目录

[1.数据准备 3](#_Toc485661725)

[1.1将原始数据加载到HDFS平台 3](#_Toc485661726)

[1.2处理数据 3](#_Toc485661727)

[2.数据分析 5](#_Toc485661728)

[2.1.MR实现 5](#_Toc485661729)

[2.2．hive实现 15](#_Toc485661730)

[3.将数据分析的结果保存到hdfs中 21](#_Toc485661731)

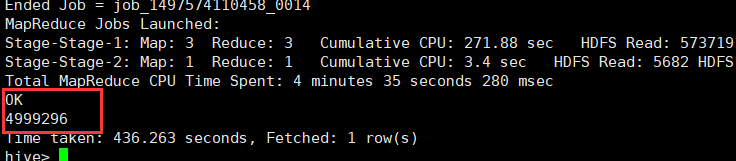
[4.将保存到hdfs中的结果文件通过Java API方式导入到HBase（一张表） 23](#_Toc485661732)

[5.查看导入的信息 26](#_Toc485661733)

## 1.数据准备

### 1.1将原始数据加载到HDFS平台

hadoop fs -put sogou.500w.utf8 /user/task



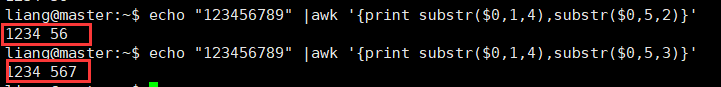
### 1.2处理数据

将原始数据中的时间字段拆分并拼接，添加年、月、日、小时字段。使用awk文本处理工具对数据进行处理。

（1）用到的主要知识点：

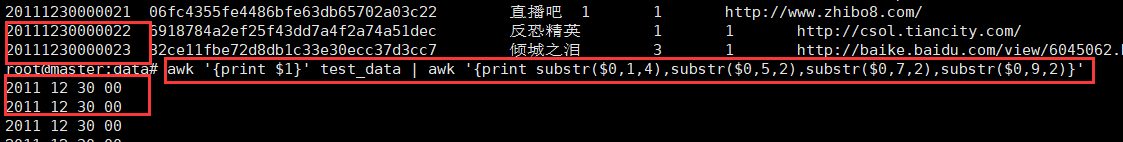
·awk截取字符：substr(str,num1,num2)

num1指的是要截取字符的开始位置，num2指从开始位置要截取num2个字

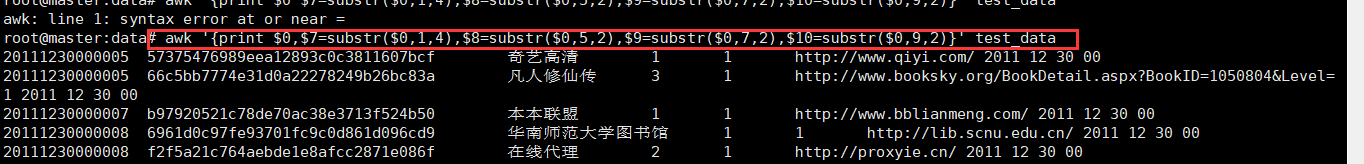


（2）先测试提取出年月日时

awk '{print $1}' test\_data | awk '{print substr($0,1,4),substr($0,5,2),substr($0,7,2),substr($0,9,2)}'

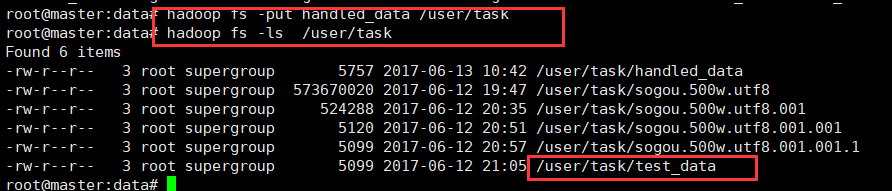
(3) 处理整个测试文档，输出满足要求的处理结果

awk '{print $0,$7=substr($0,1,4),$8=substr($0,5,2),$9=substr($0,7,2),$10=substr($0,9,2)}' test\_data

(4)测试完成，处理源数据并将其利用输出重定向将处理好后的文本保存到新的文件

data# awk '{print $0,$7=substr($0,1,4),$8=substr($0,5,2),$9=substr($0,7,2),$10=substr($0,9,2)}' sogou.500w.utf8 > handle\_data

（5）将处理好的数据加载到HDFS平台

hadoop fs -put handle\_data /user/tas

## 2.数据分析

### 2.1.MR实现

(1) 查询总条数

主要代码：

**public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {

@Override

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, Text>.Context context)

**throws** IOException, InterruptedException {

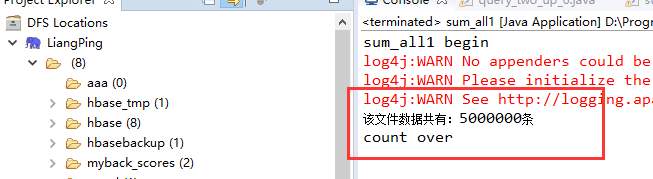
*sum*++;

