

OpenResty

在腾讯游戏营销技术中的 应用和实践

Shawn 顾小平
2018.11.17



SPEAKER

INTRODUCE

- 华为嵌入式软件工程师
- UT斯达康高级工程师
- 2012年加入腾讯
- IEG市场营销技术后台负责人
- 全“沾”工程师

内容介绍

- **OpenResty在腾讯游戏营销网关中的应用**
- **OpenResty在腾讯游戏广告投放系统中的应用**

OpenResty

在腾讯游戏营销API网关中的应用



+



引入API网关的业务背景



营销推广活动页，小程序，小游戏等

引入API网关的业务背景



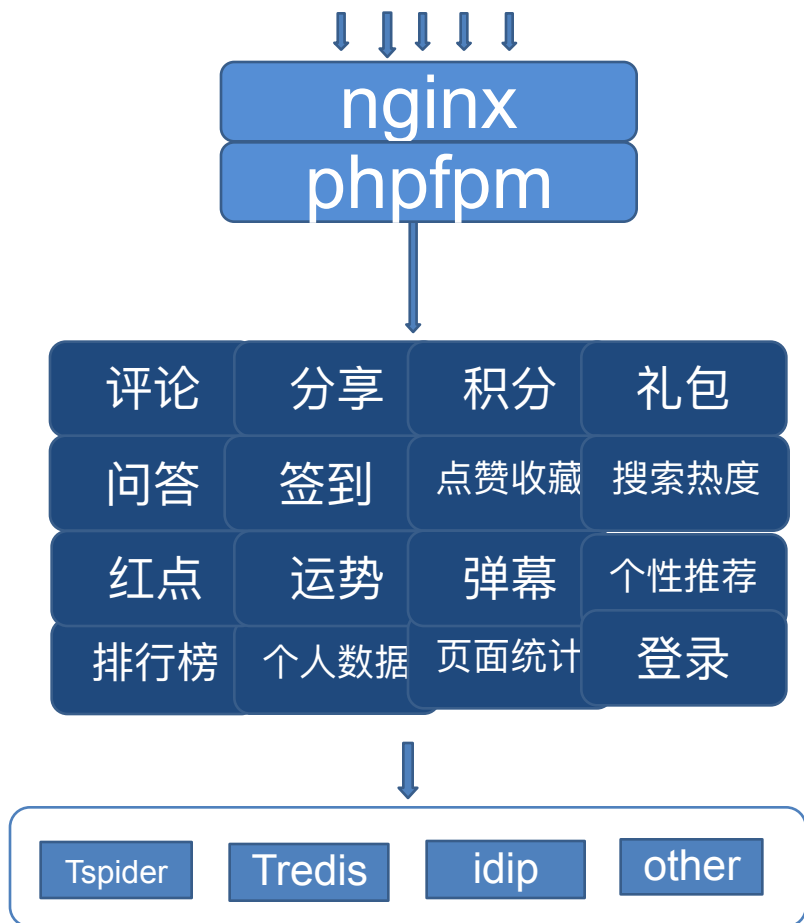
微社区

赛事直播



60+款游戏，API访问流量超过3亿

初始架构后的问题



开发阶段问题

大量非功能性重复开发

- 身份验证/登陆校验
- 流量控制/频次限制
- 安全，黑白名单/sql注入/参数校验等

线上运行阶段问题

异常用户行为控制和分析

- 刷礼包、刷积分、发广告等
- 秒杀消峰、防雪崩、熔断由前端控制
- 无基于业务的监控，统计
- 无调用路径分析，定位排查耗时

业界API网关方案汇总

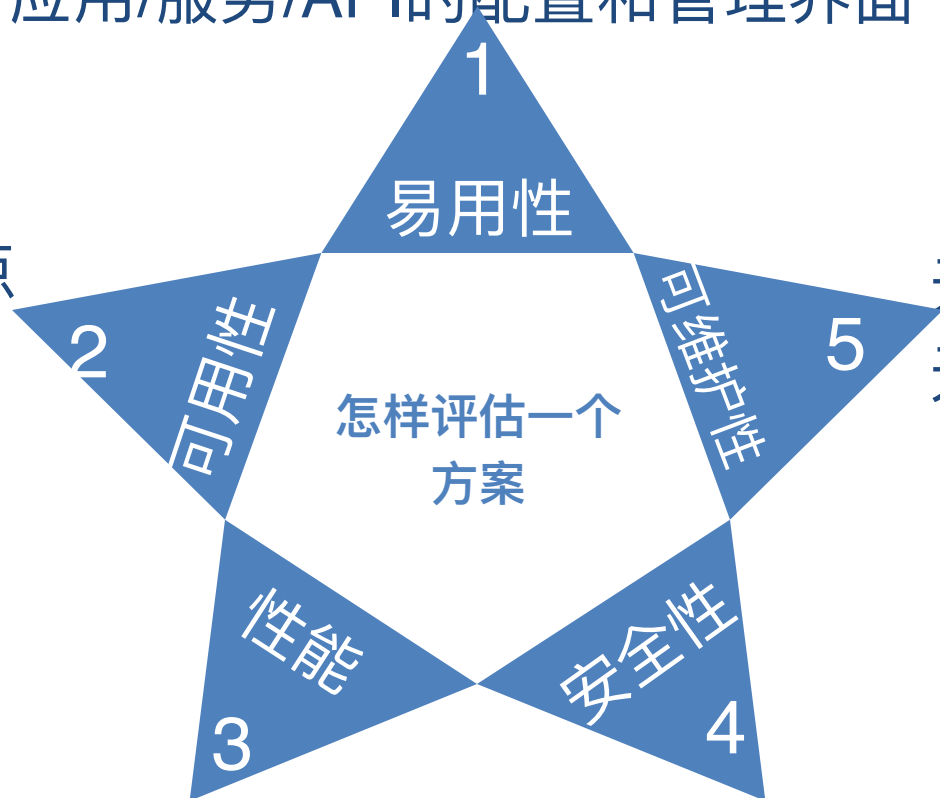
	开发语言	支持功能	处理模型	优点	缺点
	Openresty+ Lua	认证/安全/API管理/ 分流、代理/AB测 试	事件驱动 +协程	可扩展性强/配置 灵活/自带 dashboard	插件功能相对 简单
	openresty +Lua	认证/安全/分流、 代理/日志落地	事件驱动+协 程	插件丰富/平台兼 容性好/社区活跃	版本间API差异 大/插件有捆版 收费功能
	golang	多种认证方式\流 量管控\数据分析 \报文转换等	事件驱动 +协程	功能全面/安装 简单	配置复杂/收 费功能多
	Java, groovy	验证与安全保障/审查与 监控/动态路由/负载分配/ 压力测试等	异步状态机	配套全/扩展性好/ 社区活跃	功能较少/稳定 性和性能有待 考证
  	未开源	功能齐全、 商业化	未开源	无需搭建、 付费可用	黑盒操作/ API私密性

