广西电信智能营销服务支撑平台

实时营销设计方案

陕西中海华晟信息技术有限公司

2017年12月

目录

[一、平台背景 3](#_Toc500399596)

[二、业务场景 3](#_Toc500399597)

[三、方案分析 4](#_Toc500399598)

[3.1 实现方案 4](#_Toc500399599)

[平台活动提交 4](#_Toc500399600)

[平台活动更新 5](#_Toc500399601)

[Kafka消息订阅（平台侧） 5](#_Toc500399602)

[消息推送服务 5](#_Toc500399603)

[Spark处理流程 6](#_Toc500399604)

[3.2 数据格式 6](#_Toc500399605)

[3.1.1 平台侧 6](#_Toc500399606)

[Kafka消息格式 10](#_Toc500399607)

[Redis数据格式 10](#_Toc500399608)

[Spark内存数据格式 12](#_Toc500399609)

[，03.3 集成方案 13](#_Toc500399610)

[Redis集群 13](#_Toc500399611)

[Spark 14](#_Toc500399612)

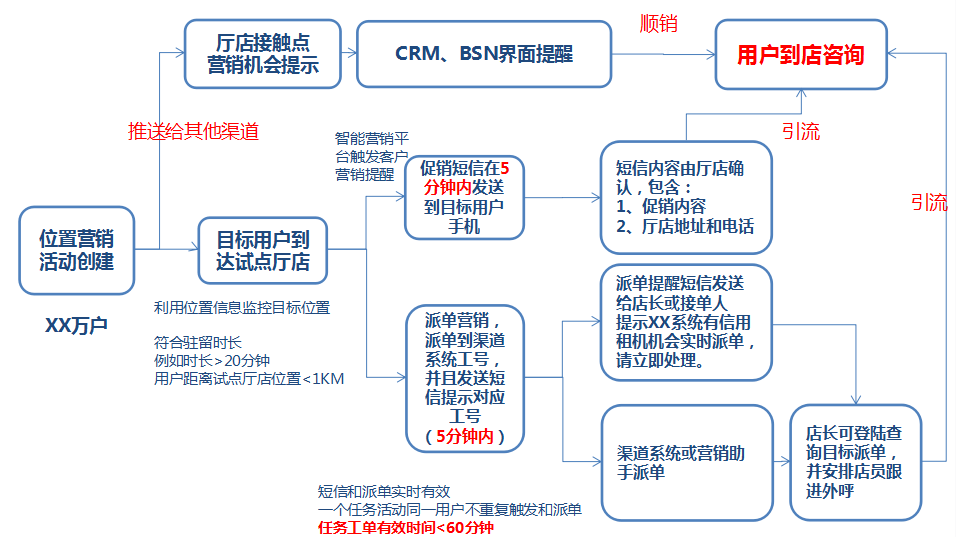
[核心技术模块 15](#_Toc500399613)

# 一、平台背景

智能营销平台目前囊括了营销资源库、营销策划、营销执行、客户挖掘、报表图表等多个基础功能，并能派单至多级地市区县，为其提供指导，提升运营效率。

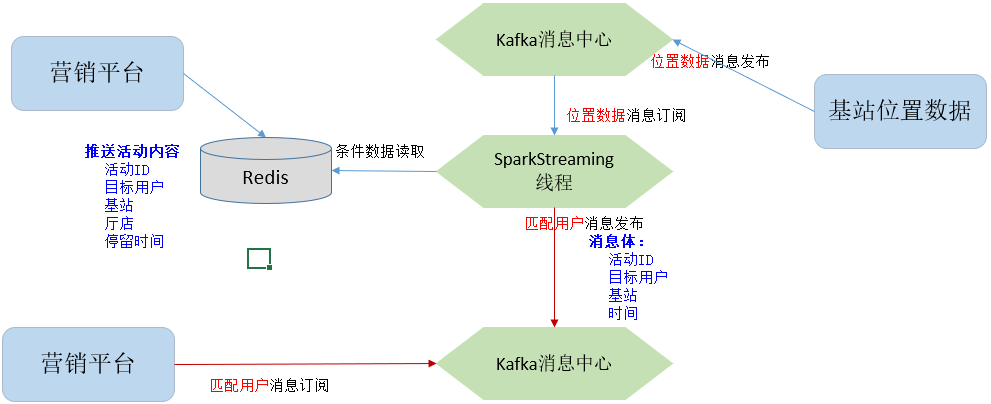
平台的用户权限相对复杂、所以平台在本质上也有一套比较成熟的权限机制，以及灵活的参数、系统、模块、组织机构的配置功能，同时也有详细的日志监控，当中运行的核心平台是可以满足多个平台的正常运转，故对系统架构做了简要的概述。