}

}

因为是查询总数，所以只利用map即可，每行数据就map一次，定义一个全局变量sum，每次加，就完成了统计总数据条数的要求。

执行结果：



(2)非空查询条数ok

主要代码：

**public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {

@Override

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, Text>.Context context)

**throws** IOException, InterruptedException {

String[] str = value.toString().split("\t");

**if** (str[2]!=**null**)

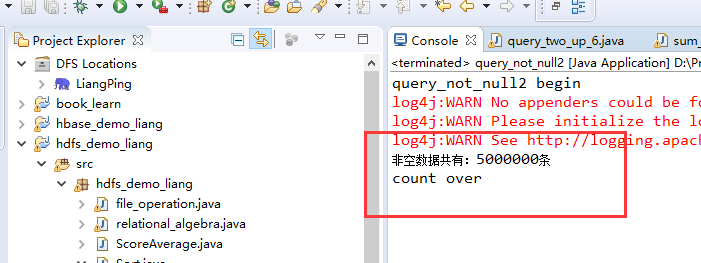
*sum*++;

}

}

非空查询，我们首先要清楚什么时空数据，在原来的数据中，当查询内容不存在时就界定为此数据为空数据，所以我在map中取出了每行数据的第三列，当该列数据不为空时全局变量计数器sum就加1，这样便可获得所有的非空数据，在这里我们同样没有用到reduce函数。

执行结果：



(3)无重复总条数ok

主要代码：

**public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {

@Override

**protected** **void** map(Object key, Text value, Context context)

**throws** IOException, InterruptedException {

context.write(**new** Text(value.toString()), **new** Text(""));

}

}

**public** **static** **class** MyReducer **extends** Reducer<Text, Text, Text, Text> {

@Override

**protected** **void** reduce(Text arg0, Iterable<Text> arg1, Context arg2)

**throws** IOException, InterruptedException {

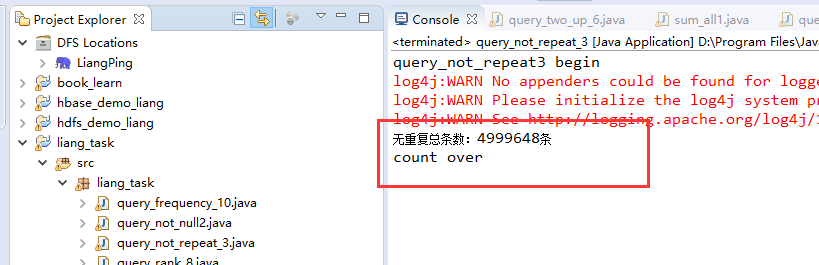
sum++;

}

}

Map函数将每条记录中需要关注的属性组合作为关键字，将空字符串作为值，生成的<key-value>对作为中间值输出。然后，reduce函数则将输入的中间结果的键值作为新的键值，value字符串仍然取空字符串，输出结果。因为所有键值相同的key都被送到了同一reducer，而reducer只输出了一个键值，所以，这样的一个过程就是去重过程。

执行结果：



(4)独立UID总数。

主要代码：

**public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {

@Override

**protected** **void** map(Object key, Text value, Context context)

**throws** IOException, InterruptedException {

String[] str = value.toString().split("\t");

context.write(**new** Text(str[1]), **new** Text(""));

}

}

**public** **static** **class** MyReducer **extends** Reducer<Text, Text, Text, Text> {

@Override

**protected** **void** reduce(Text arg0, Iterable<Text> arg1, Context arg2)

**throws** IOException, InterruptedException {

*sum*++;

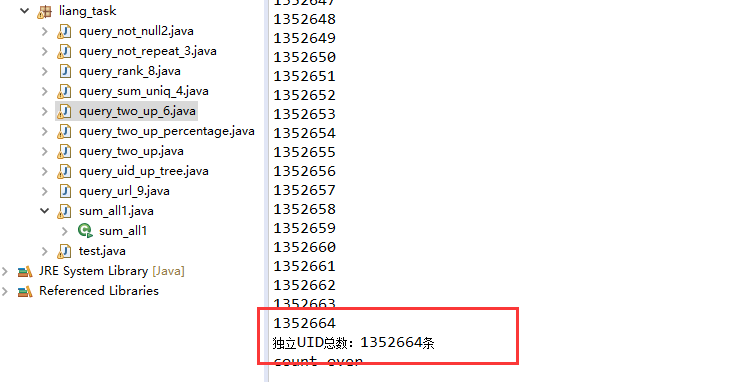
//arg2.write(arg0, new Text(""));

}

}

查询对立uid数基本思想和去重计数类似，只是返回的结果不是整条数据了而是从数据中提取出来的uid

执行结果：



(6)查询次数大于2次的用户总数 。

主要代码：

**public** **static** **class** MyMaper **extends** Mapper<Object, Text, Text, IntWritable> {

@Override

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)

**throws** IOException, InterruptedException {

String[] str = value.toString().split("\t");

Text word;

