XX 电商用户行为分析系统（二）

目录

1、模块开发--数据生成

1.1、使用 JS 进行埋点生成到服务器记录到日志中

详情可参见 Google 的 ga.js 或者新版本的 analatics.js 或者自定义的 JS 进行埋点生成和收集数据

1.2、利用 Nginx 自生成日志

详情参见 Nginx 安装和使用基本手册

2、模块开发--数据采集

2.1、需求

数据采集的需求广义上来说分为两大部分。
第一：是在页面采集用户的访问行为，具体开发工作：
1、开发页面埋点 JS，采集用户访问行为
2、后台接受页面 JS 请求记录日志
此部分工作也可以归属为“数据源”，其开发工作通常由 web 开发团队负责

第二：是从 web 服务器上汇聚日志到 HDFS，是数据分析系统的数据采集，此部分工作由数据分析平台建设团队负责，具体的技术实现有很多方式：

1、Shell 脚本

优点：轻量级，开发简单

缺点：对日志采集过程中的容错处理不便控制

2、Java 采集程序

优点：可对采集过程实现精细控制

缺点：开发工作量大

3、Flume 日志采集框架

成熟的开源日志采集系统，且本身就是 Hadoop 生态体系中的一员，与 Hadoop 体系中的各种框架组件具有天生的亲和力，可扩展性强

2.2、技术选型

在点击流日志分析这种场景中，对数据采集部分的可靠性、容错能力要求通常不会非常严苛，因此使用通用的 flume 日志采集框架完全可以满足需求。

本项目即使用 flume 来实现日志采集。

2.3、flume 实现

1、数据源信息

本项目分析的数据用 nginx 服务器所生成的流量日志，存放在各台 nginx 服务器上，如：/var/log/nginx/access.log

2、数据内容样例

数据的具体内容在采集阶段其实不用太关心。

```
58.215.204.118 - - [18/Sep/2013:06:51:35 +0000] "GET /wp-includes/js/jquery/jquery.js?ver=1.10.2 HTTP/1.1" 304 0 "http://blog.fens.me/nodejs-socketio-chat/" "Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0"
```

字段解析：

1、访客 ip 地址：58.215.204.118

2、访客用户信息：- -

3、请求时间：[18/Sep/2013:06:51:35 +0000]

4、请求方式：GET

5、请求的 url：/wp-includes/js/jquery/jquery.js?ver=1.10.2

6、请求所用协议：HTTP/1.1

7、响应码：304

8、返回的数据流量：0

9、访客的来源 url：http://blog.fens.me/nodejs-socketio-chat/

10、访客所用浏览器：Mozilla/5.0 (Windows NT 5.1; rv:23.0) Gecko/20100101 Firefox/23.0

3、日志文件生成规律

基本规律为：

当前正在写的文件为 access_log

文件体积达到 256M，或时间间隔达到 60 分钟，即滚动重命名切换成历史日志文件；
形如：
access_log.2018-11-10-13-00.log
access_log.2018-11-10-14-00.log
access_log.2018-11-10-15-00.log

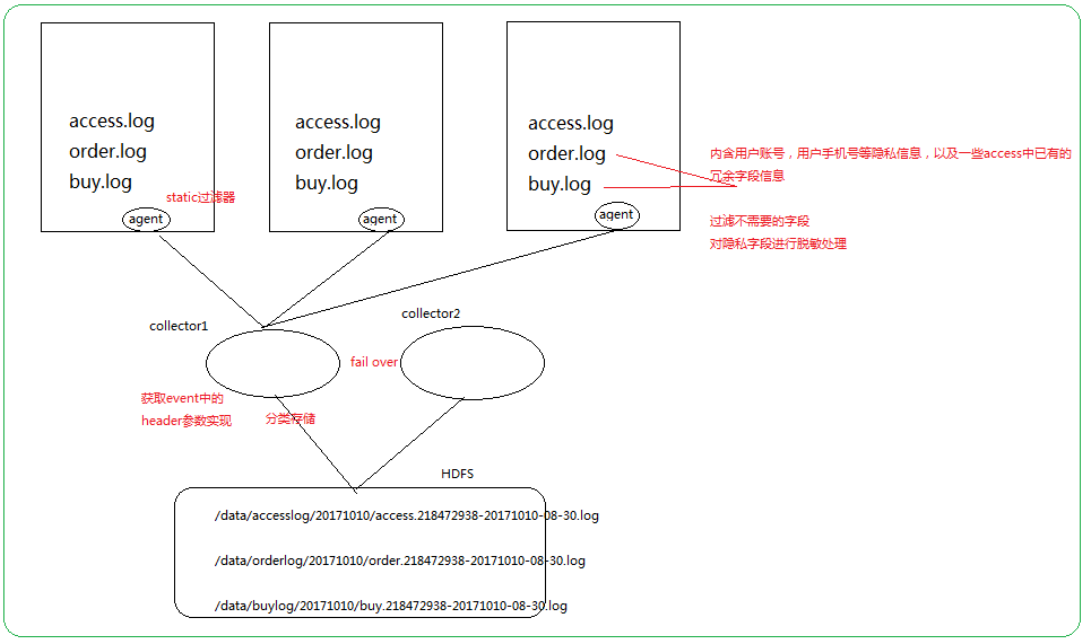
当然，每个公司的 web 服务器日志策略不同，可在 web 程序的 log4j.properties 中定义，如下：

```
log4j.appender.logDailyFile = org.apache.log4j.DailyRollingFileAppender
log4j.appender.logDailyFile.layout = org.apache.log4j.PatternLayout
log4j.appender.logDailyFile.layout.ConversionPattern = [%-5p][%-22d{yyyy/MM/dd HH:mm:ssS}][%l]%n%m%n
log4j.appender.logDailyFile.Threshold = DEBUG
log4j.appender.logDailyFile.ImmediateFlush = TRUE
log4j.appender.logDailyFile.Append = TRUE
log4j.appender.logDailyFile.File = /var/logs/access_log
log4j.appender.logDailyFile.DatePattern = '.'yyyy-MM-dd-HH-mm'.log'
log4j.appender.logDailyFile.Encoding = UTF-8
```

4、Flume 采集实现

Flume 采集系统的搭建相对简单：
1、在个 web 服务器上部署 agent 节点，修改配置文件
2、启动 agent 节点，将采集到的数据汇聚到指定的 HDFS 目录中

基本思路可以参考 Flume 的高可用架构进行部署，比如：



版本选择：apache-flume-1.8.0
采集规则设计：
1、采集源：Nginx 服务器日志目录
2、存放地：HDFS 目录/weblogs/

采集方案配置详情：nginx.properties

```
agent1.sources = source1
agent1.sinks = sink1
agent1.channels = channel1

# Describe/configure spooldir source1
#agent1.sources.source1.type = spooldir
#agent1.sources.source1.spoolDir = /var/log/nginx/
#agent1.sources.source1.fileHeader = false

# Describe/configure tail -F source1
#使用 exec 作为数据源 source 组件
agent1.sources.source1.type = exec
#使用 tail -F 命令实时收集新产生的日志数据
agent1.sources.source1.command = tail -F /var/log/nginx/access_log
agent1.sources.source1.channels = channel1

#configure host for source
#配置一个拦截器插件
agent1.sources.source1.interceptors = i1
agent1.sources.source1.interceptors.i1.type = host
#使用拦截器插件获取 agent 所在服务器的主机名
agent1.sources.source1.interceptors.i1.hostHeader = hostname

#配置 sink 组件为 hdfs
agent1.sinks.sink1.type = hdfs
#a1.sinks.k1.channel = c1
#agent1.sinks.sink1.hdfs.path=hdfs://myha01/weblog/nginx-access/%y-%m-%d/%H%M%S
#指定文件 sink 到 hdfs 上的路径
agent1.sinks.sink1.hdfs.path=
hdfs://myha01/weblog/nginx-access/%y-%m-%d/%H-%M_%hostname
#指定文件名前缀
agent1.sinks.sink1.hdfs.filePrefix = access_log
agent1.sinks.sink1.hdfs.maxOpenFiles = 5000
#指定每批下沉数据的记录条数
agent1.sinks.sink1.hdfs.batchSize= 100
agent1.sinks.sink1.hdfs.fileType = DataStream
agent1.sinks.sink1.hdfs.writeFormat =Text
#指定下沉文件按 1G 大小滚动
agent1.sinks.sink1.hdfs.rollSize = 1024*1024*1024
#指定下沉文件按 1000000 条数滚动
agent1.sinks.sink1.hdfs.rollCount = 1000000
#指定下沉文件按 30 分钟滚动
```

```
agent1.sinks.sink1.hdfs.rollInterval = 30
#agent1.sinks.sink1.hdfs.round = true
#agent1.sinks.sink1.hdfs.roundValue = 10
#agent1.sinks.sink1.hdfs.roundUnit = minute
agent1.sinks.sink1.hdfs.useLocalTimeStamp = true

# Use a channel which buffers events in memory
#使用 memory 类型 channel
agent1.channels.channel1.type = memory
agent1.channels.channel1.keep-alive = 120
agent1.channels.channel1.capacity = 500000
agent1.channels.channel1.transactionCapacity = 600

# Bind the source and sink to the channel
agent1.sources.source1.channels = channel1
agent1.sinks.sink1.channel = channel1
```

启动采集
在部署了 flume 的 nginx 服务器上，启动 flume 的 agent，命令如下：
flume-ng agent --conf $FLUME_HOME/conf -f $FLUME_HOME/conf/nginx.properties -n agent
注意：启动命令中的-n 参数要给配置文件中配置的 agent 名称

