尚硅谷大数据项目之在线教育-用户画像

(作者：尚硅谷大数据研发部)

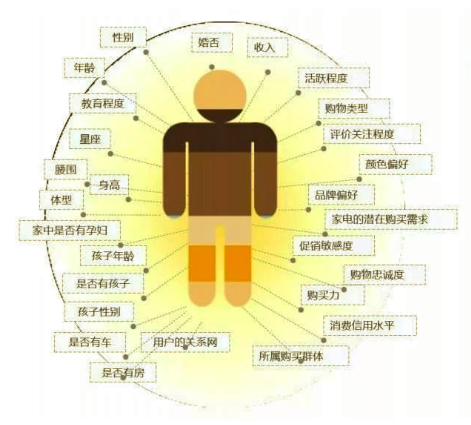
版本：V1.0

# 第1章 用户画像概念

## 1.1 什么是用户画像

1. 百度百科：用户画像又称用户角色，作为一种勾画目标用户、联系用户诉求与设计方向的有效工具，用户画像在各领域得到了广泛的应用。
2. 说人话：用数字化的标签描述用户个性特征的一种方法。再简单点理解就是给用户打标签！可以算是数据仓库中的DM层！
3. 图示：

## 



## 1.2 用户画像应用方向

1. **根据标签快速筛选人群**
2. 精准营销
3. 新注册用户赠送首次购买VIP优惠券
4. VIP快到期用户发放折扣券
5. 新学年快开始时，给非VIP用户发放下一年级课程包优惠券

# 第2章 项目(用户分群系统)需求及架构设计

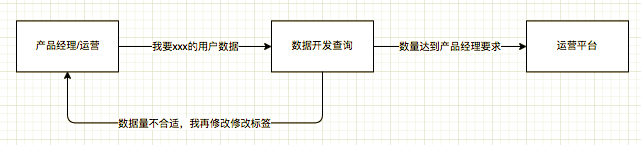
## 2.1 项目需求分析

1. 通过Hive/Sparksql后端生成标签库
2. 前端通过ui选择标签，通过JDBC形式提交到后端SparkSQL引擎，快速统计人数
3. 可以将符合筛选标签的用户ID快速导入运营平台或CRM (客户关系管理系统)，达到精准运营(比如平时我们收到的各种短信)





## 2.2 为什么会有这个需求？没有用户分群系统时是什么样的流程

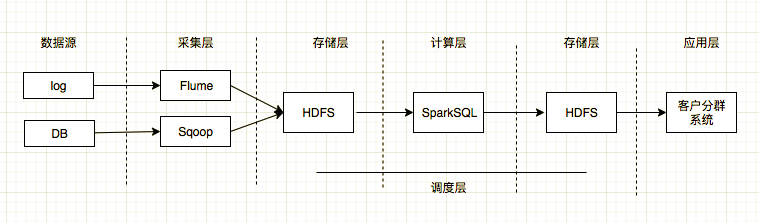


通过用户分群系统，产品经理和运营等同事不再需要向数据开发反复提需求，只需自己在平台操作，并且可以将符合需求的用户直接导入到其它平台，全部流程不再需要手动执行，大大提高生产效率！

## 2.3 项目主要模块

1. 标签定义模块
2. 标签开发模块
3. 任务调度模块
4. 用户分群UI模块

### 2.2.2 系统架构图设计



### 2.2.3 框架版本选型

|  |  |
| --- | --- |
| 产品 | 版本 |
| Hadoop | 2.7.2 |
| Flume | 1.7.0 |
| Hive | 1.2.1 |
| Sqoop | 1.4.6 |
| MySQL | 5.6.24 |
| Azkaban | 2.5.0 |
| Java | 1.8 |
| Zookeeper | 3.4.10 |

注意事项：框架选型尽量不要选择最新的框架，选择最新框架半年前左右的稳定版。

### 2.2.4 集群资源规划设计

|  |  |  |  |
| --- | --- | --- | --- |
|  | 服务器hadoop102 | 服务器hadoop103 | 服务器hadoop104 |
| HDFS | NameNode  DataNode | DataNode | DataNode  SecondaryNameNode |
| Yarn | NodeManager | Resourcemanager  NodeManager | NodeManager |
| Zookeeper | Zookeeper | Zookeeper | Zookeeper |
| Flume(采集日志) | Flume | Flume |  |
| Kafka | Kafka | Kafka | Kafka |
| Flume（消费Kafka） |  |  | Flume |
| Hive | Hive |  |  |
| MySQL | MySQL |  |  |
| Presto | Presto |  |  |

# 第3章 标签设计模块

## 3.1 将所有标签分为如下两类

用户属性标签、用户行为标签和外部标签(数据交换)

