APACHE APISIX

Apache APISIX 在虎牙多云接入层实践

周健

虎牙中间件团队

APACHE APISIX

个人介绍

- 虎牙中间件负责人
- Nacos commitor
- APISIX contributor

虎牙公司在实时内容创作与直播互动技术领域持续创新，推动直播平台多元发展

APACHE APISIX

CONTENT

APACHE APISIX

01 虎牙多云接入背景

直播场景的 web 接入，在可用性，灵活性和成本之间权衡

APACHE APISIX

虎牙多云接入关注点

灵活适应不同场景

- 非app流量，涉及内外部web业务和api请求， qps约20万
- 特定地域和云上接入场景

灵活运用云厂商资源

- 快速扩容，应对突发流量
- 不绑定厂商，资源与成本权衡

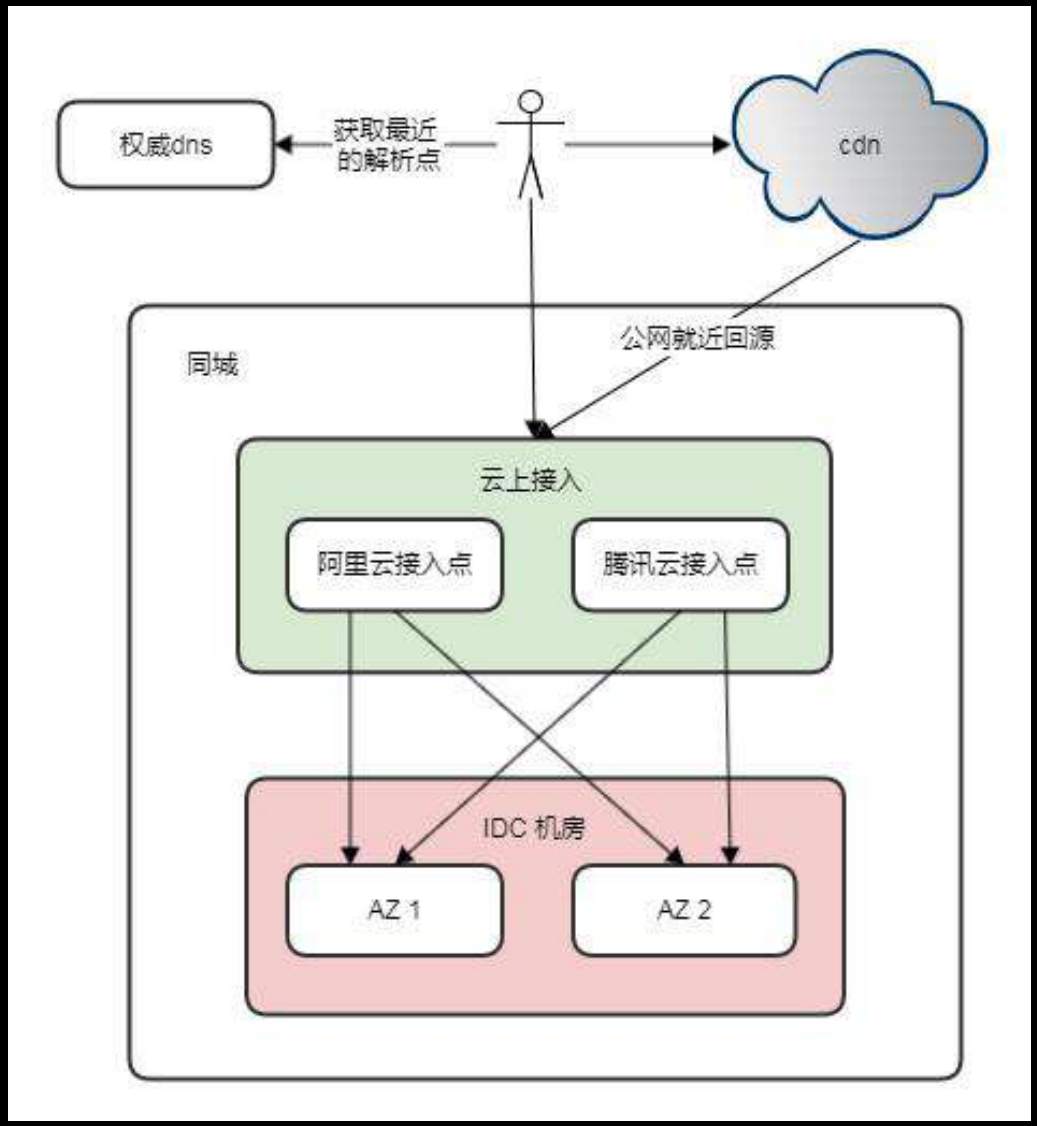
可用性保证

- 单云故障可切换，同城多活

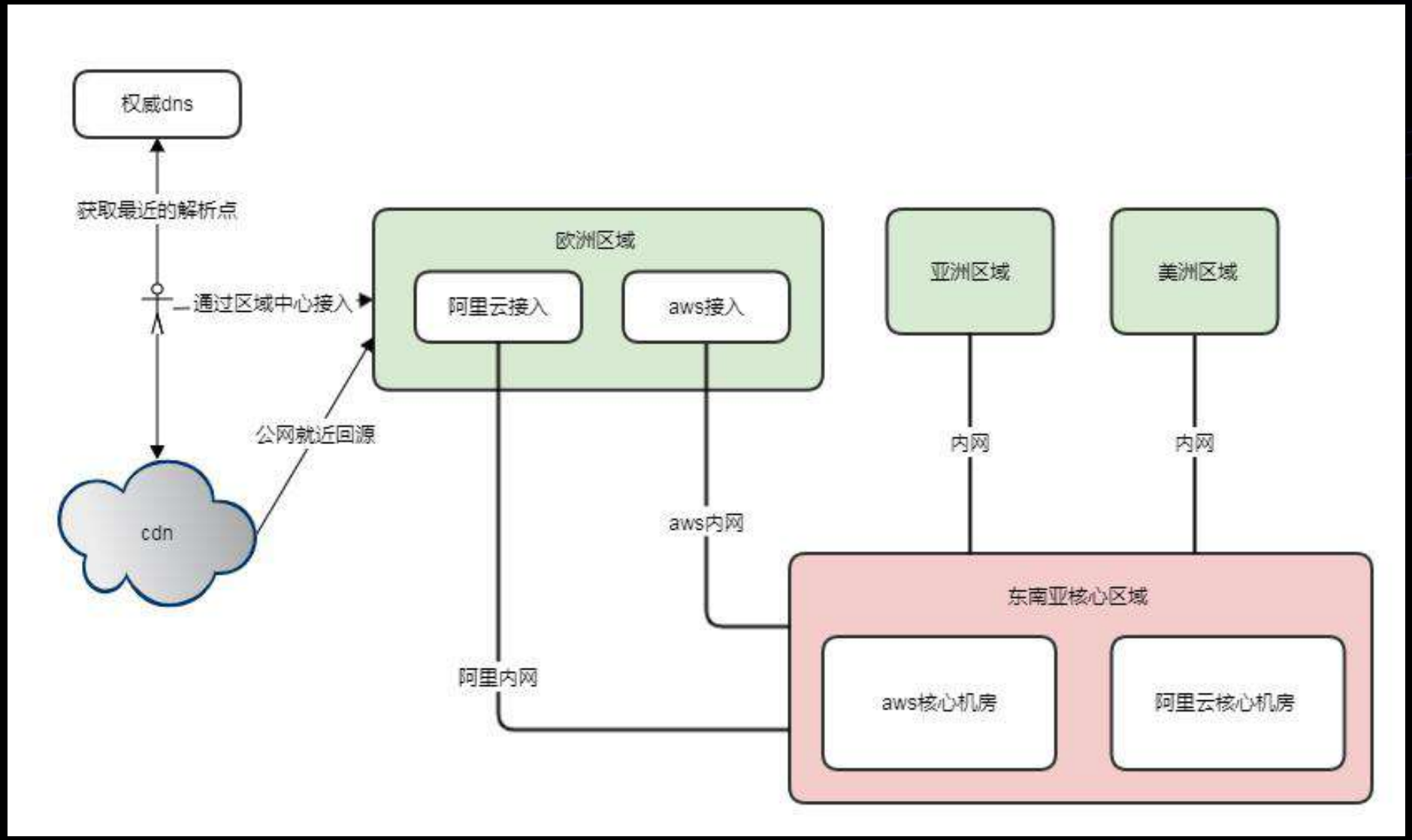
攻击防护， 阻挡异常流量

- 云上防护服务
- 7层异常流量拦截

虎牙多云接入- 国内


权威dns
获取最近的解析点
cdn
公网就近回源
同城
云上接入
阿里云接入点
腾讯云接入点
IDC 机房
AZ 1
AZ 2

虎牙多云接入- 海外


权威dns
获取最近的解析点
通过区域中心接入
欧洲区域
阿里云接入
aws接入
亚洲区域
美洲区域
公网就近回源
cdn
内网
内网
aws内网
东南亚核心区域
阿里内网
aws核心机房
阿里云核心机房