开源Orange方案不足

无业务开发人员熟悉的
应用/服务/API的配置和管理界面

网关和配置节点
均需自己保证

无错误和染色日志
无调用跟踪和定位



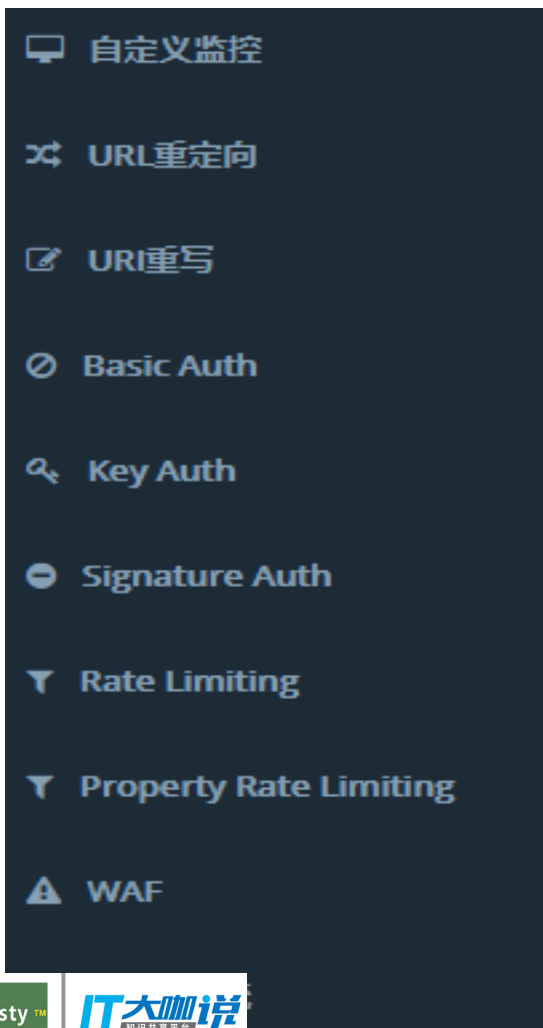
随规则数增加

仅有黑白名单及简单认证

OpenResty™ | IT大咖说 能下降明显
知识共享平台

1 易用性优化

面向业务而不是面向技术



应用->服务->API->认证/校验/安全/流控/统计

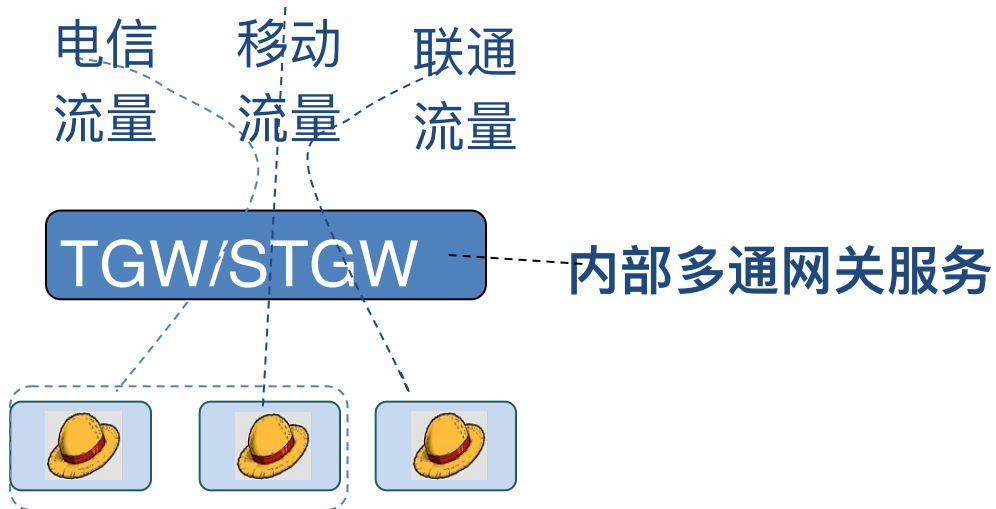
VS.

所有应用

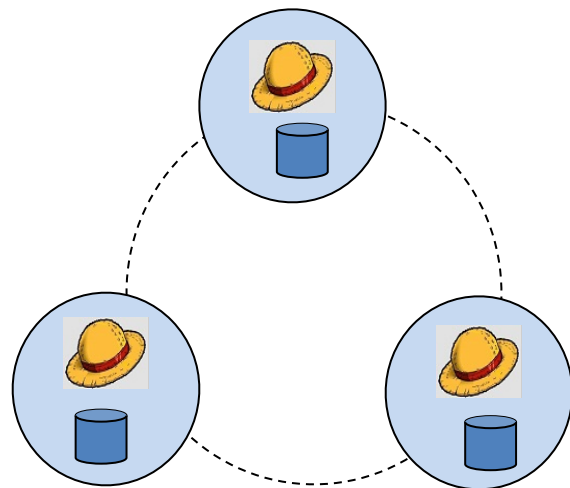
- ▶ 奇迹暖暖微社区
- ▶ 雷霆战机微社区
- ▼ 王者荣耀微社区 **应用**
 - ▶ 分享服务
 - ▶ 红点服务
 - ▶ digg interestcenter 趣味中心
 - ▶ smoba 王者荣耀个人中心服务
 - pvp域名服务
 - ▶ app域名服务
 - ▼ comment 评论服务 **服务**
 - 40102 -- 删除评论
 - 40104 -- 删除回复 **API**
 - 40115 -- 查询红点
 - 40116 -- 查询回复信息
 - 40122 -- 查询回复消息列表
 - 40114 -- 查询资源列表统计数
 - 40118 -- 查询玩家的评论列表
- ▶ 王者荣耀活动
- ▶ 王者荣耀赛事
- ▶ 王者荣耀同人站

2可用性优化

网关本身的可用性优化



配置节点的可用性优化



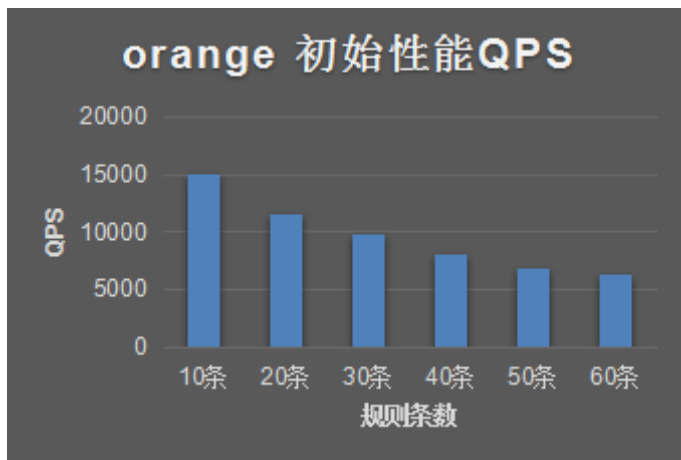
- 把网关作为TGW/STGW的后端
- 3台机器，跨机房部署

- 独立部署mysql/redis
- 集成的mysql/sqlite存放配置
- 集成的ETCD存放配置

3性能优化：初步测试

Orange奇怪的性能数据

QPS吞吐量数据



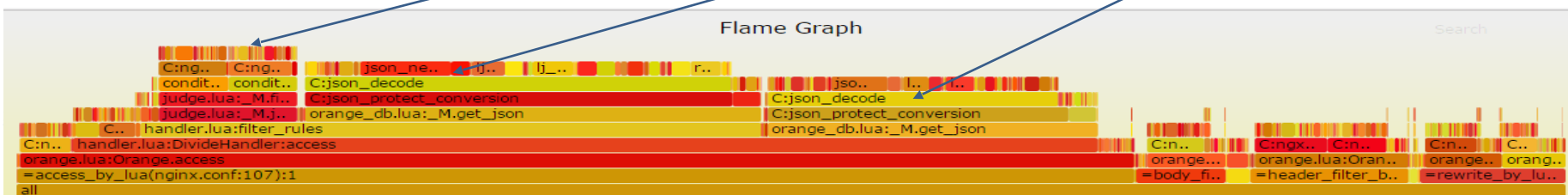
CPU消耗量数据

超过60%CPU耗时

正则匹配
操作

2次json系
列化操作

Flame Graph



3性能优化：初步优化

最大的优化是不做或柔性平衡

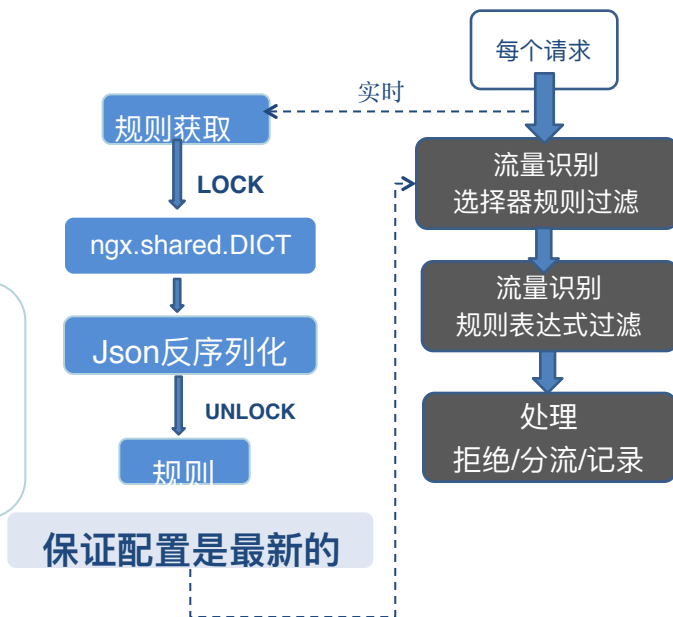
第1反应

cJSON

RapidJSON

- 性能优势明显
- 内存占用低
- 确实有比较大提升

为什么做？



能否不做？

性能

实时性

Timer

worker定时从共享内存获取配置信息

3性能优化：进一步优化



环境准备：

- ✓测试机性能，压测工具
- ✓被测试机同生产环境
- ✓测试机和被测试机网络延迟

配置调优：

- ✓系统和最大打开文件数
- ✓连接复用，TIME_WAIT回收
- ✓worker个数和cpu核数相同

静态检查：

- ✓使用local函数替代引用外部函数
- ✓避免使用pairs做循环
- ✓日志打印的时候少用..连接操作
- ✓通过FFI来调C函数而不是lua C的方式
- ✓采用FFI数据结构替代lua table（频繁使用的数据结构）
- ✓杜绝使用NYI操作（生成日志排查）

