ArchSummit
全球架构师峰会 2017

主办方 Geekbang 极客邦科技 InfoQ

优酷广告投放引擎优化实践

张云锋
优酷广告系统中心高级技术经理

SPEAKER
INTRODUCE

张云锋

优酷广告系统中心高级技术经理

2008年入职优酷，经历了优酷广告平台从小到大发展演进的整个过程，主导了优酷广告引擎的多次大型重构项目，并围绕引擎推出了广告诊断系统ad-doctor、广告缓存推荐系统ADP等一些辅助系统，目前主要致力于广告引擎架构优化、性能优化、广告投放异常诊断自动化等方面的工作，并在优化实践工作中取得了不错的收益。

TABLE OF

CONTENTS 大纲

优酷广告业务简介

优酷广告系统架构

广告引擎优化实践

- 广告链路优化
- 引擎性能优化
- 用户体验优化

优酷广告业务简介

- 投放媒体：优酷、土豆、阿里数娱、闲鱼、虾米、UC
- 广告类型：前贴、中插、后贴、暂停、角标、开屏、信息流、播放页banner、常规页面等数十种广告类型，以贴片广告为主。
- 业务规模：日曝光量数十亿。
- 收入构成：以品牌广告为主，效果广告为辅。

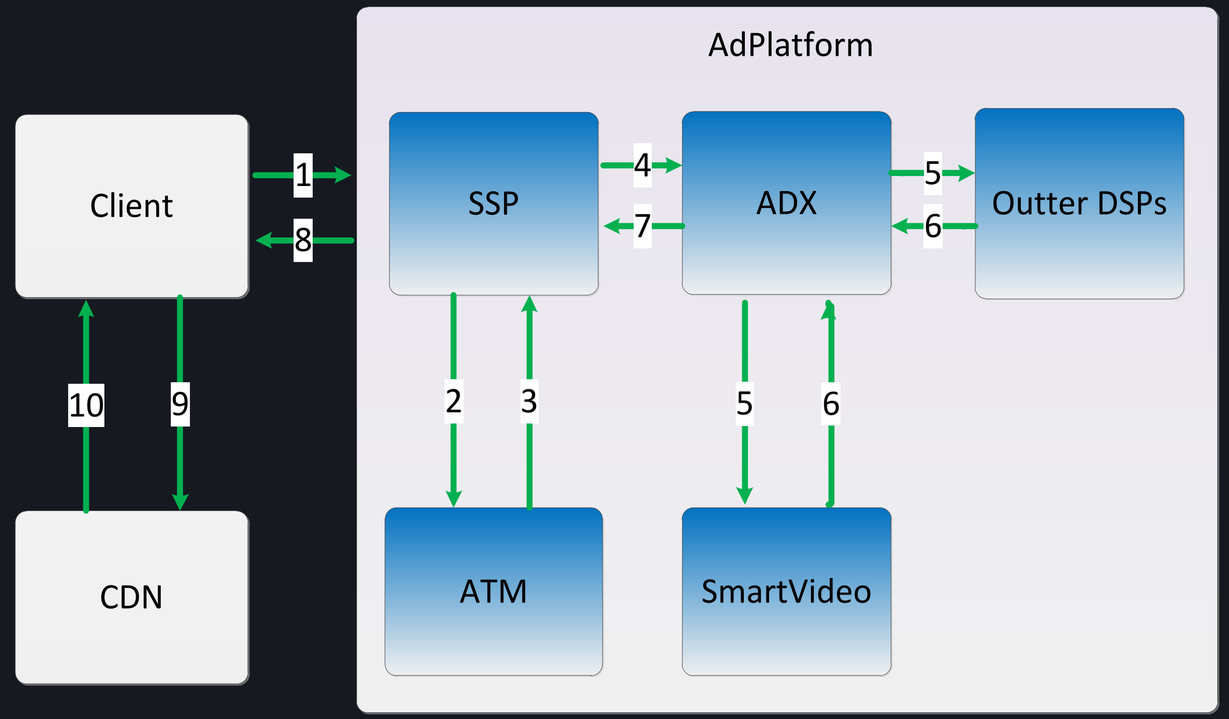
优酷广告业务特点

- 售卖方式：CPM、CPC、CPD，等，以CPM为主
- 定向方式：用户（地区、人群标签等），内容（视频组、视频时长、频道、关键字、清晰度等）、客户端（设备类型、操作系统、客户端类型、版本号等），共数十种定向方式
- 频控方式：整周期N次频控、每M天（小时）N次频控、多屏打通频控
- 其它特点：广告时长、个数限制

