委以重任的Node.js——百亿级访问实践

迪波威

个人介绍

迪波威

p 职务

原生商业推广部

Feed广告前端负责人

p Node.js实践历程

- 14年 应用于业务管理系统，实现前后端分离
- 15年 收银台使用Node后端渲染，减少白屏和转圈
- 16年 Feed广告业务建立，接手凤巢前端PHP服务(VUI)后，对前端局限性太强，着手设计Node服务
- 17年 Feed广告前端架构升级，Node替换PHP，创建Render服务
- 18年 百亿流量下，保证性能，提升研发效率，降低维护成本的持续优化

架构升级的苦与乐

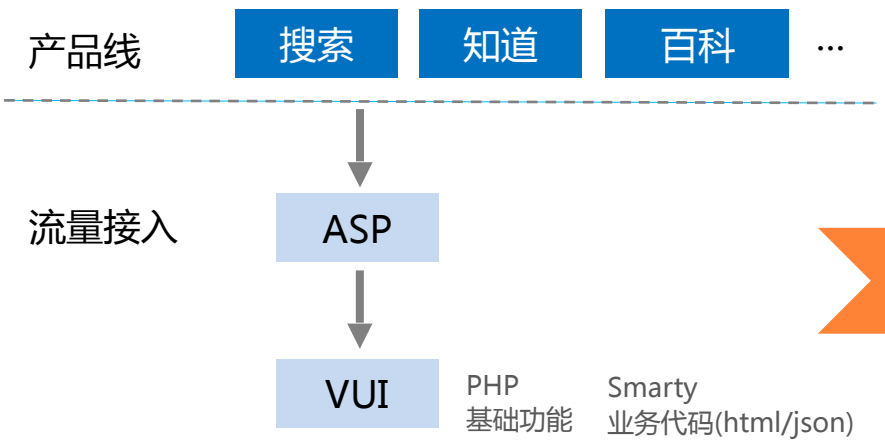
p 收益

- ✓ 技术体系化，通用性强
- ✓ 代码质量提升，Bug率降低
- ✓ 开发效率高
- ✓ 积极性提升
- ✓ 性能提升，节约机器
- ✓ 招聘容易，培养容易

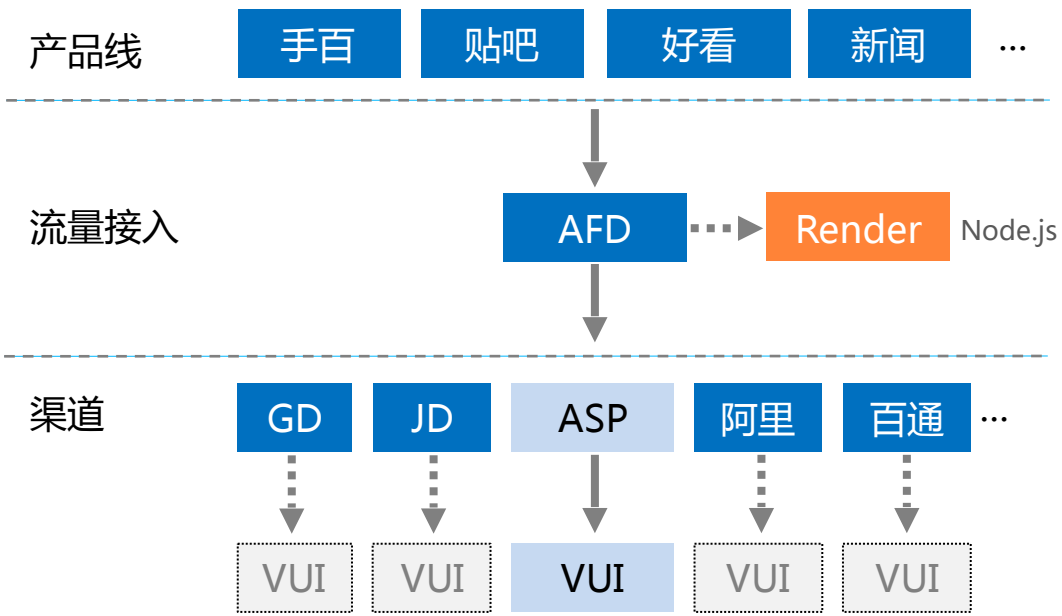
p 挑战

- 立项
- 申请资源，推动项目
- 整体排期，设立里程碑
- 减少延期，控制风险
- 打平所有业务指标，通过实验进行验证

Node取代PHP的背景



扩大规模



p 问题

✓ 技术栈偏离

✓ 与凤巢耦合

✓ 代码质量差

p 解决方案

✓ 使用Node在上游做统一物料处理及模板渲染

技术选型

技术选型 网盟render server 对比 FEX yog2

指标	render	yog2	render评分(满分5)	yog评分(满分5)	指标权重(1~10)	评价	render	yog2
默认支持模板语言	etpl,etpl	swig	4	4	10		40	40
运维平台	无	ORP	0	5	10		0	50
内核升级难度	未知，应该是只升级sojs	npm install yog2-kernel	3	4	10		30	40
RTT(无业务)	4+-1ms	6+-1ms	4	3	9		36	27
成熟度	网盟线上运行2年	百度多产品线线上运行2年	3	4	9		27	36
http/https	个人实现	express	4	4	8		32	32
协议支持	baidurpc/http/https	http/https	5	2	8		40	16
protobuf	protobufjs库(非原生C++)	protobufjs库(非原生C++)	2	2	8		16	16
进程管理	个人实现，主进程监控子进程工作	forever	4	2	8		32	16
日志	sojs.log分级日志，独立进程	yog.log分级日志，共享进程	5	4	8		40	32
模块规范	个人实现，sojs.define，sojs.using	server CommonJs, client AMD	4	5	7		28	35
集群管理	无	ORP能力，node cluster实现	0	4	7		0	28
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]		[illegible]	[illegible]
Data Fetch	无	node-ral	0	5	1		0	5
求和			65	85			443	570

省略更多对比项...

