# 数据存储

数据存储是DMP平台专门负责数据读写的模块。对外提供一个抽象的读写接口。数据处理模块和数据应用模块的程序可能会调用数据存储中已有的数据，而这两个模块的输出结果部分也会写入数据存储模块中。

数据存储单独作为一个模块的目的是：1. 不同的数据应用共享同一数据源，避免同一数据源的冗余存储。 2. 更新数据源时，保证不同的应用中版本一致，例如知识库的统一管理。3. 便于和任务平台隔离，以免因为存储模块消耗内存过多而使任务终端或平台崩溃。4. 底层使用分布式存储，根据数据量大小灵活添加或移除机器，使数据存储具有良好的扩展性。5. 使开发人员能专注于开发高效的实时读写服务，为用户定向、广告匹配等提供服务。6. 便于提供一系列简单的调用接口，简化数据处理模块和数据应用模块程序的调用。

### 接口描述

数据存储的接口分为http接口和java接口两种。Http接口简单直接，便于跨语言调用。Java接口满足开发高效的数据应用。Http接口规范旨在指定请求url和响应的格式。Java 接口旨在指定该模块api。

* Http接口

提供如下REST风格的HTTP接口：

http://<ip>:<port>/<method>?{query\_string}

query\_string由系统级参数部分和具体Open API调用参数部分组成，以key1=value1&key2=value2&…表示，对于采用POST请求的Open API，query\_string部分则是在POST请求体里。所有查询类的Open API接口既支持POST，也支持GET方式，提交类的OPEN API接口仅支持POST方式。

* 表格1 系统级参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **是否必需** | **描述** |
| method | string | 是 | 采取userinfo.query这样的命名空间方式制定方法 |
| call\_id | int | 否 | 时间戳，系统时间的秒值,同个应用的不同api请求的time值应该是递增的, 用于防replay攻击 |
| format | string | 否 | 响应包格式，可以是xml（默认）或json |
| ie | string | 否 | API调用请求包的编码类型，支持UTF-8和GBK |
| ip | string | 是 | 提供存储查询服务的ip |
| port | int | 是 | 提供存储查询服务的端口 |

* 用户数据查询接口
* 参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **是否必需** | **描述** |
| port | int | 是 | 6579 用户数据查询 |
| method | string | 是 | userinfo.query 查询用户特征 |
| uid | string | 是 | 用户标识，例如RADIUSID、COOKIE |
| feattype\* | int | 否 | 特征类型，0 百度搜索  1 百度搜索关键词  2 电商title  3 电商title关键词  4 电商搜索  5 电商搜索关键词 |
| starttime | long | 否 | 起始时间 |
| endtime | long | 否 | 结束时间 |

注：\*表示可以有0-n个

* 返回值
* Response XML 示例

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<clickwise\_userinfo\_response xmlns=[*http://clickwise.cn*](http://clickwise.cn) xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>

<user>

<uid>7d1cf3105a7388753da162a802d78055009</uid>

<infolist>

<feat>

<feattype>bdsearch</feattype>

<conent>植物大战僵尸小游戏</conent>

</feat>

<feat>

<feattype>bdkeywords</feattype>

<conent>植物大战僵尸 小游戏</conent>

</feat>

<feat>

<feattype>ectitle</feattype>

<conent>春秋平底平跟流苏单鞋唐卡豆豆鞋牛筋底磨砂真牛皮豹纹小白鞋女鞋 </conent>

</feat>

</infolist>

</user>

</clickwise\_userinfo\_response>

* Json格式的字符串内容是XML输出数据所对应的PHP数组的标准JSON字符串
* CookieMap 接口
* 参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **是否必需** | **描述** |
| port | int | 是 | 6578 cookie map |
| method | string | 是 | cookiemap.map 映射用户id |
| src\_uid | string | 是 | 源用户标识，例如RADIUSID、COOKIE |
| src\_type | int | 是 | 源用户标识类型  0 RADIUSID  1 COOKIE  2 Email |
| dest\_type+ | int | 是 | 目标用户标识类型  0 RADIUSID  1 COOKIE  2 Email |