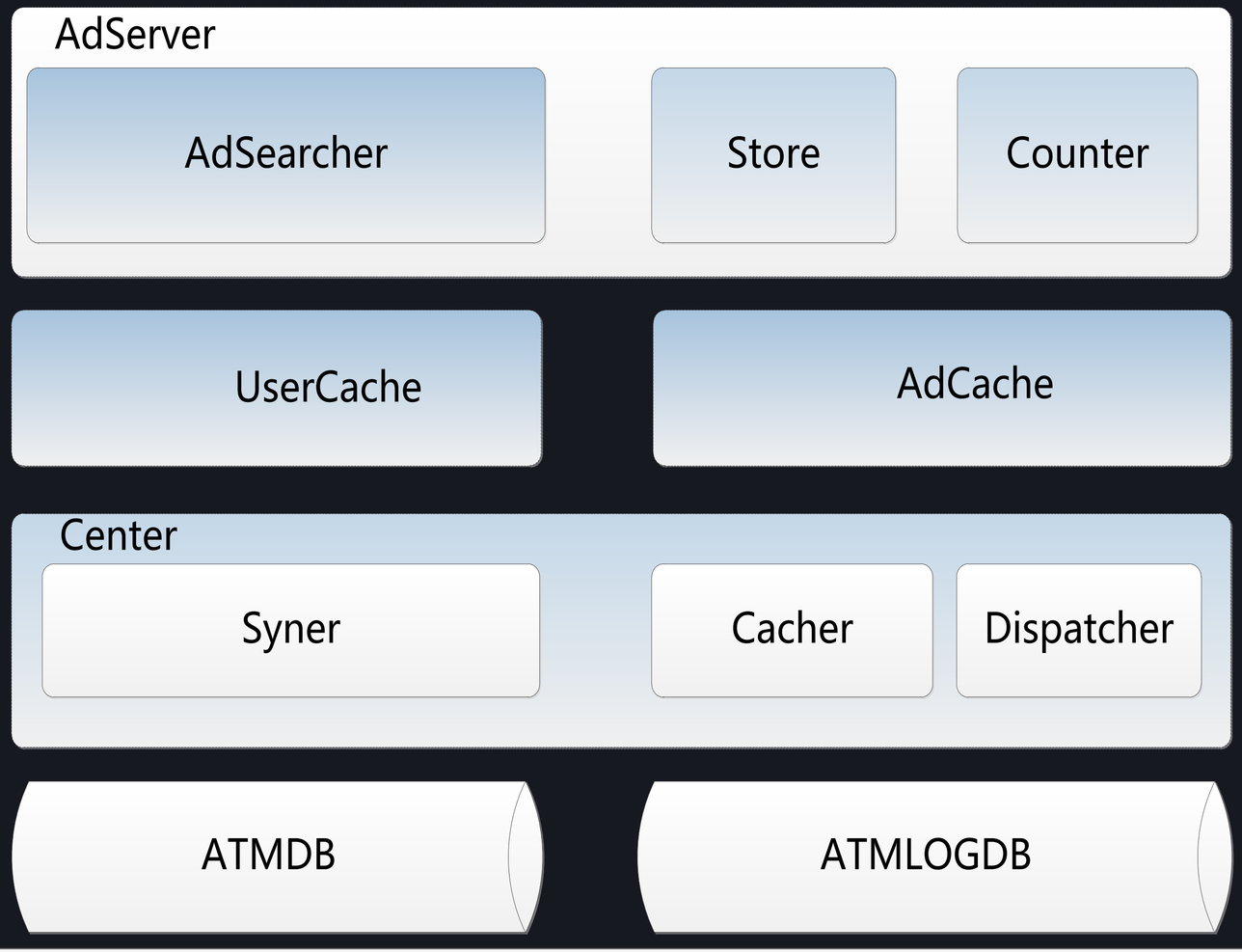
优酷广告系统架构

系统架构说明：

- SSP ：优酷SSP（ Sell-Side Platform，供应方平台）
- ADX：优酷Ad Exchange
- ATM：优酷品牌广告引擎
- SmartVideo：优酷效果广告引擎（InnerDSP）
- OutterDSPs：对接优酷ADX的外部DSP
- Client：客户端，直接发送广告请求或通过其后台服务间接发送广告请求
- CDN： 内容分发网络，视频正片和广告素材的存储服务



ATM系统架构


AdServer
AdSearcher
Store
Counter
UserCache
AdCache
Center
Syner
Cacher
Dispatcher
ATMDB
ATMLOGDB