场景思考：
公司业务系统的 web 服务器可能有多台，每台上产生的日志有多种类型，比如点击日志，搜索日志，购买日志等，则在采集时该如何设计、配置采集系统？

3、模块开发--数据预处理

3.1、主要目的

过滤“不合规”数据
格式转换和规整
根据后续的统计需求，过滤分离出各种不同主题的基础数据

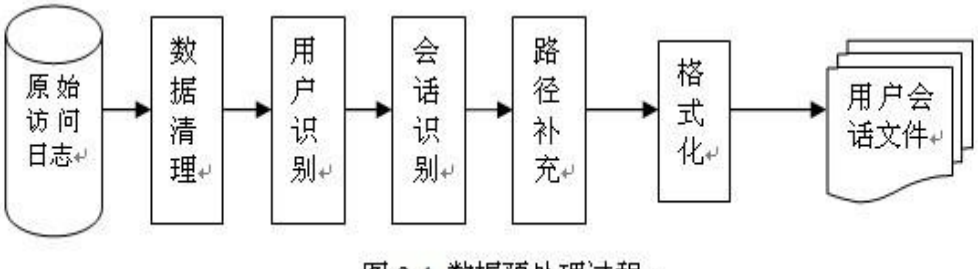


图 2.4 数据预处理过程

场景演化：
考虑多类日志数据（点击、搜索、购买）时，预处理的需求就会多种多样

3.2、实现方式

一般使用 MapReduce 做数据清洗

部分代码如下：

```
package com.mazh.aura.pre;

import java.io.IOException;
import java.util.HashSet;
import java.util.Set;

import com.mazh.aura.mrbean.WebLogBean;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/**
 * 处理原始日志，过滤出真实pv 请求 转换时间格式 对缺失字段填充默认值
 * 对记录标记valid 和invalid
 */
public class WeblogPreProcess {

    static class WeblogPreProcessMapper extends Mapper<LongWritable, Text, Text,
NullWritable> {
        // 用来存储网站url 分类数据
        Set<String> pages = new HashSet<String>();
        Text k = new Text();
        NullWritable v = NullWritable.get();

        /**
         * 从外部配置文件中加载网站的有用url 分类数据 存储到maptask 的内存中，用来对日志
数据进行过滤
         */
        @Override
```

```
        protected void setup(Context context) throws IOException,
InterruptedException {
            pages.add("/about");
            pages.add("/black-ip-list/");
            pages.add("/cassandra-clustor/");
            pages.add("/finance-rhive-repurchase/");
            pages.add("/hadoop-family-roadmap/");
            pages.add("/hadoop-hive-intro/");
            pages.add("/hadoop-zookeeper-intro/");
            pages.add("/hadoop-mahout-roadmap/");
        }

        @Override
        protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

            String line = value.toString();
            WebLogBean webLogBean = WebLogParser.parser(line);
            if (webLogBean != null) {
                // 过滤js/图片/css 等静态资源
                WebLogParser.filtStaticResource(webLogBean, pages);
                /* if (!webLogBean.isValid()) return; */
                k.set(webLogBean.toString());
                context.write(k, v);
            }
        }
    }

    public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);

        job.setJarByClass(WeblogPreProcess.class);

        job.setMapperClass(WeblogPreProcessMapper.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);

//      FileInputFormat.setInputPaths(job, new Path(args[0]));
//      FileOutputFormat.setOutputPath(job, new Path(args[1]));
        FileInputFormat.setInputPaths(job, new Path("c:/weblog/input/"));
        FileOutputFormat.setOutputPath(job, new Path("c:/weblog/preout/"));
```

```
        job.setNumReduceTasks(0);

        boolean res = job.waitForCompletion(true);
        System.exit(res ? 0 : 1);
    }
}
```

运行 MapReduce 程序对数据进行处理：

hadoop jar weblog.jar com.mazh.aura.pre.WeblogPreProcess /weblog/input /weblog/preout

3.3、点击流模型数据梳理

由于大量的指标统计从点击流模型中更容易得出，所以在预处理阶段，可以使用 mr 程序来生成点击流模型的数据

3.3.1 点击流模型 pageviews 表

Pageviews 表模型数据生成

MapReduce 代码：

```
package com.mazh.aura.clickstream;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Date;
import java.util.Locale;
import java.util.UUID;

import com.mazh.aura.mrbean.WebLogBean;
import org.apache.commons.beanutils.BeanUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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.output.FileOutputFormat;
```

```
/**
 * 将清洗之后的日志梳理出点击流 pageviews 模型数据
 *
 * 输入数据是清洗过后的结果数据
 *
 * 区分出每一次会话，给每一次 visit（session）增加了 session-id（随机 uuid）
 * 梳理出每一次会话中所访问的每个页面（请求时间，url，停留时长，以及该页面在这次 session 中的序号）
 * 保留 referral_url，body_bytes_send，useragent
 */
public class ClickStreamPageView {

   static class ClickStreamMapper extends Mapper<LongWritable, Text, Text, WebLogBean> {

      Text k = new Text();
      WebLogBean v = new WebLogBean();

      @Override
      protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

         String line = value.toString();

         String[] fields = line.split("\001");
         if (fields.length < 9) return;
         //将切出来的各字段 set 到 weblogbean 中
         v.set("true".equals(fields[0]) ? true : false, fields[1], fields[2], fields[3], fields[4], fields[5], fields[6], fields[7], fields[8]);
         //只有有效记录才进入后续处理
         if (v.isValid()) {
            k.set(v.getRemote_addr());
            context.write(k, v);
         }
      }
   }

   static class ClickStreamReducer extends Reducer<Text, WebLogBean, NullWritable, Text> {

      Text v = new Text();

      @Override
```

```
    protected void reduce(Text key, Iterable<WebLogBean> values, Context context)
throws IOException, InterruptedException {
        ArrayList<WebLogBean> beans = new ArrayList<WebLogBean>();

        // 先将一个用户的所有访问记录中的时间拿出来排序
        try {
           for (WebLogBean bean : values) {
              WebLogBean webLogBean = new WebLogBean();
              try {
                 BeanUtils.copyProperties(webLogBean, bean);
              } catch(Exception e) {
                 e.printStackTrace();
              }
              beans.add(webLogBean);
           }
           //将 bean 按时间先后顺序排序
           Collections.sort(beans, new Comparator<WebLogBean>() {

              @Override
              public int compare(WebLogBean o1, WebLogBean o2) {
                 try {
                    Date d1 = toDate(o1.getTime_local());
                    Date d2 = toDate(o2.getTime_local());
                    if (d1 == null || d2 == null)
                       return 0;
                    return d1.compareTo(d2);
                 } catch (Exception e) {
                    e.printStackTrace();
                    return 0;
                 }
              }
           });

           /**
            * 以下逻辑为：从有序 bean 中分辨出各次 visit，并对一次 visit 中所访问的 page 按
顺序标号 step
            * 核心思想：
            *  就是比较相邻两条记录中的时间差，如果时间差<30 分钟，则该两条记录属于同一个
session
            * 否则，就属于不同的 session
            */
           int step = 1;
           String session = UUID.randomUUID().toString();
           for (int i = 0; i < beans.size(); i++) {
```

```
                WebLogBean bean = beans.get(i);
                // 如果仅有1条数据，则直接输出
                if (1 == beans.size()) {

                    // 设置默认停留市场为60s
                    v.set(session+"\001"+key.toString()+"\001"+bean.getRemote_user()
+ "\001" + bean.getTime_local() + "\001" + bean.getRequest() + "\001" + step +
"\001" + (60) + "\001" + bean.getHttp_referer() + "\001" + bean.getHttp_user_agent()
+ "\001" + bean.getBody_bytes_sent() + "\001"
                            + bean.getStatus());
                    context.write(NullWritable.get(), v);
                    session = UUID.randomUUID().toString();
                    break;
                }

                // 如果不止1条数据，则将第一条跳过不输出，遍历第二条时再输出
                if (i == 0) {
                    continue;
                }

                // 求近两次时间差
                long      timeDiff      =      timeDiff(toDate(bean.getTime_local()),
toDate(beans.get(i - 1).getTime_local()));
                // 如果本次-上次时间差<30分钟，则输出前一次的页面访问信息

                if (timeDiff < 30 * 60 * 1000) {

                    v.set(session+"\001"+key.toString()+"\001"+beans.get(i          -
1).getRemote_user()  +  "\001"  +  beans.get(i  -  1).getTime_local()  +  "\001"  +
beans.get(i - 1).getRequest() + "\001" + step + "\001" + (timeDiff / 1000) + "\001"
+ beans.get(i - 1).getHttp_referer() + "\001"
                            +   beans.get(i   -   1).getHttp_user_agent()   +   "\001"   +
beans.get(i - 1).getBody_bytes_sent() + "\001" + beans.get(i - 1).getStatus());
                    context.write(NullWritable.get(), v);
                    step++;
                } else {

                    // 如果本次-上次时间差>30分钟，则输出前一次的页面访问信息且将step重置，
以分隔为新的visit
                    v.set(session+"\001"+key.toString()+"\001"+beans.get(i          -
1).getRemote_user()  +  "\001"  +  beans.get(i  -  1).getTime_local()  +  "\001"  +
beans.get(i  -  1).getRequest()  +  "\001"  +  (step)  +  "\001"  +  (60)  +  "\001"  +
beans.get(i - 1).getHttp_referer() + "\001"
                            +   beans.get(i   -   1).getHttp_user_agent()   +   "\001"   +
```

```
beans.get(i - 1).getBody_bytes_sent() + "\001" + beans.get(i - 1).getStatus());
                context.write(NullWritable.get(), v);
                // 输出完上一条之后，重置 step 编号
                step = 1;
                session = UUID.randomUUID().toString();
            }

            // 如果此次遍历的是最后一条，则将本条直接输出
            if (i == beans.size() - 1) {
                // 设置默认停留市场为 60s
                v.set(session+"\001"+key.toString()+"\001"+bean.getRemote_user()
+ "\001" + bean.getTime_local() + "\001" + bean.getRequest() + "\001" + step +
"\001" + (60) + "\001" + bean.getHttp_referer() + "\001" + bean.getHttp_user_agent()
+ "\001" + bean.getBody_bytes_sent() + "\001" + bean.getStatus());
                context.write(NullWritable.get(), v);
            }
          }

        } catch (ParseException e) {
          e.printStackTrace();
        }
    }

    private String toStr(Date date) {
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.US);
        return df.format(date);
    }

    private Date toDate(String timeStr) throws ParseException {
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.US);
        return df.parse(timeStr);
    }

    private long timeDiff(String time1, String time2) throws ParseException {
        Date d1 = toDate(time1);
        Date d2 = toDate(time2);
        return d1.getTime() - d2.getTime();
    }

    private long timeDiff(Date time1, Date time2) throws ParseException {
        return time1.getTime() - time2.getTime();
    }
```

```
    }

    public static void main(String[] args) throws Exception {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);

        job.setJarByClass(ClickStreamPageView.class);

        job.setMapperClass(ClickStreamMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(WebLogBean.class);

        job.setReducerClass(ClickStreamReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

//      FileInputFormat.setInputPaths(job, new Path(args[0]));
//      FileOutputFormat.setOutputPath(job, new Path(args[1]));
        FileInputFormat.setInputPaths(job, new Path("c:/weblog/preout"));
        FileOutputFormat.setOutputPath(job, new Path("c:/weblog/pageviews"));

        boolean isDone = job.waitForCompletion(true);
        System.exit(isDone ? 0 : 1);

    }
}
```