APACHE APISIX

02

技术选型和部署

当前场景的挑战， 发挥 APISIX 的优势

APACHE APISIX

多云接入点管理

1、云设施

- 区域性网络问题是常态
- 资源在特定时间和区域会有瓶颈

2、实时更新

- 同城直接回源业务节点，节点更新变化及时性
- 弱网环境，多层代理，及时屏蔽异常线路

3、统一配置发布方案

- 配置中心 + agent
- 中心化 vs 集群化， 成本和高可用的权衡

APACHE APISIX

多类型资源部署

1、资源类型

- 容器、物理机、虚拟机
- 云厂商，IDC

2、版本，信息管理

- 线上，研测
- 代码版本， 灰度版本

3、用途划分

- 业务隔离
- 公网， 内网

基于原生 nginx 构建

1、管理复杂

- 配置以域名维度划分，管理分散
- 证书更新，每个节点更新文件，并且 reload

2、流量调度困难

- 4000+域名，配置批量修改， 生效缓慢， 可靠性难以保证
- Reload 后，影响流量均衡

3、难以满足各类场景，进行扩展

- 异常流量拦截
- 动态白名单
- 原生 lua 插件维护困难

APACHE APISIX

技术选型

功能	APISIX	Kong	Envoy
动态 router	支持	支持	支持
动态 upstream	支持	支持	支持
健康检查	支持	支持	支持
定制负载均衡策略	支持	不支持	不支持
4层 & 7层 proxy	支持	支持	支持
扩展方案	插件，多语言	插件	插件，多语言
配置生效时间	基于事件触发，ms 级生效	拉数据库， 5 秒	通过基于 grpc stream 通道的 xds api，ms 级生效
架构方案	Nginx + etcd	Nginx + postgress	Sidecar + proxy
单核 qps	18000	1700	15000
配置复杂度	低	低	高