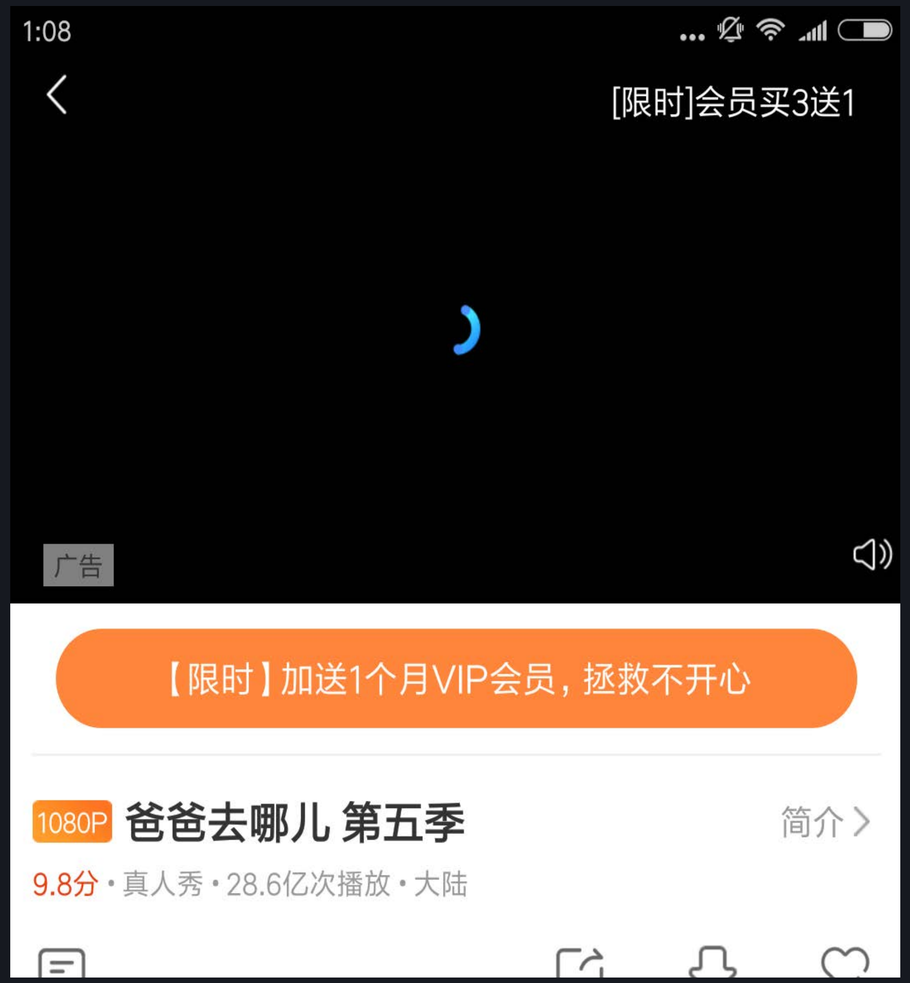
广告引擎优化实践 – 视频广告常见问题

问题描述：

- 广告加载太慢，播放窗口小圆圈一直转

优化思路：

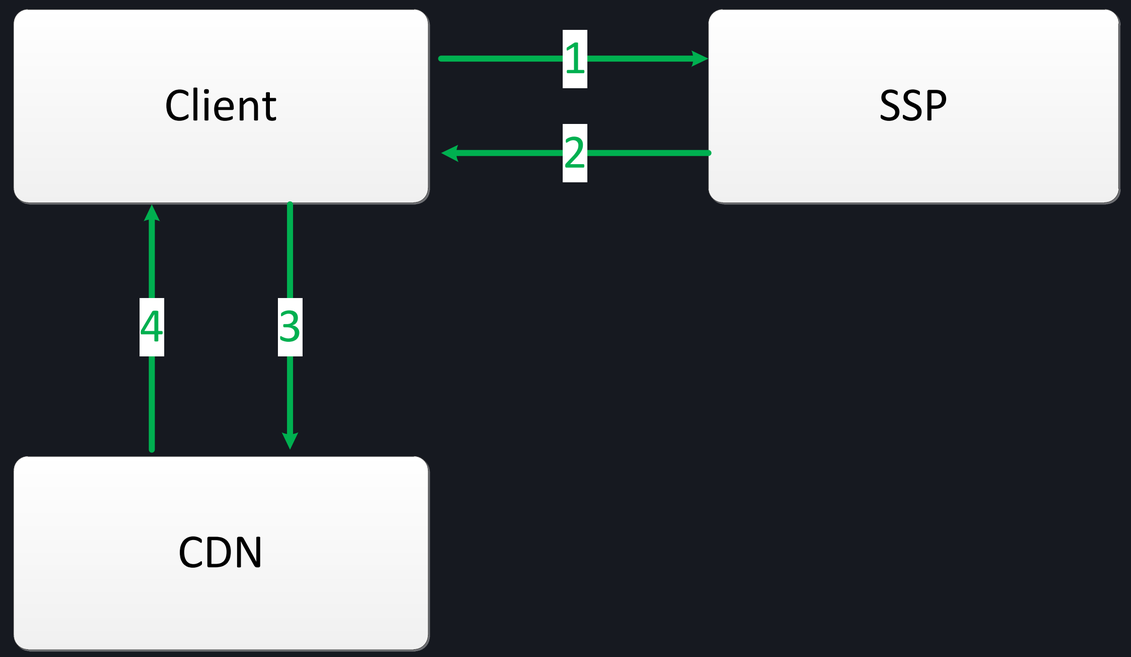
- 修条新路 - 广告链路优化
- 老路提速 - 引擎性能优化



广告链路优化 - 广告加载流程分析

问题诊断：

- 广告加载分两阶段：一、向广告系统请求广告代码，二、根据代码中的URL从CDN加载广告素材
- 加载广告素材加载耗时过长，导致用户跳出或跳转播放页比例高