提交任务运行：

**hadoop jar weblog.jar **
**com.mazh.aura.clickstream.ClickStreamPageView **
/weblog/preout /weblog/pageviews/

3.3.2 点击流模型 visit 信息表

注：“一次访问”=“N 次连续请求”
直接从原始数据中用 HQL 语法得出每个人的“次”访问信息比较困难，可先用 MapReduce 程序分析原始数据得出“次”信息数据，然后再用 HQL 进行更多维度统计

用 MapReduce 程序从 PageViews 数据中，梳理出每一次 visit 的起止时间、页面信息

```
package com.mazh.aura.clickstream;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

import com.mazh.aura.mrbean.PageViewsBean;
import com.mazh.aura.mrbean.VisitBean;
import org.apache.commons.beanutils.BeanUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
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.output.FileOutputFormat;

/**
 * 输入数据：pageviews 模型结果数据
 * 从 pageviews 模型结果数据中进一步梳理出 visit 模型
 * sessionid  start-time   out-time   start-page   out-page   pagecounts  ......
 */
public class ClickStreamVisit {

    // 以 session 作为 key，发送数据到 reducer
    static class ClickStreamVisitMapper extends Mapper<LongWritable, Text, Text,
PageViewsBean> {

        PageViewsBean pvBean = new PageViewsBean();
        Text k = new Text();

        @Override
        protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

            String line = value.toString();
            String[] fields = line.split("\001");
            int step = Integer.parseInt(fields[5]);
            //(String session, String remote_addr, String timestr, String request,
int step, String staylong, String referal, String useragent, String bytes_send,
String status)
            //299d6b78-9571-4fa9-bcc2-f2567c46df3472.46.128.140-2013-09-18
```

```
07:58:50/hadoop-zookeeper-intro/160"https://www.aura.cn/""Mozilla/5.0"14722200
            pvBean.set(fields[0], fields[1], fields[2], fields[3], fields[4],
step, fields[6], fields[7], fields[8], fields[9]);
            k.set(pvBean.getSession());
            context.write(k, pvBean);
        }
    }

    static class ClickStreamVisitReducer extends Reducer<Text, PageViewsBean,
NullWritable, VisitBean> {

        @Override
        protected void reduce(Text session, Iterable<PageViewsBean> pvBeans,
Context context) throws IOException, InterruptedException {

            // 将 pvBeans 按照 step 排序
            ArrayList<PageViewsBean> pvBeansList = new ArrayList<PageViewsBean>();
            for (PageViewsBean pvBean : pvBeans) {
                PageViewsBean bean = new PageViewsBean();
                try {
                    BeanUtils.copyProperties(bean, pvBean);
                    pvBeansList.add(bean);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            Collections.sort(pvBeansList, new Comparator<PageViewsBean>() {

                @Override
                public int compare(PageViewsBean o1, PageViewsBean o2) {
                    return o1.getStep() > o2.getStep() ? 1 : -1;
                }
            });

            // 取这次 visit 的首尾 pageview 记录，将数据放入 VisitBean 中
            VisitBean visitBean = new VisitBean();
            // 取 visit 的首记录
            visitBean.setInPage(pvBeansList.get(0).getRequest());
            visitBean.setInTime(pvBeansList.get(0).getTimestr());
            // 取 visit 的尾记录
            visitBean.setOutPage(pvBeansList.get(pvBeansList.size() -
1).getRequest());
            visitBean.setOutTime(pvBeansList.get(pvBeansList.size() -
```

```
1).getTimestr());
            // visit 访问的页面数
            visitBean.setPageVisits(pvBeansList.size());
            // 来访者的 ip
            visitBean.setRemote_addr(pvBeansList.get(0).getRemote_addr());
            // 本次 visit 的 referal
            visitBean.setReferal(pvBeansList.get(0).getReferal());
            visitBean.setSession(session.toString());

            context.write(NullWritable.get(), visitBean);
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf);

        job.setJarByClass(ClickStreamVisit.class);

        job.setMapperClass(ClickStreamVisitMapper.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(PageViewsBean.class);

        job.setReducerClass(ClickStreamVisitReducer.class);
        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(VisitBean.class);

//        FileInputFormat.setInputPaths(job, new Path(args[0]));
//        FileOutputFormat.setOutputPath(job, new Path(args[1]));
      FileInputFormat.setInputPaths(job, new Path("c:/weblog/pageviews"));
      FileOutputFormat.setOutputPath(job, new Path("c:/weblog/visitout"));

        boolean res = job.waitForCompletion(true);
        System.exit(res ? 0 : 1);
    }
}
```

提交任务执行：

```
hadoop jar weblog.jar \
com.mazh.aura.clickstream.ClickStreamVisit \
/weblog/pageviews /weblog/visitout
```

然后，在 hive 仓库中建点击流 visit 模型表：

```
create database if not exists weblog;
```

```
use weblog;
drop table if exist click_stream_visit;
create table weblog.click_stream_visit(
session         string,
remote_addr string,
inTime          string,
outTime         string,
inPage          string,
outPage         string,
referal         string,
pageVisits      int)
partitioned by (datestr string);
```

然后，将 MR 运算得到的 visit 数据导入 visit 模型表

```
load data inpath '/weblog/visitout' into table click_stream_visit partition(datestr='2013-09-18');
```

查询验证：

```
select * from weblog.click_stream_visit limit 10;
```

3.4、预处理任务调度

将以上各预处理 MapReduce 程序写成 shell 脚本，配置成 azkaban 的 job，并上传到 azkaban 服务器进行定时调度

4、模块开发--数据仓库设计

注意：
1、数据仓库设计中的 **星型模型** 和 **雪花型模**
2、**事实表** 和 **维度表**

4.1、事实表

基础数据表：time_local 和 request 都没有拆分，也就是原始表：ods_weblog_origin

原始数据表:t _weblog_origin		
valid	string	是否有效
remote_addr	string	访客 ip
remote_user	string	访客用户信息
time_local	string	请求时间
request	string	请求 url

status	string	响应码
body_bytes_sent	string	响应字节数
http_referer	string	来源 url
http_user_agent	string	访客终端信息

Request 进行了拆分：

ETL 中间表：t_etl_referurl		
valid	string	是否有效
remote_addr	string	访客 ip
remote_user	string	访客用户信息
time_local	string	请求时间
request	string	请求 url
request_host	**string**	**请求的域名**
status	string	响应码
body_bytes_sent	string	响应字节数
http_referer	string	来源 url
http_user_agent	string	访客终端信息
host	**string**	**外链 url 的域名**
path	**string**	**外链 url 的路径**
query	**string**	**外链 url 的参数**
query_id	**string**	**外链 url 的参数值**

访问日志明细宽表：t_ods_access_detail		
valid	string	是否有效
remote_addr	string	访客 ip
remote_user	string	访客用户信息
time_local	string	请求时间
request	string	请求 url 整串
request_level1	**string**	**请求的一级栏目**
request_level2	**string**	**请求的二级栏目**
request_level3	**string**	**请求的三级栏目**
status	string	响应码
body_bytes_sent	string	响应字节数
http_referer	string	来源 url
http_user_agent	string	访客终端信息
http_user_agent_browser	string	访客终端浏览器
http_user_agent_sys	string	访客终端操作系统
http_user_agent_dev	string	访客终端设备
host	**string**	**外链 url 的域名**
path	**string**	**外链 url 的路径**
query	**string**	**外链 url 的参数**
query_id	**string**	**外链 url 的参数值**
daystr	string	日期整串

tmstr	string	时间整串
year	**string**	**年**
month	**string**	**月份**
day	**string**	**日**
hour	**string**	**时**
minute	**string**	**分**
mm	string	分区字段--月
dd	string	分区字段--日

4.2、维度表

时间维度 v_year_month_date
year
month
day
hour
minute

访客地域维度 t_dim_area
北京
上海
广州
深圳
河北
河南

终端类型维度 t_dim_termination
uc
firefox
chrome
safari
ios
android

网站栏目维度 t_dim_section
跳蚤市场
房租信息
休闲娱乐
建材装修
本地服务
人才市场

5、模块开发--ETL

该项目的数据分析过程在 Hadoop 集群上实现，主要应用 hive 数据仓库工具，因此，采集并经过预处理后的数据，需要加载到 hive 数据仓库中，以进行后续的挖掘分析。

5.1、创建原始数据表

5.1.1、创建存储预处理之后的原始数据表

在 Hive 仓库中建原始数据表：**ods_weblog_origin** 存储清洗后的日志

表结构：

```
0: jdbc:hive2://hadoop05:10000> desc ods_weblog_origin;
+--------------------------+----------------------+----------------------+
|         col_name         |       data_type      |        comment       |
+--------------------------+----------------------+----------------------+
| valid                    | string               |                      |
| remote_addr              | string               |                      |
| remote_user              | string               |                      |
| time_local               | string               |                      |
| request                  | string               |                      |
| status                   | string               |                      |
| body_bytes_sent          | string               |                      |
| http_referer             | string               |                      |
| http_user_agent          | string               |                      |
| datestr                  | string               |                      |
|                          | NULL                 | NULL                 |
| # Partition Information  | NULL                 | NULL                 |
| # col_name               | data_type            | comment              |
|                          | NULL                 | NULL                 |
| datestr                  | string               |                      |
+--------------------------+----------------------+----------------------+
15 rows selected (0.08 seconds)
```

建表到导入数据的语句：

```
create database if not exists weblog;
use weblog;
drop table if exists weblog.ods_weblog_origin;
create table weblog.ods_weblog_origin(
valid string,
remote_addr string,
remote_user string,
time_local string,
request string,
status string,
body_bytes_sent string,
http_referer string,
http_user_agent string)
partitioned by (datestr string)
row format delimited
fields terminated by '\001';
```