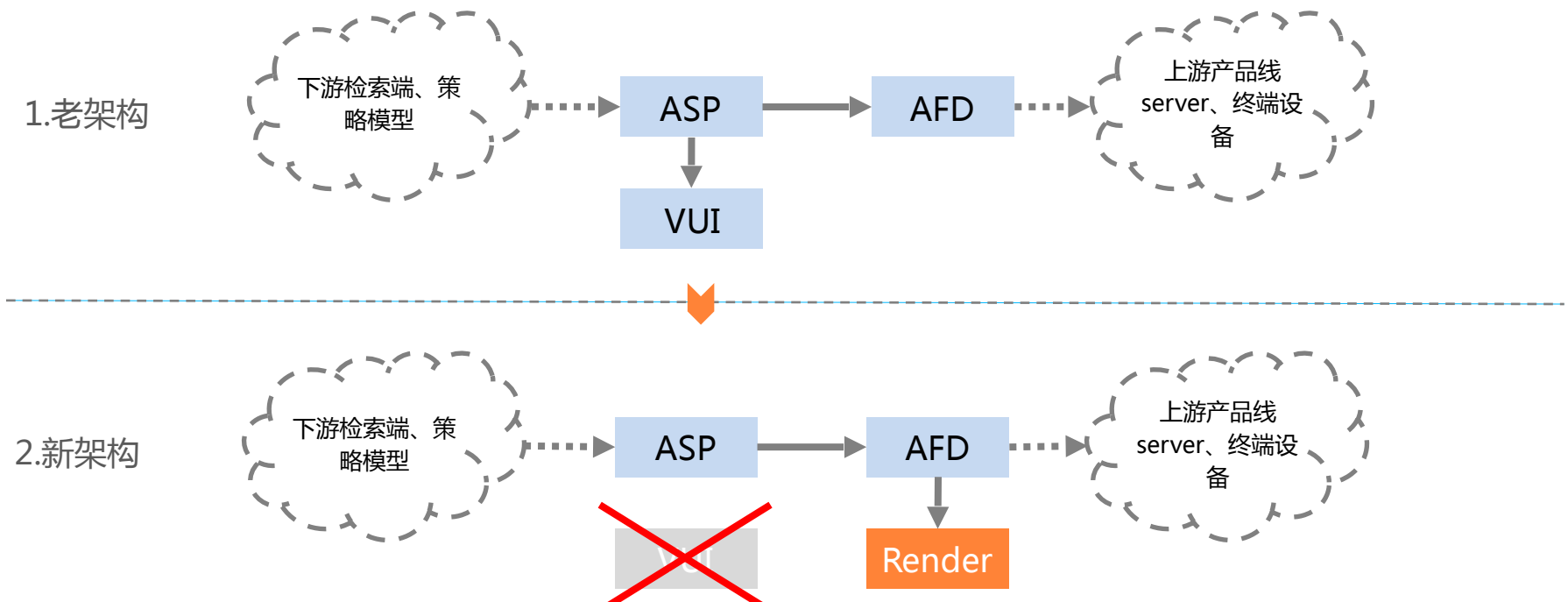
给行驶中的汽车换轮子

p 目标

- ✓ 平滑迁移 迁移过程不停工；
- ✓ 指标打平 迁移后展现、曝光、点击、收入、转化率无gap；
- ✓ 平响<10ms 终端超时限制610ms，上下游均有开销，Render 最大可用开销10ms；