IntWritable one = **new** IntWritable(1);

word = **new** Text(str[1]);

context.write(word, one);

// 执行完毕后就是一个单词 对应一个value(1)

}

}

**public** **static** **class** MyReducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {

@Override

**protected** **void** reduce(Text arg0, Iterable<IntWritable> arg1,

Reducer<Text, IntWritable, Text, IntWritable>.Context arg2) **throws** IOException, InterruptedException {

// arg0是一个单词 arg1是对应的次数

**int** sum = 0;

**for** (IntWritable i : arg1) {

sum += i.get();

}

**if**(sum>2){

*total*=*total*+1;

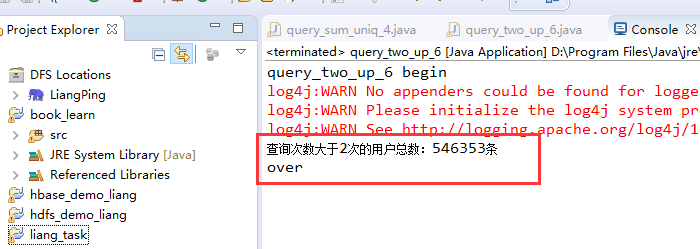
}

}

}

此程序类似于wordcount，将uid从map中返回，然后在reducer中统计次数，当统计的次数大于2时，将全局变量加1，得到最终结果。

执行结果：



(7)查询次数大于2次的用户占比。

执行代码（6）的基本相同

**public** **static** **class** MyMaper **extends** Mapper<Object, Text, Text, IntWritable> {

@Override

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)

**throws** IOException, InterruptedException {

*total2*++;

String[] str = value.toString().split("\t");

Text word;

IntWritable one = **new** IntWritable(1);

word = **new** Text(str[1]);

context.write(word, one);

// 执行完毕后就是一个单词 对应一个value(1)

}

}

**public** **static** **class** MyReducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {

@Override

**protected** **void** reduce(Text arg0, Iterable<IntWritable> arg1,

Reducer<Text, IntWritable, Text, IntWritable>.Context arg2) **throws** IOException, InterruptedException {

// arg0是一个单词 arg1是对应的次数

**int** sum = 0;

**for** (IntWritable i : arg1) {

sum += i.get();

}

**if**(sum>2){

*total1*++;

}

arg2.write(arg0, **new** IntWritable(sum));

}

}

结果输出：

System.***out***.println("total1="+*total1*+"\ttotal2="+*total2*);

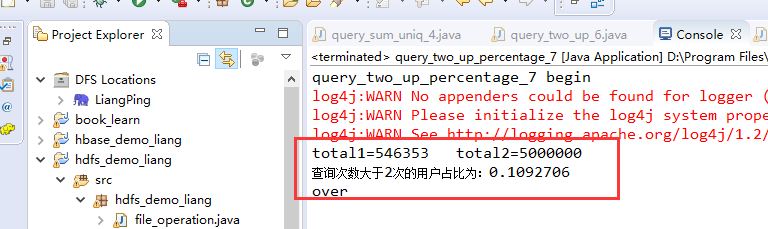
**float** percentage = (**float**)*total1*/(**float**)*total2*;

System.***out***.println("查询次数大于2次的用户占比为：" + percentage);

System.***out***.println("over");

在这里total1是uid总数，total2是数据总数。

执行结果：



(8)Rank在10以内的点击次数占比。

主要代码:

**public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {

@Override

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, Text>.Context context)

**throws** IOException, InterruptedException {

*sum2*++;

String[] str = value.toString().split("\t");

**int** rank = Integer.*parseInt*(str[3]);

**if**(rank<11)

{

*sum1*=*sum1*+1;

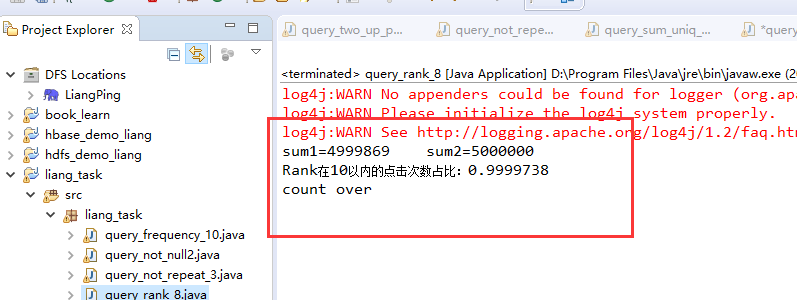
}

}

}

将每条数据中的rank列取出，然后再比较是否小于11，满足条件就加1。只利用map函数即可完成。sum1是指满足条件的数量，sum2指总数量。然后输出结果。

执行结果：



(9)直接输入URL查询的比例

主要代码：

**public** **static** **int** *sum1* = 0;

**public** **static** **int** *sum2* = 0;

**public** **static** **class** MyMapper **extends** Mapper<Object, Text, Text, Text> {

@Override

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, Text>.Context context)

**throws** IOException, InterruptedException {

String[] str = value.toString().split("\t");