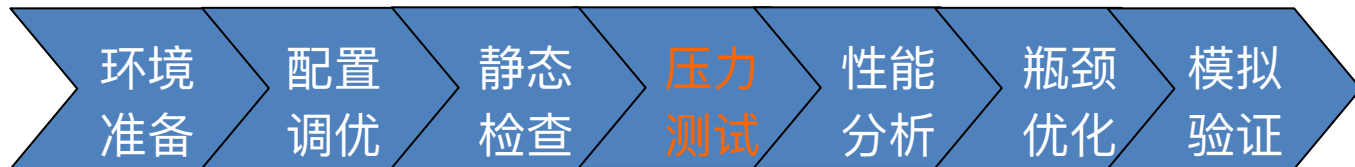
CPU资源性等

OpenResty™

IT大咖说
知识共享平台

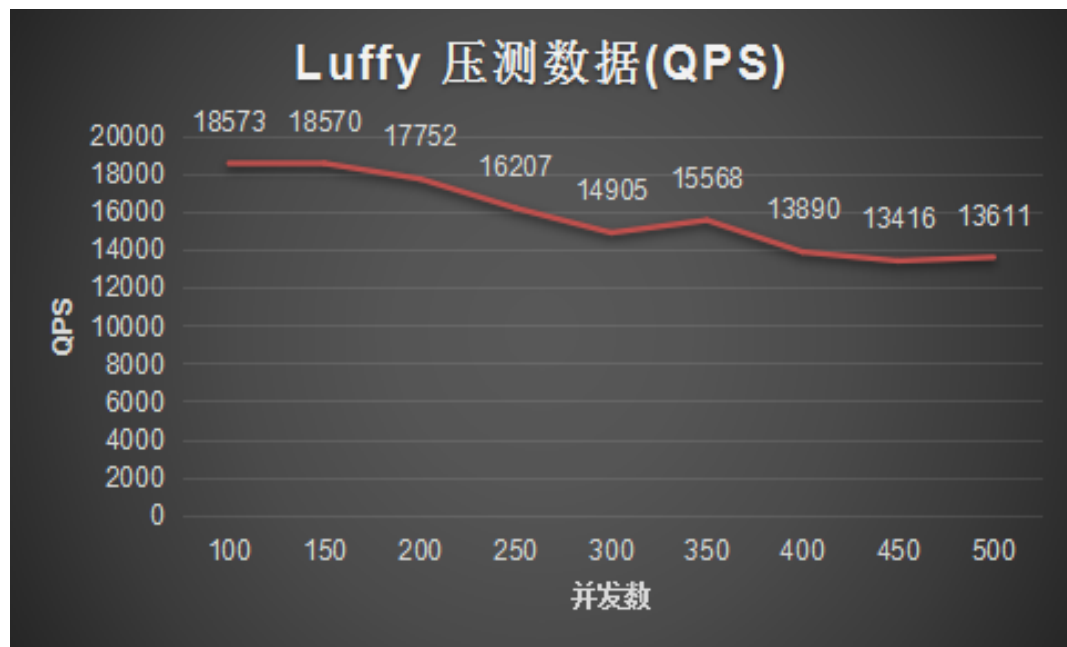


3性能优化：压力测试



压力测试（模拟现网流量）：

- ◆7个query参数提取
- ◆7个参数正则表达式匹配操作
- ◆通过I5向真实后端服务发送TCP请求
- ◆对后端数据json序列化、反序列化
- ◆基于该接口的流控（每秒10000条）
- ◆基于api/svc/app实时统计和离线入库
- ◆每个请求lua各阶段处理耗时统计；



内部机型c1



腾讯游戏
Tencent Games

用心创造快乐!

3性能优化：性能分析

json(3+4)占9.18%

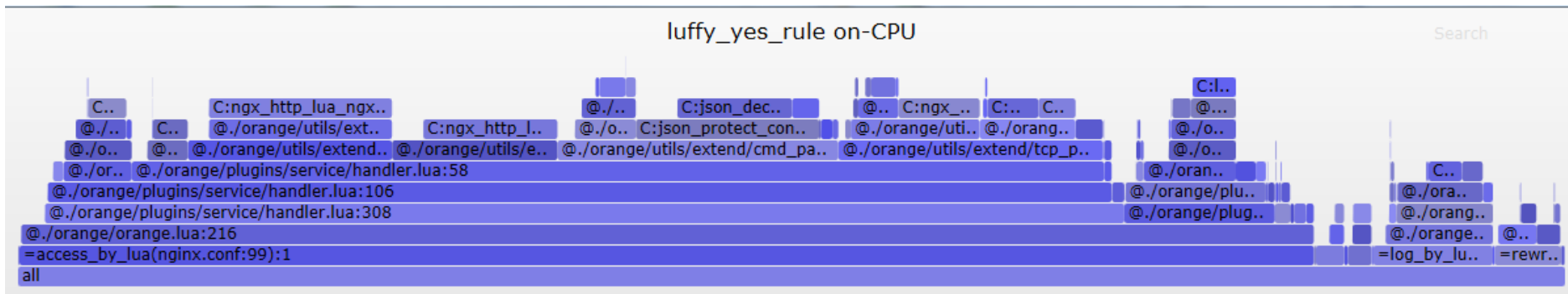
l5(4+7)占4.74%

1	Function: C:ngx_http_luangx_req_get_uri_args (1,018 samples, 11.83%)
2	Function: C:ngx_http_luangx_re_find (746 samples, 8.67%)
3	Function: C:json_decode (638 samples, 7.42%)
4	Function: C:json_encode (151 samples, 1.76%)
5	Function: C:_ZL13L5ApiGetRouteP9lua_State (213 samples, 2.48%)
6	Function: C:ngx_http_lua_shdict_incr (184 samples, 2.14%)
7	Function: C:lj_cf_string_gmatch_aux (145 samples, 1.69%)
8	Function: C:_ZL22L5ApiRouteResultUpdateP9lua_State (194 samples, 2.26%)
9	Function: C:ngx_http_lua_var_get (117 samples, 1.36%)

正则匹配(2+6)
占10.36%

其他占15.33%

CPU消耗占比Top10



忽略火焰图颜色

3性能优化：瓶颈优化（1）

I5(4+7)占4.74%

L5消耗瓶颈优化

- 查看I5agent CPU利用率，日志，出错情况
- Lua C/API替换为FFI的调用方式
- I5调用是阻塞的UDP调用
- 换成异步的I5接口？
- 用lua的cosocket方式实现I5 API？

→

正常

→

提升不大

→

注意超时参数

→

每worker一个新线程

→

协议不了解，麻烦

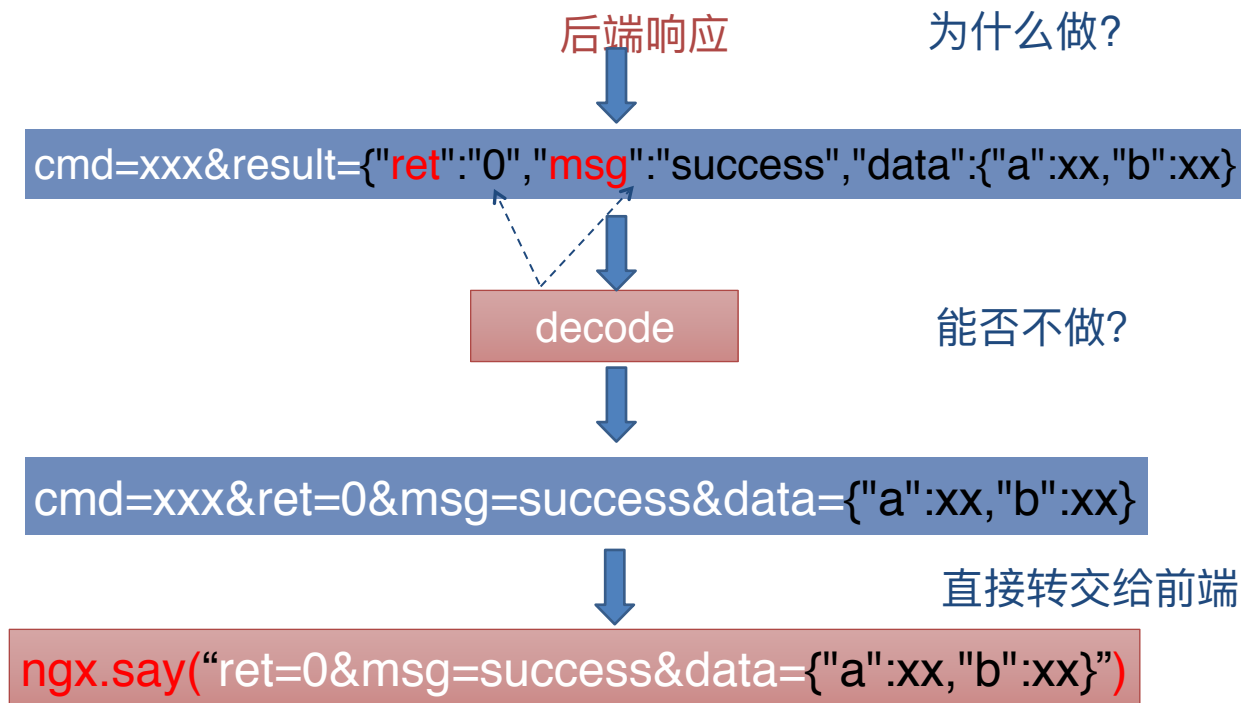


放弃优化，无提升

3性能优化：瓶颈优化（2）

json(3+4)占9.18%

Json消耗瓶颈优化



性能直接提升9.18%↑

3性能优化：瓶颈优化 (3)