5.1.2、创建点击流数据表

表结构：

```
0: jdbc:hive2://hadoop05:10000> desc click_stream_pageviews;
+--------------------------+----------------------+------------------------+
|         col_name         |       data_type      |        comment         |
+--------------------------+----------------------+------------------------+
| session                  | string               |                        |
| remote_addr              | string               |                        |
| remote_user              | string               |                        |
| time_local               | string               |                        |
| request                  | string               |                        |
| visit_step               | string               |                        |
| page_staylong            | string               |                        |
| http_referer             | string               |                        |
| http_user_agent          | string               |                        |
| body_bytes_sent          | string               |                        |
| status                   | string               |                        |
| datestr                  | string               |                        |
|                          | NULL                 | NULL                   |
| # Partition Information  | NULL                 | NULL                   |
| # col_name               | data_type            | comment                |
|                          | NULL                 | NULL                   |
| datestr                  | string               |                        |
+--------------------------+----------------------+------------------------+
17 rows selected (0.087 seconds)
```

点击流模型 pageviews 表：**click_stream_pageviews** 存储点击流模型中的 pageviews 数据

```
create database if not exists weblog;
use weblog;
drop table if exists weblog.click_stream_pageviews;
create table weblog.click_stream_pageviews (
session string,
remote_addr string,
remote_user string,
time_local string,
request string,
visit_step string,
page_staylong string,
http_referer string,
http_user_agent string,
body_bytes_sent string,
status string)
partitioned by (datestr string)
row format delimited
fields terminated by '\001';
```

5.1.3、创建点击流访客表

表结构：

```
0: jdbc:hive2://hadoop05:10000> desc click_stream_visit;
+--------------------------+----------------------+----------------------+
|         col_name         |       data_type      |        comment       |
+--------------------------+----------------------+----------------------+
| session                  | string               |                      |
| remote_addr              | string               |                      |
| intime                   | string               |                      |
| outtime                  | string               |                      |
| inpage                   | string               |                      |
| outpage                  | string               |                      |
| referal                  | string               |                      |
| pagevisits               | int                  |                      |
| datestr                  | string               |                      |
|                          | NULL                 | NULL                 |
| # Partition Information  | NULL                 | NULL                 |
| # col_name               | data_type            | comment              |
|                          | NULL                 | NULL                 |
| datestr                  | string               |                      |
+--------------------------+----------------------+----------------------+
14 rows selected (0.072 seconds)
```

点击流 visit 模型表：click_stream_visit 存储点击流模型中的 session 汇总数据

```
create database if not exists weblog;
use weblog;
drop table if exists weblog.click_stream_visit;
create table weblog.click_stream_visit(
session         string,
remote_addr string,
inTime          string,
outTime         string,
inPage          string,
outPage         string,
referal         string,
pageVisits    int)
partitioned by (datestr string);
```

5.2、导入数据

注：通常情况下，应将以下数据 load 命令，写在脚本中，然后配置在 azkaban 中定时运行运行的时间点： 应该在预处理完成之后

1、导入清洗结果数据到数据表 ods_weblog_origin

load data inpath '/weblog/preout' overwrite into table ods_weblog_origin partition(datestr='2013-09-18');

2、导入点击流模型 pageviews 数据到 click_stream_pageviews 表

load data inpath '/weblog/pageviews' overwrite into table click_stream_pageviews partition(datestr='2013-09-18');

3、导入点击流模型 visit 数据到 click_stream_visit 表

load data inpath '/weblog/visitout' overwrite into table click_stream_visit partition(datestr='2013-09-18');

4、查询验证：

select * from ods_weblog_origin limit 3; **// 原始数据表**

select * from click_stream_pageviews limit 3; **// pageviews 表**

select * from click_stream_visit limit 3; **// visit 表**

5.3、生成 ODS 层明细宽表

5.3.1、需求概述

整个数据分析的过程是按照数据仓库的层次分层进行的，总体来说，是从 ODS 原始数据中整理出一些中间表（比如，为后续分析方便，将原始数据中的时间、url 等非结构化数据作结构化抽取，将各种字段信息进行细化，形成明细表），然后再在中间表的基础之上统计出各种指标数据

5.3.2、ETL 实现

建表：明细表 **ods_weblog_detail** (源：ods_weblog_origin) (目标：ods_weblog_detail)

```
create database if not exists weblog;
use weblog;
drop table if exists weblog.ods_weblog_detail;
create table weblog.ods_weblog_detail(
valid                string comment "有效标识",
remote_addr          string comment "来源 IP",
remote_user          string comment "用户标识",
time_local           string comment "访问完整时间",
daystr               string comment "访问日期",
timestr              string comment "访问时间",
year                 string comment "访问年",
month                string comment "访问月",
day                  string comment "访问日",
hour                 string comment "访问时",
request              string comment "请求的 url",
status               string comment "响应码",
body_bytes_sent      string comment "传输字节数",
http_referer         string comment "来源 url",
```

```
ref_host              string comment "来源的 host",
ref_path              string comment "来源的路径",
ref_query             string comment "来源参数 query",
ref_query_id          string comment "来源参数 query 的值",
http_user_agent       string comment "客户终端标识"
)
partitioned by(datestr string);

// 标红的字段表示目标表 ods_weblog_detail 从源表 ods_weblog_origin 新增加的字段
```

解析 URL 中的信息：

1、抽取 refer_url 到中间：**t_ods_tmp_referurl**

2、将来访 url 分离出 host　path　query　query id

```
create database if not exists weblog;
use weblog;
drop table if exists weblog.t_ods_tmp_referurl;
create table weblog.t_ods_tmp_referurl as
SELECT a.*, b.*
FROM ods_weblog_origin a
LATERAL VIEW parse_url_tuple(regexp_replace(http_referer, "\"", ""), 'HOST', 'PATH', 'QUERY',
'QUERY:id') b
as host, path, query, query_id;
select * from weblog.t_ods_tmp_referurl a where a.host is not null limit 3;
```

解析时间字符串字段：

1、抽取转换 time_local 字段到中间表明细表：**t_ods_tmp_detail**

```
create database if not exists weblog;
use weblog;
drop table if exists weblog.t_ods_tmp_detail;
create table weblog.t_ods_tmp_detail as
select b.*,substring(time_local,0,10) as daystr,
substring(time_local,11) as tmstr,
substring(c.time_local,0,4) as year,
substring(time_local,5,2) as month,
substring(time_local,8,2) as day,
substring(time_local,11,2) as hour
From t_ods_tmp_referurl b;
select * from weblog.t_ods_tmp_detail where month is not null limit 3;
```

上面执行的代码也可以这么写：

```
insert into table weblog.ods_weblog_detail partition(datestr='2013-09-18')
select c.valid,c.remote_addr,c.remote_user,c.time_local,
substring(c.time_local,0,10) as daystr,
substring(c.time_local,12) as tmstr,
```

```
substring(c.time_local,0,4) as year,
substring(c.time_local,6,2) as month,
substring(c.time_local,9,2) as day,
substring(c.time_local,11,3) as hour,
c.request,c.status,c.body_bytes_sent,c.http_referer,c.ref_host,c.ref_path,
c.ref_query,c.ref_query_id,c.http_user_agent
from
(SELECT
a.valid,a.remote_addr,a.remote_user,a.time_local,
a.request,a.status,a.body_bytes_sent,a.http_referer,a.http_user_agent,
b.ref_host,b.ref_path,b.ref_query,b.ref_query_id
FROM weblog.ods_weblog_origin a LATERAL VIEW
parse_url_tuple(regexp_replace(http_referer, "\"", ""), 'HOST', 'PATH','QUERY', 'QUERY:id') b
as ref_host, ref_path, ref_query, ref_query_id) c;
```

检查验证结果：

```
select * from weblog.ods_weblog_detail limit 10;
```

6、模块开发--统计分析

注：每一种统计指标都可以跟各维度表进行钻取
为了在前端展示时速度更快，可以把每一个指标都事先算出各维度结果存入 MySQL

在实际生产中，究竟需要哪些统计指标通常由数据需求相关部门人员提出，而且会不断有新的统计需求产生，以下为网站流量分析中的一些典型指标示例。

6.1、流量统计

6.1.1、多维度流量统计

1、按时间维度统计

按小时统计 PV

```
select year, month, day, hour, count(*) from weblog.ods_weblog_detail group by year, month, day, hour;
```

```
0: jdbc:hive2://hadoop05:10000> select year, month, day, hour, count(*) from weblog.ods_weblog_detail group by year, month, day, hour;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine
+-------+--------+------+-------+-------+
| year  | month  | day  | hour  |  _c4  |
+-------+--------+------+-------+-------+
| 2013  | 09     | 18   |  06   | 111   |
| 2013  | 09     | 18   |  07   | 1010  |
| 2013  | 09     | 18   |  08   | 2052  |
| 2013  | 09     | 18   |  09   | 1374  |
| 2013  | 09     | 18   |  10   | 568   |
| 2013  | 09     | 18   |  11   | 571   |
| 2013  | 09     | 18   |  12   | 621   |
| 2013  | 09     | 18   |  13   | 531   |
| 2013  | 09     | 18   |  14   | 514   |
| 2013  | 09     | 18   |  15   | 759   |
| 2013  | 09     | 18   |  16   | 475   |
| 2013  | 09     | 18   |  17   | 382   |
| 2013  | 09     | 18   |  18   | 262   |
| 2013  | 09     | 18   |  19   | 390   |
| 2013  | 09     | 18   |  20   | 211   |
| 2013  | 09     | 18   |  21   | 213   |
| 2013  | 09     | 18   |  22   | 351   |
| 2013  | 09     | 18   |  23   | 382   |
| 2013  | 09     | 19   |  00   | 312   |
| 2013  | 09     | 19   |  01   | 324   |
| 2013  | 09     | 19   |  02   | 546   |
| 2013  | 09     | 19   |  03   | 552   |
| 2013  | 09     | 19   |  04   | 569   |
| 2013  | 09     | 19   |  05   | 540   |
| 2013  | 09     | 19   |  06   | 150   |
+-------+--------+------+-------+-------+
25 rows selected (21.382 seconds)
```

2、计算该处理批次（一天）中的各小时 pvs

```
drop table if exists dw_pvs_hour;
create table dw_pvs_hour(year string, month string, day string, hour string, pvs bigint) partitioned by(datestr string);
insert into table weblog.dw_pvs_hour partition(datestr='2013-09-18')
select a.year as year, a.month as month, a.day as day, a.hour as hour, count(1) as pvs
from ods_weblog_detail a
where a.datestr='2013-09-18'
group by a.year, a.month, a.day, a.hour;
select * from dw_pvs_hour limit 10;
```