Pattern p = Pattern.*compile*("(\\w?)+\\.(com|cn|net|org|biz|info|cc|tv|top)");

Matcher matcher = p.matcher(str[2]);

matcher.find();

**try** {

**if**(matcher.group()!=**null**)

*sum1*++;

*sum2*++;

} **catch** (Exception e) {

*sum2*++;

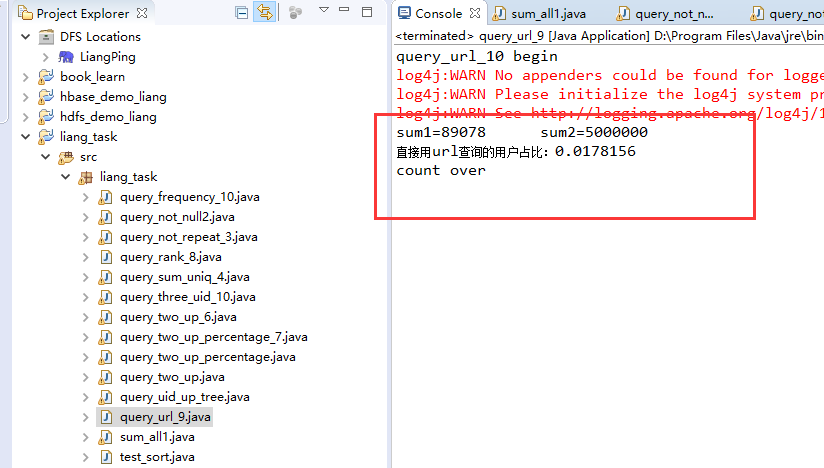
}

}

}

将查询结果取出来，然后利用正则表达式判断是否符合网站的条件，满足则加1.

执行结果：



(10)查询搜索过”仙剑奇侠传“的uid，并且次数大于3

主要代码：

**public** **static** String *queryContent*="仙剑奇侠传";

**public** **static** **class** MyMaper **extends** Mapper<Object, Text, Text, IntWritable> {

@Override

// key是行地址

// value是一行字符串

**protected** **void** map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)

**throws** IOException, InterruptedException {

String[] str = value.toString().split("\t");

Text word;

IntWritable one = **new** IntWritable(1);

**if**(str[2].equals(*queryContent*)){

word = **new** Text(str[1]);

context.write(word, one);

}

// 执行完毕后就是一个单词 对应一个value(1)

}

}

// 头两个参数表示的是输入数据key和value的数据类型

// 后两个数据表示的就是输出数据key和value的数据类型

**public** **static** **class** MyReducer **extends** Reducer<Text, IntWritable, Text, IntWritable> {

@Override

**protected** **void** reduce(Text arg0, Iterable<IntWritable> arg1,

Reducer<Text, IntWritable, Text, IntWritable>.Context arg2) **throws** IOException, InterruptedException {

// arg0是一个单词 arg1是对应的次数

**int** sum = 0;

**for** (IntWritable i : arg1) {

sum += i.get();

}

**if**(sum>3){

System.***out***.println(arg0 + ":" + sum);

}

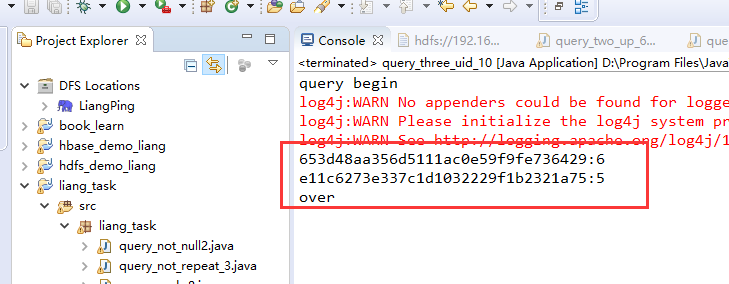
arg2.write(arg0, **new** IntWritable(sum));

}

}

该思想和wordcount类似，满足条件后输出结果。

执行结果：



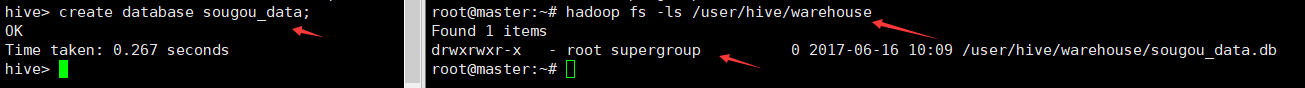
### 2.2．hive实现

#### 2.2.1．数据准备

首先将hdfs上的数据导入到hive表

在hive中新建一个表（根据数据特点建表）

创建数据库：



创建表：

hive> CREATE EXTERNAL TABLE sougou\_table(

> time STRING,

> uid STRING,

> keyword STRING,

> rank INT,

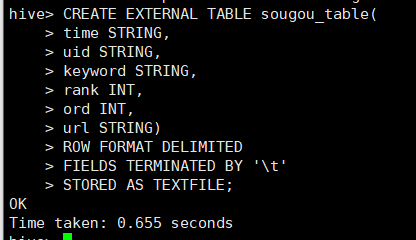
> ord INT,

> url STRING)