# 二、业务场景



# ·三、方案分析

## 3.1 实现方案



### 3.1.1 平台活动提交

【处理步骤】

1. 厅店配置

点击保存后，后台服务负责根据客户选择的厅店，将厅店信息保存到PL\_IM\_REALTIME\_ACTIVITY\_HALL（实时营销活动厅店信息表）

1. 触发规则

点击保存后，后台服务负责将基站范围、停留时间，保存到实时营销活动配置表（PL\_IM\_REALTIME\_ACTIVITY\_CONF）

1. 触发事件

点击保存后，后台服务负责将客户选择的事件触发规则保存到实时营销活动配置表（PL\_IM\_REALTIME\_ACTIVITY\_CONF）

1. 系统派单提交
   1. 保存活动信息数据到实时营销活动信息表（PL\_IM\_MARKETING\_ACTIVITY）
   2. 保存活动用户数据到实时营销活动用户信息表（PL\_IM\_REALTIME\_ACTIVITY\_USER\_INFO）
   3. 基站计算

保存后，后台服务根据用户所选的基站范围，计算出厅店和对应的基站关系，将数据保存到PL\_IM\_REALTIME\_ACTIVITY\_HALL（实时营销活动厅店信息表）；

* 1. 更新Redis数据

调用后台的Redis数据刷新服务，将"活动信息、活动用户、活动基站"更新到Redis集群中，具体的数据格式，请参考章节《Redis数据格式》

### 3.1.2 平台活动更新

【处理步骤】

活动支持暂停、取消等操作，执行后会调用后台的Redis数据刷新服务，将"活动信息”中活动状态刷新为指定值，具体请参考章节《Redis数据格式》中"活动信息”定义

### 3.1.3 Kafka消息订阅（平台侧）

【处理步骤】

1. 平台侧提供Kafka消息订阅负责，订阅Topic名称：TargetUserTopic
2. 当接收到Spark推送的“活动ID,用户手机号,厅店ID, starttime”消息时，负责调用”消息推送服务”

### 3.1.4 消息推送服务

【处理步骤】

1. 服务被调用后，更新消息数据到实时营销活动用户信息表（PL\_IM\_REALTIME\_ACTIVITY\_USER\_INFO）
2. 调用接口，负责推送相关营销数据到该目标用户、指定的厅店负责人（或全员）

### 3.1.5 Spark处理流程

1. 调用redis中活动信息

将这一操作设计成懒加载的模式，只有在使用的时候才去加载他（根据活动id查找目标用户、根据活动id获取基站对应厅店的列表），当活动状态为1的时候，获取正在执行活动id的列表。

1. 获取基站信息列表

根据活动id循环读取基站信息的数据，对基站和厅店求笛卡尔积之后得到：基站+厅店+活动id+要求停留时间，将转化的基站列表格式化成rdd类型。这样的基站列表信息设置每5分钟加载一次。

1. 获取kafka中的信令数据

设置每10秒去读取kafka的数据来处理，将获取的数据用#分割得到基站id、电话tel、开始时间starttime，之后进行map处理，将基站id作为key，tel、starttime作为value。