APACHE APISIX

基于 APISIX 流量治理

1、转发屏蔽

- Upstream node 元数据打上机房信息
- 自定义负载均衡 + 全局插件， 屏蔽特定机房流量

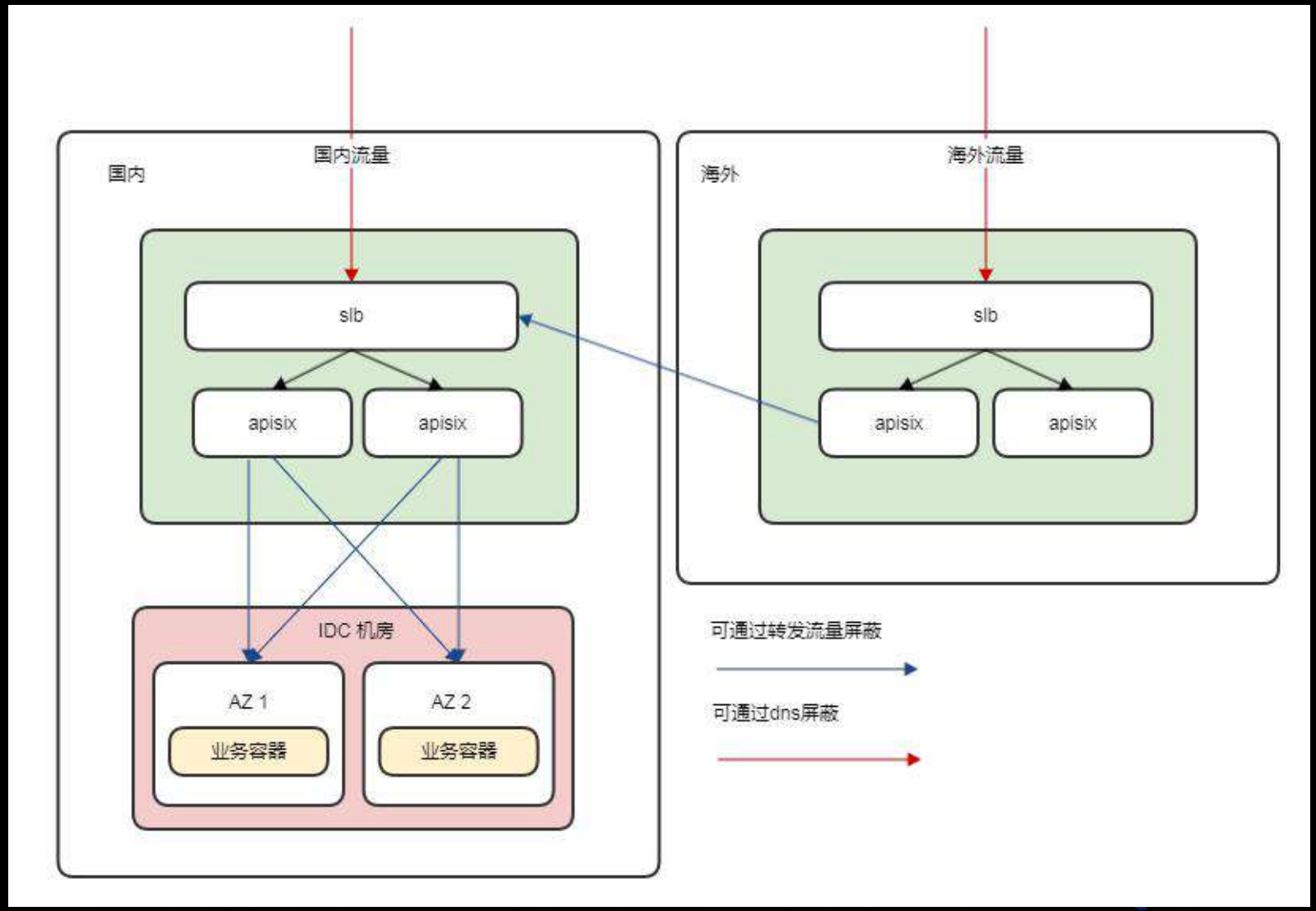
2、异常流量屏蔽

- 多维度流量控制和屏蔽
- 默认配置 tcp 和 http 探测逻辑

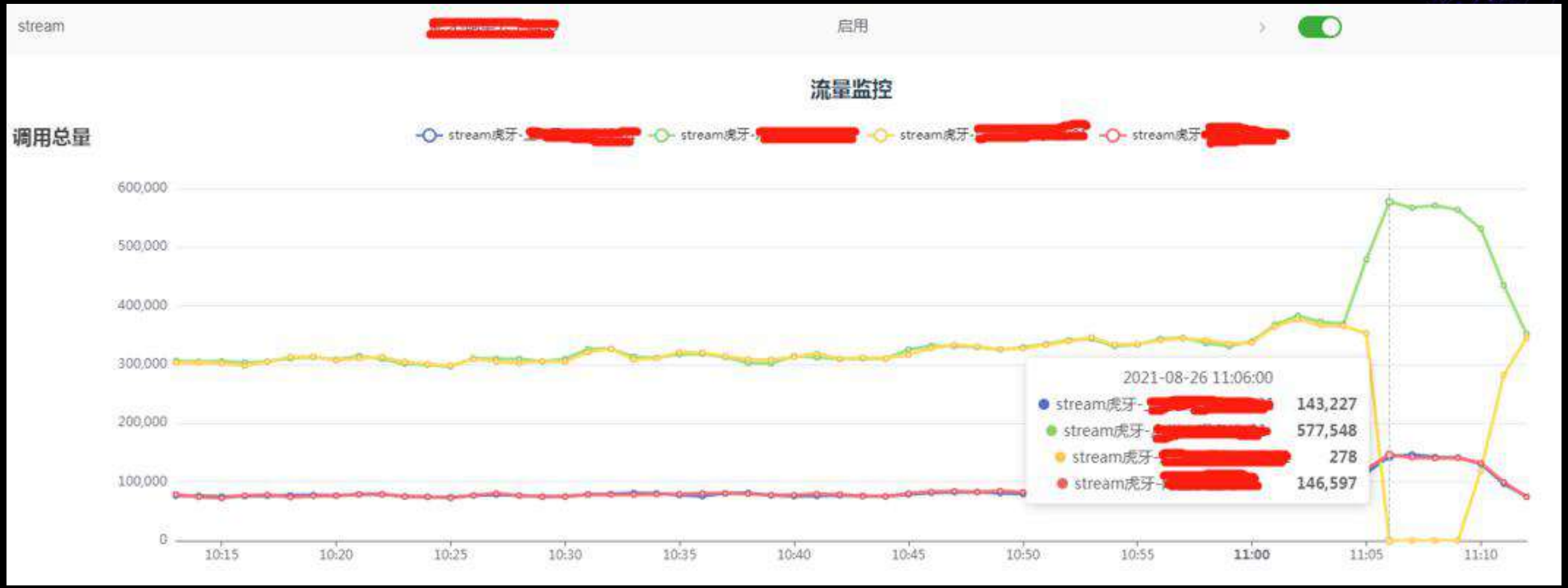
3、 海外流量调度

- 路由策略下发
- 用户标签转发

基于 APISIX 流量治理



基于 APISIX 流量治理



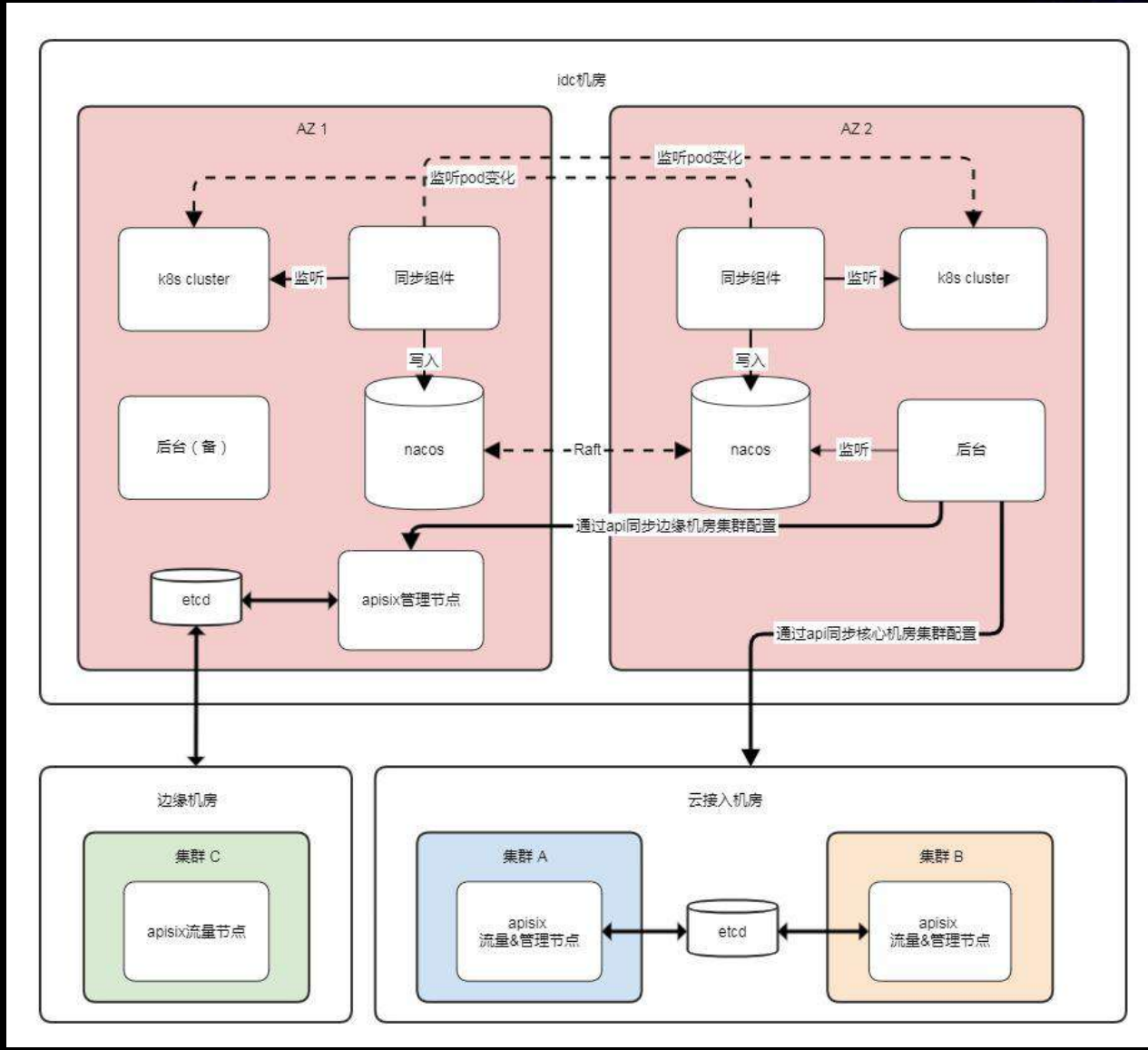
基于 APISIX 管理部署

1、统一发布

- 核心机房统一监听节点变化
- 后台统一发布，记录发布版本

2、快速屏蔽，复制

- 配置发布异常， 屏蔽接入点
- 快速复制当前集群



APACHE APISIX

基于 APSIX 资源管理

1、节点管理

- 流量节点可以同时为管理节点， 可以部署在边缘机房
- 管理节点位于核心机房， 与 etcd 同机房
- 区分集群 etcd.prefix: /apisix/${{CLUSTER_NAME}}

2、通过 endpoint 管理信息

- /env
- /v1/healthcheck

3、证书更新

- 通过 api，更新默认证书和特定证书

APACHE APISIX

03 Apache APISIX 落地实践

拥抱开源， 解决实际场景问题

APACHE APISIX

实践分享

1、版本升级，全量覆盖

- 关键节点
- 移植开源ci任务

2、历史 nginx 功能兼容

- 兼容 nginx location 匹配规则
- 配置文件功能转化为插件

3、问题定位

- 死循环
- 插件新旧版本兼容

APACHE APISIX

APISIX 落地实践

整体落地时间线

2021.02

逐步灰度

2.3 版本

2021.04

升级到 2.5 版本