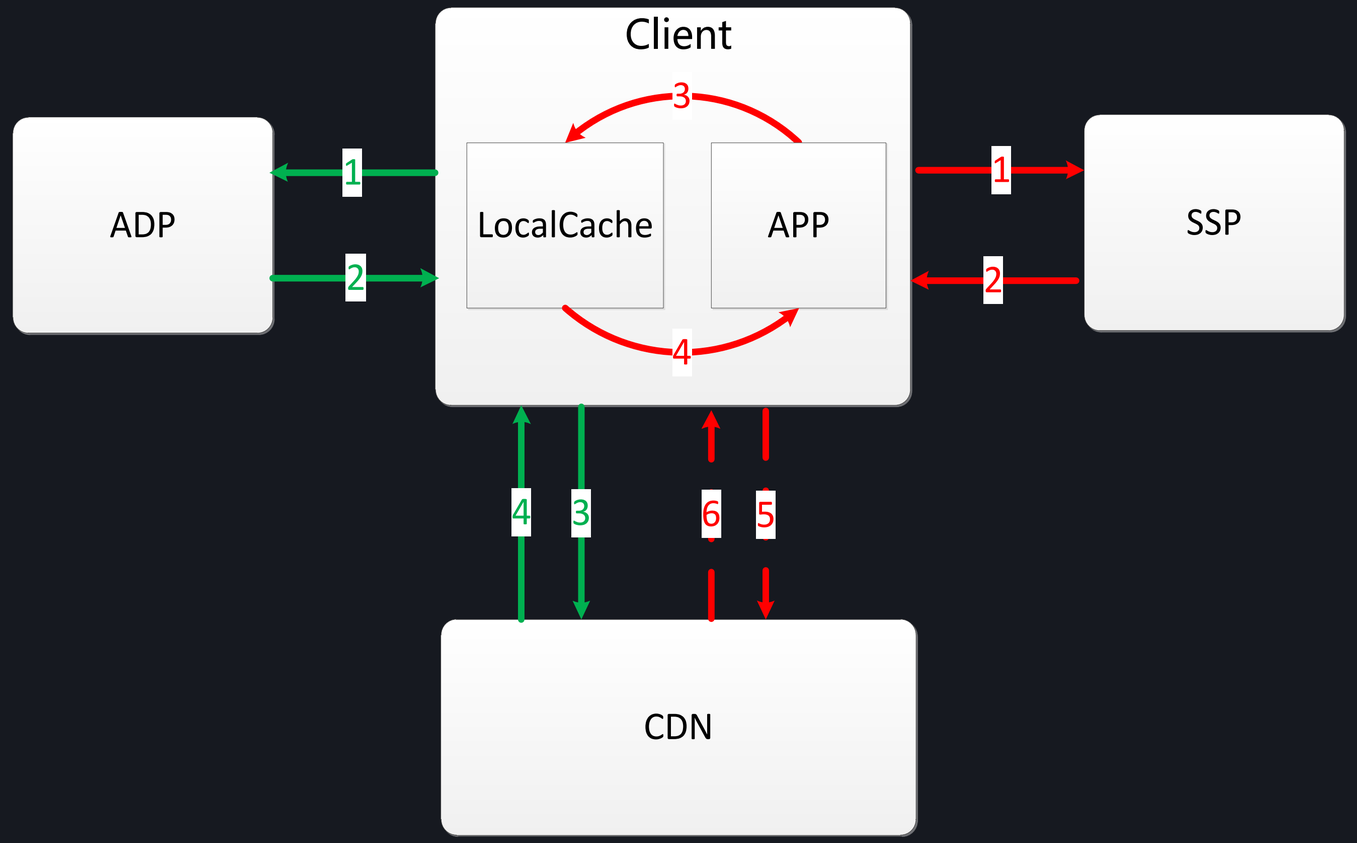
广告链路优化 – 广告缓存推荐原理

基本原理：

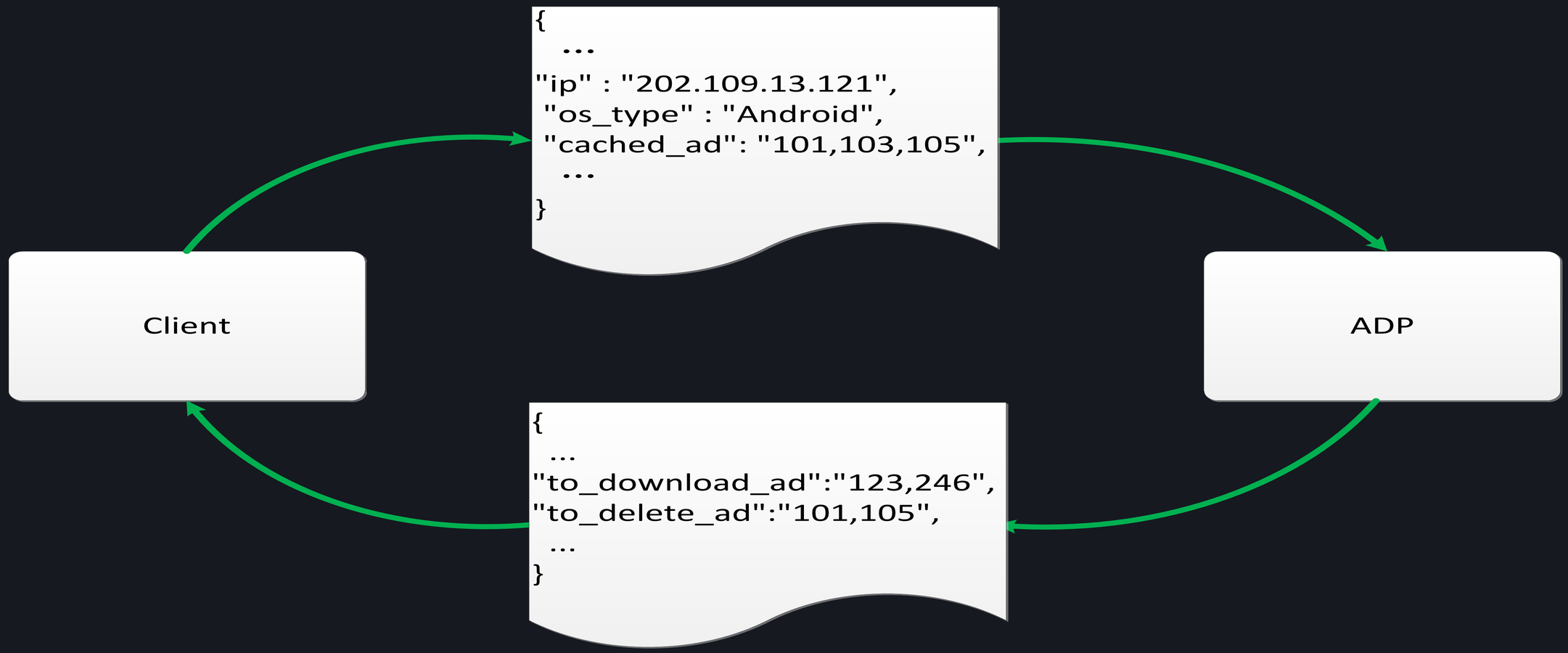
- 通过缓存推荐服务ADP提前下载广告素材到本地
- 将用户在线观看过的广告素材也缓存到本地
- 播广告时先查本地缓存，未命中时才从CDN加载

优化前后链路对比：

- 无缓存推荐：1->2->5->6
- 缓存推荐命中：1->2->3->4
- 缓存推荐未命中：1->2->3->4->5->6
- 新增一条独立的缓存推荐链路：1->2->3->4