1. 过滤信令数据

得到信令数据中的tel，然后去查找redis中活动用户中是否有这条数据并且用户的状态为0，如果这条数据不符合前面的条件，则直接过滤。

1. Join操作

将第2步得到的基站rdd和第4步过滤出来的rdd进行join操作，之后再进行map处理，得到带有活动id、tel的key，厅店id、基站id、starttime、要求停留时间的value。

1. 计算停留时间
2. 首先判断这条数据是否是第一次得到的，根据厅店id是否为0来判断，如果为0则表示第一次进入，将步骤5的value值保存下来
3. 如果不是第一次进入，则判断当前厅店和保存的状态的厅店是否相等，如果不相等则更新状态中厅店的信息
4. 如果厅店相等，计算累计数据，用状态中的累计滞留时间+(当前获取的时间-状态中保存的时间)，判断累计滞留时间是否大于等于要求停留时间，如果不大于，则更新滞留时间到状态中
5. 如果累计滞留时间大于等于要求停留时间，则更新redis中活动用户的状态为1，并且将当前的状态移除，然后输出这条记录
6. 向kafka中写入

启动kafka，向kafka中写入结果数据。

## 3.2 数据格式

### 3.2.1 平台侧

#### 1. 实时营销活动信息表

**存储位置：**

Oracle

**数据存储格式：**

**PL\_IM\_MARKETING\_****ACTIVITY**

|  |  |  |
| --- | --- | --- |
| ACTIVITY\_ID | VARCHAR2(255) | 活动标识 |
| ACTIVITY\_NAME | VARCHAR2(255) | 活动名称 |
| ACTIVITY\_DESC | NVARCHAR2(2000) | 活动说明 |
| ACTIVITY\_TYPE\_ID | VARCHAR2(255) | 活动分类ID（PL\_IM\_ACTIVITY\_TYPE） |
| GROUP\_ID | VARCHAR2(255) | 原始客户群编号 |
| PRIORITY | VARCHAR2(10) | 活动优先级（参照数据字典504） |
| EXECUTE\_TYPE | VARCHAR2(1) | 执行时间间隔：[每天 |
| MONTH | VARCHAR2(5) | 月 |
| WEEK | VARCHAR2(5) | 周 |
| DAY | VARCHAR2(5) | 日 |
| HOUR | VARCHAR2(5) | 时 |
| MINUTE | VARCHAR2(5) | 分 |
| CREATE\_PERIOD | VARCHAR2(1) | 建表周期[2：每月(固定)] |
| SOURCE\_TYPE | VARCHAR2(5) | 数据来源类型（1:场景、2:模型、3:标签、4:导入）详见字典表501 |
| SOURCE\_ID | VARCHAR2(255) | 数据来源ID |
| UPDATE\_DATE | DATE | 数据更新时间 |
| CLIENT\_COUNT | NUMBER | 客户数 |
| START\_DATE | DATE | 开始日期 |
| END\_DATE | DATE | 结束日期 |
| USER\_ID | VARCHAR2(255) | 创建人 |
| CREATE\_TIME | DATE | 创建时间 |
| AUDITOR\_USER | VARCHAR2(255) | 审核人 |
| AUDITOR\_TIME | DATE | 审核时间 |
| ORG\_ID | VARCHAR2(255) | 所属组织机构 |
| ENABLED | VARCHAR2(1) | 状态（1 |
| STATUS | VARCHAR2(10) | 业务状态（0：驳回待处理、1：创建活动、2：选择数据源、3：获取目标客户、4：选择营销策略，5：渠道配置、6：待审核、7：已审核；8：营销执行、9：营销结束 |
| REMARKS | NVARCHAR2(2000) | 备注 |
| AUDITOR\_DESC | LONG | 审核意见 |
| LABEL\_ID | VARCHAR2(255) | 标签基础表Id |
| ACTIVITY\_CATEGORY | VARCHAR2(1) | 活动类别（0：普通活动；1：实时活动） |
| LOOP | VARCHAR2(255) | 循环频率 |
| UNIT | VARCHAR2(255) | 循环单位(1 |
| YEAR | VARCHAR2(5) | 年 |
| DATA\_TYPE | VARCHAR2(1) | 数据类型（0：活动；1：场景） |
| IS\_PUBLIC | VARCHAR2(1) | 0：私有；1：公共 |
| AUDITOR\_STATUS | VARCHAR2(1) | 审核状态（0：临时保存、1：提交待审核、2：已生效、4：退回、5：作废） |
| EXECUTE\_PROGRESS | NUMBER | 派单执行进度 |
| IF\_TEST | VARCHAR2(10) | T时代表是测试活动，为空时代表正式活动 |
| IF\_TEST | VARCHAR2(10) | T时代表是测试活动，为空时代表正式活动 |