修复死循环问题

2021.08

升级到 2.8 版本，修复插件兼容性

问题，支持独立的 keepalive pool

和自定义负载均衡

2021.11

升级到2.9版本

real ip 支持，全量覆盖

APACHE APISIX

持续版本迭代

```
huya_test() {
    ## apisix核心功能回归
    ## FIXME 这里剔除掉ipv6和有外部依赖，而且我们没有用到的功能
    ## /node/upstream-domain-with-special-dns.t, /cli/cli.t, ipv6 与ipv6功能相关
    ##  /misc/,   /core/config_etcd  和  etcd tls相关, 需要外部依赖
    ##  /discovery/  /control/discovery.t 和服务发现相关, 需要外部依赖
    ## /stream-plugin/ip-restriction, 需要ip-restriction支持stream
    export FLUSH_ETCD=1
    export PERL5LIB=.:$PERL5LIB
    find t -name '*.t'  \
    | grep -v /node/upstream-domain-with-special-dns.t \
    | grep -v ipv6 \
    | grep -v radixtree-sni \
    | grep -v /cli/cli.t\
    | grep -v /plugin/\
    | grep -v /stream-plugin/ip-restriction*\
    | grep -v /misc/\
    | grep -v /discovery/\
    | grep -v /control/discovery.t\
    | grep -v /core/config_etcd.t\
    | xargs  prove -Itest-nginx/lib
```