正则匹配(2+6)占10.36%CPU

PCRE正则引擎

各语言，工具使用

只支持块模式

1次只编译1个规则

一个输入多个规则需匹配多次

性能相对慢

VS.

Hyperscan正则引擎

DPI/IDS/IPS产品使用

支持块模式和流模式

1次编译多个规则

一个输入多个规则只匹配1次

性能极快

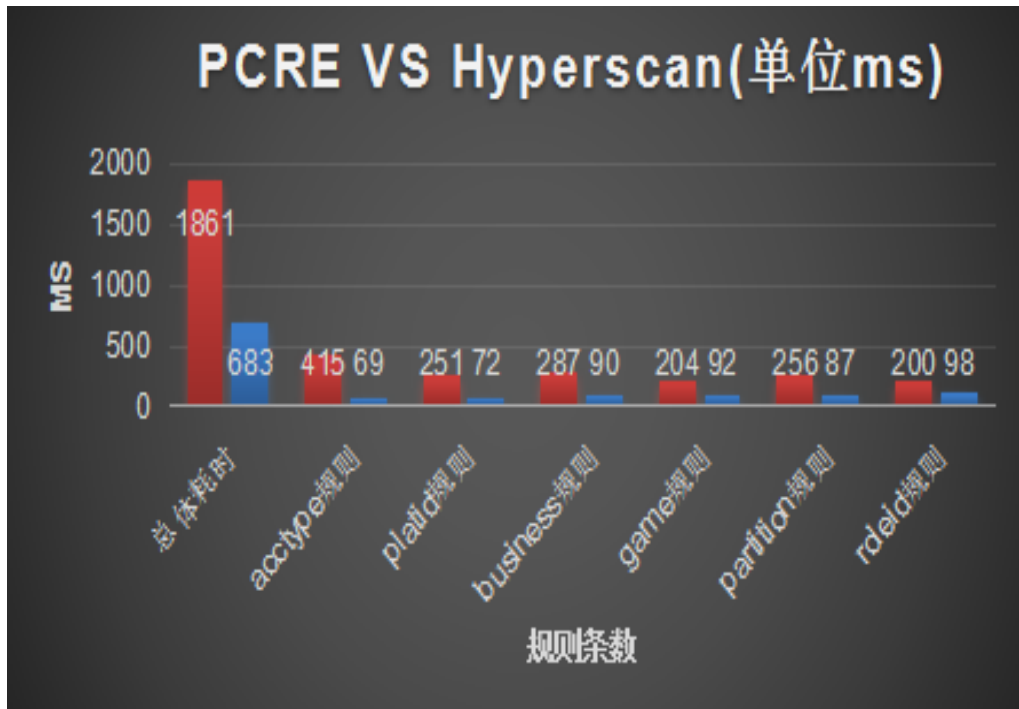
3性能优化：瓶颈优化 (3)

正则匹配(2+6)占10.36%CPU

带参数的测试请求url文本数据，
每个参数匹配300K次

具体测试规则（参数校验）：

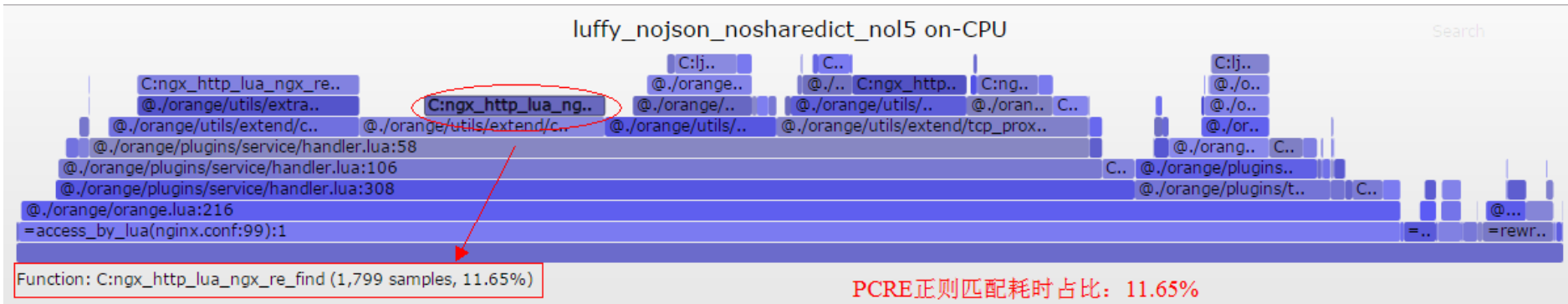
- 1) acctype, 取值1和2, 规则表达式: `^[1|2]$`
- 2) platid, 取值0和1, 规则表达式: `^[0|1]$`
- 3) business, 纯字母, 规则表达式: `^[A-Za-z]+$`
- 4) game, 纯字母, 规则表达式: `^[A-Za-z]+$`
- 5) partiton, 纯数字, 规则表达式: `^d+$`
- 6) roleid, 纯数字, 规则表达式: `^d+$`
- 7) openid, 数字+字母+下划线, 规则表达式: `^w+$`



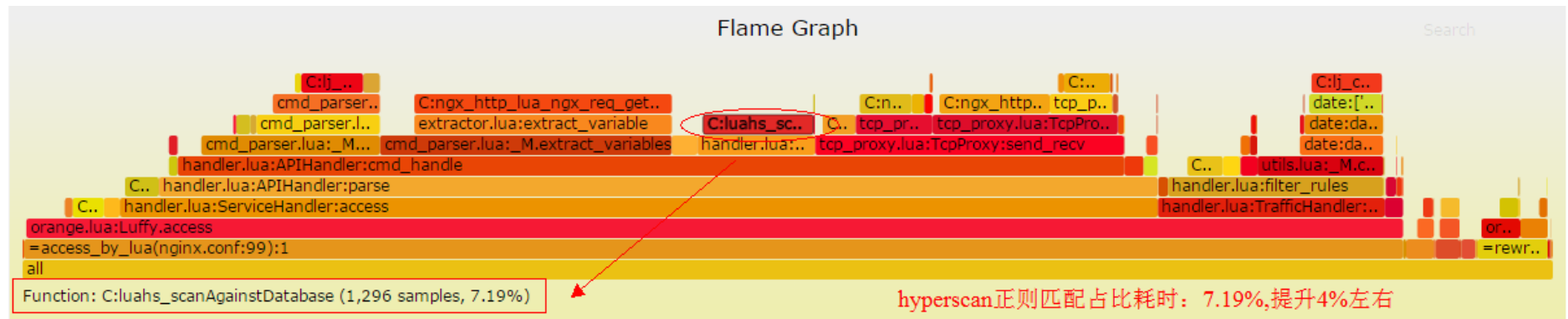
块模式下

hyperscan总耗时是PCRE **36.7%** ↓

3性能的优化：模拟验证



优化前PCRE消耗火焰图



集成hyperscan到网关中后

3性能的优化：是否还可以优化

是否还可以优化？

```
graph TD; A[是否还可以优化?] --> B[能否一次匹配所有参数规则]; A --> C[规则表达式编写难度的问题];
```

