业务理解:

* 一些名词:

广告主:简单来说就是我们的顾客

第三方:是介于我们公司和广告主之间的监测机构

归因周期:一个安装从点击到安装的时间

最大归因周期:同一个产品归因周期的最大值

* 公司业务

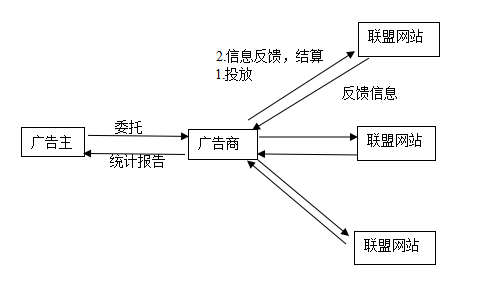
简单来说，我们公司的业务就是将广告主的委托投放到许多媒体上来展示，我们付给媒体一定的费用，以此来赚取差价。(我们公司的角色就是广告商)

* 付费方式

基本上分为:展示付费，安装付费，交易付费，点击付费

* 广告中的信息不对称

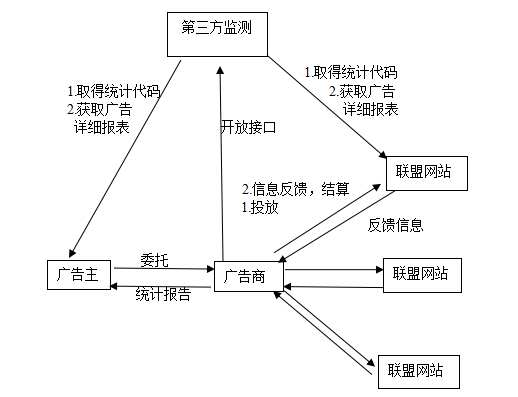
较早的广告业务是由广告主委托广告商投放广告，然后广告商将数据(安装数，点击数等)反馈给广告主。



这样就会造成许多问题:从广告主角度来看，广告商给出的统计报告只会有点击数，点击时间等，并没有投放的效果，也不知道有没有恶意刷量；从联盟网站来看，广告商给出的信息反馈也不一定准确，有可能会刻意扣除掉一些点击，来提高广告商自身的收益。

* 引入第三方

由于广告行业中有太多的不对称的信息，所以引入了一个较为公正的平台:第三方。第三方来监测广告投放之后的信息，这个来说相对公正，公平。



* 安装付费带来的问题

当一个广告被投放，用户看到了投放的广告，有些用户会点击这些广告，点击之后就会下载app，然后还有一些用户会选择安装app，这样就为我们公司带来了利益。

这里面会存在一个问题: 有些用户当时点击了却没有安装，后来才去下载了app，这样就没有办法判断这个安装是哪个点击带来的，也没有办法判断这个安装时是个广告商带来的。

这样就引入了一个归因方式，一个用户安装了app，第三方会判断出这个用户安装的app的最后一次点击是哪一个，把这个安装归到最后一个点击上面，点击是哪个网站发出的是可以知道的，这个安装给哪个广告商也就可以知道了。所以会有归因周期的说法。

注:归因周期最大为30天，如果一个安装的点击在30天以外，那么第三方就不会匹配。

不同的单子(app)会有不同的归因周期

掐量系统理解:

* 名词:

CVR:安装数目/点击数目(点击转化成安装的比率)

* 什么是尾量:

如果广告主停止了广告的投放，广告主的理想状态是停止，就没有安装了。但实际上用户还是会安装，第三方还是会做归因来匹配点击，这样匹配的点击时间在停止投放之间，安装时间在停止时间之后的安装叫做尾量。

* 掐量:

由于会有尾量，广告主一定不希望产生尾量，所以我们会想办法去掉尾量，也就是掐量。

* 如何进行掐量:

第三方会把一个安装归因到一个点击上面，如果说我们在安装之前给第三方发送一个点击，并且让第三方不能通过点击来找到我们公司，那么这个安装也就不能归因到我们公司。我们通过这种方式来进行掐量。

* 掐量点击:

我们执行掐量所发送的点击。掐量的点击发送的数量是一个问题：发送的过多，公司收入就会减少，并且可能有被第三方察觉的风险；发送的少，掐量效果不好。

现在我们的掐量系统是根据单子的最大归因周期来发出点击的。

理论上来说，如果一个单子停掉，那么停止时间到停止时间之前的一个最大归因周期的点击都有可能产生尾量(有安装并且第三方匹配到了这个点击就产生尾量),所以我们就会发送点击‘顶掉’这段时间的点击，以防止尾量的产生。

* 如何评估掐量效果:

当一个单子做了掐量之后，我们需要知道它做的怎么样，是否有效，所以我们要评估掐量的成功率(掐掉的安装/预估正常的尾量数目)。

首先尾量的数目我们是知道的，那么如果我们知道假如不掐量的话安装是多少，我们就可以评估成功率，所以评估掐量效果，主要就是预测如果不掐量的话安装有多少。

一个单子停掉，那么停止时间到停止时间之前的一个最大归因周期的点击都有可能产生尾量，我们知道这段时间的点击数目，那么通过CVR就可以算出会产生多少安装，再通过CTI就可以算出这些安装有哪些是在停止时间之后，就能够算出不掐量的尾量。

* 如何算出CVR和CTI(CTI定义下文有):

现在问题已经转化成算出单子的CVR的CTI。

理论上来说，我们可以取单子的停止时间的前两个最大归因周期到前一个最大归因周期来算出单子的CVR和CTI。因为这段时间的安装和点击都是正常的(我们没有进行掐量，所以算出CVR相对准确)，并且安装和点击的数目我们都知道。

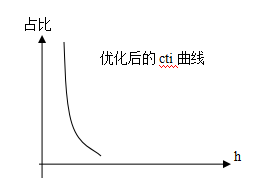
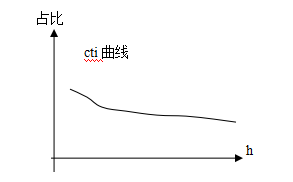
大部分单子可以这样求出，但是有些单子前两个最大归因周期到前一个最大归因周期并没有安装和点击，现在的策略是直接用现有的安装和点击算出CVR，再求一个渠道的总体CVR和CTI,用较大的CVR。

但是这样算出的CVR总是会有问题，希望你可以想一个更好的办法...

CTI优化系统理解:

* CTI曲线:反应一个单子归因周期占比的曲线(安装时间-点击时间)

理论上来说安装时间-点击时间在1小时之内的占比是最高的(因为大部分人都是点击没多久就安装了)。安装时间和点击时间之差越大，占比越小。



* CTI目标:

有些广告主希望CTI曲线的占比尽量在24小时内达到100%(安装时间和点击时间之差在24小时之内)，所以会设定一个目标首小时占比多少，24小时占比多少。

* CTI优化:

有了CTI目标，我们要做的就是优化CTI曲线使CTI曲线达到广告主的要求值。

* 如何优化CTI曲线:

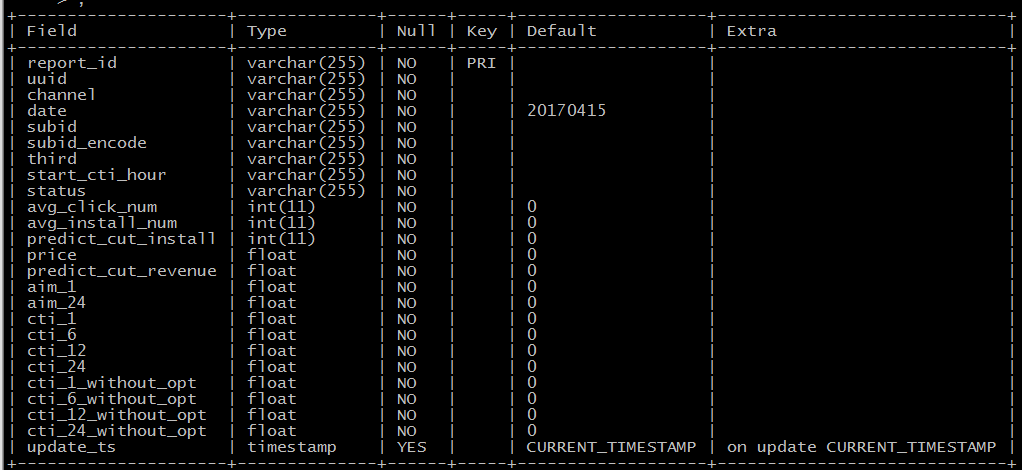
现在CTI曲线的优化方式主要有三种:掐量，去devid和appsflyer参数控制