广告链路优化 –ADP的缓存推荐接口

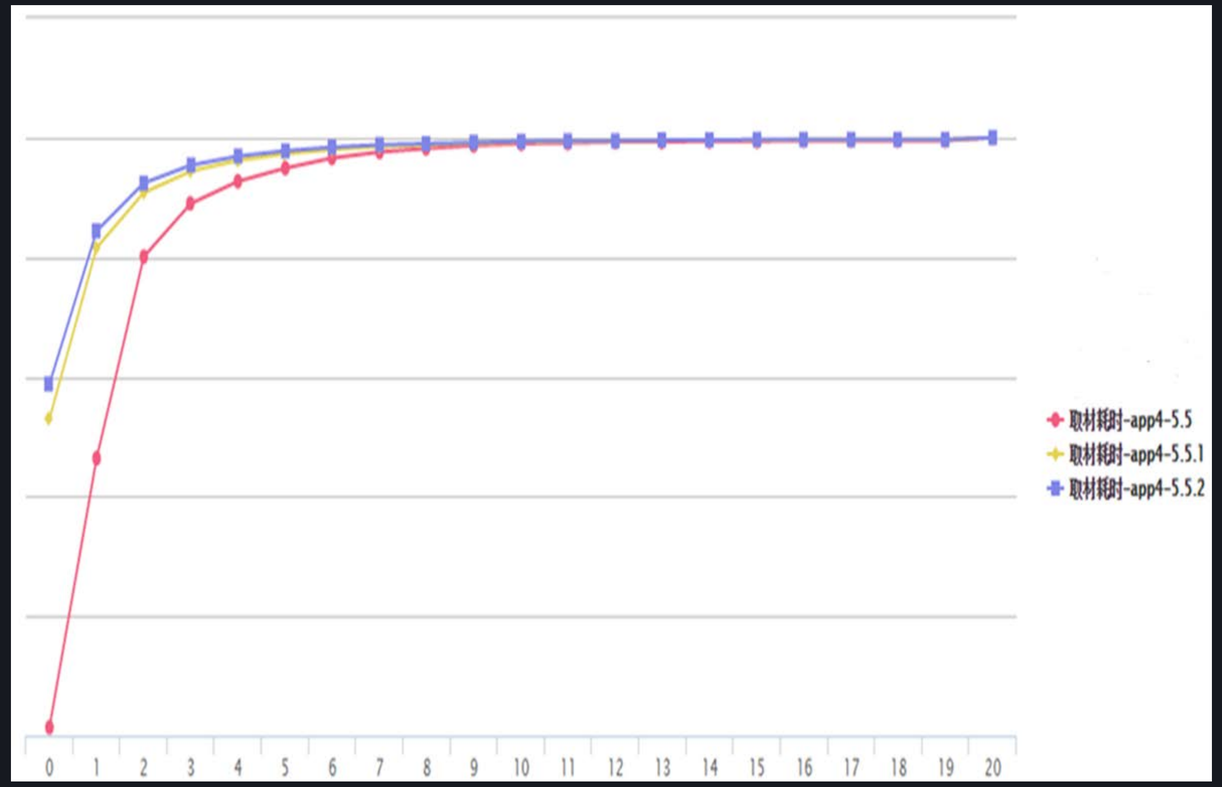


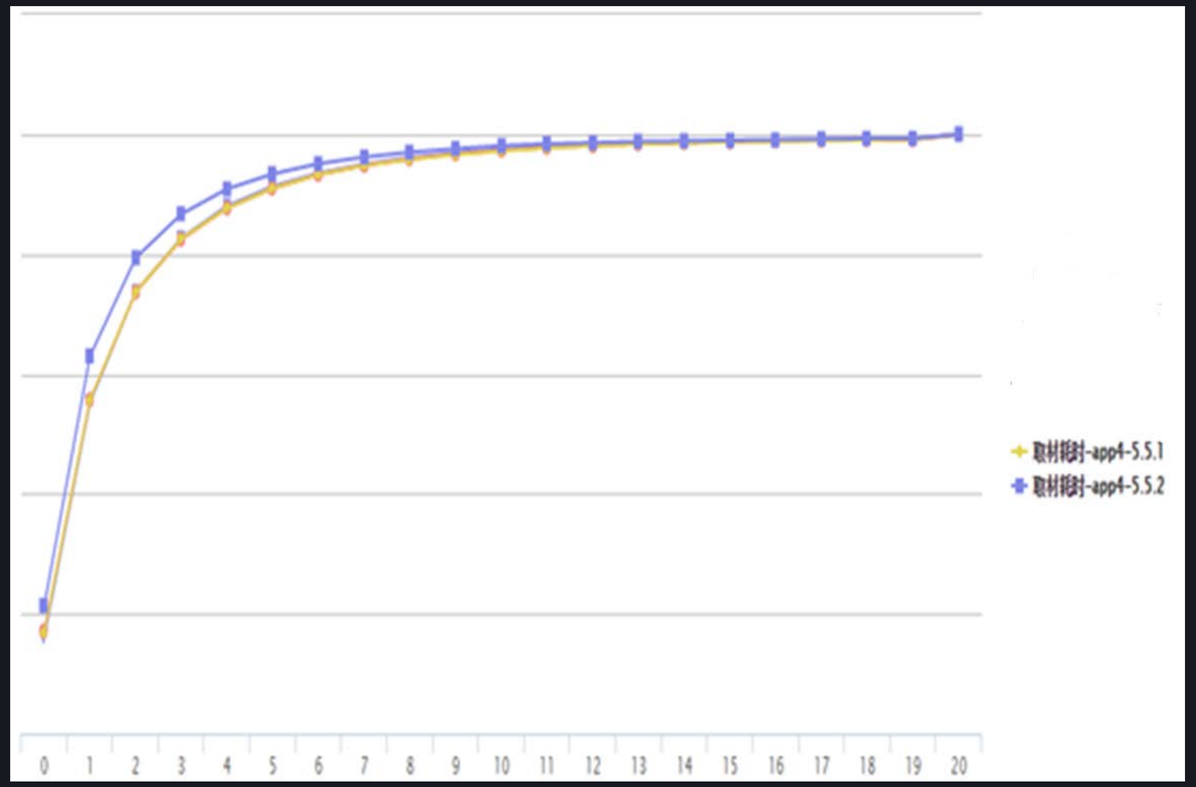
广告链路优化 - 缓存推荐相关规则

- 广告推荐模式：基于定向条件粗筛+算法预判模式
- 广告推荐时机：APP启动完成后
- 本地缓存大小：考虑机器硬件差异

广告缓存推荐 - 广告加载速度的提升效果

如下图：每条曲线对应一个APP版本在某个时间范围内（x轴）完成广告素材加载的比例（y轴），左图为iPhoneApp端，右图为AndroidApp端。iOS端1秒内完成广告加载的比例提升约35倍。