结果：

```
0: jdbc:hive2://hadoop05:10000> drop table if exists dw_pvs_hour;
No rows affected (0.099 seconds)
0: jdbc:hive2://hadoop05:10000> create table dw_pvs_hour(year string, month string, day string, hour string, pvs bigint) partitioned by(datestr string);
No rows affected (0.068 seconds)
0: jdbc:hive2://hadoop05:10000> insert into table weblog.dw_pvs_hour partition(datestr='2013-09-18')
. . . . . . . . . . . . . . . > select a.year as year, a.month as month, a.day as day, a.hour as hour, count(1) as pvs from ods_weblog_detail a where a.d
nth, a.day, a.hour;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, te
No rows affected (21.643 seconds)
0: jdbc:hive2://hadoop05:10000> select * from dw_pvs_hour limit 10;
+--------------------+---------------------+--------------------+---------------------+--------------------+-----------------------+
| dw_pvs_hour.year   | dw_pvs_hour.month   | dw_pvs_hour.day    | dw_pvs_hour.hour    | dw_pvs_hour.pvs    | dw_pvs_hour.datestr   |
+--------------------+---------------------+--------------------+---------------------+--------------------+-----------------------+
| 2013               | 09                  | 18                 |  06                 | 111                | 2013-09-18            |
| 2013               | 09                  | 18                 |  07                 | 1010               | 2013-09-18            |
| 2013               | 09                  | 18                 |  08                 | 2052               | 2013-09-18            |
| 2013               | 09                  | 18                 |  09                 | 1374               | 2013-09-18            |
| 2013               | 09                  | 18                 |  10                 | 568                | 2013-09-18            |
| 2013               | 09                  | 18                 |  11                 | 571                | 2013-09-18            |
| 2013               | 09                  | 18                 |  12                 | 621                | 2013-09-18            |
| 2013               | 09                  | 18                 |  13                 | 531                | 2013-09-18            |
| 2013               | 09                  | 18                 |  14                 | 514                | 2013-09-18            |
| 2013               | 09                  | 18                 |  15                 | 759                | 2013-09-18            |
+--------------------+---------------------+--------------------+---------------------+--------------------+-----------------------+
10 rows selected (0.089 seconds)
```

维度：按天统计

```
drop table if exists weblog.dw_pvs_day;
create table weblog.dw_pvs_day(year string, month string, day string, pvs bigint);
insert into table weblog.dw_pvs_day
select a.year as year, a.month as month, a.day as day, count(1) as pvs
from weblog.ods_weblog_detail a
```

group by a.year, a.month, a.day;
select * from weblog.dw_pvs_day;

```
0: jdbc:hive2://hadoop05:10000> select * from weblog.dw_pvs_day;
+------------------+-------------------+-----------------+-----------------+
| dw_pvs_day.year  | dw_pvs_day.month  | dw_pvs_day.day  | dw_pvs_day.pvs  |
+------------------+-------------------+-----------------+-----------------+
| 2013             | 09                | 18              | 10777           |
| 2013             | 09                | 19              | 2993            |
+------------------+-------------------+-----------------+-----------------+
2 rows selected (0.079 seconds)
```

--或者，从之前算好的小时结果中统计
Insert into table weblog.dw_pvs_day
select sum(pvs) as pvs, year, month, day
from weblog.dw_pvs_hour
group by year, month, day having day='18';

结果如下：
select * from weblog.dw_pvs_day;

```
0: jdbc:hive2://hadoop05:10000> select * from weblog.dw_pvs_day;
+-----------------+-------------------+-----------------+
| dw_pvs_day.pvs  | dw_pvs_day.month  | dw_pvs_day.day  |
+-----------------+-------------------+-----------------+
| 10777           | 09                | 18              |
+-----------------+-------------------+-----------------+
1 row selected (0.112 seconds)
```

维度：按月统计

drop table if exists weblog.dw_pvs_month;
create table if not exists dw_pvs_month(year string, month string, pvs bigint);
insert into table weblog. dw_pvs_month
select a.year, a.month, count(*) as pvs
from weblog.ods_weblog_detail a
group by a.year, a.month;
select * from dw_pvs_month limit 10;

结果：

```
0: jdbc:hive2://hadoop05:10000> select * from dw_pvs_month limit 10;
+---------------------+----------------------+--------------------+
| dw_pvs_month.year   | dw_pvs_month.month   | dw_pvs_month.pvs   |
+---------------------+----------------------+--------------------+
| 2013                | 09                   | 13770              |
+---------------------+----------------------+--------------------+
1 row selected (0.071 seconds)
```

2、按终端维度统计

<方式一：>
直接在原始数据中对终端字段进行 group by 分组聚合统计

<方式二：>
注：探索数据中的终端类型

```
drop table if exists t_user_agent;
create table t_user_agent as select distinct(http_user_agent) from ods_weblog_detail where http_user_agent like '%Mozilla%';
select * from t_user_agent limit 100;
```

```
0: jdbc:hive2://hadoop05:10000> select * from t_user_agent limit 10;
+----------------------------------------------------+
|           t_user_agent.http_user_agent             |
+----------------------------------------------------+
| "DiggFeedFetcher1.0(Mozilla/5.0(Macintosh;IntelMacOSX10_7_1)AppleWebKit/534.48.3(KHTML,likeGecko)Version/5.1Safari/534.48.3)" |
| "Mozilla/4.0"                                      |
| "Mozilla/4.0(compatible;)"                         |
| "Mozilla/4.0(compatible;MSIE5.00;Windows98)"       |
| "Mozilla/4.0(compatible;MSIE6.0;AmericaOnlineBrowser1.1;rev1.2;WindowsNT5.1;SV1;.NETCLR1.1.4322)" |
| "Mozilla/4.0(compatible;MSIE6.0;MSIE5.5;Windows95)Opera7.03[de]" |
| "Mozilla/4.0(compatible;MSIE6.0;Windows98)"        |
| "Mozilla/4.0(compatible;MSIE6.0;WindowsNT5.1)"     |
| "Mozilla/4.0(compatible;MSIE6.0;WindowsNT5.1;SV1)" |
| "Mozilla/4.0(compatible;MSIE6.0;WindowsNT5.1;SV1;)" |
+----------------------------------------------------+
10 rows selected (0.211 seconds)
```

3、按页面维度统计

4、按地域维度统计

5、按来源维度统计

6.1.2、按 referer 维度统计 pv 总量

需求：按照来源及时间维度统计 pv，并按照 pv 大小倒序排序

1、按照小时粒度统计，查询结果存入：weblog.dw_pvs_referer_h

```
drop table if exists dw_pvs_referer_h;
create table weblog.dw_pvs_referer_h(referer_url string, referer_host string, month string, day string, hour string, pv_referer_cnt bigint) partitioned by(datestr string);

insert into table dw_pvs_referer_h partition(datestr='2013-09-18')
select http_referer,ref_host,month,day,hour,count(1) as pv_referer_cnt
from weblog.ods_weblog_detail
group by http_referer,ref_host,month,day,hour
having ref_host is not null
order by hour asc,day asc,month asc,pv_referer_cnt desc;
```

2、按小时粒度统计各来访域名的产生的 pv 数并排序

```
drop table if not exists weblog.dw_ref_host_visit_cnts_h;
create table weblog.dw_ref_host_visit_cnts_h(ref_host string,month string,day string,hour string,ref_host_cnts bigint) partitioned by(datestr string);

insert into table weblog.dw_ref_host_visit_cnts_h partition(datestr='2013-09-18')
```

```
select ref_host,month,day,hour,count(1) as ref_host_cnts
from weblog.ods_weblog_detail
group by ref_host,month,day,hour
having ref_host is not null
order by hour asc,day asc,month asc,ref_host_cnts desc;
```

补充：
还可以按照地域，或者用户终端维度进行统计
试想怎么实现？

6.1.3、统计 PV 总量最大的来源 TopN

需求描述：按照时间维度，比如，统计一天内各小时产生最多 pv 的来源 TopN

需要用到 row_number 窗口函数
以下语句对每个小时内的来访 host 次数倒序排序标号：

```
select ref_host,ref_host_cnts,concat(month,hour,day),
row_number() over (partition by concat(month, hour, day) order by ref_host_cnts desc) as od
from weblog.dw_ref_host_visit_cnts_h
```

效果如下：

ref_host	ref_host_cnts	_c2	od
blog.fens.me	68	Sep1806	1
www.angularjs.cn	3	Sep1806	2
www.google.com	2	Sep1806	3
www.baidu.com	1	Sep1806	4
cos.name	1	Sep1806	5
blog.fens.me	706	Sep1807	1
www.angularjs.cn	20	Sep1807	2
www.google.com.hk	20	Sep1807	3
www.dataguru.cn	10	Sep1807	4
r.dataguru.cn	6	Sep1807	5
www.fens.me	6	Sep1807	6
cnodejs.org	5	Sep1807	7
cos.name	5	Sep1807	8
www.google.com	3	Sep1807	9
f.dataguru.cn	2	Sep1807	10
blog.chinaunix.net	2	Sep1807	11
it.dataguru.cn	2	Sep1807	12
image.baidu.com	1	Sep1807	13
www.google.fr	1	Sep1807	14
74.125.128.160	1	Sep1807	15
www.weibo.com	1	Sep1807	16
www.baidu.com	1	Sep1807	17
blog.fens.me	1550	Sep1808	1
www.fens.me	26	Sep1808	2

根据上述 row_number 的功能，可编写 HQL 取各小时的 ref_host 访问次数 TopN：

```
drop table if exists weblog.dw_pvs_refhost_topn_h;
create table if not exists weblog.dw_pvs_refhost_topn_h(
```

```
hour string,
toporder string,
ref_host string,
ref_host_cnts string
) partitioned by(datestr string);

insert into table weblog.dw_pvs_refhost_topn_h partition(datestr='2013-09-18')
select t.hour,t.od,t.ref_host,t.ref_host_cnts from
(select ref_host,ref_host_cnts,concat(month,day,hour) as hour,
row_number() over (partition by concat(month,day,hour) order by ref_host_cnts desc) as index
from weblog.dw_ref_host_visit_cnts_h) t where index<=3;
```