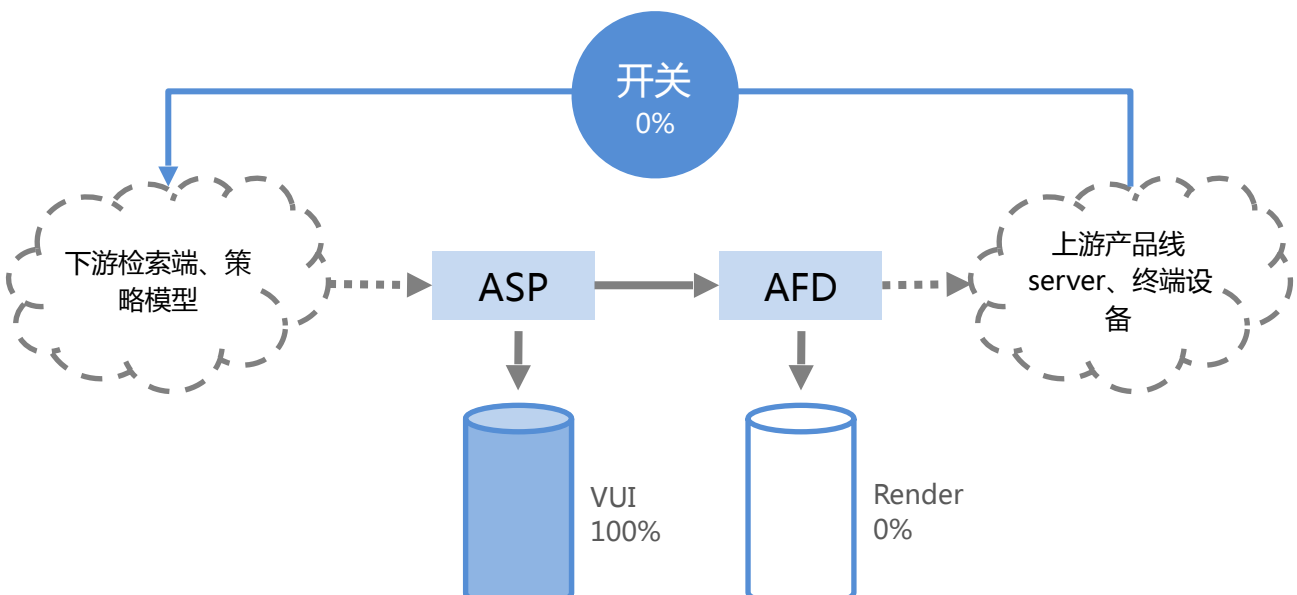
平滑迁移

- 如何确保迁移过程不停工，并可以灰度发布测试？



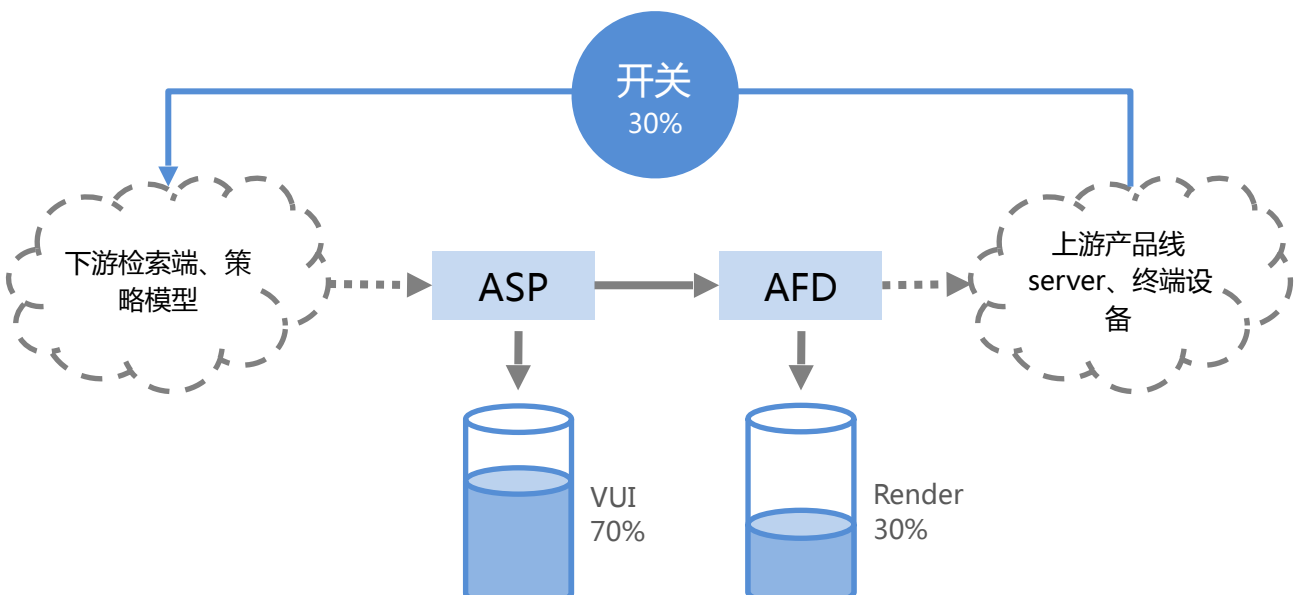
平滑迁移

- 解决方案
  - ✓ 架空旧模块 在VUI内部增加透传模板；
  - ✓ 开关联动 控制透传模板流量大小，关联上下游；



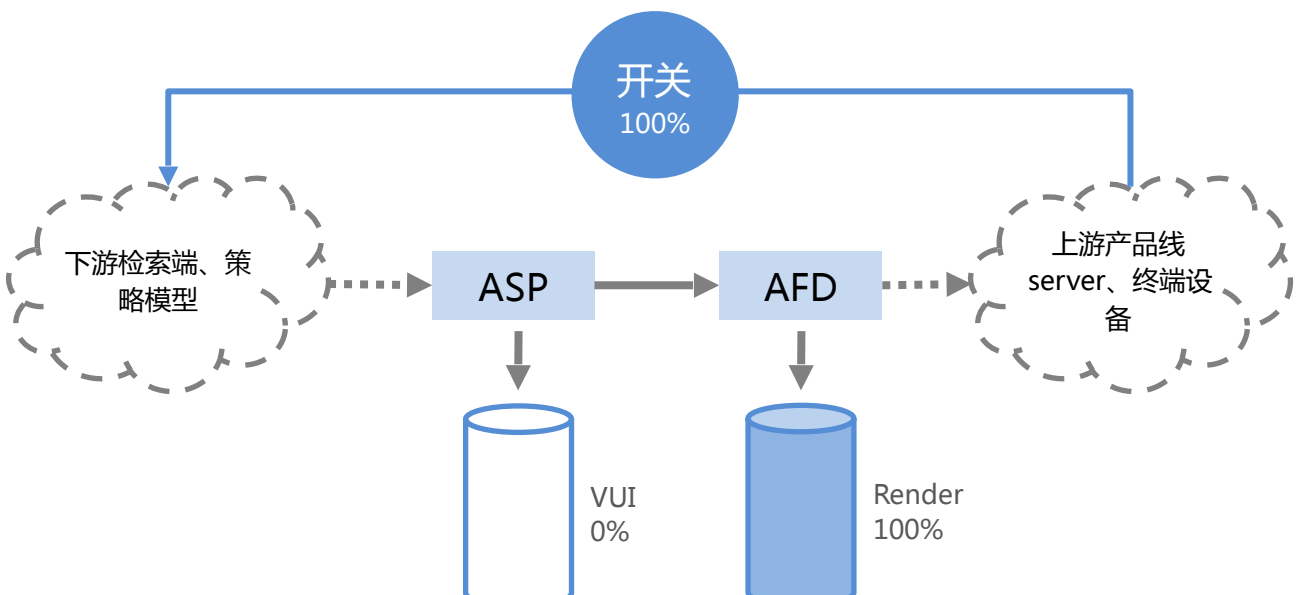
平滑迁移

- 解决方案
  - ✓ 架空旧模块 在VUI内部增加透传模板；
  - ✓ 开关联动 控制透传模板流量大小，关联上下游；



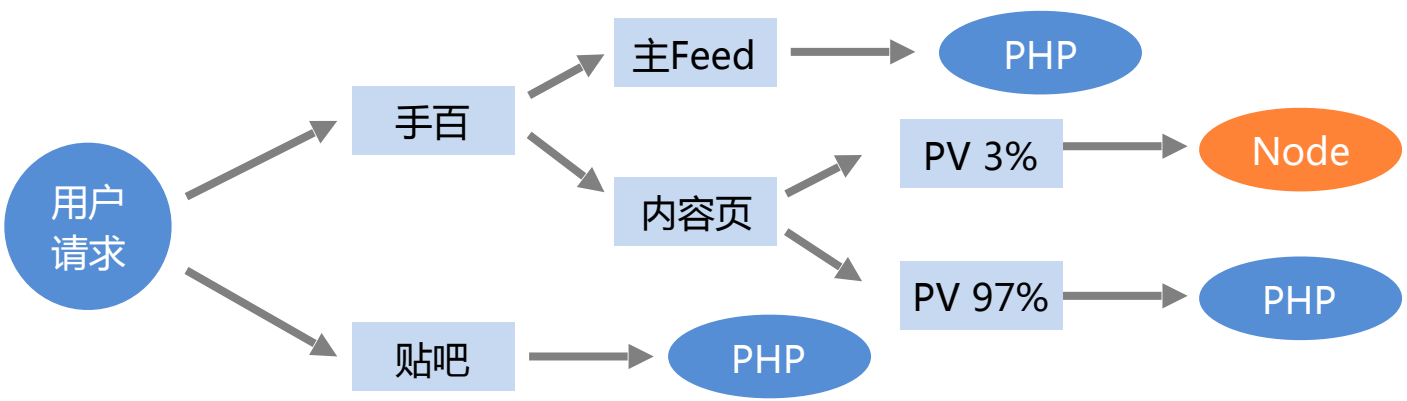
平滑迁移

- 解决方案
  - ✓ 架空旧模块 在VUI内部增加透传模板；
  - ✓ 开关联动 控制透传模板流量大小，关联上下游；



平滑迁移

- 开关设计
  - ✓ 关联产品线 可针对任意具体产品线、广告位进行控制；
  - ✓ 随机抽流 可按随机PV/CUID/BAIDUID进行分流；

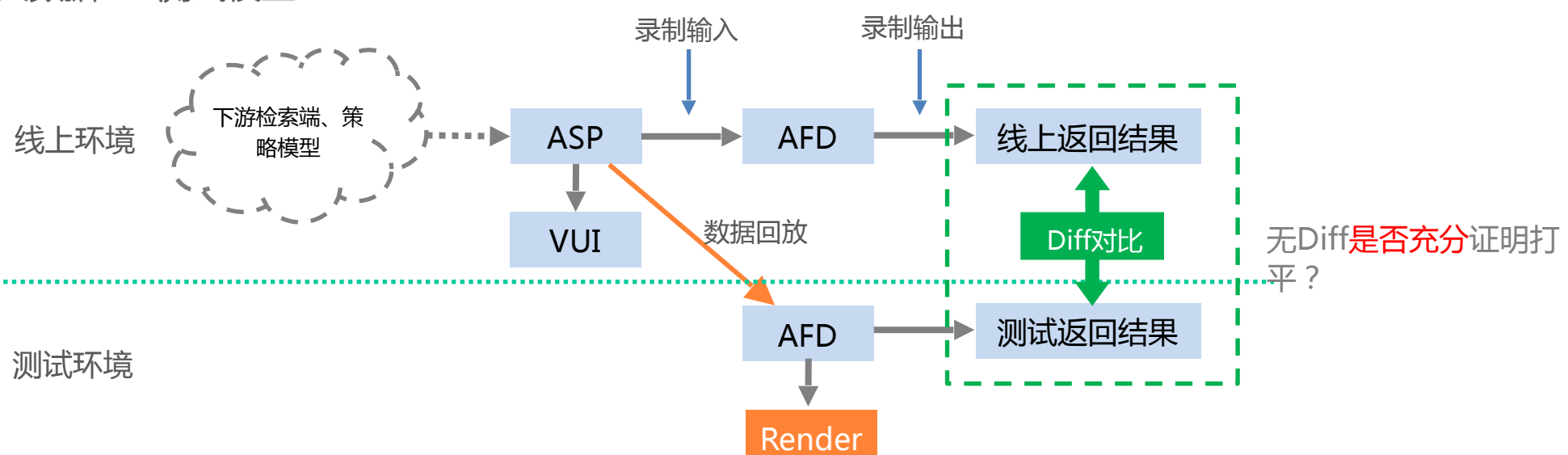


指标打平

- 解决方案
  - ✓ 线下
    - 业务逻辑梳理与迁移
    - 大数据Diff测试
    - 代码覆盖率测试
  - ✓ 线上
    - 小流量实验

指标打平

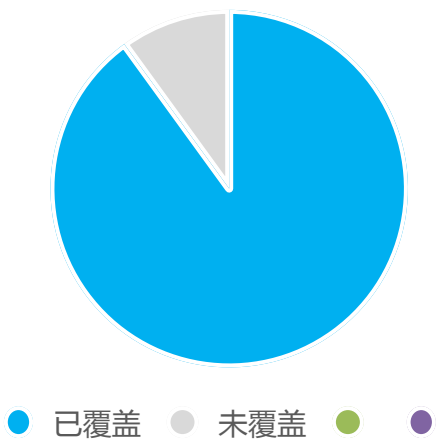
- p 大数据Diff测试
  - ✓ 原理 线上所有用户的请求分别对应一个唯一结果，将这些请求在新架构中回放，新结果与旧结果应无diff；
  - ✓ 大数据Diff测试模型



指标打平

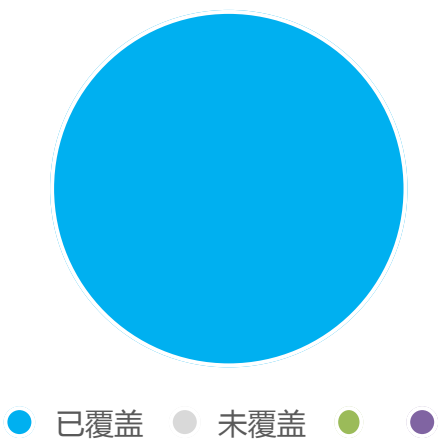
- 代码覆盖率测试

覆盖率



✗ 未覆盖的部分可能存在Diff

覆盖率



√ 无Diff

指标打平

- 使用Node自研覆盖率测试工具

/

50.15% Statements 982/1959 **31.96%** Branches 263/823 **30.11%** Functions 81/269 **52.35%** Lines 971/1855