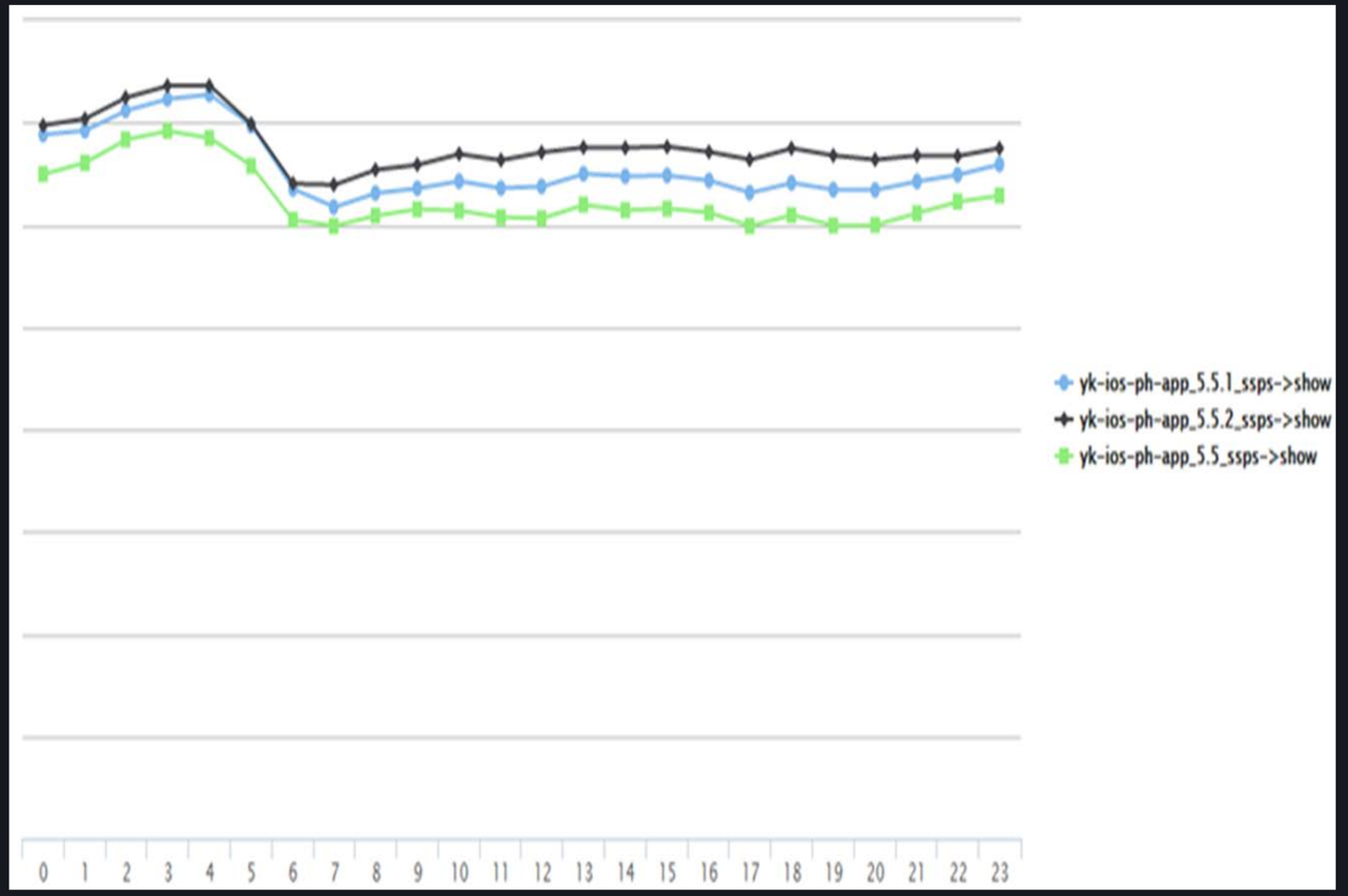




广告缓存推荐 – 贴片曝光率提升效果

优化效果：

- iOS端，优化后的版本（V5.5.1和V5.5.2）比优化前的版本（V5.5）曝光率提升明显，其中V5.5.2比V5.5提升约3.75%。



广告缓存推荐 – 踩过的“坑”

- Q：Android使用了缓存推荐，为嘛有时广告的损耗反而增加了？
- A：原因：Android端文件下载失败几率相对iOS高，客户端尝试播放错误的缓存文件会增大损耗比例。解决办法：下载完后添加文件完整性校验功能。

