一. 背景 对于代理商而言，生存与否很大程度上取决于广告主对自己的需求程度。除了提供基础的业务服务（创建广告，管理广告，观察统计），我们也在思考，还有什么东西是我们可以提供而Facebook不能或者不屑于去做的？自管理系统的诞生背景就来自于leader对于这个问题的设想：如何能够帮广告主节省管理广告的时间，让广告主只设定宏观的优化目标，而让机器自动的去调整广告状态（价格，预算，定向，开关）来达到这一宏观状态。

二. 问题 1.相对而言，自管理系统的任务目标是极其抽象的，因为不同于广告管理的服务，后者的输入输出是很明确的，广告管理服务就是接受用户的设定作为输入，在Facebook上创建完广告作为输出，当广告创建完成后，它的工作就已经完成了，它是一次性的（用大家更熟悉的web服务来描述，一次http request/response结束后，对client的服务就结束了，后续的任务从模式上是之前的重复），但对自管理系统而言，广告主只是设定了宏观的成本（CPA）目标，而系统要时刻根据需要去调整接近这一目标，只要这个广告还在投放，管理就要持续，它不是一次性的操作，在程序上怎么去做实现？ 2.Facebook广告按包含层次分为Campaign/Adset/Ad，为了让广告主宏观控制和节省时间，我们希望广告主的设定只发生在Campaign级别，而一个Campaign内可能有多个Adset和Ad，如何把Campaign级别的目标设定转换到Adset/Ad级别的操作？（在我们的系统里，Adset和Ad在创建时就保持了1to1关系，因此后文Adset/Ad部分用Adset替代） 3.广告投放对质和量都是有要求的，如果单纯的控制成本（比如通过极低的出价）而缺乏量级，广告主同样不会接受，因此自管理系统需要一套变化而非固定的参数（价格、量级或其它）设置，才可能提供广告主需要的效果。 综上所述，自管理系统在最开始设计时面临的问题，既有技术实现问题，也有业务问题。 另外比较麻烦的是，因为要解决问题本身的特点，不同于如何优化Django Server，或者如何优化MySQL访问这样明确的技术问题，我们不太可能从网上找到直接参考的资料。

三. 问题解决 1.这是系统最基本的问题，相当于如何搭架子。我们想到，或许系统最终的目标是超过人，但最开始要先做到模拟人的行为。设想系统自己是一个广告优化师，它应该根据当前的情况，决定对广告下一步的操作。这很像是机器人的行为（当然真实机器人问题要复杂更多），所以我们也得以借鉴机器人问题里的方案：状态机。 每个进入自管理系统的campaign都相当于被一个机器人管理，在系统里我们把它称为worker，当然对系统而言，因为有多个campaign所以需要有多个worker，但这不是太大的问题，worker之间是独立工作的，因为campaign彼此的管理是独立的，因此我们主要要解决worker该干什么。 先不考虑后面的问题，按状态机的设计，每个worker都会有不同的状态（status），我们要定义两点：

1. 指定status下，worker的行为  
2. status之间的转换条件

只要这两点定义完成，我们的系统就可以处于自运行状态，因为每个worker或者处于某个status，或者在某些条件下进行status的跳转。 状态机的实现，我们参考了设计模式中的State模式完成。

2.有了状态模式，我们解决了最基础的问题，但worker是一个campaign级别的概念，而最终的广告管理却要发生在adset/ad级别，因此直接在worker里使用状态模式是不合理的：在同一个campaign下的不同adset很可能有不同的表现，因而应该对应采取不同的行为。 为此我们定义了一个介于campaign和adset中间的概念：group，它包含了一组应该被近似视为一致的adset统一管理，status作用在group上，具体campaign, adset和worker, group, status的关系如下：

worker和campaign是一对一的  
每个campaign有一个或多个adset  
每个campaing有至少一个group  
每个group同一时刻有且只有一个status  
每个adset属于且只属于一个group。

借用group的概念，我们把adset根据投放情况分组，按组级别进行状态管理。

3.投放效果的问题，最开始和最重要的做法就是，向广告优化师了解和学习。他们是最直接接触业务的人，也可以说是投放经验最丰富的人，也就是说，他们是决定“什么条件”下该对广告“做什么事”的人。这中间的问题在于，由于广告优化师不是技术人员，不可能用技术的语言去描述自己的做法，有些做法甚至是偏感性的，我们要做的就是把他们的描述翻译成技术的语言，最终转化成可以执行的代码逻辑。 更多一点要说明的是，我们希望自管理系统可以提供人不能提供的功能，这种不能提供并不一定是人不能理解，而是人力无法完成：举个例子，在半夜时根据投放成本及时关停广告，人不能24小时盯着广告，但机器可以。