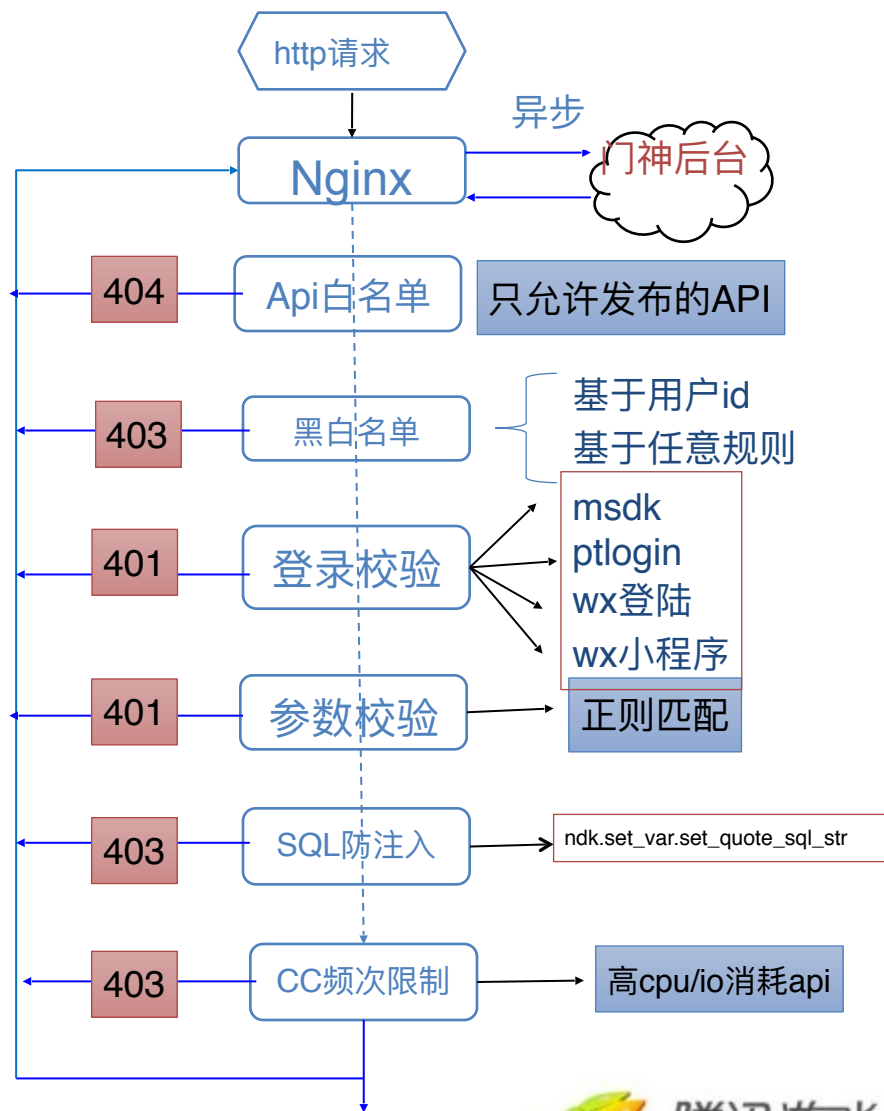
能否一次匹配所有参数规则

规则表达式编写难度的问题

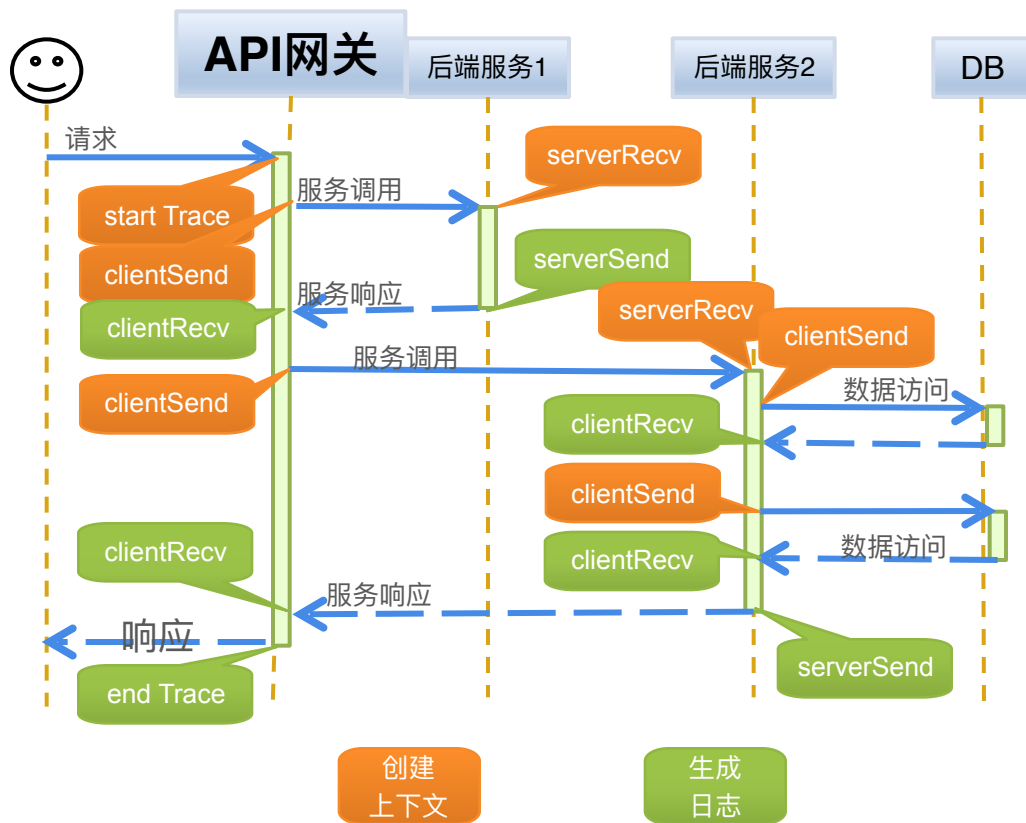
4安全性优化

个性化安全策略

- 把门神集成到api网关
- 基于API的白名单
- 基于用户id和规则的黑白名单
- 双平台（微信/QQ）登录校验
- 参数类型和正则校验
- sql注入问题