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY '\t'

> STORED AS TEXTFILE;



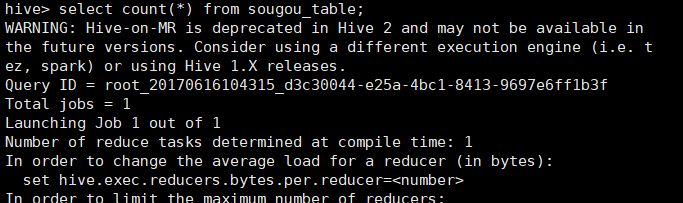
将hdfs中存储的数据导入到上步创建的表中：

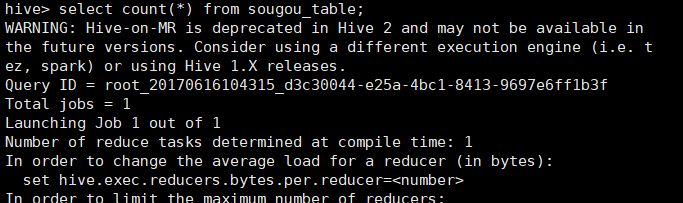
hive> load data inpath '/user/task/sogou.500w.utf8' into table sougou\_table;

#### 2.2.2.分析

( 1)查询总条数

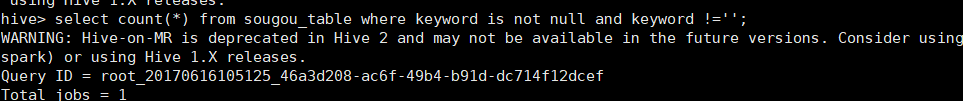
select count(\*) from sougou\_table;

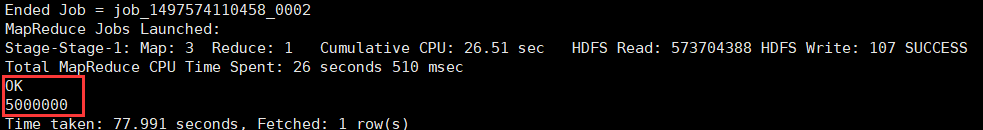




(2) 非空查询条数

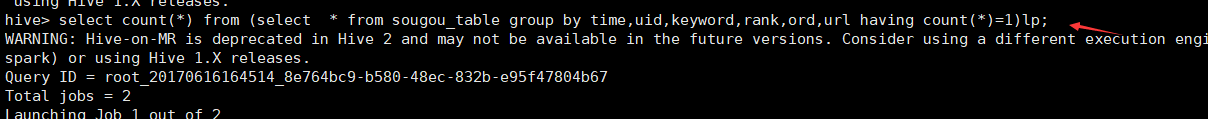
select count(\*) from sougou\_table where keyword is not null and keyword !='';

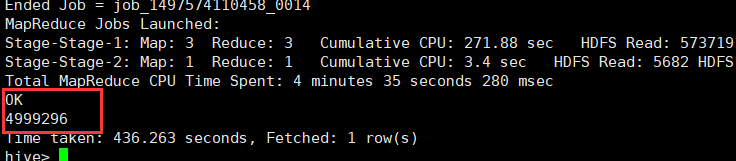




(3) 无重复总条数

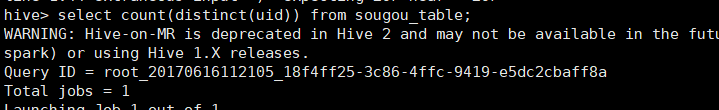
select count(\*) from (select \* from sougou\_table group by time,uid,keyword,rank,ord,url having count(\*)=1)lp;

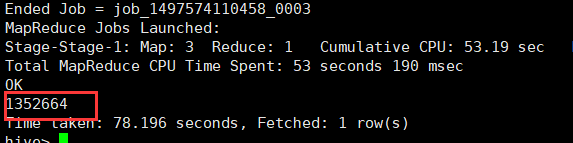




(4) 独立UID总数

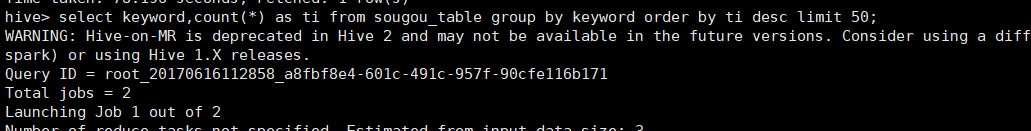
select count(distinct(uid)) from sougou\_table;

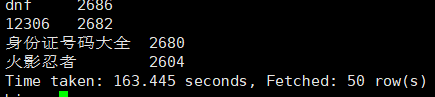




(5) 查询频度排名（频度最高的前50词）

select keyword,count(\*) as ti from sougou\_table group by keyword order by ti desc limit 50;