四. 系统架构 根据上述问题的解决设想，我们设计了自管理系统的框架：

enter image description here

enter image description here

概念含义：

1.controller：系统总体的控制类，因为我们有很多campaign需要管理，而每个campaign都需要一个对应的worker，因此可以将controller视为workers的容器，它定时调度，启动每个worker的操作。  
同时，controller还会复杂一些公有模块的加载和维护工作，比如离线模型中读取的信息。  
2.worker：负责campaign管理，每个worker包含一个campaign的信息，它调度groups进行状态转换  
3.group/status：每个group包含一组adsets，处于一个status，在指定status下，group下的adset将有特定的行为，我们最终要根据业务实现的策略，也就是对status行为的描述和status跳转的描述  
4.adsets operation：最终广告管理的行为将转为Facebook API的调用作用于Facebook，同时会向db做写入的同步。

图中右侧描述了新campaign被添加到自管理系统的流程：

worker, group, status将被依次创建和初始化，最终以worker的形式被加入到controller中。

为了提高效率，我们做了如下设计： 1.没有让每次controller操作均读取数据库，而是在内存里维护一个campaign:worker的字典，只有在服务重启时，才会从数据库读取相关设置去重建字典，其它时间，controller只会根据用户操作（添加campaign，删除campaign）去更新字典；数据同步上，我们也只会同步adset和group的部分字段；通过这样的设计，我们避免不必要的读写操作 2.不同worker之间的运行是多线程调用 3.新campaign通过消息队列添加到controller

在完成宏观的设计后，我们可以把问题集中到每个group的status管理上，这也是业务逻辑发挥作用的地方：

1. 指定status下，group的行为  
2. status之间的转换条件

不同的状态组合可以被视为不同的策略，下面是我们的一个策略转换关系：

enter image description here

enter image description here

因为这是一篇关于自管理系统设计的博客，我们不打算解释每个策略名词的细节，但希望强调的是，用户可以非常方便的定义自己的策略，或者在已有策略里添加新的逻辑，只需要完成上面提到的两点内容。在实际应用里，这种框架设计对我们后续策略添加的效率上也提供了很大的帮助。

五. 策略准则 这里我们希望讨论的是在指定状态下，可能用到的影响用户行为的方法，把它们分为以下几类：

1.基于规则的：满足某条件下，进行某操作，例如：今天CPA高于广告主设定目标，且花费大于N时，关闭adset  
2.基于时间的：与时间有关的，例如：当投放时间为凌晨时，暂停广告  
3.基于变化的：使用增量信息，这是人无法观察到的信息，反馈调整，例如：自上次调整出价后，花费M后成本高于目标，调低出价，反之，提高出价  
4.基于离线信息的：读取离线数据，例如：某个creative在历史投放上表现较差，降低对它的容限

在不同状态下，策略可以视为对上述几类行为的组合，当然最终来说，它们还是来自广告优化师实际投放的案例。

六.系统演进 演进从技术和业务两方面划分，在业务方面，主要是新策略的添加，在技术方面，主要包括下述两方面：

1.增加离线策略  
2.增加实验对比

离线策略 最早期的自管理系统是基于状态机实现的纯在线系统，每个worker只会使用到它自己管理adsets的数据信息，而不会被其它worker所影响。但在实际问题中，比如广告素材的好坏，在不同campaign定向上也是有相通性的，如果已知一个创意较差或者较好，在新campaign投放上也可以采用不同的出价策略。为此，我们开发了一系列离线策略，它们定期执行，将结果保存到文件，再交由controller里新增的reader读取，作为在线调整的依据。

实验对比 实验对比主要有两方面作用：新功能的小流量测试和A/B测试。我们增加了实验参数的配置，可以使得指定代码逻辑只应用在部分campaign或部分account上。 但这里需要说明的是，在代理商级别，Facebook广告没有办法做严格的A/B测试，因为不能保证用户只看到使用A策略或B策略的广告。

实现这两部分功能，自管理系统框架添加了如下模块: enter image description here

七.结语 上面是我们关于自管理系统功能设计的介绍。实际应用上，更多的问题来源于策略，因为理想策略在实践中往往有未曾预料的负效应，比如反馈调整出价的设计，过于频繁的价格调整可能会引起Facebook内部广告排序的负向权重，这些问题，我们在不断总结和改进之中。 欢迎关于策略方面的讨论，在将来我们也可能对策略做更深入的讨论。