5可维护性优化



优化1：日志和统计
4xx/5xx用户染色日志
基于API/服务/应用的统计

优化2：API网关作为调用链跟踪系统的起点：

- 生成Traceid，创建调用链上下文，结束调用链
- 控制大流量API上报调用跟踪的频率

OpenResty

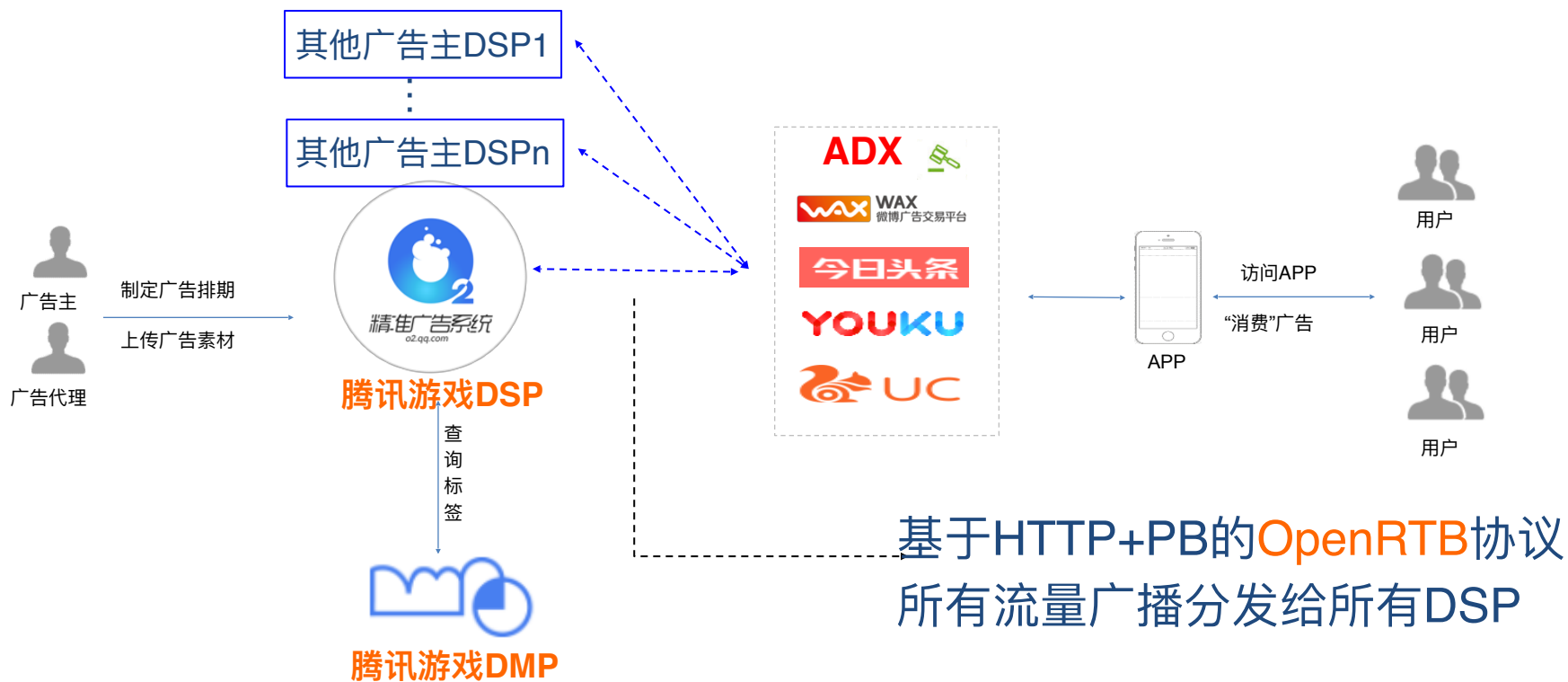
在腾讯游戏广告投放系统中的应用



+



实时竞价广告流程介绍



- ① 媒体通过广告交易平台**ADX**把用户访问流量广播给广告主的服务器**DSP**
- ② **DSP**根据广告主存放在**DMP**数据管理平台中的自有数据（eg.到访/注册/活跃/流失付费等）
- ③ 实时选择和出价，同时和其他广告主实时竞争来获得每次广告曝光机会