File ▲	Statements		Branches		Functions		Lines	
app/common/	60%	18/30	42.86%	3/7	11.11%	1/9	81.82%	18/22
app/common/action/	66.67%	46/69	60%	21/35	87.5%	7/8	68.85%	42/61
app/common/lib.gql/	22.24%	129/580	6.61%	17/257	1.23%	1/81	24.07%	129/536
app/common/lib/	52.41%	305/582	39.23%	102/260	43.9%	36/82	55.37%	299/540
app/niceapp/	100%	15/15	100%	0/0	100%	0/0	100%	15/15
app/niceapp/lib/	83.93%	47/56	77.42%	24/31	100%	3/3	83.93%	47/56
app/niceapp/model/feed/16511/	100%	40/40	66.67%	16/24	100%	3/3	100%	39/39
app/niceapp/model/feed/16513/	100%	64/64	63.64%	14/22	100%	6/6	100%	64/64
app/niceapp/model/feed/16515/	100%	65/65	75%	18/24	100%	7/7	100%	65/65
app/niceapp/model/feed/16518/	100%	61/61	83.33%	20/24	100%	7/7	100%	61/61
app/niceapp/model/feed/16525/	100%	56/56	68.18%	15/22	100%	6/6	100%	56/56
app/niceapp/model/feed/3177/	100%	29/29	75%	6/8	100%	3/3	100%	29/29
conf/plugins/	67.74%	21/31	0%	0/2	0%	0/4	67.74%	21/31
conf/ral/	100%	14/14	66.67%	4/6	100%	0/0	100%	14/14
plugins/bdservice/	100%	4/4	100%	0/0	100%	0/0	100%	4/4