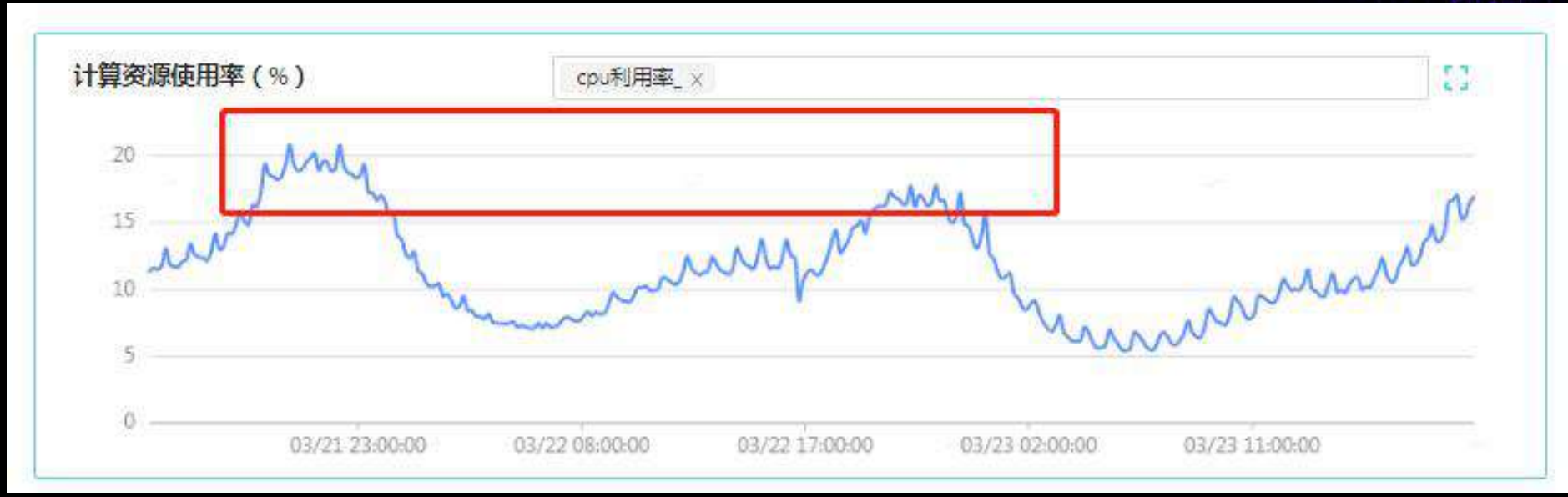
参照 linux_openresty_common_runner.sh，将开源 ci 用例迁移到内部

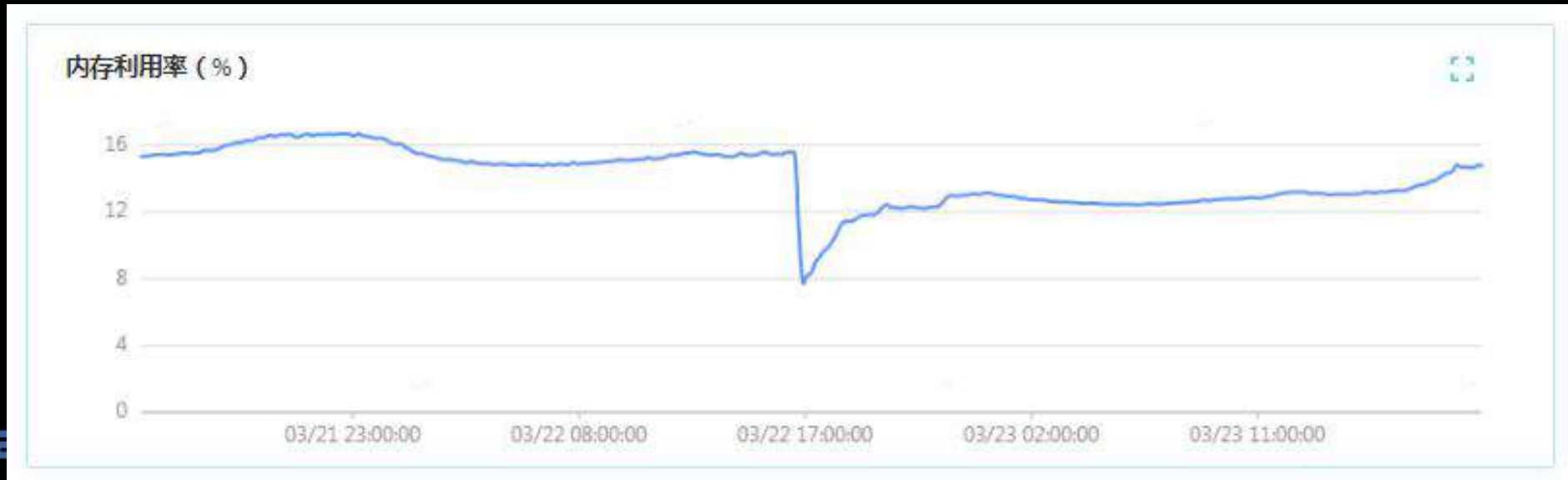
- 开源 1~2 个月一个版本
- 内部插件和定制的 openresty 镜像在新版本稳定
- 参与开源社区， 同步最新版本