结果如下：

```
0: jdbc:hive2://localhost:10000> select * from dw_ref_host_topn_h;
+----------------------------+-------------------------------+--------------------------------+---------------------------------+
| dw_ref_host_topn_h.hour    | dw_ref_host_topn_h.toporder   | dw_ref_host_topn_h.ref_host    | dw_ref_host_topn_h.ref_host_cnts |
+----------------------------+-------------------------------+--------------------------------+---------------------------------+
| Sep1806                    | 1                             | blog.fens.me                   | 68                              |
| Sep1806                    | 2                             | www.angularjs.cn               | 3                               |
| Sep1806                    | 3                             | www.google.com                 | 2                               |
| Sep1807                    | 1                             | blog.fens.me                   | 706                             |
| Sep1807                    | 2                             | www.angularjs.cn               | 20                              |
| Sep1807                    | 3                             | www.google.com.hk              | 20                              |
| Sep1808                    | 1                             | blog.fens.me                   | 1550                            |
| Sep1808                    | 2                             | www.fens.me                    | 26                              |
| Sep1808                    | 3                             | www.baidu.com                  | 15                              |
| Sep1809                    | 1                             | blog.fens.me                   | 1037                            |
| Sep1809                    | 2                             | www.baidu.com                  | 27                              |
| Sep1809                    | 3                             | www.google.com.hk              | 11                              |
| Sep1810                    | 1                             | blog.fens.me                   | 306                             |
| Sep1810                    | 2                             | www.baidu.com                  | 15                              |
| Sep1810                    | 3                             | www.angularjs.cn               | 7                               |
| Sep1811                    | 1                             | blog.fens.me                   | 245                             |
| Sep1811                    | 2                             | fens.me                        | 12                              |
| Sep1811                    | 3                             | www.angularjs.cn               | 7                               |
| Sep1812                    | 1                             | blog.fens.me                   | 394                             |
| Sep1812                    | 2                             | cnodejs.org                    | 3                               |
| Sep1812                    | 3                             | f.dataguru.cn                  | 2                               |
| Sep1813                    | 1                             | blog.fens.me                   | 243                             |
| Sep1813                    | 2                             | www.baidu.com                  | 27                              |
| Sep1813                    | 3                             | cos.name                       | 6                               |
| Sep1814                    | 1                             | blog.fens.me                   | 259                             |
| Sep1814                    | 2                             | www.angularjs.cn               | 6                               |
```

6.1.4、人均浏览页面数/人均流量

需求描述：比如，今日所有来访者，平均请求的页面数
计算方式：总页面请求数/去重总人数

计算 HQL 语句如下：

```
drop table if exists weblog.dw_avgpv_user_d;
create table if not exists weblog.dw_avgpv_user_d(
day string,
avgpv string);

insert into table weblog.dw_avgpv_user_d
select '2013-09-18',sum(b.pvs)/count(b.remote_addr) from
(select remote_addr,count(1) as pvs from weblog.ods_weblog_detail
where datestr='2013-09-18' group by remote_addr) b;
```

6.2、受访分析

6.2.1、各页面统计

各页面 PV
各页面 UV
各页面 IP

6.2.2、热门页面统计

统计每日最热门的页面

6.3、访客分析

6.3.1、独立访客

需求描述：按照时间维度比如小时来统计独立访客及其产生的 pvCnts
对于独立访客的识别，如果在原始日志中有用户标识，则根据用户标识即很好实现;
此处，由于原始日志中并没有用户标识，以访客 IP 来模拟，技术上是一样的，只是精确度相对较低

时间维度：时

6.3.2、每日新访客

需求描述：将每天的新访客统计出来
实现思路：创建一个去重访客累积表，然后将每日访客对比累积表

6.4、访客 Visit 分析

6.4.1、回头客/单次访客统计

需求描述：查询今日所有回头访客及其访问次数
实现思路：上表中出现次数>1 的访客，即回头访客；反之，则为单次访客

6.4.2、人均访问频次

需求：统计出每天所有用户访问网站的平均次数（visit）
总 visit 数/去重总用户数

6.4.3、访客忠诚度分析

6.4.4、访客活跃度分析

6.5、关键路径转化率分析

转化：在一条指定的业务流程中，各个步骤的完成人数及相对上一个步骤的百分比

6.5.1、需求分析

step	numbs	rate		
1	10000	100%		
2	4000	40%	40%	
3	2000	20%	50%	
4	1000	10%	50%	
5	500	5%	50%	
6	200	2%	40%	

6.5.2、模型设计

定义好业务流程中的页面标识，以电商的顾客购买商品的场景为例：
那基本步骤为：

Step1、	/index	首页
Step2、	/category	类别页
Step3、	/item	商品详情页
Step4、	/order	下单页面

6.5.3、开发实现

分以下步骤进行开发：

1、查询每一个步骤的总访问人数

```
create table dw_order_trans_number as
select 'step1' as step,count(distinct remote_addr)    as numbs from ods_click_pageviews where request like '/index%'
union
select 'step2' as step,count(distinct remote_addr) as numbs from ods_click_pageviews where request like '/category%'
union
select 'step3' as step,count(distinct remote_addr) as numbs from ods_click_pageviews where request like '/item%'
union
select 'step4' as step,count(distinct remote_addr)    as numbs from ods_click_pageviews where request like '/order%';
```

效果：

route_num.step	route_num.numbs
step1	4716
step2	4592
step3	3929
step4	2841

2、查询每一个步骤相对于路径起点人数的比例

思路：利用 join

```
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs
from dw_order_trans_number rn
inner join dw_order_trans_number rr;
```

效果：

rnstep	rnnumbs	rrstep	rrnumbs
step1	4716	step1	4716
step2	4592	step1	4716
step3	3929	step1	4716
step4	2841	step1	4716
step1	4716	step2	4592
step2	4592	step2	4592
step3	3929	step2	4592
step4	2841	step2	4592
step1	4716	step3	3929
step2	4592	step3	3929
step3	3929	step3	3929
step4	2841	step3	3929
step1	4716	step4	2841
step2	4592	step4	2841
step3	3929	step4	2841
step4	2841	step4	2841

```
select tmp.rnstep,tmp.rnnumbs/tmp.rrnumbs as ratio
from(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs
from dw_order_trans_number rn
inner join
dw_order_trans_number rr) tmp
```

```
where tmp.rrstep='step1';
```

效果：

```
+-------------+--------------------+--+
| tmp.rnstep  |       ratio        |  |
+-------------+--------------------+--+
| step1       | 1.0                |  |
| step2       | 0.9737065309584394 |  |
| step3       | 0.8331212892281594 |  |
| step4       | 0.6024173027989822 |  |
+-------------+--------------------+--+
```

3、查询每一步骤相对于上一步骤的漏出率

```
select tmp.rrstep as rrstep,tmp.rrnumbs/tmp.rnnumbs as ration
from(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs
from dw_order_trans_number rn
inner join
dw_order_trans_number rr) tmp
where cast(substr(tmp.rnstep,5,1) as int)=cast(substr(tmp.rrstep,5,1) as int)-1;
```

效果：

```
+---------+--------------------+--+
| rrstep  |       ration       |  |
+---------+--------------------+--+
| step2   | 0.9737065309584394 |  |
| step3   | 0.8556184668989547 |  |
| step4   | 0.7230847543904302 |  |
+---------+--------------------+--+
```

4、汇总以上两种指标

```
select abs.step,abs.numbs,abs.ratio as abs_ratio,rel.ratio as rel_ratio
from
(
select tmp.rnstep as step,tmp.rnnumbs as numbs,tmp.rnnumbs/tmp.rrnumbs as ratio
from
(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs
from dw_order_trans_number rn
inner join
dw_order_trans_number rr) tmp
where tmp.rrstep='step1'
) abs
left outer join
(
select tmp.rrstep as step,tmp.rrnumbs/tmp.rnnumbs as ratio
from
(
select rn.step as rnstep,rn.numbs as rnnumbs,rr.step as rrstep,rr.numbs as rrnumbs
from dw_order_trans_number rn
```

```
inner join
dw_order_trans_number rr) tmp
where cast(substr(tmp.rnstep,5,1) as int)=cast(substr(tmp.rrstep,5,1) as int)-1
) rel
on abs.step=rel.step
```

效果：

abs.step	abs.numbs	abs_ratio	rel_ratio
step1	4716	1.0	NULL
step2	4592	0.9737065309584394	0.9737065309584394
step3	3929	0.8331212892281594	0.8556184668989547
step4	2841	0.6024173027989822	0.7230847543904302

7、模块开发--结果导出

报表统计结果，由 Sqoop 从 Hive 表中导出到 MySQL
Sqoop 对 Hive 与 MySQL 之间数据互导的主要命令如下：

```
1、列出 mysql 数据库中的所有数据库命令
sqoop list-databases \
--connect jdbc:mysql://hadoop02:3306/ \
--username root \
--password root

2、连接 mysql 并列出数据库中的表命令
sqoop list-tables \
--connect jdbc:mysql://hadoop02:3306/weblog \
--username root \
--password root
命令中的 test 为 mysql 数据库中的 test 数据库名称
username password 分别为 mysql 数据库的用户密码

3、将关系型数据的表结构复制到 hive 中
sqoop create-hive-table \
--connect jdbc:mysql://hadoop02:3306/weblog \
--table ods_weblog_origin \
--username root \
--password root \
--hive-table ods_weblog_origin
其中
--table ods_weblog_origin 为 MySQL 中的数据库 weblog 中的表的表名
--hive-table   ods_weblog_origin  为 hive 中新建的表名称
```

```
4、从关系数据库导入文件到 hive 中
sqoop import \
--connect jdbc:mysql://hadoop02:3306/weblog \
--username root \
--password root \
--table ods_weblog_origin \
--hive-import

5、将 Hive 表的数据导出到 MySQL
sqoop export \
--connect jdbc:mysql://hadoop:3306/weblog \
--username root \
--password root \
--table uv_info \
--export-dir /user/hive/warehouse/uv/dt=2013-09-18 \
--input-fields-terminated-by '\t'
```