任务报告

任务信息

创建时间	备注	render地址	筛选项	运行数/全部数据	无diff数据数/运行数
2018-07-06 10:08:50	王斐-宏替换升级	1[redacted]5.154:8210	shoubaiapp/all	42699/42603	26167/42699 61%

diff结果

分字段diff结果　全部diff结果

monitor_url　prefetch_html　prefetch_upload　_0

筛选字段：**ads[0].content[0].prefetch_html**
每个字段最多显示五条典型结果

物料id	操作
5b3de3a7ec1775c2333c06c2	查看
5b3de3e4ec1775c2333c1f4d	查看
5b3de444ec1775c2333c30c1	查看

< 1 >

任务报告

diff结果

拷贝基准req　拷贝基准res

基准　对照


```
{
    ads: [
        0: {
            content: [
                0: {
                    prefetch_html: "http://jingtai.youyuan.com/html/2017/yw/201707/31/page_1_ay/index.html?from=104552",
                    prefetch_upload: "1"
                },
            ],
        },
    ],
}
```


确认

指标打平

- 使用Node自研覆盖率测试工具

任务报告

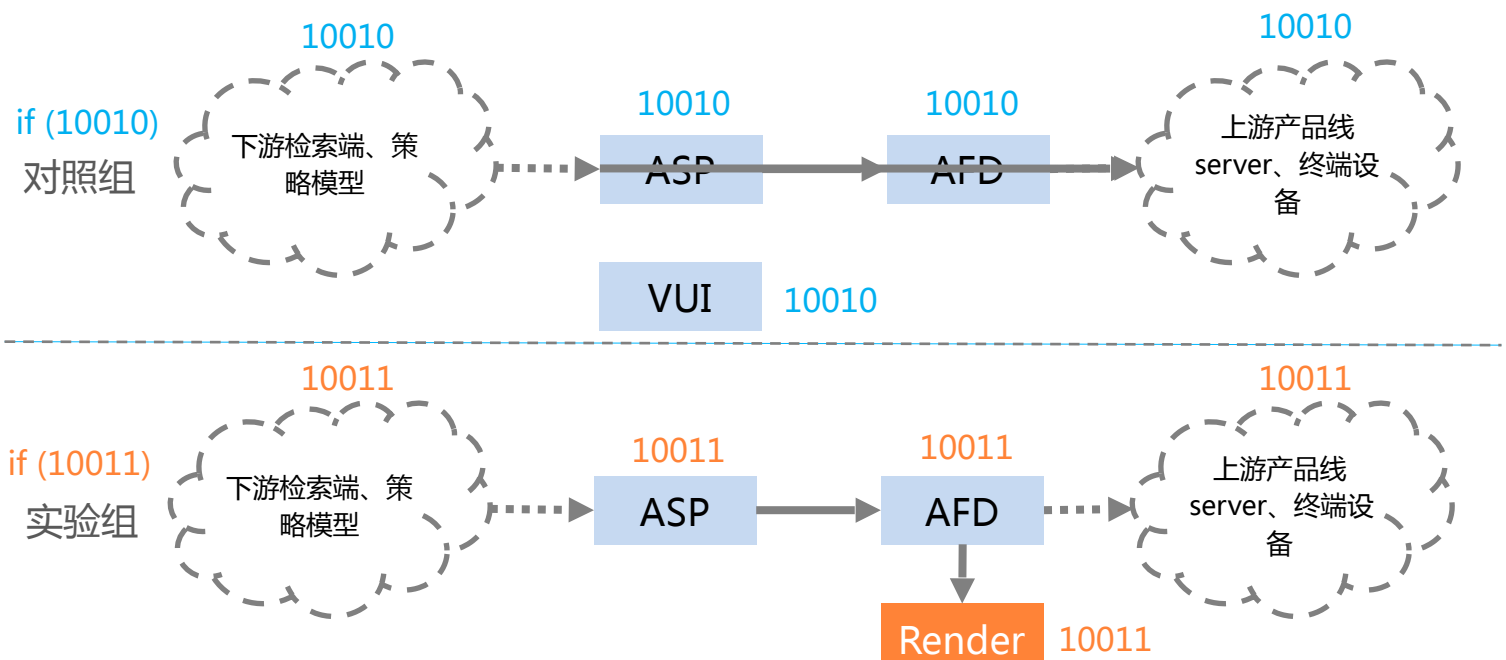
任务信息

创建时间	备注	render地址	筛选项	运行数/全部数据	无diff数据数/运行数
2018-07-09 05:33:38	bp环境	gz[redacted]-orp-feed-a[redacted]29.gzhxy:8260	niceapp/all	1230/1230	1230/1230

确定

指标打平

- p 实验平台
  - ✓ 抽流 按实验平台的配置抽取指定流量的实验组和对照组，下发实验ID
  - ✓ 关联日志 上下游的所有模块打日志都需要带上实验ID，数据可对比



对比指标是否打平

10011 : 10010

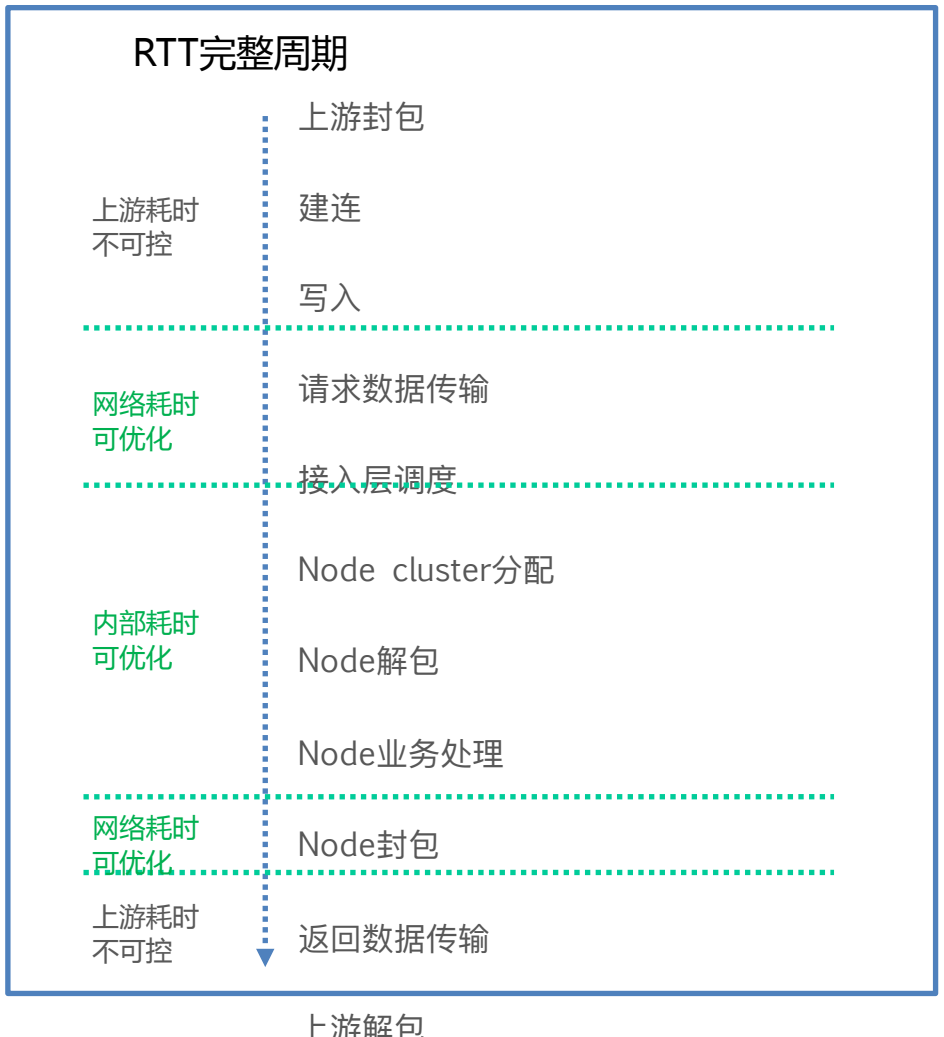
pv:	0.0001%
epv:	0.0143%
show:	0.0164%
eshow:	0.0136%
click:	0.0108%
charge:	0.0173%
ctr:	-0.0047%