兼容 nginx location 匹配规则

选用 radixtree_host_uri

- 4000+ 域名， 大部分名字的是默认路径 /
- 与 radixtree_host 相比， cpu 下降 30%， 内存下降 15%





APACHE APISIX

兼容 nginx location 匹配规则

- 禁止内部重定向， 否则会复杂化路由匹配， 难以管理
- Location 唯一

localtion	uri	Vars match	priority
= match	match		
^~ match	/match*		
~ or ~* match	/*	Uri ~ or ~* match	从1000开始 ,优先级按照配置顺序累加
match	/*	Uri ^~ match	从0开始 ,优先级按照match的长度累加

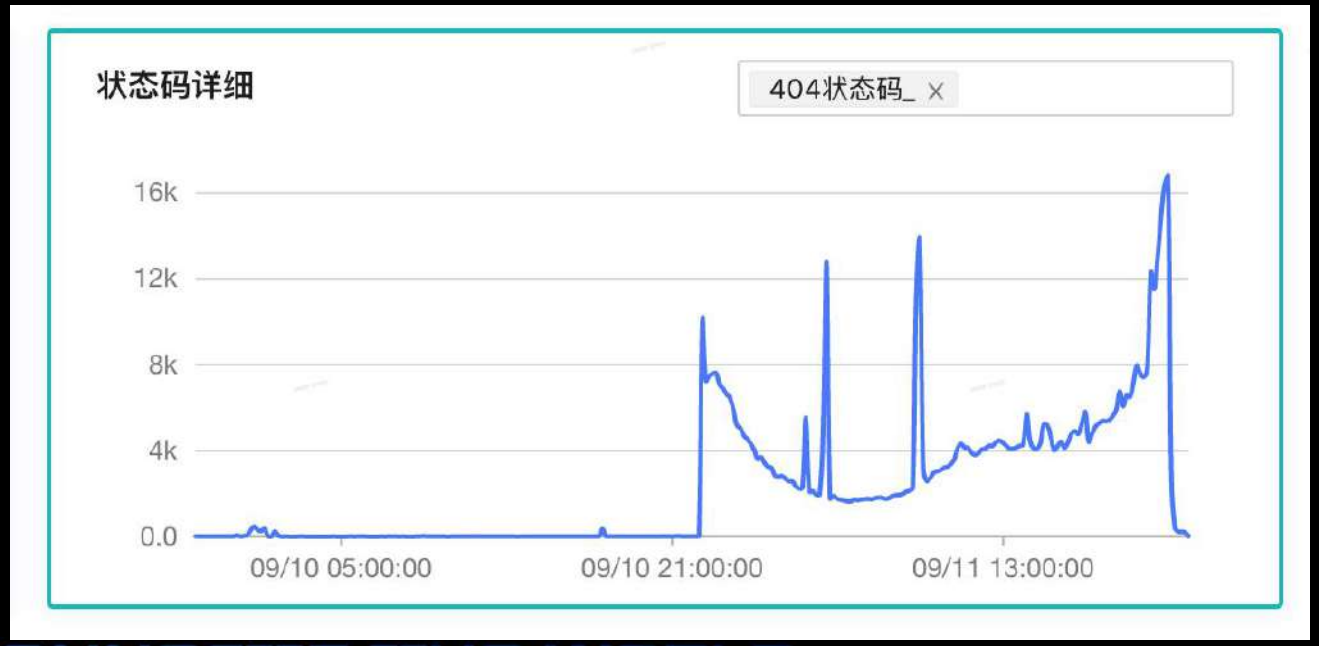
线上问题定位

- 2.5 版本 升级 到 2.8 版本， 逐个节点灰度， 插件添加字段后， 旧节点无法更新成功

- Etcd 触发 compact 的时候， 触发全量数据加载， 会导致部分 route 加载失败，出现 404

```
failed to fetch data from etcd: failed to check item data of [/apisix/****/routes] err:failed to check the configuration of plugin ip-restriction err: value should match only one schema, but matches none, etcd key: /apisix/****/routes
```

```
local schema = {
    type = "object",
    properties = {
        message = {
            type = "string",
            minLength = 1,
            maxLength = 1024,
            default = "Your IP address is not allowed"
        },
        whitelist = {
            type = "array",
            items = {anyOf = core.schema.ip_def},
            minItems = 1
        },
        blacklist = {
            type = "array",
            items = {anyOf = core.schema.ip_def},
            minItems = 1
        },
    },
    oneOf = {
        {required = {"whitelist"}},
        {required = {"blacklist"}},
    },
    additionalProperties = false,
}
```