实时竞价广告技术挑战

数据

- 标签挖掘
- 人群扩散
- 画像分析

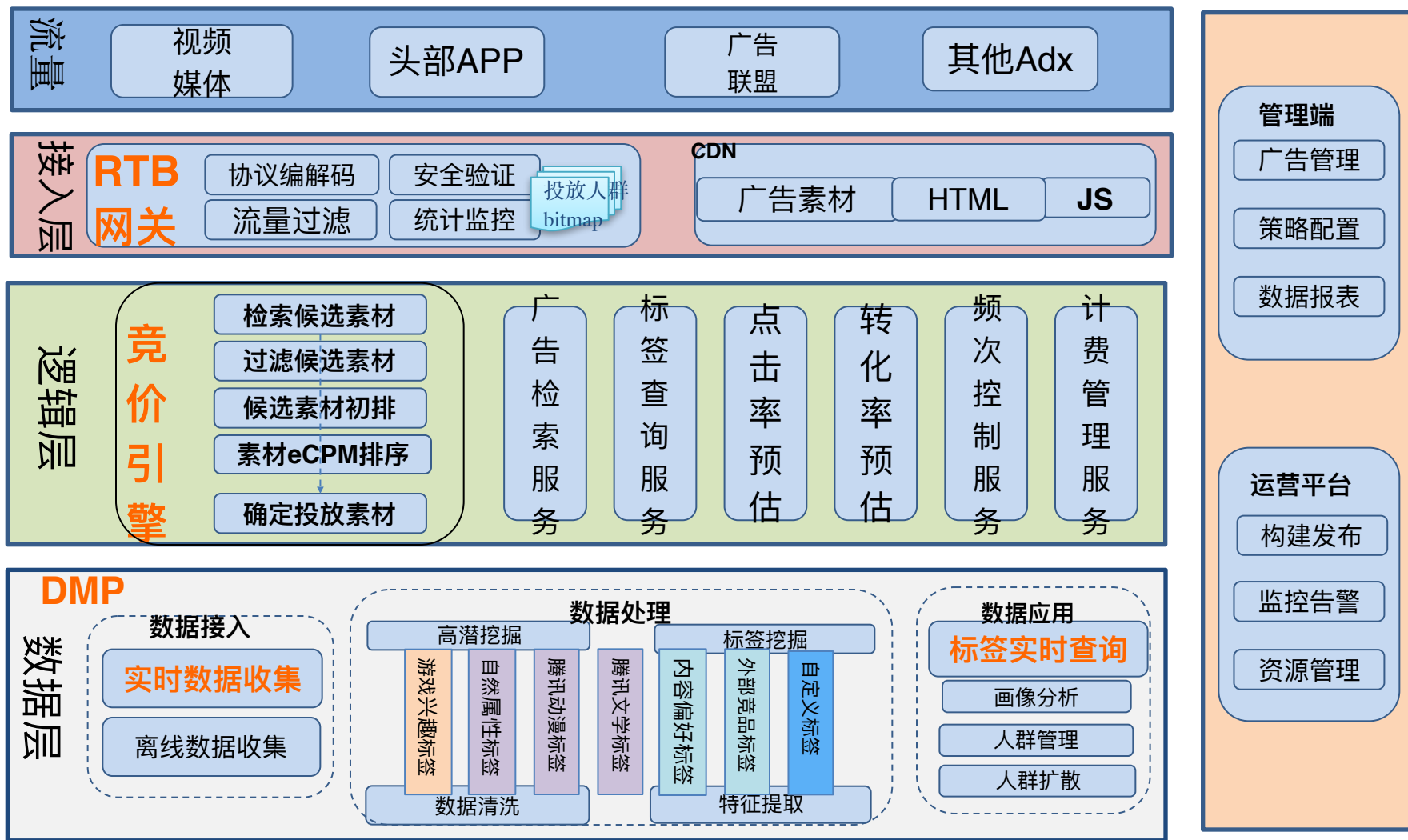
算法

- pCTR
- pCVR

系统

- 高吞吐
- 低延时

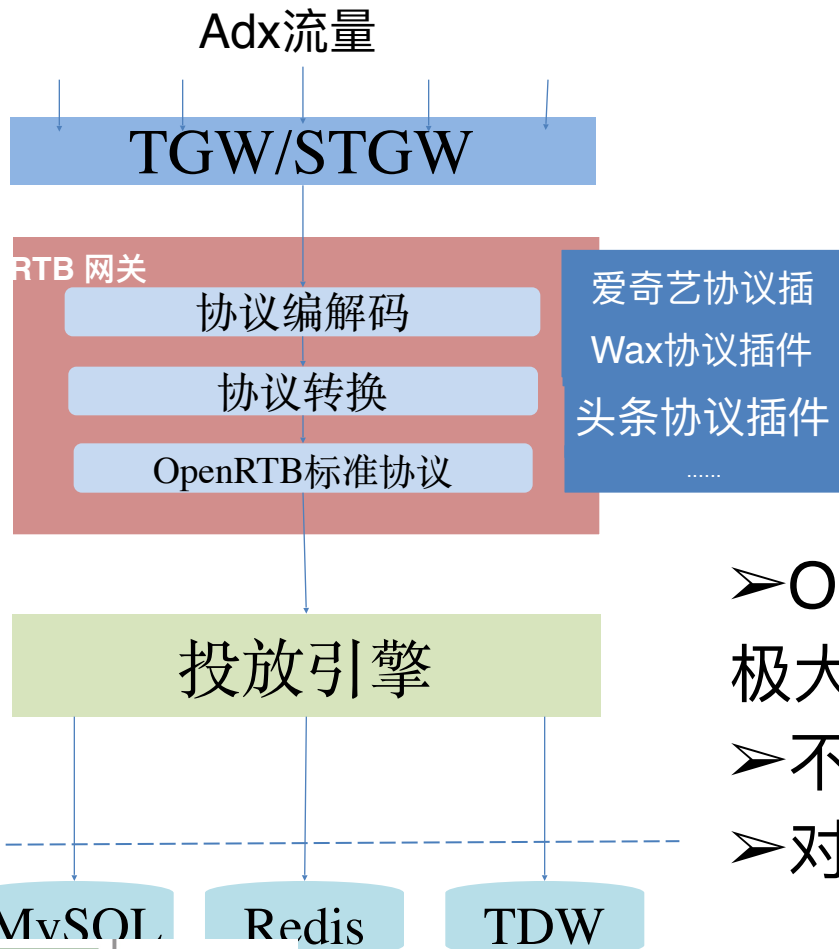
实时竞价广告系统侧架构简图



橙黄色为采用OpenResty重构和优化过

接入层：OpenResty定制RTB网关

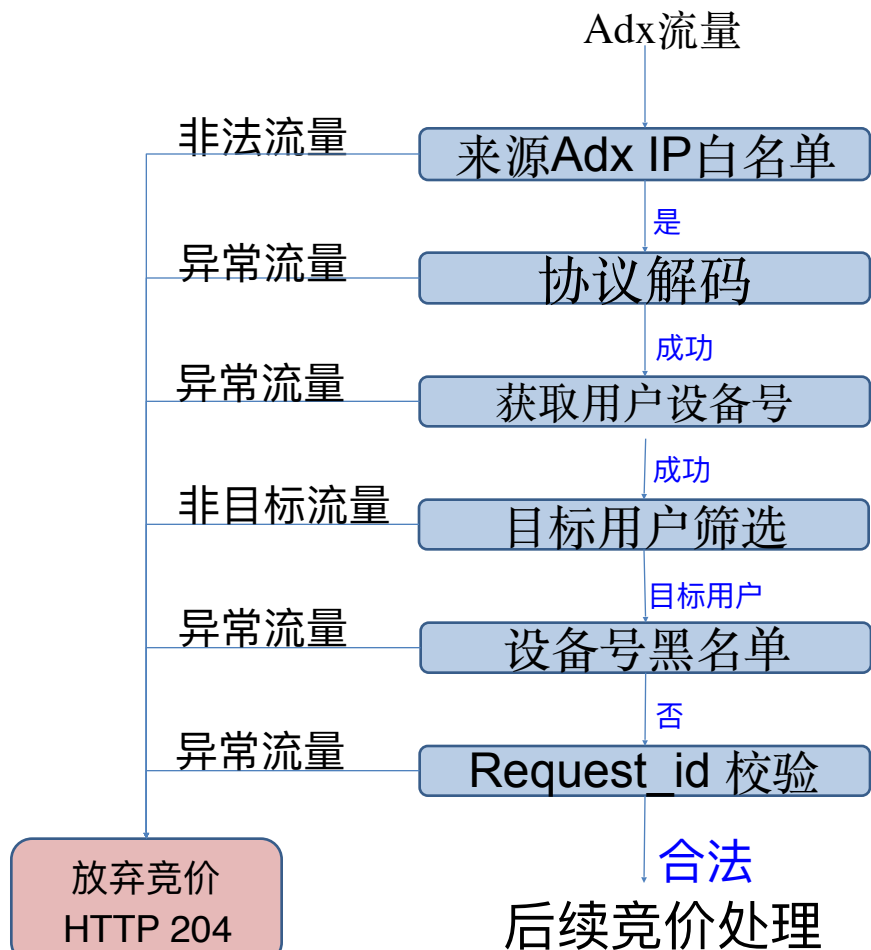
插件化不同编解码协议



- OpenResty插件化，解释性，极大提高开发和对接效率
- 不同媒体不同的RTB协议编解码插件
- 对下竞价引擎提供统一的调用

接入层：OpenResty定制RTB网关

集成安全验证策略



➤ Adx IP 白名单，设备号黑名单

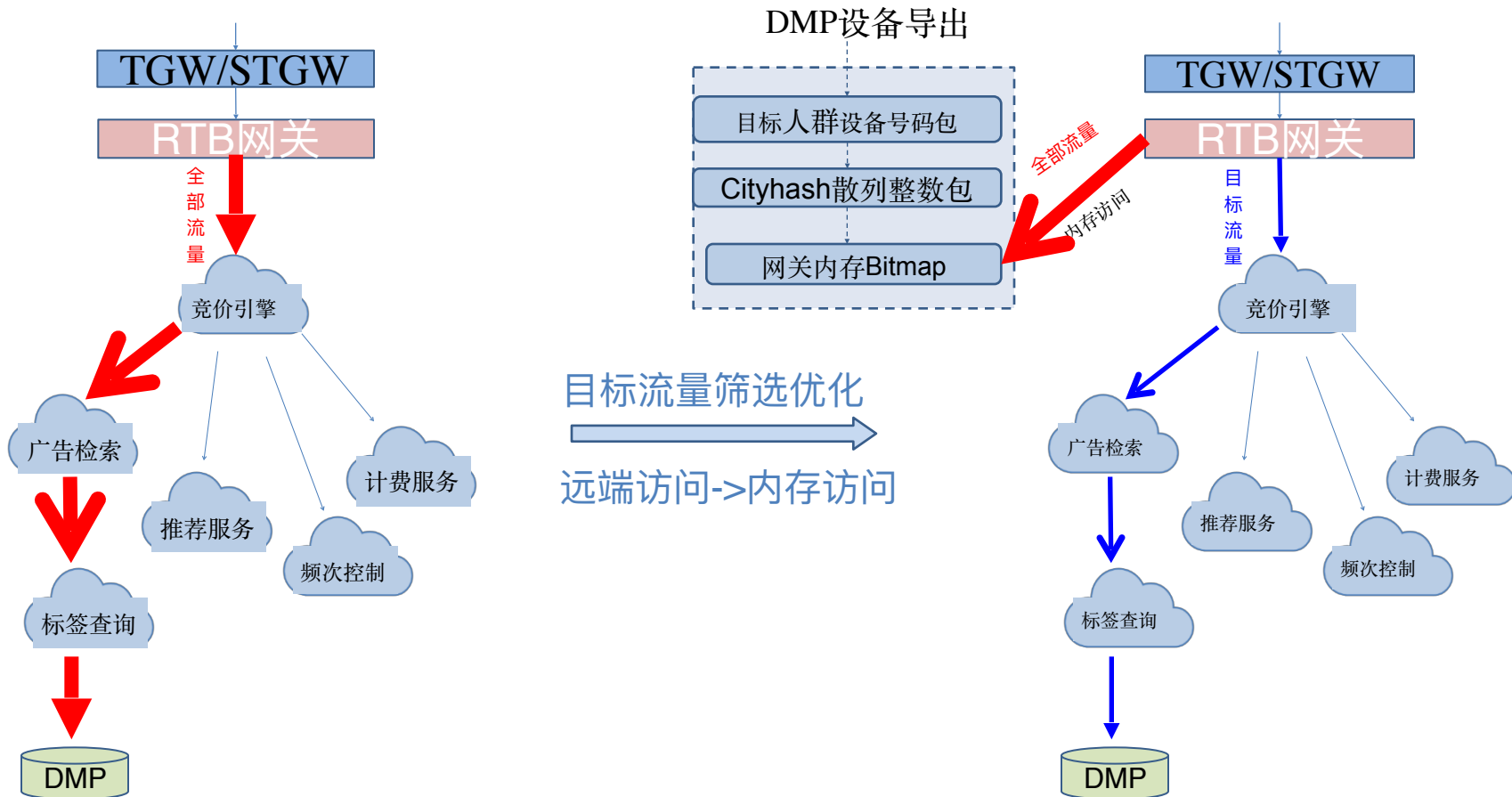
➤ request_id 在win-notice/曝光/点击阶段的校验