#### 2. 实时营销活动厅店信息表

**存储位置：**

Oracle

**数据存储格式：**

**PL\_IM\_REALTIME\_****ACTIVITY\_HALL**

|  |  |  |
| --- | --- | --- |
| ACTIVITY\_ID | VARCHAR2(30) | 活动标识 |
| HALL\_ID | VARCHAR2(20) | 厅店ID |
| CELL\_ID | VARCHAR2(20) | 基站ID |

**数据说明：**

1个活动对应多个HALL\_ID，1个HALL\_ID对应多个CELL\_ID

#### 3. 实时营销活动配置表

**存储位置：**

Oracle

**数据存储格式：**

**PL\_IM\_REALTIME\_ACTIVITY\_CONF**

|  |  |  |
| --- | --- | --- |
| ACTIVITY\_ID | VARCHAR2(30) | 活动标识 |
| HALL\_RANGE | NUMBER | 厅店范围 |
| INTERVAL | NUMBER | 停留时间（单位：分钟） |
| IS\_SEND\_USER | NUMBER | 0:不发送,1:发送 |
| USER\_MSG\_ID | VARCHAR2(30) | 用户短信ID |
| IS\_SEND\_STAFF | NUMBER | 0:不发送,1:发送 |
| SEND\_STAFF\_TYPE | NUMBER | 1:厅店负责人,2:所有人 |
| STAFF\_MSG\_ID | VARCHAR2(30) | 用户短信ID |
| VALID\_TIME | NUMBER | 工单有效时间(分钟) |

#### 4. 实时营销活动用户信息表

**存储位置：**

Oracle

**数据存储格式：**

**PL\_IM\_REALTIME\_ACTIVITY\_USER\_INFO**

|  |  |  |
| --- | --- | --- |
| ACTIVITY\_ID | VARCHAR2(30) | 活动标识 |
| SERV\_ID | VARCHAR2(30) | 客户ID |
| HALL\_ID | VARCHAR2(30) | 用户推送的厅店ID |
| CELL\_ID | VARCHAR2(20) | 用户出现基站ID |
| PUSH | NUMBER | 0:未推送,1:已推送 |
| PUSH\_TIME | DATE | 推送时间 |

### 3.2.2 Kafka消息格式

**流程：**

Spark匹配到目标用户后，发布用户信息到Kafka，然后平台消费该条消息

**Topic名称：**

TargetUserTopic

**消息格式：（string）**

“活动ID,用户手机号,厅店ID, starttime”

### 3.2.3 Redis数据格式

**推送数据：**

活动ID 目标用户 基站 厅店

活动ID 停留时间 活动状态

**存储位置：**

Redis

**数据存储格式：**

（活动ID： 41132）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **内容** | **类型** | **hkey** | **格式示例** | **备注** |
| 活动信息 | Hash | activityInfo | (41132, "1,10"),(41133, "1,10")… |  |
| 活动用户 | Hash | 41132\_user | (user1,0), (user2,0), (user3,1)… |  |
| 活动基站 | Hash | 41132\_lac | (lac1, "厅店1,厅店2"),( lac2, "厅店2,厅店7,厅店11")… |  |