更多的 Sqoop 的导入导出技巧请参考：
https://blog.csdn.net/zhongqi2513/article/category/6455001

8、模块开发--工作流调度

将整个项目的数据处理过程，从数据采集到数据分析，再到结果数据的导出，一系列的任务分割成若干个 azkaban 的 job 单元，然后调度执行

9、模块开发--数据展示

在企业的数据分析系统中，前端展现工具有很多：
1、独立部署专门系统的方式：以 Business Objects(BO,Crystal Report),Heperion(Brio),Cognos 等国外产品为代表的，它们的服务器是单独部署的，与应用程序之间通过某种协议沟通信息

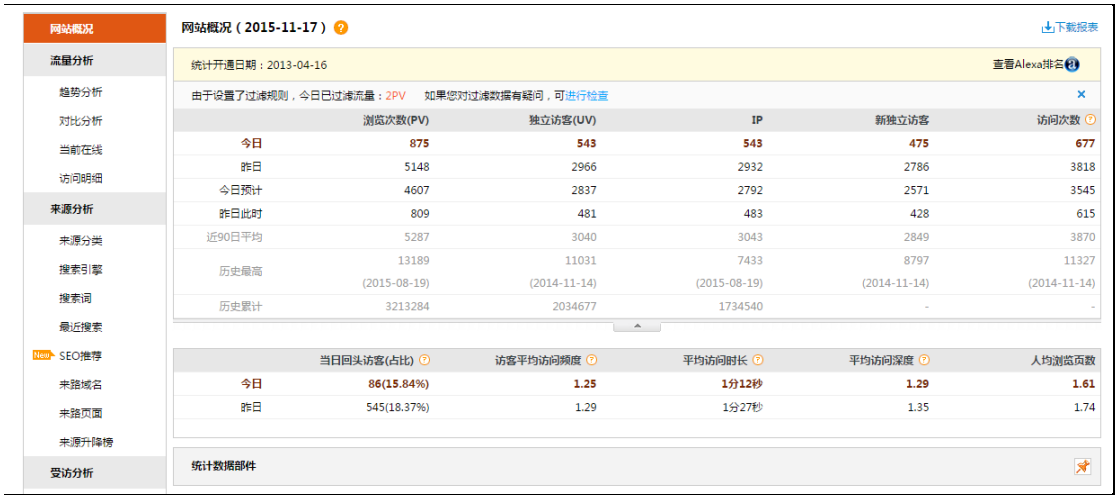
2、有 WEB 程序展现方式：通过独立的或者嵌入式的 JavaWeb 系统来读取报表统计结果，以网页的形式对结果进行展现，如，100%纯 Java 的润乾报表

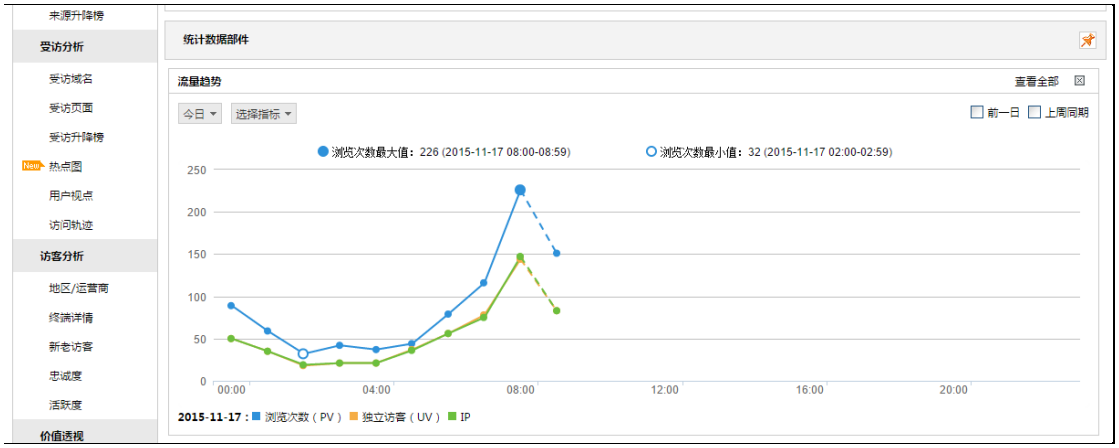
本用户行为日志分析项目采用自己开发 web 程序展现的方式
Web 展现程序采用的技术框架：
Jquery + Echarts + SpringMVC + Spring + MyBatis + MySQL
展现的流程：
1、使用 SSM 从 MySQL 中读取要展现的数据

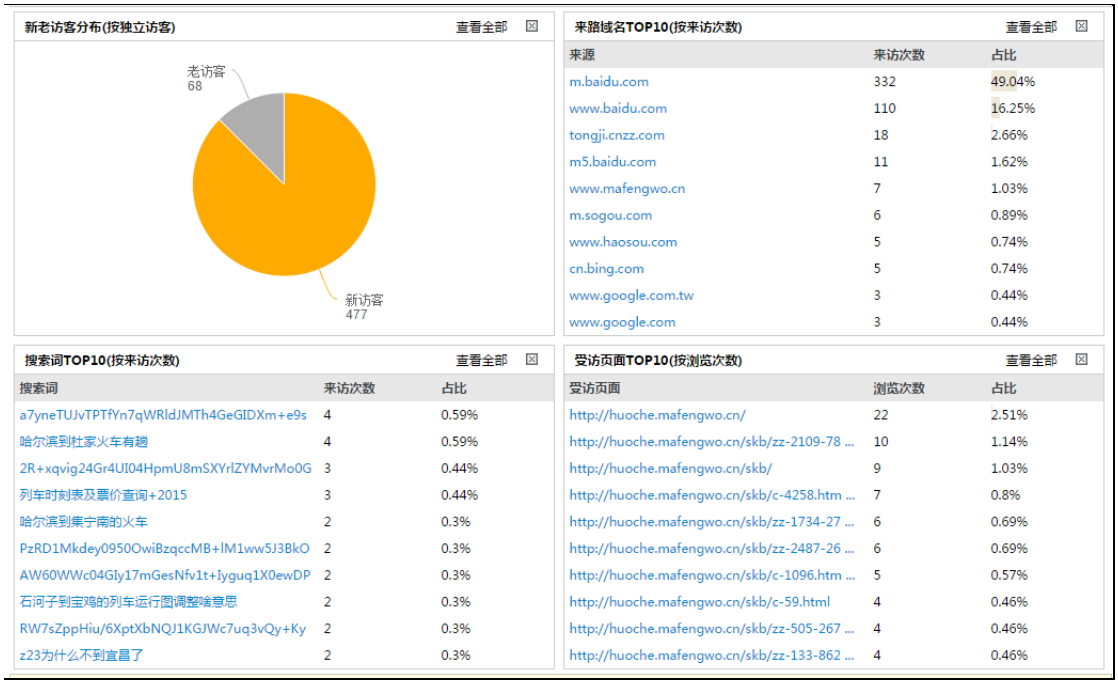
2、使用 JSON 格式将读取到的数据返回给页面

3、在页面上用 Echarts 对 JSON 解析并形成图表

9.1、网站概况







新老访客分布(按独立访客) 查看全部

来路域名TOP10(按来访次数) 查看全部

来源	来访次数	占比
m.baidu.com	332	49.04%
www.baidu.com	110	16.25%
tongji.cnzz.com	18	2.66%
m5.baidu.com	11	1.62%
www.mafengwo.cn	7	1.03%
m.sogou.com	6	0.89%
www.haosou.com	5	0.74%
cn.bing.com	5	0.74%
www.google.com.tw	3	0.44%
www.google.com	3	0.44%

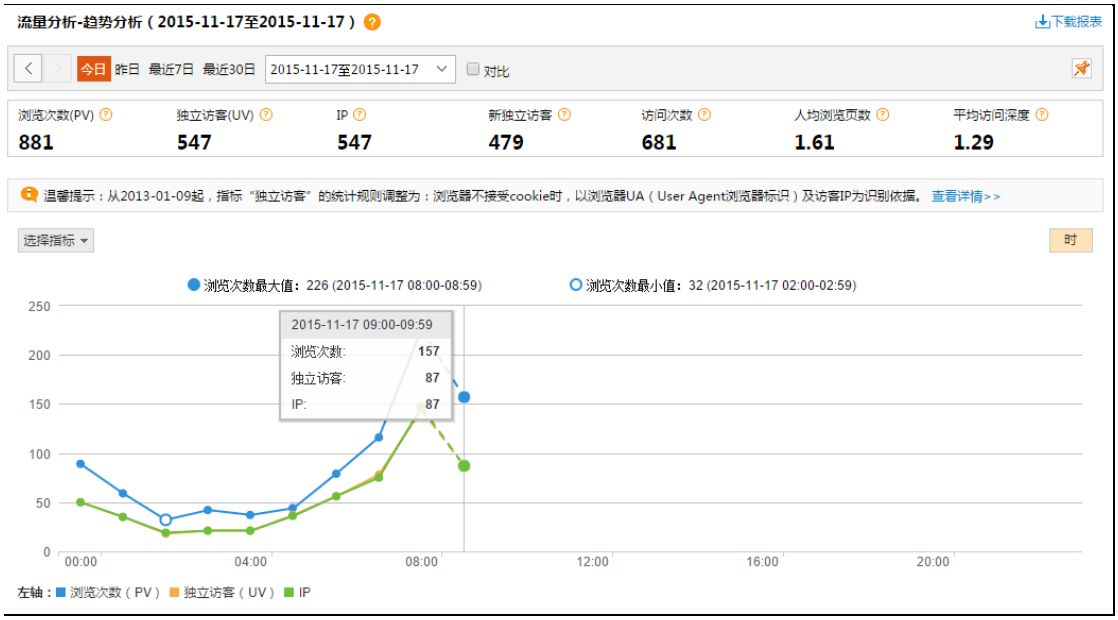
搜索词TOP10(按来访次数) 查看全部

搜索词	来访次数	占比
a7yneTUJvTPTfYn7qWRldJMTh4GeGIDXm+e9s	4	0.59%
哈尔滨到牡家火车有趟	4	0.59%
2R+xqvig24Gr4UI04HpmU8mSXYrlZYMvrMo0G	3	0.44%
列车时刻表及票价查询+2015	3	0.44%
哈尔滨到集宁南的火车	2	0.3%
PzRD1Mkdey0950OwiBzqccMB+lM1ww5J3BkO	2	0.3%
AW60WWc04GIy17mGesNfv1t+Iyguq1X0ewDP	2	0.3%
石河子到宝鸡的列车运行图调整啥意思	2	0.3%
RW7sZppHiu/6XptXbNQJ1KGJWc7uq3vQy+Ky	2	0.3%
z23为什么不到宜昌了	2	0.3%