注：+表示可以有1-n个

* 返回值
* Response XML 示例

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<clickwise\_cookiemap\_response xmlns=[*http://clickwise.cn*](http://clickwise.cn) xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>

<src\_uid>

<uid\_type>cookie</uid\_type>

<uid\_value>7d1cf3105a7388753da162a802d78055009</uid\_value>

</src\_uid>>

<dest\_uids>

<uid\_item>

<uid\_type>radius</uid\_type>

<uid\_value>854f8dddc446a9822572aa47f8f75500e</uid\_value>

</uid\_item>

<uid\_item>

<uid\_type>email</uid\_type>

<uid\_value>test@126.com</uid\_value>

</uid\_item>

</dest\_uids>

</clickwise\_cookiemap\_response>

* Java API

*To do*

### 设计实现

数据存储共分为数据存储介质、数据存储驱动、存储查询服务三个子层次。



数据存储介质子层是实际存储数据的各种类型的数据库或文件系统。根据响应时间分为非实时存储介质，如文本文件、hdfs、hive、mysql等，和实时存储介质，如hbase、redis、ardb、memcache等。根据可扩展性分为本地存储介质，如文本文件、mysql、redis等，和分布式存储介质,如hdfs、hive、hbase等。根据所要存储的数据量大小、数据格式以及响应时间选择实际的存储介质。一些查询速度快的介质，如redis、内存等可以用来做本地缓存。

数据存储驱动子层是数据存储介质子层上面封装的一系列接口，该子层的存在目的是：1. 统一管理已有数据，能够方便的浏览已有的表、数据库以及各自的格式 2. 记录读写日志，以分析数据的质量情况、读写的速度和稳定性。3. 将高并发实时查询分配到多台机器上，平衡各机器之间的负载。4. 随机的热点数据的本地缓存，以提高查询效率。5. 将连接数据库、读写数据库等过程封装成统一接口，便于存储查询服务中调用。

存储查询子层服务提供不同数据源的读写接口，例如用户数据、CookieMap、知识库等。该子层根据不同数据源的数据特性，调用数据存储子层的一系列接口，对数据应用层和数据加工层提供读写服务，对外提供Http接口和Java API。

### 安装部署

*To do*

# 数据应用

## DSP平台界面

### 接口描述

### 设计实现

### 安装部署

## DSP对接

DMP平台为DSP平台提供数据和计算支持。DMP平台利用已有的用户数据、计算能力、实现算法，对特定上下文中的某一用户优化广告投放，从而提高广告的点击率和转化率。DMP平台的用户数据来自本身的数据采集层和广告投放日志。为了制定广告投放策略，DMP平台需要维持并迅速更新本地广告库，使本地广告库和DSP平台广告库同步。DMP平台的DSP对接模块依托数据存储层的用户和广告数据，首先根据业务需求生成候选广告集合，然后调用已实现算法选择出点击率和转化率可能比较高的广告。DSP对接模块还负责跟踪广告投放过程，统计当前的广告投放效果，收集广告投放日志等。

DMP平台主要分两部分内容，一是业务逻辑，包括同步广告库、广告投放策略实现、跟踪广告投放日志、统计广告投放效果；二是广告算法，该部分根据广告投放日志和用户数据，不断优化广告投放效果。

### 接口描述

该模块和DSP平台进行交互，所以提供http接口，从而使DSP平台可以直接调用。同样提供如下REST风格的HTTP接口：

http://<ip>:<port>/<method>?{query\_string}

* 用户广告匹配接口

* 参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **是否必需** | **描述** |
| port | int | 是 | 6577 ad match |
| method | string | 是 | dsp.admatch 对用户选择广告 |
| uid | string | 是 | 用户标识，例如RADIUSID、COOKIE |
| uid\_type | int | 是 | 用户标识类型  0 RADIUSID  1 COOKIE  2 Email |
| host | string | 否 | 用户正在访问的网站域名 |
| uri | string | 否 | 用户正在浏览的网页 |
| title | string | 否 | 用户正在浏览的网页title |
| vtime | long | 否 | 用户访问时间 |
| adloc | double[] | 否 | 广告位的位置 |
| adform | int | 否 | 广告位的形式 |
| adsize | double[] | 否 | 广告位的尺寸 |
| adnum | int | 否 | 返回广告的最大数目 |