数据库表及字段说明:

* cti\_report表:

对应脚本: cti\_report\_v2.py

主要存放cti优化单子的相关的信息



* report\_id:用uuid, channel, date生成的主键
* uuid:单子名称
* channel:渠道
* date: 日期
* subid:子渠道 现在此表用不到，都设置为 all
* subid\_encode:子渠道 现在此表用不到，都设置为 all
* third:第三方
* start\_cti\_hour: 单子开始执行cti优化的时间
* status: 三种状态 overdue代表单子昨天优化单子今天不优化了 running代表单子正在优化

new 代表单子没有在优化(不需要优化或者不支持优化)

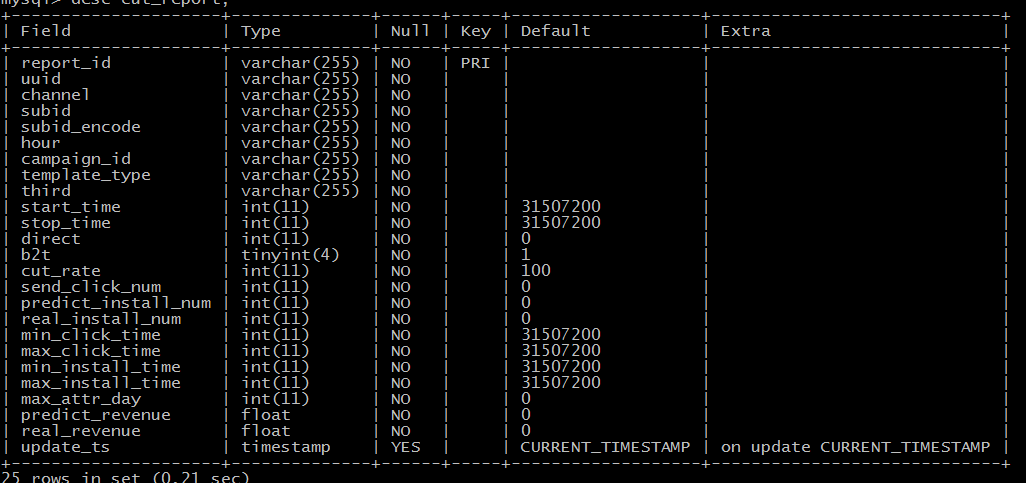
* avg\_click\_num: 平均每天点击数目
* avg\_install\_num: 平均每天安装数目
* predict\_cut\_install: 预计平均每天掐掉安装
* price: 单子单价
* predict\_cut\_revenue: 预计平均每天掐掉的收益
* aim\_1: 单子优化1小时的占比目标
* aim\_24: 单子优化24小时的占比目标
* cti\_1: 优化后1小时占比
* cti\_6: 优化后6小时占比
* cti\_12: 优化后12小时占比
* cti\_24: 优化后24小时占比
* cti\_1\_without\_opt: 优化前(没有优化)1小时占比
* cti\_6\_without\_opt: 优化前(没有优化)6小时占比
* cti\_12\_without\_opt: 优化前(没有优化)12小时占比
* cti\_24\_without\_opt: 优化前(没有优化)24小时占比
* update\_ts: 更新时间

注:这里的cti和平均都是用单子前7天安装和点击求出。

* cut\_report表:

对应脚本: cut\_report\_v4.py

主要存放掐量单子的相关的信息

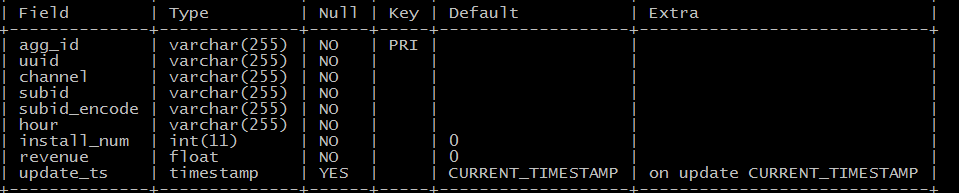


* report\_id:用uuid, channel, subid, subid\_encode, stop\_time, hour生成的主键
* uuid:单子名称
* channel:渠道
* subid:子渠道
* subid\_encode:子渠道
* hour: 时间(小时级)
* campaign\_id: uuid+channel的单子名称
* template\_type: 掐量方式 SUBID或SAMEPKG
* third: 第三方
* start\_time: 单子开始跑的时间
* stop\_time: 单子停止时间
* direct: 是否是直客单(暂时没有用)
* b2t: b2t占比 字段有错误
* cut\_rate: 掐量比例都是100%
* send\_click\_num: 发送点击的数目 字段未填充
* predict\_install\_num: 预测尾量
* real\_install\_num: 真实尾量
* min\_click\_time: 单子点击的最小时间 字段未填充
* max\_click\_time: 单子点击的最大时间 字段未填充
* min\_install\_time: 单子安装的最小时间 字段未填充
* max\_install\_time: 单子安装的最大时间 字段未填充
* max\_attr\_day:最大归因周期
* predict\_revenue: 预测尾量的收益(一小时安装的总收益，不是单价)
* real\_revenue: 真实尾量的收益(一小时安装的总收益，不是单价)
* update\_ts: 更新时间
* real\_tail\_install\_hourly表:

这个表是从SSSPortal和cut\_history中匹配取得的单子，SSSPortal的单子同时也在cut\_history中

对应脚本: real\_tail\_installs\_v3.py

主要存放掐量之后单子尾量的相关信息

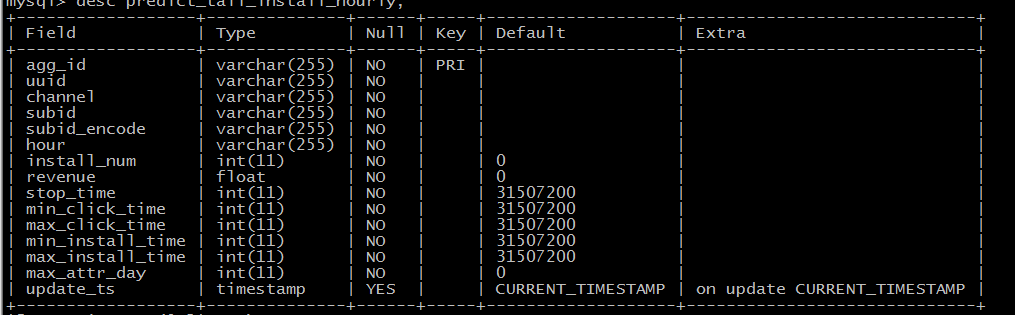


* agg\_id: 用uuid, channel, subid, subid\_encode, hour 生成的主键
* uuid:单子名称
* channel:渠道
* subid:子渠道
* subid\_encode:子渠道
* hour: 时间(小时级)
* install\_num: 真实尾量数目
* revenue: 真实尾量的收益(一小时安装的总收益，不是单价)
* update\_ts: 更新时间
* predict\_tail\_install\_hourly表:

从trigger\_history和cut\_history中匹配取得单子，trigger\_history的单子同时也在cut\_history中

对应脚本: predict\_tail\_install\_cti\_v3.py

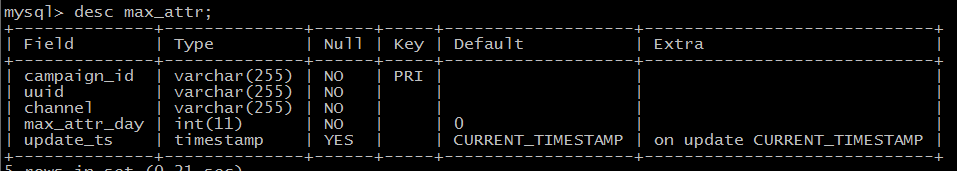
主要存放掐量之后单子预测的尾量的相关信息



* agg\_id: 用uuid, channel, subid, subid\_encode, hour 生成的主键
* uuid:单子名称
* channel:渠道
* subid:子渠道
* subid\_encode:子渠道
* hour: 时间(小时级)
* install\_num: 预测的尾量数目
* revenue: 预测的尾量单(注意:是单价)
* stop\_time: 单子停止时间
* min\_click\_time: 单子点击的最小时间 字段未填充
* max\_click\_time: 单子点击的最大时间 字段未填充
* min\_install\_time: 单子安装的最小时间 字段未填充
* max\_install\_time: 单子安装的最大时间 字段未填充
* max\_attr\_day:最大归因周期
* update\_ts: 更新时间
* max\_attr表:

对应脚本: real\_tail\_installs\_v3.py

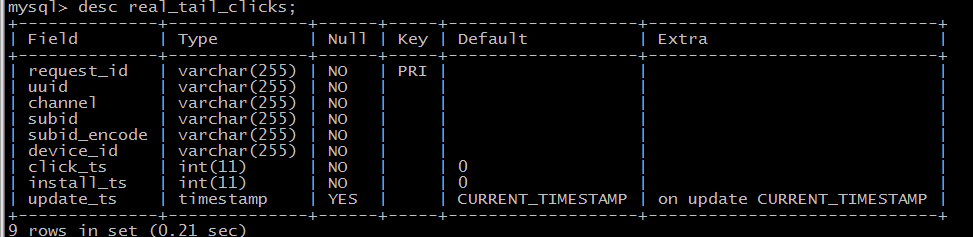
主要存放单子的最大归因周期



* campaign\_id: uuid+channel的单子名称(此表主键)
* uuid:单子名称
* channel:渠道
* max\_attr\_day: 最大归因周期
* update\_ts: 更新时间
* real\_tail\_clicks表:

对应脚本: real\_tail\_installs\_v3.py

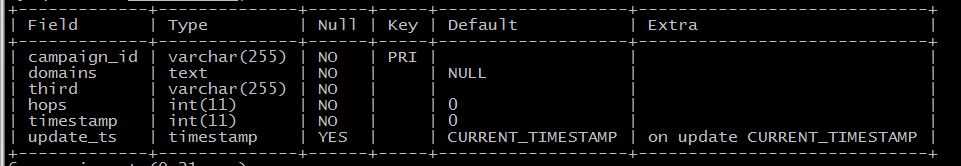
主要存放尾量点击的device\_id



* request\_id: 点击的标识id(此表主键)
* channel:渠道
* subid:子渠道
* subid\_encode:子渠道
* device\_id: 点击的一段标识id
* click\_ts: 安装的点击时间
* install\_ts: 安装时间
* update\_ts: 更新时间
* detect\_main表:

对应脚本: detect\_third\_hops.py

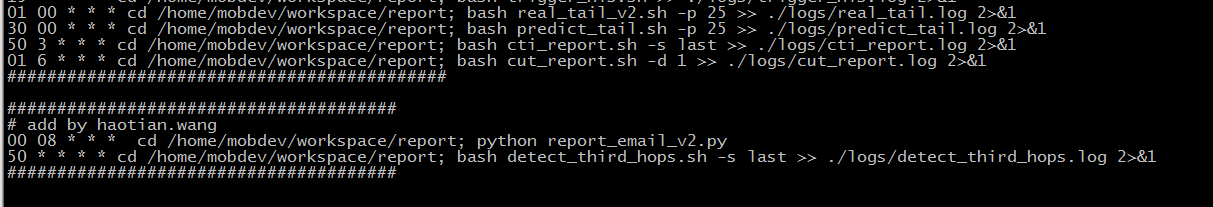
主要存放单子的第三方和跳转次数



* campaign\_id: uuid+channel的单子名称(此表主键)
* domains:探测的点击跳转信息
* third: 第三方
* hops: 我们域名到第三方跳转次数(为1是直客单，2是二手单,以此类推...)
* timestamp: 探测的时间戳
* update\_ts: 更新时间

重要脚本:

crontab上的脚本及日志文件



重要脚本请查看 重要脚本.xlsx

注:所有脚本均在git上面有保存

每天需要做的事情及可能遇到的问题处理:

* 每天需要做的事

1. azkaban上失败任务的重跑
2. 查看cti优化的情况并找出优化情况比较差的单子(相关sql语句已经在txt文本里)
3. 查看掐量成功率，找出成功率低的单子，并排查原因(相关sql语句已经在txt文本里)
4. 每周需要出一份cti周报和掐量周报(相关sql语句已经在txt文本里，大部分的数据已经在每天的邮件里给出)