指标打平

- 实验方法
- ✓ 如何确保实验数据置信
  - 第一步 抽流：抽取两组等量的流量分配实验id，流量大小需确保样本充足
  - 第二步 空跑：实验组与对照组做无代码diff的指标比较，确定波动范围，波动太大需加大流量
  - 第三步 实验上线：让实验组走新的代码逻辑，对比数据是否打平，如果在空跑波动内，进入下一步
  - 第四步 反向对照：将实验与对照进行对调，如果gap颠倒说明问题与实验组相关

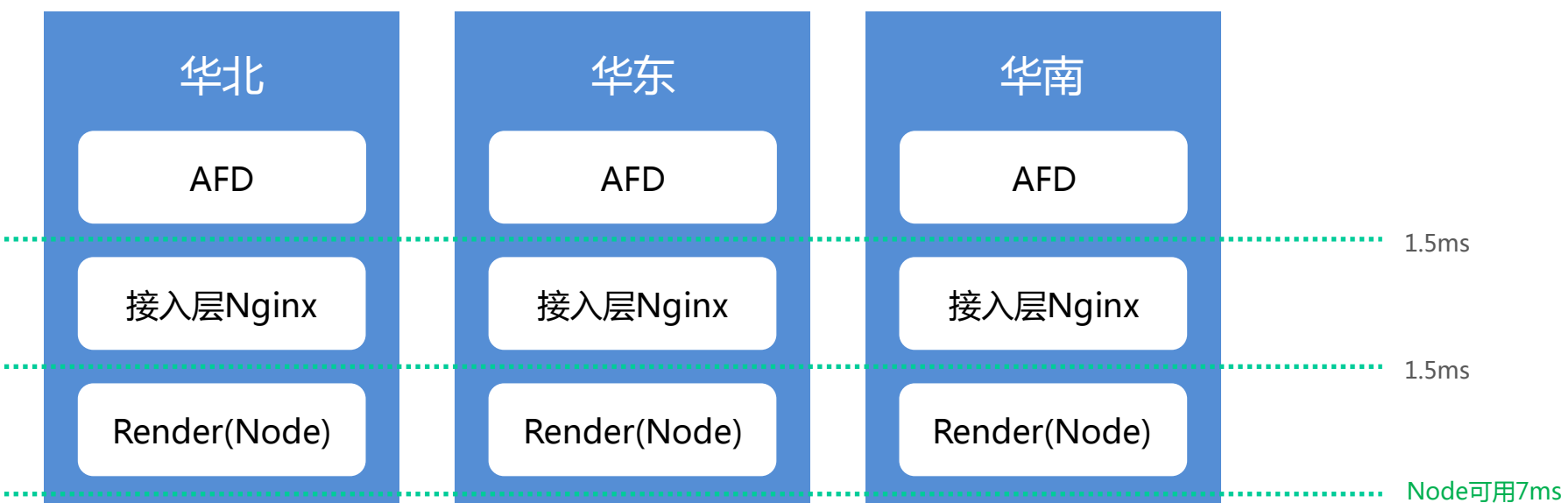
平响<10ms

- p RTT拆解
  - ✓ 网络耗时
  - ✓ Node内部耗时



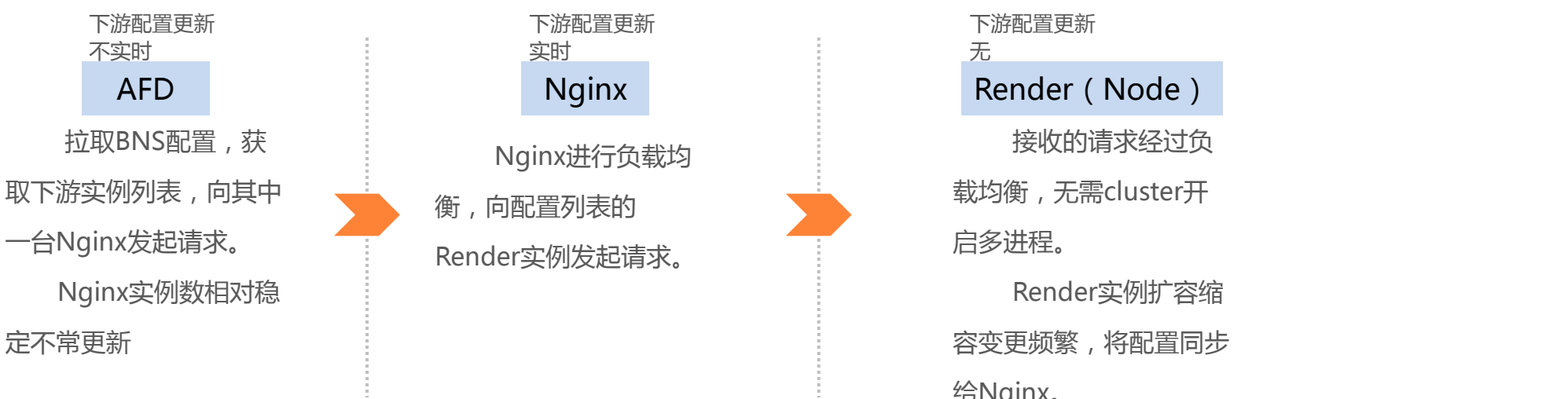
网络耗时解决方案

- 同机房部署
- ✓ 上下游关联模块
  - 上游 - AFD
  - 下游 - 无



网络耗时解决方案

- 接入层的作用
  - ✓ 负载均衡 – 确保每次请求分配到相对空闲的实例上，避免出现热点机
  - ✓ 探活 – 当Render服务故障时，流量调到其他机器
  - ✓ 缓冲配置 – 减少AFD下游的配置变动，提高稳定性
  - ✓ 拷贝流量 – TCPCopy指定QPS到BP环境，用于演练