* 返回值
* Response XML 示例

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<clickwise\_admatch\_response xmlns=[*http://clickwise.cn*](http://clickwise.cn) xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>

<matchinfo>

<uid>7d1cf3105a7388753da162a802d78055009</uid>

<uid\_type>cookie</uid\_type>

<matchads>

<matchad>

<aid>1001</aid>

<score>0.7</score>

</matchad>

<matchad>

<aid>1002</aid>

<score>0.5</score>

</matchad>

</matchads>

</matchinfo>

</clickwise\_admatch\_response>

* 广告日志接口

* 参数

|  |  |  |  |
| --- | --- | --- | --- |
| **参数名** | **类型** | **是否必需** | **描述** |
| port | int | 是 | 6576 ad log |
| method | string | 是 | dsp.adlog 对用户选择广告 |
| uid | string | 是 | 用户标识，例如RADIUSID、COOKIE |
| uid\_type | int | 是 | 用户标识类型  0 RADIUSID  1 COOKIE  2 Email |
| aid | string | 是 | 广告id ,能够唯一确定广告库里的某一广告 |
| host | string | 否 | 用户正在访问的网站域名 |
| uri | string | 否 | 用户正在浏览的网页 |
| title | string | 否 | 用户正在浏览的网页title |
| vtime | long | 否 | 用户访问时间 |
| adloc | double[] | 否 | 广告位的位置 |
| adform | int | 否 | 广告位的形式 |
| adsize | double[] | 否 | 广告位的尺寸 |
| click | boolean | 是 | 是否点击 |

* 返回值
* Response XML 示例

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<clickwise\_status\_response xmlns=[*http://clickwise.cn*](http://clickwise.cn) xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>

<error\_code>0</error\_code>

<error\_msg>success</error\_msg>

</clickwise\_status\_response>

### 设计实现

DSP对接模块主要有三个子模块，广告匹配服务、业务逻辑及广告算法、用户及广告数据更新。



广告匹配服务是DSP对接模块与其它模块交互的窗口。广告匹配服务提供广告匹配http接口，DSP程序可以方便的调用。并提供存储广告日志的http接口，方便DSP程序记录广告日志。广告匹配服务要考虑高并发的匹配请求，并且保证每个广告匹配的实时性。这就要求业务逻辑实现及广告算法要高效，并且用户和广告的查询速度要满足实时要求。

业务逻辑及广告算法，业务逻辑是广告主制定的广告匹配策略，例如对特定host或特定地域的人群投放，时间选择、黑名单、白名单等。通过业务逻辑的过滤，形成候选广告集合。对候选广告集合的广告根据预测点击率进行排序。例如从用户广告点击日志建立LR模型。

用户和广告数据更新在后台进行，DSP对接模块每隔特定时间更新用户和广告的数据库。用户数据的更新间隔时间可以长一些，例如两小时更新一次，每次更新的数据量也比较大。广告数据库要更新的数据库相对较小，更新间隔时间要短一些，例如五分钟更新一次。

### 安装部署

# 数据采集

## RADIUS数据采集

RADIUS是运营商用来验证用户身份、动态分配ip使用的协议。每个用户有唯一的RADIUSID。从RADIUS原始报文里可以提取出RADIUSID、IP、STATUS字段。STATUS 有三种1 上线，2 下线，3 心跳。依次记录用户每次会话原始报文里的这三个字段以及这次会话的时间戳，就可以知道一个RADIUSID在特定时间使用的IP是多少或者一个IP在特定时间是被分配到哪个RADIUSID。

RADIUS数据采集负责从SIG RDS或RADIUS CENTER读取上述三个字段。SIG RDS或RADIUS CENTER发出的数据流是从原始报文实时解析得到的，所以可以把读取某条数据的时间作为该条数据的时间戳。每条数据包括RADIUSID、IP、STATUS三个字段。

### 接口描述

RADIUS数据采集对外提供的数据有两种，实时数据、离线数据。当前有RADIUS数据的省份有两个浙江、江苏。虽然浙江发出RADIUS数据流的是SIG RDS，江苏发出RADIUS数据流的是RADIUS CENTER。但是对外提供的数据接口基本上是一致的。