➤ win-notice/曝光/点击链接参数加密，防盗链作弊

接入层：OpenResty定制RTB网关

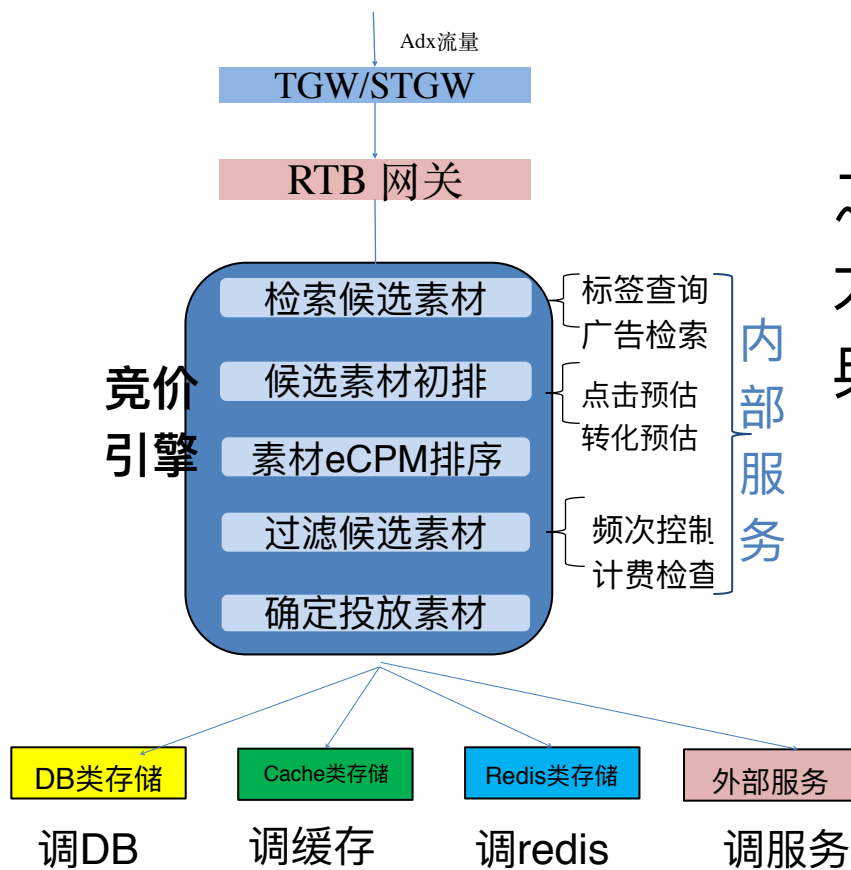
思维固化下的业界处理流程

非精确柔性过滤无效流量



逻辑层：OpenResty重构竞价引擎

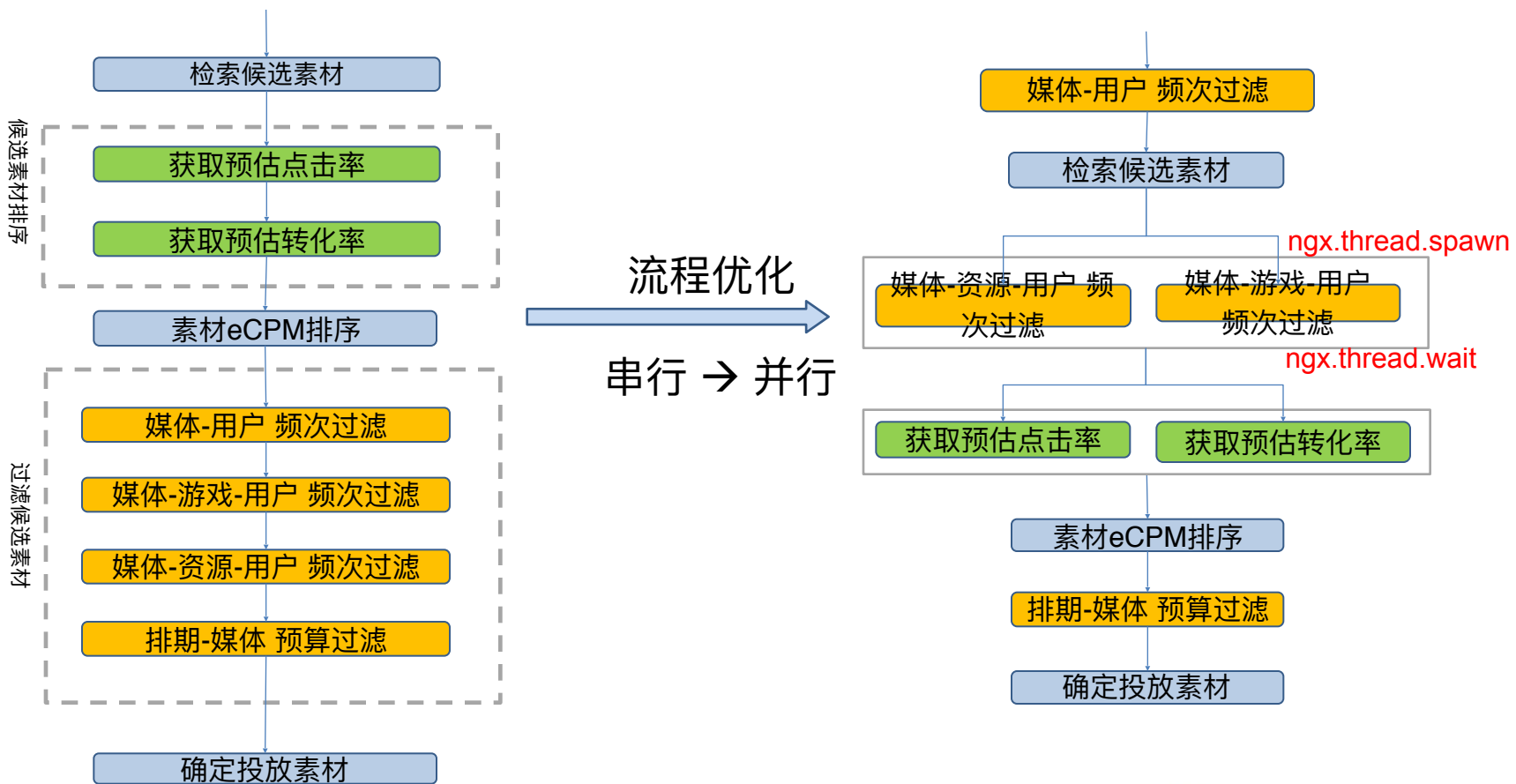
大胆重构极大提高并发和吞吐



之前采用C++同步多线程框架
大量线程等待I/O
典型IO密集型服务

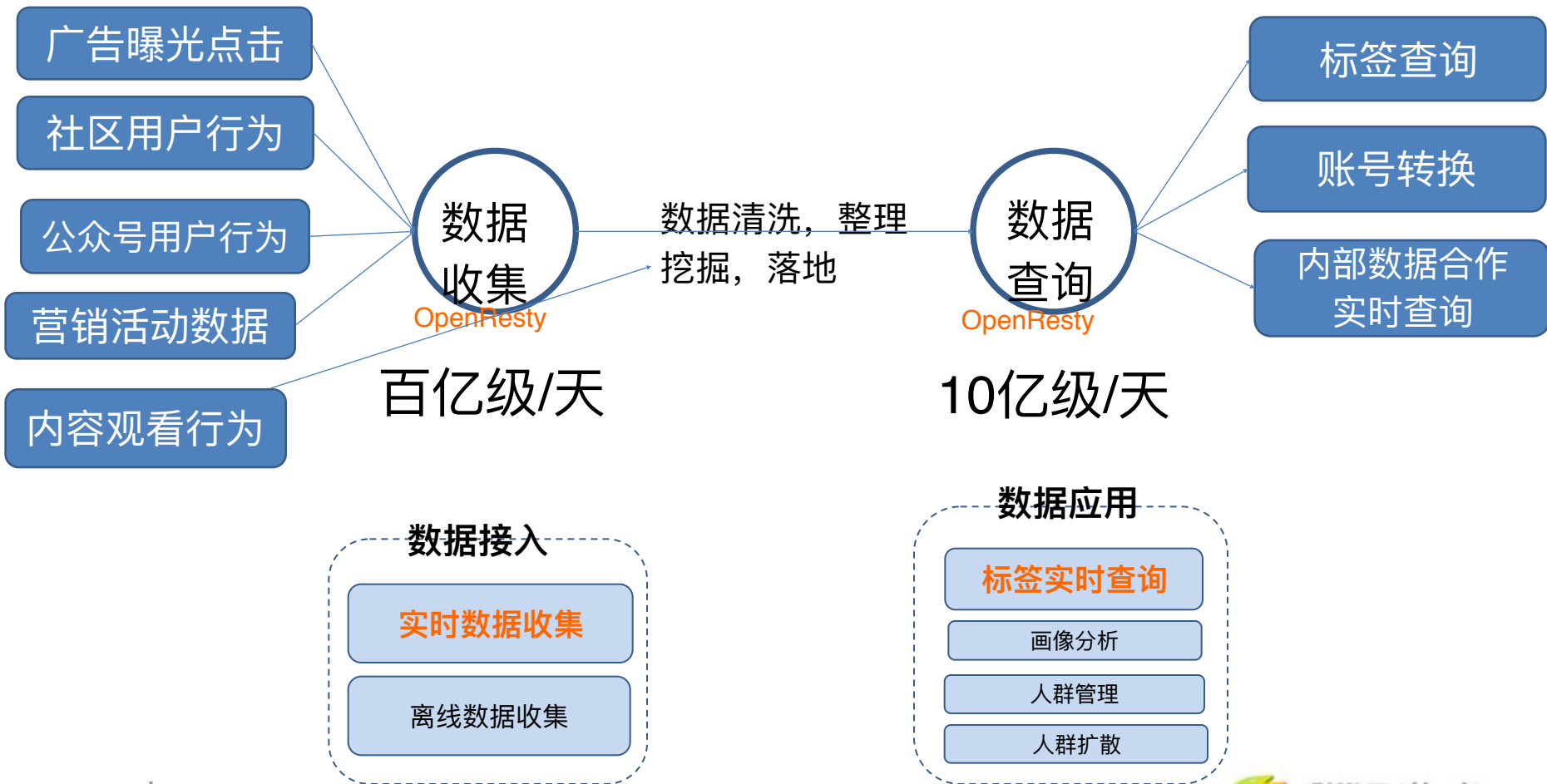
逻辑层：OpenResty重构竞价引擎

并行流程优化降低时延



数据层：OpenResty重构数据相关服务

轻松搞定百亿级流量



One More Thing...

| 不是9527

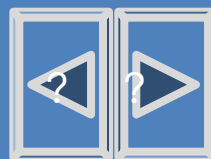
3



5



2



7



| Q&A