统计结果：

百度 38441

baidu 18312

人体艺术 14475

4399小游戏 11438

qq空间 10317

优酷 10158

新亮剑 9654

馆陶县县长闫宁的父亲 9127

公安卖萌 8192

百度一下 你就知道 7505

百度一下 7104

4399 7041

魏特琳 6665

qq网名 6149

7k7k小游戏 5985

黑狐 5610

儿子与母亲不正当关系 5496

新浪微博 5369

李宇春体 5310

新疆暴徒被击毙图片 4997

hao123 4834

123 4829

4399洛克王国 4112

qq头像 4085

nba 4027

龙门飞甲 3917

qq个性签名 3880

张去死 3848

cf官网 3729

凰图腾 3632

快播 3423

金陵十三钗 3349

吞噬星空 3330

dnf官网 3303

武动乾坤 3232

新亮剑全集 3210

电影 3155

优酷网 3115

两次才处决美女罪犯 3106

电影天堂 3028

土豆网 2969

qq分组 2940

全国各省最低工资标准 2872

清代姚明 2784

youku 2783

争产案 2755

dnf 2686

12306 2682

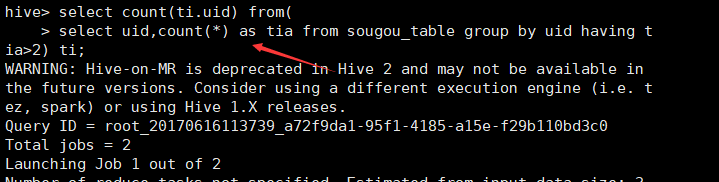
身份证号码大全 2680

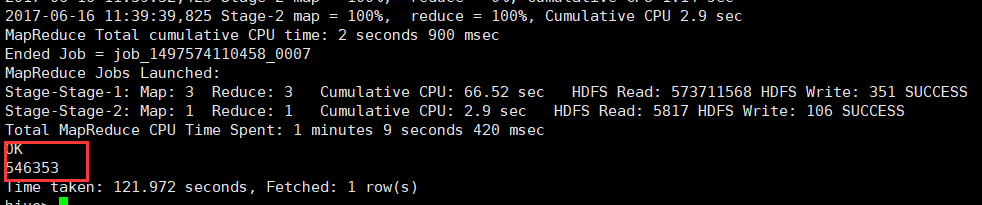
火影忍者 2604

(6) 查询次数大于2次的用户总数

> select count(ti.uid) from(

> select uid,count(\*) as tia from sougou\_table group by uid having tia>2) ti;





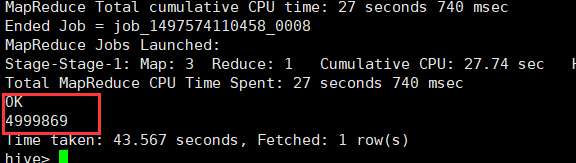
(7)查询次数大于2次的用户占比

将（6）中查询出的结果除以（1）的结果

546353/50000000=0.1092706

(8)Rank在10以内的点击次数占比

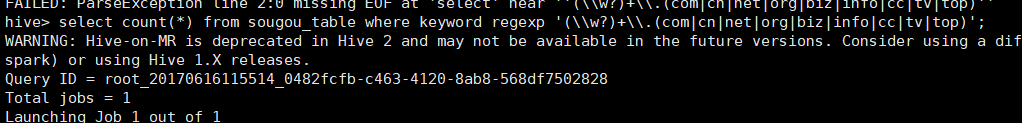
select count(\*) from sougou\_table where rank <11;

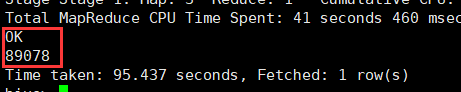


然后用49999869/50000000=0.9999738

(9)直接输入URL查询的比例

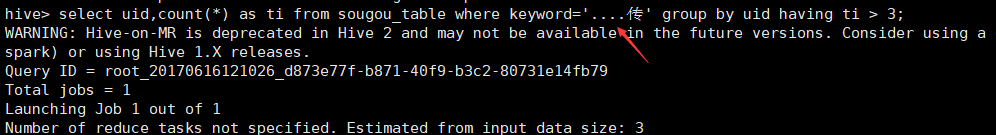
select count(\*) from sougou\_table where keyword regexp '(\\w?)+\\.(com|cn|net|org|biz|info|cc|tv|top)';



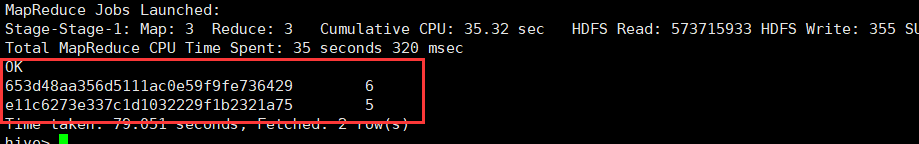


(10)查询搜索过”仙剑奇侠传“的uid，并且次数大于3

select uid,count(\*) as ti from sougou\_table where keyword='....传' group by uid having ti > 3;