**数据格式说明：**

【活动信息】

Key 41132 活动ID

Value “1,10” 即status,interval

status

0 新增活动

1 正在执行，当spark活动读取该活动数据时，需要更新后活动状态

2 活动结束（平台侧更新）

3 活动失败（平台侧更新）

4 活动暂停（平台侧更新）

5 活动取消（平台侧更新）

interval

10 用户停留的时间间隔，单位（分钟）

【活动用户】

user1 用户手机号

1. 标识未推送该用户
2. 标识已推送

【活动基站】

lac1 基站编码

"厅店1,厅店2" 基站对应的厅店ID

### 3.2.4 Spark内存数据格式

**数据来源：**

Redis：活动信息、活动用户、活动基站

Kafka：信令数据

**数据格式：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **内容** | **类型** | **key** | **格式示例** | **备注** |
| 活动基站 | Map | lac1 | (lac1, "厅店1 活动1 停留时间")  (lac1, "厅店2 活动1 停留时间")  …… | Value中的值以空格分割 |
| Kafka信令数据 | Map | lac1 | (lac1, “user1 starttime”)  (lac1, “user2 starttime”) | Value中的值以空格分割 |
| 活动基站和信令join操作 | Map | 活动1+User | (活动1 user1,"厅店1 lac1 starttime 停留时间")  ……. | 组合的key |
| 保存的状态 | Map | 活动1+User | （活动1 user1，“厅店1 lac1 starttime 停留时间 累计滞留时间”） |  |
| Kafka写入 | String |  | （活动1 user1 厅店1 starttime） |  |

**数据格式说明：**

【活动基站】

Key：基站id

Value：从redis中读取有效的活动id，根据活动id得到基站对应的厅店，并加上对应的活动id和要求停留时间

【Kafka信令数据】

Key：基站id

Value：从kafka中获取信令数据，得到用户号码和开始时间

【活动基站和信令join操作】

Key：由活动id和user来确定唯一

Value：活动基站和信令数据的join操作可以匹配到目标用户对应的厅店id和活动id，

【保存的状态】

Key：由活动id和user来确定唯一

Value：同一用户的厅店id和基站id是可变的，每次得到的数据都和保存的状态数据去匹配，如果不同，则更新value的值。当累计停留时间大于条件时，输出这条记录

【Kafka写入】

得到数据后以string格式向kafka中写入，平台得到数据后进行消息推送

## 3.3 集成方案

### 3.3.1 硬件配置需求分析

**Redis集群**

Redi负责存储实施营销活动相关的用户数据、基站数据、厅店数据等，这些数据除了在内存中，同时需要持久化到磁盘，根据Redis自身特性和网络带宽等因素考虑，峰值并发可达5W条/秒。

* 服务器数量

3台，做集群，同时提供高可靠和高并发（HA）

* 峰值并发：5W条/秒
* 目标用户：2000W
* 目标基站：200W
* 活动信息：10W
* 则Redis服务器要求内存为：   
  2000W \* 50B ~= 1GB 用户数据

200W \* 100B ~= 200M 基站数据

10W \* 100B ~= 10M 活动数据

总内存：

1GB + 100M + 10M ~= 1.2GB

* Redis预留40%空闲内存，系统本身要消耗30%的系统资源，则Redis服务占用内存为：   
  1.2GB/60%/70% ~=3GB
* 服务器的实际资源占用即“系统忙”不应大于70%，则实际要求内存为：3GB / 70% ~= 4.5GB
* 磁盘资源：40GB

考虑满足1 年内业务系统的服务能力，因此建议Redis集群3台服务器每台器配置为4核服务器：4CPU、6GB内存、40GB磁盘。

**Spark**

Spark负责实时通过Kafka接收基站数据，然后跟活动相关数据进行匹配，然后将目标用户通过Kafka推送到智能营销平台。

* Executor数量：20个
* 单Executor内存：1GB
* 目标用户：2000W
* 目标基站：200W
* 活动信息：10W
* 则Spark应用要求内存为：   
  20 \* 1GB = 20GB

考虑满足1 年内业务系统的服务能力，因此建议Spark集群能够提供20GB的内存空间以供计算使用。

### 3.3.2 网络拓扑