浙江的实时RADIUS存储在Memcache中，江苏的实时RADIUS存储在Redis中，以RADIUSID作为key，可以查出此时该RADIUSID使用的IP，以IP为key，可以查出此时该IP被分配到哪个RADIUSID。

离线RADIUS数据存储在radius log 文件中，文件每一行格式如下：

IP地址\t 在线\离线状态\t 用户ID\t 接收时间戳

例如：

183.155.88.212 1 shJHkuoXXAqo59/qEDFFnQ== 2014-01-23 15:27:16

122.242.84.167 1 shJHkuoXXAoYqJLyjlcMpA== 2014-01-23 15:27:16

183.152.8.53 2 c0sdDEpvEh39huVbnNjTHQ== 2014-01-23 15:27:16

### 设计实现

浙江从SIG RDS发出RADIUS数据流，江苏从RADIUS CENTER发出RADIUS数据流。两个数据流的格式不同，而且SIG RDS和RADIUS CENTER认证数据流接收端的方式也不相同。所以两个地方的RADIUS数据接收端是使用不同的程序实现。



浙江RADIUS接收端通过TCP协议连接SIG RDS，发送验证信息。SIG RDS确认客户端身份后，向接收端发送应答信息并开始发送Radius数据流。

验证信息的格式如下：

typedef struct key

{

unsigned short usKeyType; /\* 固定为 1\*/

unsigned short usKeylen; /\* KEY 实际长度\*/

unsigned char ucKey[20];

}KEY;

应答信息的格式如下：

typedef struct keyRes

{

unsigned long ulKeyRes; /\* 0: OK; 1:Key Erro; 2: Max Connect;3:链接超时;\*/

unsigned long ulLastSeq; /\*超时断开时有效，最后一个Seq 。如果是认证超时，Seq为0\*/

}KEYRES;

每条RADIUS消息的格式如下：

typedef struct RadiusInfo

{

unsigned long user\_ip; /\* 用户IP\*/

unsigned short user\_status; /\* 1：start，2：stop, 3：ka\*/

unsigned short user\_name\_len; /\*帐号长度\*/

unsigned char user\_name[0]; /\* 帐号信息，大小不固定，由user\_name\_len决定\*/

} RADIUSINFO ;

每隔5秒，RADIUS接收端向SIG RDS发送一个KA信息，以通知接收端正常存在，SIG RDS定时检查KA是否更新，如果超过20s未更新则发送超时关闭信息，并断开链接。KA信息的结构如下：

typedef struct RadiusKA

{

//unsigned long ulSeq; /\* Seq\*/

uint32\_t ulSeq; /\* Seq\*/

} RADIUSKA;

RADIUS接收端收到RADIUSINFO消息，解析出RADIUSID、IP、STATUS字段，更新memcache里RADIUSID对应的IP以及IP对应的RADIUSID。并把当前时间作为时间戳，按照”IP地址\t 在线\离线状态\t 用户ID\t 接收时间戳”的格式写入日志文件。



江苏RADIUS接收端通过TCP协议连接RADIUS CENTER，身份认证是通过接收端的IP来进行。RADIUS CENTER维持一份允许的IP列表，只支持这些IP地址的接收端读取数据。RADIUS CENTER在确认接收端连接后，向接收端发送RADIUS数据流。该RADIUS数据流是RADIUS CENTER转发的原始IP报文，包括报文头和报文体。接收端读取到的每条报文头格式如下：



报文头共有16字节，前四字节等于报文体的长度加12。报文体的格式如下：



2字节长度表示报文体的总长度bodylen。User-Name的长度等于报文体的总长度减去32。在RADIUS接收端收到的报文体中，Acc-Status-Type、User-Name、Framed-IP-Address占据报文体的最后bodylen-20个字节。但是这bodylen-20个字节中Acc-Status-Type、User-Name、Framed-IP-Address的顺序是不固定的。但是已知的是Acc-Status-Type占六个字节，Framed-IP-Address也占六个字节，User-Name占据剩余的bodylen-32个字节。如果将包含Acc-Status-Type、User-Name、Framed-IP-Addres的bodylen-20用十六进制串来表示，即每一个字节用两个十六进制数字表示，那么Acc-Status-Type是以28 06开始的六个字节，而Framed-IP-Addres是以08 06开始的六个字节。Framed-IP-Addres是个去掉08 06后四个字节，每个字节转换成整数后的t1.t2.t3.t4形式的串。Acc-Status-Type是个六个字节的最后一个字节对应的整数。