1. 用户属性标签

* 性别
* 年龄
* 城市
* 年级
* 学校
* 渠道
* 注册时间
* 哪门课程VIP
* 是否是好友推荐（是否添写了邀请码）
* …

1. 用户行为标签

* 喜欢在什么系统登录
* 之前已购买VIP总时长
* 是否7日内活跃(登录过) 0 ->不活跃，1 -> 活跃
* 是否深度活跃(7日内看过5个视频以上)
* 有付费意愿(单日内3次进入订单页且没有成为会员)
* 是否乐于助人(7日内社区内回答超过3个问题)
* 是否是学霸(7日内做题超过3次且完成等级为S)
* …

## 说明

用户画像中的标签不是凭空捏造的，一定是产品经理或业务人员通过业务发现需求，提出的标签模型，在企业中开发时，需要对业务非常熟悉的人提出标签需求才能进入用户画像的开发！

# 第4章 代码实现流程

## 4.1 用户标签窄表介绍

1. Hive schema

create table if not exists dm.user\_tag\_narrow(

uid STRING comment "用户id",

str\_tag\_value STRING comment "字符型标签值",

date\_tag\_value DATE comment "日期型标签值",

int\_tag\_value INT comment "数字型标签值")

partitioned by (tag\_id STRING) row format delimited fields terminated by "\t";

1. 现在ods有从业务数据库中同步过来的users表，
2. Mysql schema如下（app库下创建）：

CREATE TABLE `users` (  
  `id` int(11) NOT NULL AUTO\_INCREMENT,  
  `uid` varchar(32) DEFAULT NULL,  
  `username` varchar(16) DEFAULT NULL,  
  `age` tinyint(4) DEFAULT NULL,  
  `gender` varchar(8) DEFAULT NULL,  
  `province` varchar(16) DEFAULT NULL,  
  `city` varchar(16) DEFAULT NULL,  
  `channel` varchar(16) DEFAULT NULL,  
  `invite\_code` varchar(16) DEFAULT NULL,  
  `register\_date` datetime DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8

插入2条测试数据：

insert into users (uid,username,age,gender,province,city,channel,invite\_code,register\_date) values("40001","Tom",18,"男","北京","北京","华为","123456789","2019-04-24 16:25")

INSERT INTO users (uid,username,age,gender,province,city,channel,register\_date) VALUES("4002","Marry",20,"女","河北","保定","头条","2019-04-23 12:25")

1. Sqoop导入语句：

注意：这里是根据字段动态生成表，防止建好表，schme字段老变化

sqoop import \

--connect jdbc:mysql://atguigu:3306/app \

--username root \

--password 123456 \

--table users \

--fields-terminated-by "\t" \

--delete-target-dir \

--target-dir /user/hive/warehouse/ods.db/users

--num-mappers 4 \

--hive-import \

--hive-database ods \

--hive-table users \

--null-string '\\N' \

--null-non-string '\\N'

1. 将用户基础属性标签写入到用户标签窄表dm.user\_tag\_narrow中

此处只模拟插入3个属性标签，但全部类型已覆盖到

from ods.users

insert overwrite table dm.user\_tag\_narrow partition(tag\_id="age")

select uid,null,null,age

insert overwrite table dm.user\_tag\_narrow partition(tag\_id="gender")

select uid,gender,null,null

insert overwrite table dm.user\_tag\_narrow partition(tag\_id="register\_date")

select uid,null,to\_date(from\_unixtime(UNIX\_TIMESTAMP(register\_date,"yyyy-MM-dd"))) as register\_date,null

......

1. 将用户行为标签写入到用户标签窄表dm.user\_tag\_narrow中

以7日活跃为例

insert overwrite table dm.user\_tag\_narrow partition(tag\_id="seven\_days\_active")

select uid,null,null,is\_active from (

select t1.uid as uid,if(t2.uid != null,1,0) as is\_active from

(select uid from ods.users) t1

left join

(select distinct uid as uid from dwd.user\_behavior where dt between ${startDay} and ${endDay} and event\_key = "startApp") t2

on t1.uid = t2.uid) tmp;

## 用户标签宽度介绍

1. 宽表schema需要由窄表有多少指标来定，大概逻辑为：

drop table if exists dm.user\_tag;create table if not exists dm.user\_tag

as

select

uid,

max(case when tag\_id="age" then int\_tag\_value end) as age,

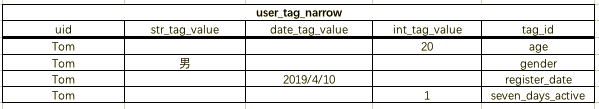
max(case when tag\_id="gender" then str\_tag\_value end) as gender,

max(case when tag\_id="register\_date" then date\_tag\_value end) as register\_date,

max(case when tag\_id="seven\_days\_active" then int\_tag\_value end) as seven\_days\_active

from dm.user\_tag\_narrow group by uid;

1. 数据转换过程如下图：





# 第五章 程序部署说明

1. 整套流程使用调度框架执行，每天凌晨执行一次
2. ods.users表，每次使用sqoop全量导入，且自动生成Hive表，而不是手动创建。因为业务数据库的users表schema不固定！
3. dm.user\_tag每日先删除再重新生成表，因为标签窄表非常可能增加或减少标签，所以schema也无法固定！