广告链路优化 - 广告异步加载

问题描述：

- 某些APP上开屏广告加载的耗时过长

优化思路：

- 改变“出行”模式 – 使用异步加载模式，颠倒广告请求和展现的顺序，先展现本地缓存的广告，再请求广告和下载广告素材

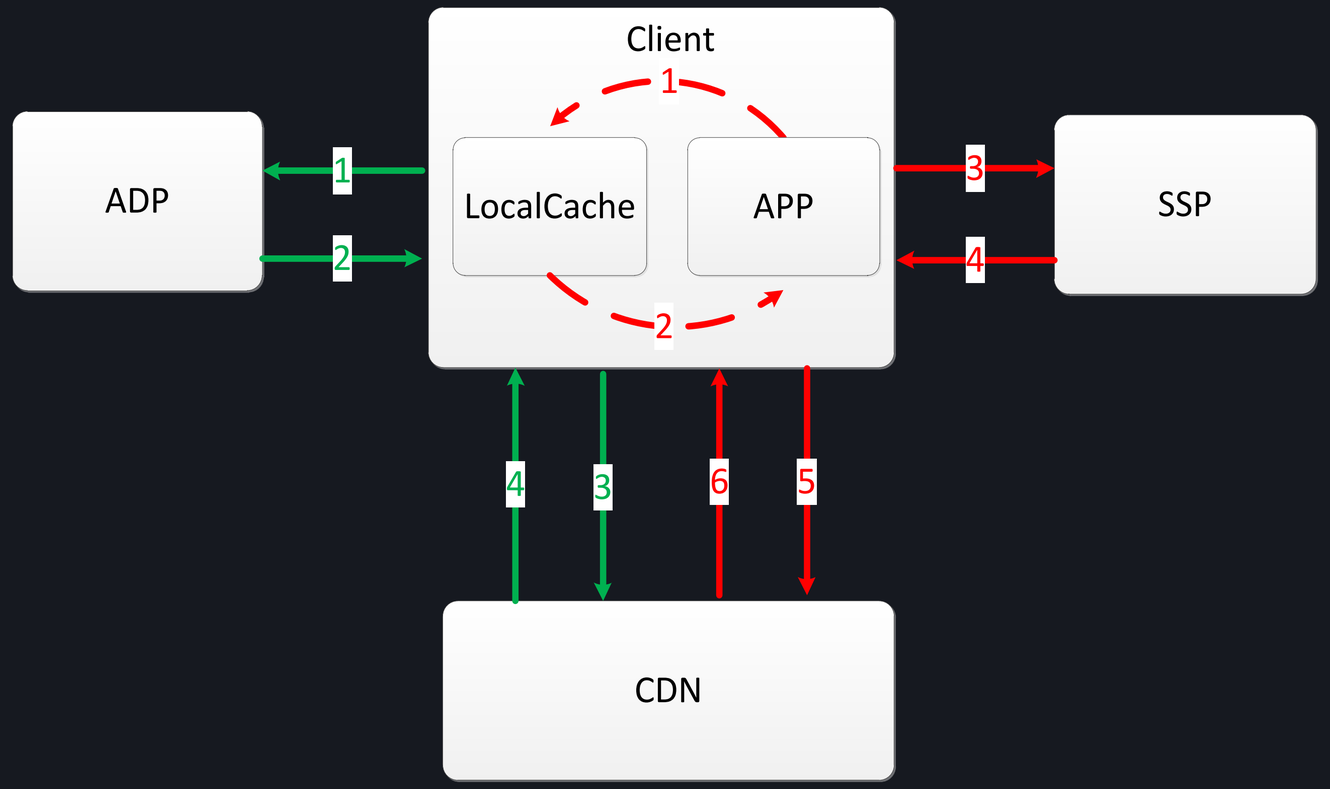
广告链路优化 - 广告异步加载流程

广告加载链路（红色）：

- 广告展现：1->2
- 异步请求：3->4->5->6

独立的广告缓存推荐链路（绿色）：

1->2->3->4



广告异步加载 – 遇到的“坑”及解决办法

Q：如何避免广告投放初期缓存命中率低导致曝光量过低

A：提前设置广告排期，提前返回N（常数）天后投放的广告，供客户端提前缓存，同时再结合缓存推荐机制，增大预缓存比例。

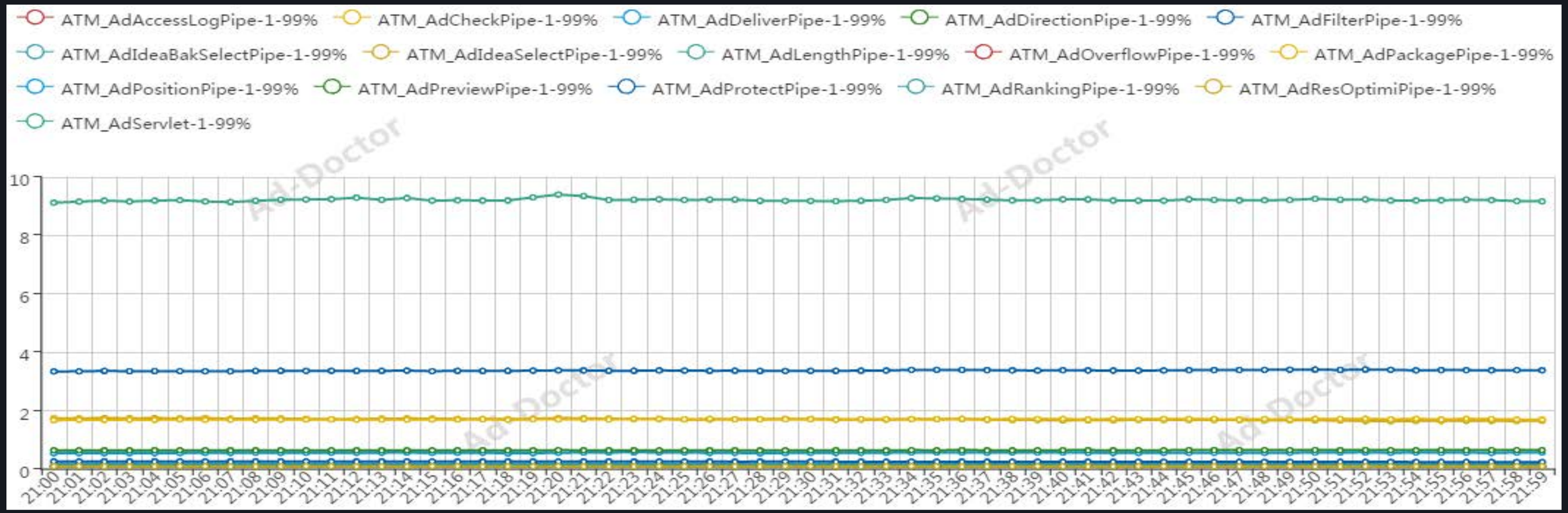
Q：如何将已缓存到客户端的广告素材的及时下线

A：基于推送服务实时通知或提高缓存推荐接口的调用频率