Render内部耗时解决方案

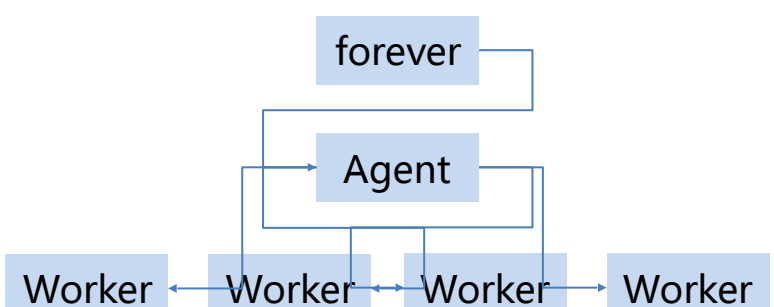
- 禁用Cluster，降低quota

  接入层已解决负载均衡和探活的问题，故障重启交给运维平台自动处理，可节省Cluster性能开销

放弃

CPU：4核

实例数：n



采用

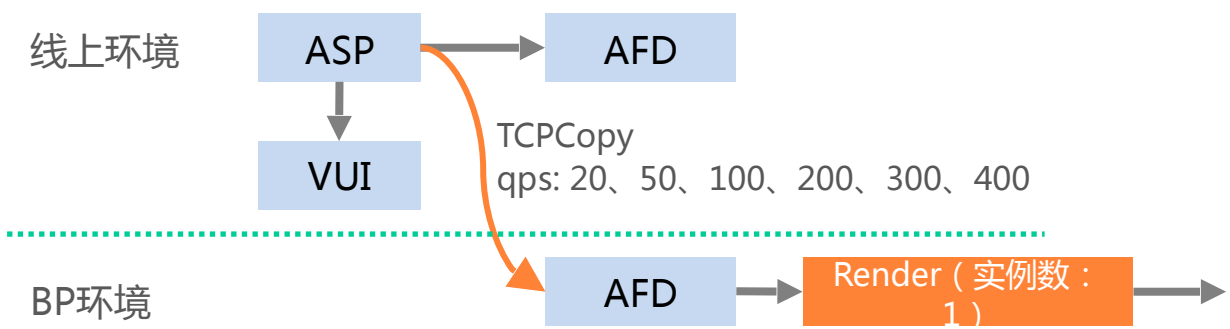
CPU：1核

实例数：4n

Master

Render内部耗时解决方案

- 压力测试
  - ✓ 使用TCPCopy拷贝线上流量，调整压力，在BP环境监控性能指标



	qps <= 400	qps > 400
内部平响	2ms	>3ms
AFD平响	5ms	>6ms
AFD成功率	99.999%	99.9%

Render内部耗时解决方案

- 影响QPS极限的决定性因素是什么
  - 木桶原理：CPU、内存、磁盘IO、网络IO的短板决定QPS极限

✓ CPU——决定性因素

木桶短板，最先撞线，重点关注指标，cpu idle（空闲），cpu load（负载）

✓ 内存——无影响

稳定不变

✓ 磁盘IO——可控

写日志操作会影响磁盘IO，可精简日志，注意物理机整体磁盘IO

✓ 网络IO——可控

控制网络请求与返回的包大小

Render内部耗时解决方案

- 单实例QPS设计上限

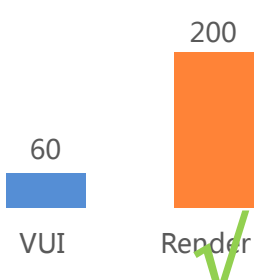
  极限qps: 400/实例

  冗余qps: 100/实例 (防止重试雪崩)

  峰值qps: 300/实例

  均值qps: 200/实例

- QPS对比



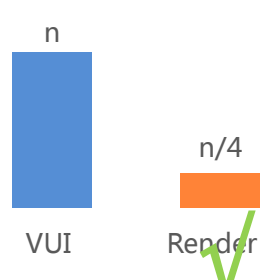
- 计算资源占用

  总qps=pv/24小时/60分钟/60秒

  单实例均值qps=200

  申请实例数=总qps/单实例均值qps

- 资源对比



百亿级流量下的考验

- 线上性能稳定性

上游请求成功率

间隔 10分钟	当天值
17:00 - 17:10	99.9997
16:50 - 17:00	99.9995
16:40 - 16:50	99.9996
16:30 - 16:40	99.9996
16:20 - 16:30	99.9996
16:10 - 16:20	99.9996
16:00 - 16:10	99.9991
15:50 - 16:00	99.9994

Render

设计上限：400QPS/实例

平响5毫秒

间隔 10分钟	当天值
17:00 - 17:10	5.1979
16:50 - 17:00	5.1795
16:40 - 16:50	5.1887
16:30 - 16:40	5.2643
16:20 - 16:30	5.8167
16:10 - 16:20	6.1095
16:00 - 16:10	6.27
15:50 - 16:00	6.2567