* 可能遇到的问题处理

问题

如果当天CTI优化和掐量情况邮件没有按时发出, 原因基本为cti\_report\_v2.py 执行失败(可以查看cti\_report.log日志文件确认)，失败原因是: /home/mobdev/workspace/

cti\_optimization/conf里当天凌晨2点优化目标数据生成延迟或没有生成。

处理

若当天2点数据不存在，找一个存在的最接近2点的数据执行, 例如6月2日发生此状况则执行python cti\_report\_v2.py 2017060203 prod (填充的是6月1日的数据)

若当天2点数据存在，则执行python cti\_report\_v2.py 2017060202 prod (填充的是6月1日的数据)

注:如果发现当天cti优化情况单子数目不对，也可以查看cti\_report.log文件，确认是否有问题，有问题就再执行一遍cti\_report\_v2.py脚本

2.

问题

查看一个单子是否能执行掐量

处理

可以用detect\_main查看单子third和hops 来判断是否是我们支持的第三方和是否是直客单；也可以用thirduuid脚本查看；或者用vta机器上(52.55.78.75)的template2脚本查看是否有模板

3.

问题

单子的掐量成功率较低(成功率=(预测尾量-真实尾量)/预测尾量)

分析

a)如果单子的channel是mobvista\_adn 可以查看单子的b2t占比，如果单子b2t占比较低，成功率低是正常情况。

b)如果这个单子执行掐量的时候，azkaban任务延迟或者失败数据没有补上，则当天的成功率会略低。

c)预测尾量过低，可以把predict\_tail\_install\_v3.py的第600行replace2db注释掉，再用脚本预测一下成功率低单子的尾量，如果输出没有“LOG: installs total get:”字样那么说明我们不能拿到单子准确的cvr。出现这种情况可以重新执行脚本把这个单子的最新结果写入到数据库中。

d)有可能是点击发送失败，到click\_status\_agg\_hourly表里面查看点击是否发出。

e)如果点击已经发出用analysis\_real\_install.py 脚本分析尾量，看是否是我们发出的点击而导致了尾量的形成。

4.

问题

某天总体掐量成功率变低

分析

a)有可能是某个单子尾量非常多(单子cvr上升或者跑了激励)，可以把这个单子从数据报表里去除，并加以说明。

b)可以查看predict\_tail.log日志文件看predict\_tail\_install\_cti\_v3.py脚本的执行是否出现了问题,如果出现问题，重跑脚本。

c)查看成功率较低的单子

接下来要做的事情:

* 掐量&CTI优化覆盖面的统计(CTI优化比较紧急)
* 背景介绍:

现在我们的掐量和CTI优化系统并不能支持所有单子，例如二手单和不支持的第三方我们就不支持。现在我们想要知道不支持的这批单子是否有支持的价值(也就是说收益高不高)，所以需要统计单子的覆盖面。

* 统计要求:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3s offer | | | adn offer | | |
| M | | dsp | M | | dsp |
| rba | b2t | rta | rba | b2t | rta |

一、尾量

1. 每日自然尾量 。从最终业务效果，综合评估覆盖面&成功率(去devid对尾量的正向影响也会被包含进来)

自然尾量，粗略的认为渠道维度 当日click=0 && install!=0。

2. 掐量覆盖面

可直接统计掐量可覆盖、不可覆盖(仅判断是否可覆盖，不一定实际触发了掐量； 去devid、appsflyer控制覆盖的也可以算成一种掐量覆盖，打个独立的标签能区分就行)的 offer数 以及 自然安装 revenue等。

3. 掐量本身指标

如点击量、成功率、预计损失等，这些现在已经有了。

二、cti

同样的思路，分整体业务结果、覆盖面、达成率来看。

注意引入adn bt，刨掉被3s bt的安装。

1. 整体install 1h 24h占比。

2. offer数、revenue覆盖面

3. 覆盖offer的达成率(目前也有了)、总体1h、24h比例。

* 统计方法(以CTI为例)(如果想到更好的方法忽略这个):

1. 从adn拿到安装信息(单价，点击时间，安装时间等)
2. 判断属于哪个系统(rba, rta, b2t)，打上标签
3. 判断是否有CTI目标，打上标签，如果有目标，写进数据库
4. 如果没有判断我们是否能够覆盖(掐量，去devid, appsflyer参数控制)，打上标签，写进数据库

Compaign\_id和subid生成key