受访页面TOP10(按浏览次数) 查看全部

受访页面	浏览次数	占比
http://huoche.mafengwo.cn/	22	2.51%
http://huoche.mafengwo.cn/skb/zz-2109-78 ...	10	1.14%
http://huoche.mafengwo.cn/skb/	9	1.03%
http://huoche.mafengwo.cn/skb/c-4258.htm ...	7	0.8%
http://huoche.mafengwo.cn/skb/zz-1734-27 ...	6	0.69%
http://huoche.mafengwo.cn/skb/zz-2487-26 ...	6	0.69%
http://huoche.mafengwo.cn/skb/c-1096.htm ...	5	0.57%
http://huoche.mafengwo.cn/skb/c-59.html	4	0.46%
http://huoche.mafengwo.cn/skb/zz-505-267 ...	4	0.46%
http://huoche.mafengwo.cn/skb/zz-133-862 ...	4	0.46%

9.2、流量分析



流量分析-趋势分析（2015-11-17至2015-11-17） 下载报表

今日 昨日 最近7日 最近30日 2015-11-17至2015-11-17 对比

浏览次数(PV)	独立访客(UV)	IP	新独立访客	访问次数	人均浏览页数	平均访问深度
881	547	547	479	681	1.61	1.29

温馨提示：从2013-01-09起，指标“独立访客”的统计规则调整为：浏览器不接受cookie时，以浏览器UA（User Agent浏览器标识）及访客IP为识别依据。 查看详情>>

选择指标 时

浏览次数最大值：226 (2015-11-17 08:00-08:59) 浏览次数最小值：32 (2015-11-17 02:00-02:59)

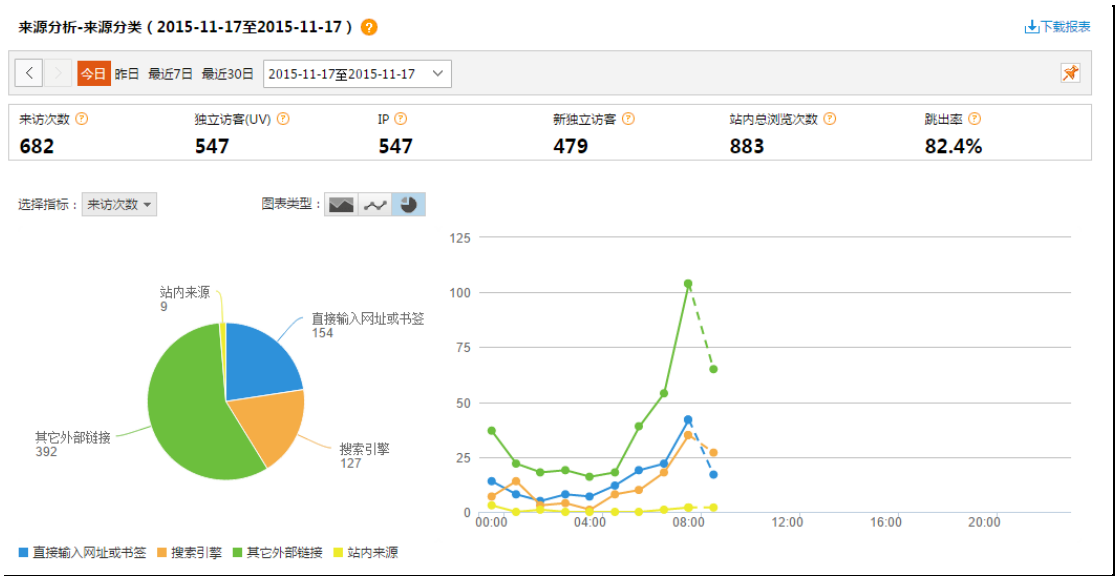
左轴：浏览次数（PV） 独立访客（UV） IP

左轴：■ 浏览次数（PV） ■ 独立访客（UV） ■ IP

查看： 统计报表 占比 ★更多指标

时段	浏览次数(PV)	独立访客(UV)	IP	新独立访客	访问次数	人均浏览页数	平均访问深度
全站总计	881	547	547	479	681	1.61	1.29
00:00-00:59	89	50	50	21	61	1.78	1.46
01:00-01:59	59	35	35	29	44	1.69	1.34
02:00-02:59	32	18	19	18	27	1.78	1.19
03:00-03:59	42	21	21	18	31	2	1.35
04:00-04:59	37	21	21	19	24	1.76	1.54
05:00-05:59	44	37	36	36	38	1.19	1.16
06:00-06:59	79	56	56	50	68	1.41	1.16
07:00-07:59	116	78	75	73	95	1.49	1.22
08:00-08:59	226	144	147	131	183	1.57	1.23
09:00-09:59	157	87	87	84	110	1.8	1.43
10:00-10:59	-	-	-	-	-	-	-

9.3、来源分析



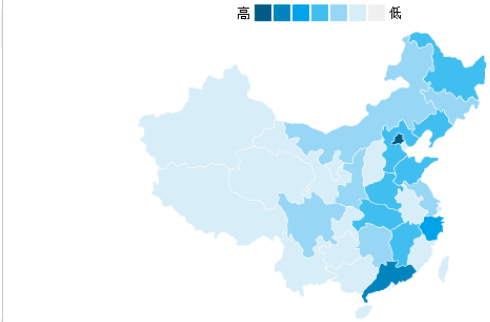
查看： 来源分类 ★更多指标

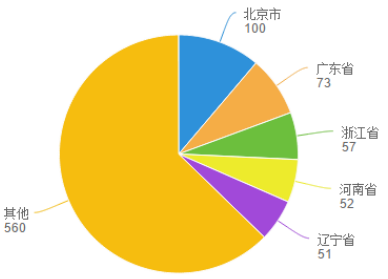
来源形式	来访次数	独立访客(UV)	IP	新独立访客	站内总浏览次数	跳出率
全站总计	682	547	547	479	883	82.4%
搜索引擎	127	97	95	87	180	81.1%
搜索引擎TOP5 查看全部						
百度	109	82	80	74	154	83.49%
搜狗	6	6	6	5	7	83.33%
360搜索	5	4	4	3	8	40%
必应	5	3	3	3	9	60%
谷歌	1	1	1	1	1	100%
搜索词TOP5 查看全部						
a7yneTUJvTPTfYn7qWRldJMTh4GeGI ...	4	1	1	1	9	50%
哈尔滨到杜家火车有趟	4	1	1	1	4	100%
2R+xqvig24Gr4UI04HpmU8mSXYrlZY ...	3	1	1	1	3	100%
列车时刻表及票价查询+2015	3	2	2	2	7	33.33%
哈尔滨到果宁南的火车	2	1	1	1	2	100%

9.4、访客分析

浏览次数(PV)	独立访客(UV)	IP	新独立访客	访问次数	跳出率
893	553	553	485	691	82.49%

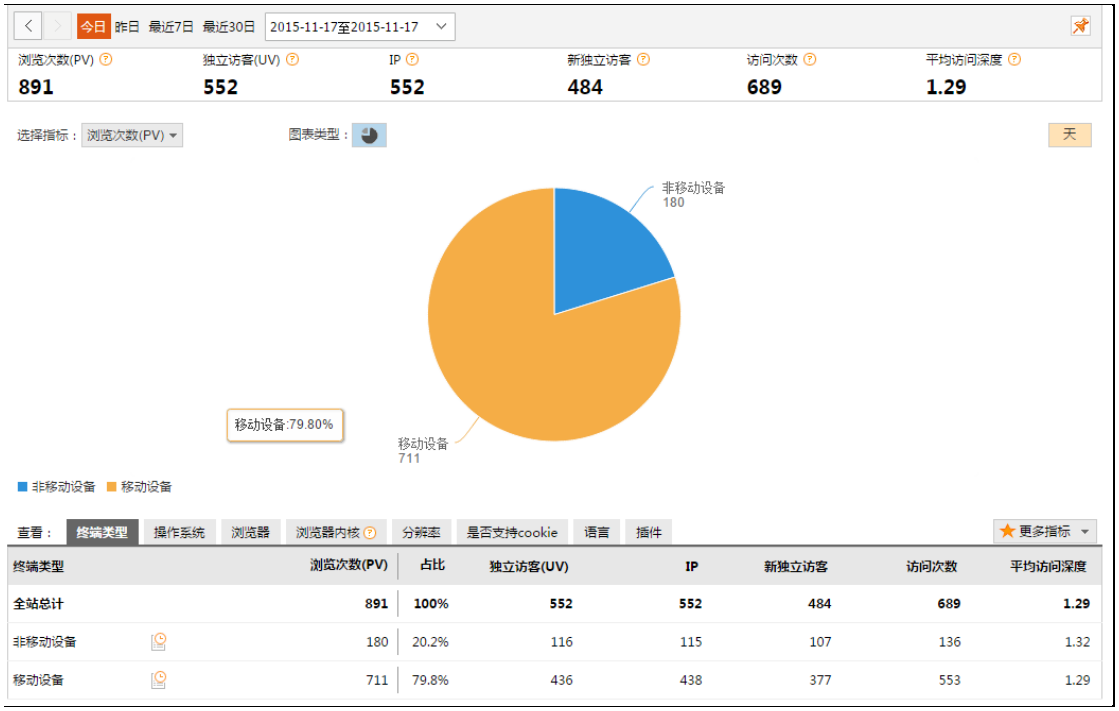
选择指标： 浏览次数（PV）





查看： 地区分布 网络运营商 ★更多指标

省份 城市	浏览次数(PV)	占比	独立访客(UV)	IP	新独立访客	访问次数	跳出率	查看运营商
全站总计	893	100%	553	553	485	691	82.49%	-
北京市	100	11.2%	60	60	50	78	84.62%	查看
广东省	73	8.17%	47	48	41	55	76.36%	查看
浙江省	57	6.38%	37	38	29	42	83.33%	查看



终端类型	浏览次数(PV)	占比	独立访客(UV)	IP	新独立访客	访问次数	平均访问深度
全站总计	891	100%	552	552	484	689	1.29
非移动设备	180	20.2%	116	115	107	136	1.32
移动设备	711	79.8%	436	438	377	553	1.29