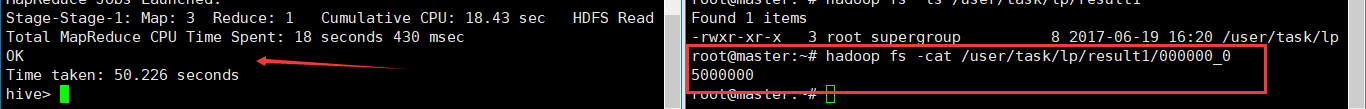
不明白字段为什么不现实全，编码问题吗？



## 3.将数据分析的结果保存到hdfs中

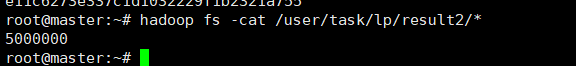
（1）查询总条数

insert overwrite directory "/user/task/lp/result1" select count(\*) from sougou\_table;



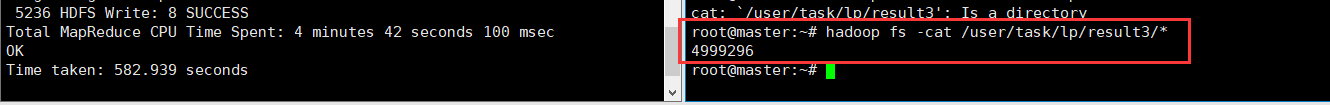
（2）非空查询条数

insert overwrite directory "/user/task/lp/result2" select count(\*) from sougou\_table where keyword is not null and keyword !='';



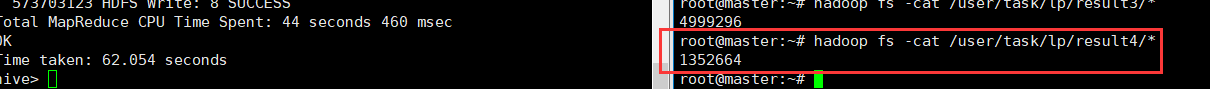
（3）无重复总条数

insert overwrite directory "/user/task/lp/result3" select count(\*) from (select \* from sougou\_table group by time,uid,keyword,rank,ord,url having count(\*)=1)lp;



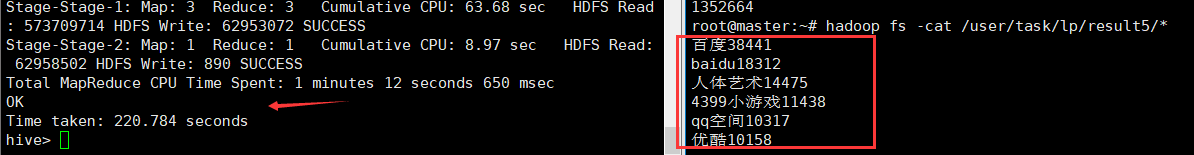
（4）独立UID总数

insert overwrite directory "/user/task/lp/result4" select count(distinct(uid)) from sougou\_table;



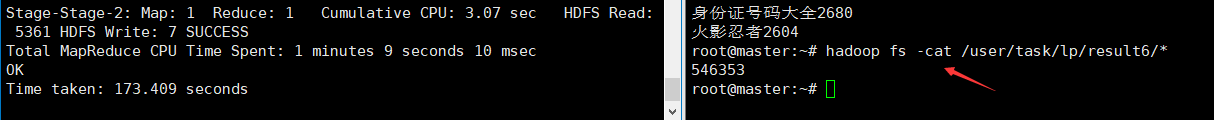
（5）查询频度排名（频度最高的前50词）

insert overwrite directory "/user/task/lp/result5" select keyword,count(\*) as ti from sougou\_table group by keyword order by ti desc limit 50;



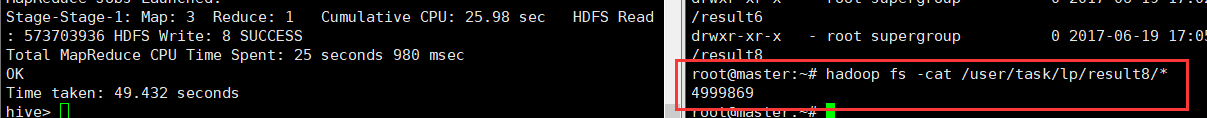
（6）查询次数大于2次的用户总数

insert overwrite directory "/user/task/lp/result6" select count(ti.uid) from(select uid,count(\*) as tia from sougou\_table group by uid having tia>2) ti;



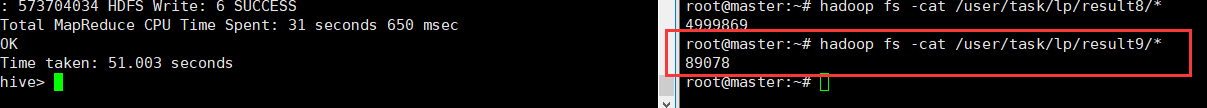
（8）Rank在10以内的点击次数占比

insert overwrite directory "/user/task/lp/result8" select count(\*) from sougou\_table where rank <11;