百亿级流量下的常规问题

p 预案与演练

✓ 作用

- 对于异常流量的模拟与应急方案的演练

✓ 方法

- TCPcopy，使用BP环境

p Trace Log

✓ 作用

- 复盘线上故障与快速定位问题

✓ 方法

- 保持录制一段时间的输入输出日志

p Graceful shut down

✓ 作用

- 防止上线过程重启服务导致成功率下降

✓ 方法

- 先停止listen，延迟kill进程

p 监控与报警

✓ 作用

- 监控线上指标波动，超出阈值人工介入

✓ 方法

- 日志接入监控报警系统

百亿级流量下的衍生问题

p 随着流量规模的扩大，伴随业务需求增加，团队规模扩大，将面对以下问题

✓ 提升开发效率

- 减少重复的开发，尽量配置化

✓ 提升代码质量

- 从灵活向严格转变
- 规范业务形态

✓ 降低维护成本

- 文档清晰且同步，业务高共识

✓ 控制影响面

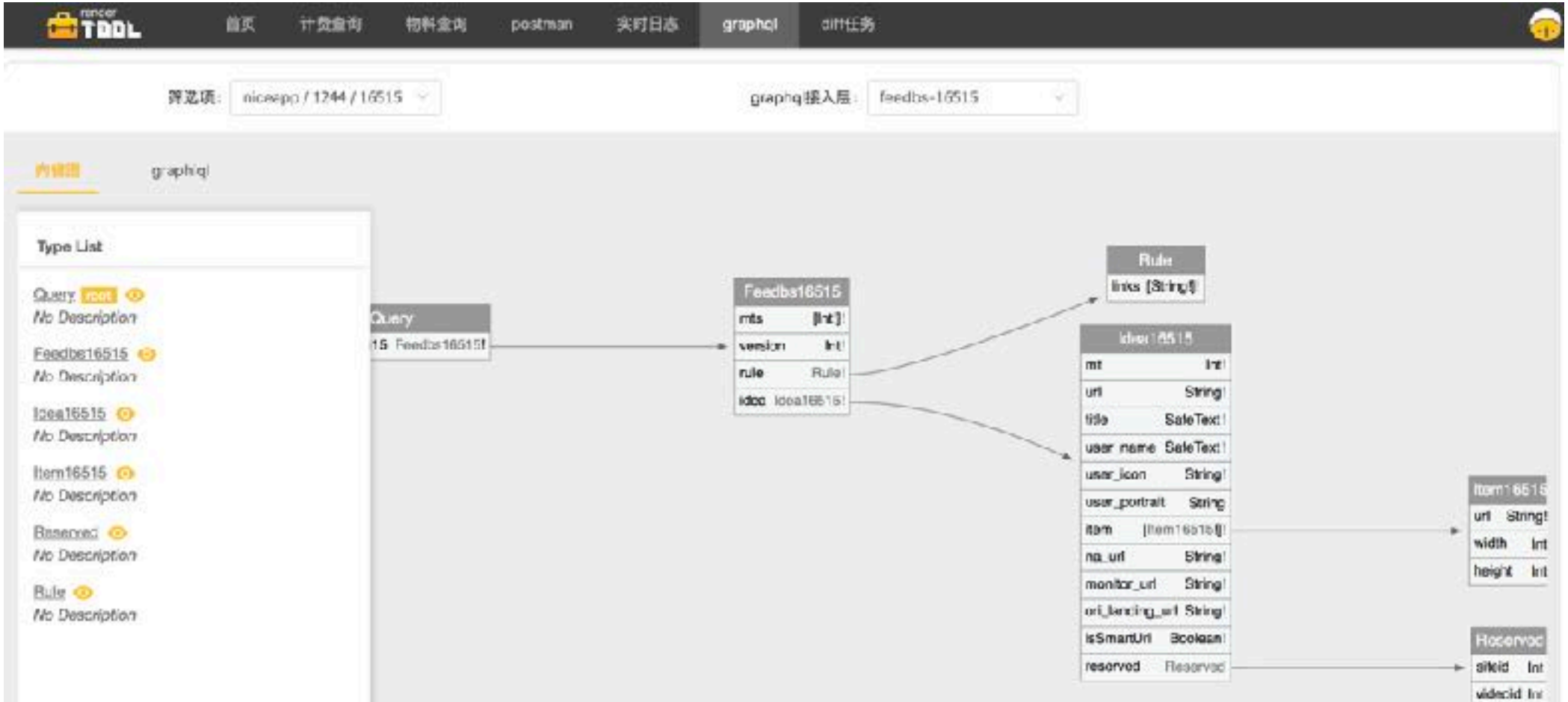
- 代码库拆分独立上线，业务层级清晰

持续优化
二次升级

Render2.0

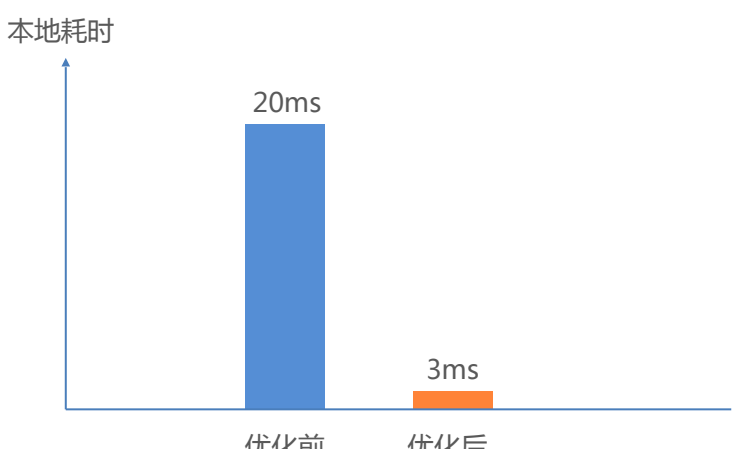
GraphQL

- 配置化开发
- 标量定义，业务流程标准化
- 代码即文档



GraphQL的性能问题

- 动态查询带来了重复计算，浪费CPU
- ✓ 解决方案
  - 内存换CPU，服务启动时预先缓存全部query路径，因为业务逻辑可数，所以内存使用量可控
- ✓ 效果



Node赋予前端的独立性

p 独立完成常规业务平台需求

- ✓ 前后端分离 独立迭代，自主上线，后端复用性高；
- ✓ 接入层自主 域名、路由自主管理，灵活创建应用；
- ✓ 开发效率提升 沟通成本低，设计灵活；

Node赋予前端的独立性

- 独立优化空间大
  - ✓ SSR 后端渲染方案灵活使用；
  - ✓ CDN通用化 开发上线对CDN无感知，上线自动映射；
  - ✓ 落地页预加载 广告主落地页打开慢，使用Puppeteer预取落地页，缓存在CDN；

Node赋予前端的独立性

- Headless的应用
  - ✓ 爬虫 从广告主落地页提取信息，供策略模型优化；
  - ✓ 自动化测试 模拟用户行为做测试；

Node赋予前端的独立性

- 自建落地页

✓ 独立存储数据 前端独立存储数据和查询，如落地页表单提交收集客户信息；

✓ 
需求，快速创建广告、落地页；

Q&A

投简历加微信群

THANKS