须是得到用户访问路径，然后计算该用户的总时间，然后再算平均。下面我们说说只单独计算PV,UID,UIP。

|  |  |
| --- | --- |
| Map1 input | 行级别的日志 [rowed, row] |
| Map1 out | <province,ip|1|uid> |
| Reduce1 in | <province,ip|1|uid> |
| Reduce1 out【mongo,hdfs,mysql】 | <cip:pv:cuip:cuid,null> |

假设按照省份单独来计算页面的平均访问时间呢？还是如此，我们首先要得到每个用户的访问时间，这必须有个MR。然后再根据省份对用户的访问时间聚合，第二个MR。

这里我们能看到，既然计算PV,UV需要1个MR，计算平均访问时间要2个MR，如果我们分开计算，那么要开3个MR，浪费时间，很明显第二个任务的第一个MR和第一个任务的MR是可以同时计算的，因此将其合并。

## 词频率统计

从map到最终的reduce经历的过程有：

|  |  |
| --- | --- |
| Map | 1 数量 [文件数量，文件大小 块大小] |
| Combine | 在送入patition前执行一次规约，较少网络传输 |
| Partition | 1 有默认值  2 如果设置了reduce个数，那么要设计hash方法，hash到reduce个数桶 |
| Reduce | 最终完成规约 |

如果要统计错误：

**public** **enum** Counter {

***MAPSKIP***, // mapper中出错的行

***COMSKIP***, // combiner中出错的行

***REDSKIP***, // reducer中出错的行

}

在catch到异常时，context.getCounter(Counter.REDSKIP).increment(1);这样在最终的mr结果里可以看到异常的总数。

## 页面PV统计[从首页过来的]

|  |  |
| --- | --- |
| Map1 in | <rowline, row> |
| Map1 out | <uid:page|ctime> |
| Reduce1 in | <uid:page|ctime> |
| Reduce1 out | 1 每次输入的是单个用户的访问记录  2 用treeMap排序  3 用一个字典保存【只保存homepage出现之后的】  {  “page”:{“pv”:10,”click”:11,”exits”:0}  }  4最终输出就是<null,page|click|pv|exits> |
| Map2 in | <null, page|click|pv|exits> exits是表示该页面是否退出 |
| Map2 out | <page,click|pv|exits> 很多行是同一page |
| Reduce2 in | <page,click|pv|exits> 同1page在同一行 |
| Reduce2 out | <page,cntPv|cntClick|cntExits|dexits> dexits为页面退出率,cntExits是退出总量 |