因为Acc-Status-Type、User-Name、Framed-IP-Address顺序不固定的，要根据十六进制串的形式来找到将三个字段分离出来，所以采用远程解析的方式。假设用ufa表示包含Acc-Status-Type、User-Name、Framed-IP-Addres的bodylen-20用十六进制串为，得到ufa的时间作为时间戳，那么得到的记录是timestamp\tufa。该记录会写入日志文件，并且同过tcp协议传输到另一台主机。另一台主机首先将该记录放入RecordPond中，已启动的多个ParseThread并发从RecordPond读取timestamp\tufa形式的数据，解析出Framed-IP-Address 、Acc-Status-Type、User-Name，并更新redis里IP对应的RADIUSID，即User-Name和RADIUSID对应的IP。

### 安装部署

* 浙江Radius接收端部署

接收端程序和memcahe位于两台机器上，其中安装程序及源码位于

183.136.168.67上，memcahe 位于183.136.168.68上。

* 程序目录

安装程序存放在183.136.168.67 /root/thz\_work/radius\_client中，目录如下：

radius\_client

|-- Makefile

|-- bin

|-- lib #依赖库及头文件

|-- log #日志目录

|-- log4cxx.properties #日志配置文件

|--mergefile.sh #归档脚本【每5分钟将output目录文件合并到/data1/radius\_data】

|-- make

|-- output #临时输出目录

|-- radius\_client

|-- run.sh

|-- src #源代码目录

|-- start.sh #启动脚本

|-- stop.sh #关闭脚本

|-- sysconfig.ini #配置文件

Memcache位于183.136.168.68上。

* 配置文件

#浙江SIG Rasius 服务器

[radius\_server]

ip=220.191.158.69

port=10000

#memcache服务器

[memcache]

memcache\_ip=192.168.10.68

memcache\_port=20000

#本地临时存储目录

[param]

outputpath=./output

* 江苏Radius接收端部署

接收端程序和实时解析程序分别位于180.96.26.203和180.96.26.204上。

* 程序目录
* 接收端程序接收并初步解析得到timestamp\tufa记录的程序位于180.96.26.203 /home/lq 目录下。目录内容如下：

|-- delete\_outdate.sh

|-- log4j.properties

|-- logs

|-- log.txt

|-- organizeFiles.sh

|-- radiusReform.jar

|-- restart\_remote\_radius\_client\_single.sh

|-- runtime.log

|-- start\_radius\_client\_remote\_single.sh

|-- sync\_radius\_hour.sh

* 接收端Crontab配置

10 00 \* \* \* root (cd /home/lq; ./organizeFiles.sh)

10 01 \* \* \* root (cd /home/lq; ./delete\_outdate.sh)

05 \* \* \* \* root (cd /home/lq; sh sync\_radius\_hour.sh today >> /tmp/log\_scp\_hour.txt 2>&1 )

28 \* \* \* \* root (cd /home/lq; sh restart\_remote\_radius\_client\_single.sh )

1 \* \* \* \* root /usr/sbin/ntpdate ntp.api.bz

15 0 \* \* \* root (cd /home/lq; ./sync\_radius.sh yesterday >> /tmp/scp\_radius.log 2>&1)

* 实时解析程序位于180.96.26.204 /home/lq 目录下。目录内容如下：

|-- delete\_outdate.sh

|-- log4j.properties

|-- log\_q.txt

|-- organizeFiles.sh

|-- parsedLogs

|-- radiusReform.jar

|-- restart\_radius\_queue\_pond.sh

|-- start\_radius\_query\_server.sh

|-- 1 22:44 start\_radius\_queue\_pond.sh

* 实时解析端Crontab配置

10 00 \* \* \* root (cd /home/lq; ./organizeFiles.sh)

10 01 \* \* \* root (cd /home/lq; ./delete\_outdate.sh)