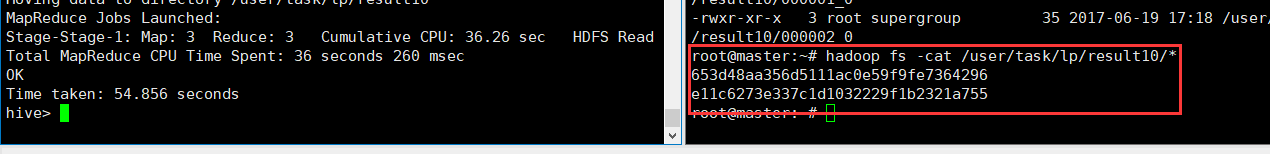
（9）直接输入URL查询的比例

insert overwrite directory "/user/task/lp/result9" select count(\*) from sougou\_table where keyword regexp '(\\w?)+\\.(com|cn|net|org|biz|info|cc|tv|top)';



（10）查询搜索过”仙剑奇侠传“的uid，并且次数大于3

insert overwrite directory "/user/task/lp/result10" select uid,count(\*) as ti from sougou\_table where keyword='仙剑奇侠传' group by uid having ti > 3;



## 4.将保存到hdfs中的结果文件通过Java API方式导入到HBase（一张表）

源码：

**package** hbase\_demo\_liang;

**import** java.io.IOException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** org.apache.hadoop.conf.Configuration;

**import** org.apache.hadoop.hbase.HBaseConfiguration;

**import** org.apache.hadoop.hbase.client.Mutation;

**import** org.apache.hadoop.hbase.client.Put;

**import** org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;

**import** org.apache.hadoop.hbase.mapreduce.TableOutputFormat;

**import** org.apache.hadoop.hbase.mapreduce.TableReducer;

**import** org.apache.hadoop.io.LongWritable;

**import** org.apache.hadoop.io.NullWritable;

**import** org.apache.hadoop.io.Text;

**import** org.apache.hadoop.mapreduce.Job;

**import** org.apache.hadoop.mapreduce.Mapper;

**import** org.apache.hadoop.mapreduce.Reducer;

**import** org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

**import** org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

**public** **class** data\_hbase {

**private** **static** String *tableName* = "lp";

**static** Configuration *conf* = **null**;

**static** {

*conf* = HBaseConfiguration.*create*();

*conf*.set("hbase.rootdir", "hdfs://192.168.23.9:9000/hbase");

*conf*.set("hbase.master", "hdfs://192.168.23.9:60000");

*conf*.set("hbase.zookeeper.property.clientPort", "2181");

*conf*.set("hbase.zookeeper.quorum", "master,slaver1,slaver2");

*conf*.set(TableOutputFormat.***OUTPUT\_TABLE***, *tableName*);

}

**public** **static** **class** MyMapper **extends**

Mapper<LongWritable, Text, LongWritable, Text> {

**protected** **void** map(LongWritable key, Text value,

Mapper<LongWritable, Text, LongWritable, Text>.Context context)

**throws** IOException, InterruptedException {

Text line = **new** Text();

line.set(value.toString());

context.write(key, line);

}

}

**public** **static** **class** MyReducer **extends**

TableReducer<LongWritable, Text, NullWritable> {

**private** String family = "data";

@Override

**protected** **void** reduce(

LongWritable arg0,

Iterable<Text> values,

Reducer<LongWritable, Text, NullWritable, Mutation>.Context context)

**throws** IOException, InterruptedException {

**for** (Text v : values) {

String[] splited = v.toString().split("\t");

String rowKey = splited[0];

Put put = **new** Put(rowKey.getBytes());

put.~~add~~(family.getBytes(), "data1".getBytes(), v.toString()

.getBytes());

context.write(NullWritable.*get*(), put);

}

}

}

**public** **static** **void** imputil(String str) **throws** IOException, ClassNotFoundException,

InterruptedException {

Job job = Job.*getInstance*(*conf*, data\_hbase.**class**.getSimpleName());

TableMapReduceUtil.*addDependencyJars*(job);

job.setJarByClass(data\_hbase.**class**);

FileInputFormat.*setInputPaths*(job,str);

job.setInputFormatClass(TextInputFormat.**class**);

job.setMapperClass(MyMapper.**class**);

job.setMapOutputKeyClass(LongWritable.**class**);

job.setMapOutputValueClass(Text.**class**);

job.setReducerClass(MyReducer.**class**);

job.setOutputFormatClass(TableOutputFormat.**class**);

job.waitForCompletion(**true**);

}

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException, IOException, InterruptedException {

String[] file={"hdfs://192.168.23.9:9000/user/task/lp/result1/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result2/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result3/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result4/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result5/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result6/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result8/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result9/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result10/000000\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result10/000001\_0",

"hdfs://192.168.23.9:9000/user/task/lp/result10/000002\_0"

};

**for** (String str:file){

*imputil*(str);

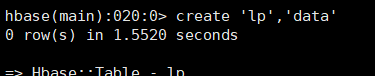
}

System.***out***.println("over");

}

}

首先建个表：



执行代码导入文件信息.

## 5.查看导入的信息