Q：如何解决CPM售卖时的超量投放问题

A：提前分配每日预缓存比例，要为投放当天预留适当比例，投放当天再做实时精准控量，且前后两次重复推送时，需将剩余量计数器中扣掉的量加回来

引擎性能优化 – 优化前广告检索服务各模块耗时



x轴：时间（单位：分钟），y轴：广告接口RT（单位：ms），每条曲线为一个处理模块的耗时曲线

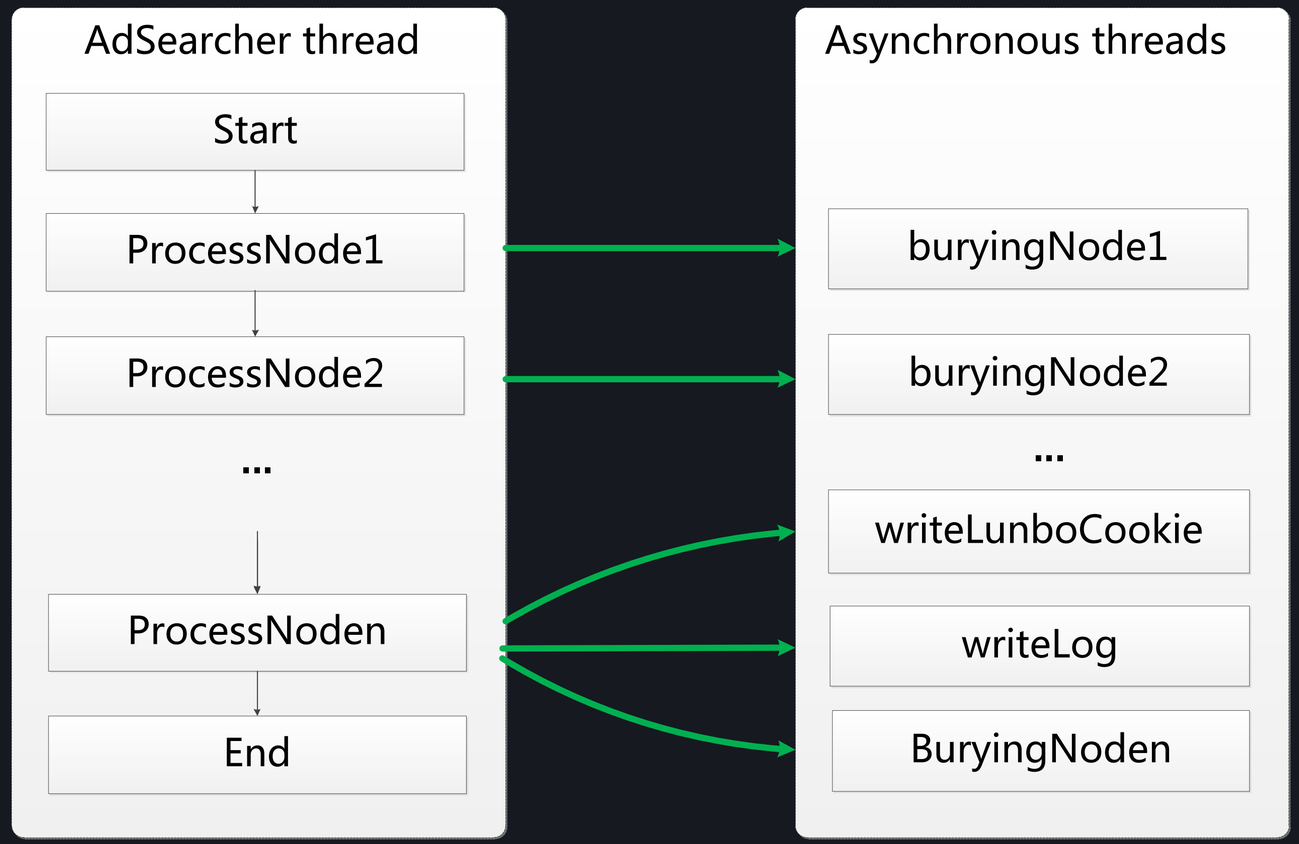
引擎性能优化 – 异步写优化

优化思路：

- 异步埋点（buryingNode）
- 异步写cookie(writeLunboCookie)
- 异步写日志（writeLog）

优化效果：

- 广告请求接口内部耗时相比优化前降低23.8%



引擎性能优化 - 延迟读优化

优化思路：

- 对性能影响较大，且不是每次请求都需要的属性，可以延迟到需要使用时再读（右图示例：Redis中的人群标签等）

优化效果：

- 广告请求接口内部耗时相比优化前降低12.3%