线上问题定位

单 cpu100%

- 流量摘除后，仍然无法恢复

- 新请求无法正常处理

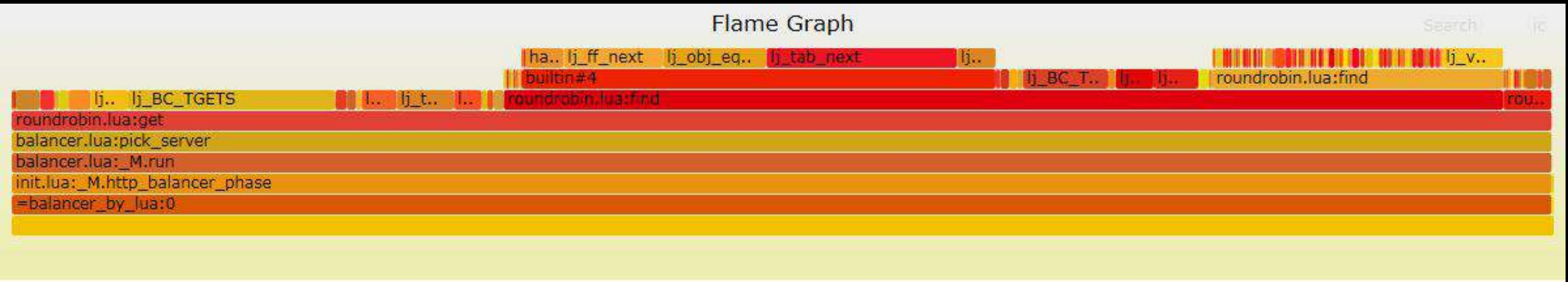
```
top - 11:30:50 up 62 days, 23:53,  0 users,  load average: 9.34, 11.49, 11.58
Tasks:  21 total,   5 running,  16 sleeping,   0 stopped,   0 zombie
%Cpu(s): 26.9 us,  3.5 sy,  0.1 ni, 66.8 id,  0.1 wa,  0.0 hi,  2.7 si,  0.0 st
KiB Mem : 13190310+total, 14281876 free, 27770880 used, 89850352 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 59665324 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
  224 www-data  20   0  971484  83600  37492 R  98.7  0.1   6271:22 openresty
  228 www-data  20   0  819800  50444  26676 R  93.0  0.0  12896:32 openresty
  230 www-data  20   0 1008160  90320  39288 R  84.3  0.1   5613:32 openresty
  225 www-data  20   0  952884  87628  37348 R  13.7  0.1   1628:01 openresty
   98 root      20   0  103812  32988   7604 S   0.3  0.0  37:28.34 host-agent-daem
```

线上问题定位

使用 systemtap 获取 Flame Graph

- 容器上需要开启特权模式
- 新版 openresty 需要加上编译选项 --without-luajit-gc64



线上问题定位

- 使用了 weight 0， 来实现 nginx 的 backup 节点功能
- 当所有 weight > 0 的节点尝试都失败的时候， 由于此时 picker:find 永远不会返回 weight 为 0 的节点， 所以 while 循环无法跳出
- 2.4版本修复了该问题

```
get = function (ctx)
    if ctx.balancer_tried_servers and ctx.balancer_tried_servers_count == nodes_count then
        return nil, "all upstream servers tried"
    end

    local server, err
    while true do
        server, err = picker:find()
        if not server then
            return nil, err
        end
        if ctx.balancer_tried_servers then
            if not ctx.balancer_tried_servers[server] then
                break
            end
        else
            break
        end
    end

    return server
end,
```

```
get = function (ctx)
    if ctx.balancer_tried_servers and ctx.balancer_tried_servers_count == nodes_count then
        return nil, "all upstream servers tried"
    end

    local server, err
    for i = 1, safe_limit do
        server, err = picker:find()
        if not server then
            return nil, err
        end
        if ctx.balancer_tried_servers then
            if not ctx.balancer_tried_servers[server] then
                break
            end
        else
            break
        end
    end

    return server
end,
```

APACHE APISIX

线上问题定位

- 低延迟，高并发请求场景会出现 “Cannot assign requested address ”
- APISIX 统一一个 upstream keepalive配置，会出现互相影响
- 2.9 版本支持在 upstream 级别自定义 keepalive 连接池

```
upstream apisix_backend {
    server 0.0.0.1;

    {% if use_apisix_openresty then %}
    keepalive {* http.upstream.keepalive *};
    keepalive_requests {* http.upstream.keepalive_requests *};
    keepalive_timeout {* http.upstream.keepalive_timeout *};
    # we put the static configuration above so that we can override it in the Lua code
```

APACHE APISIX

04 未来展望

协议转换网关， 接入流量治理

APACHE APISIX

融入微服务

1、协议转换

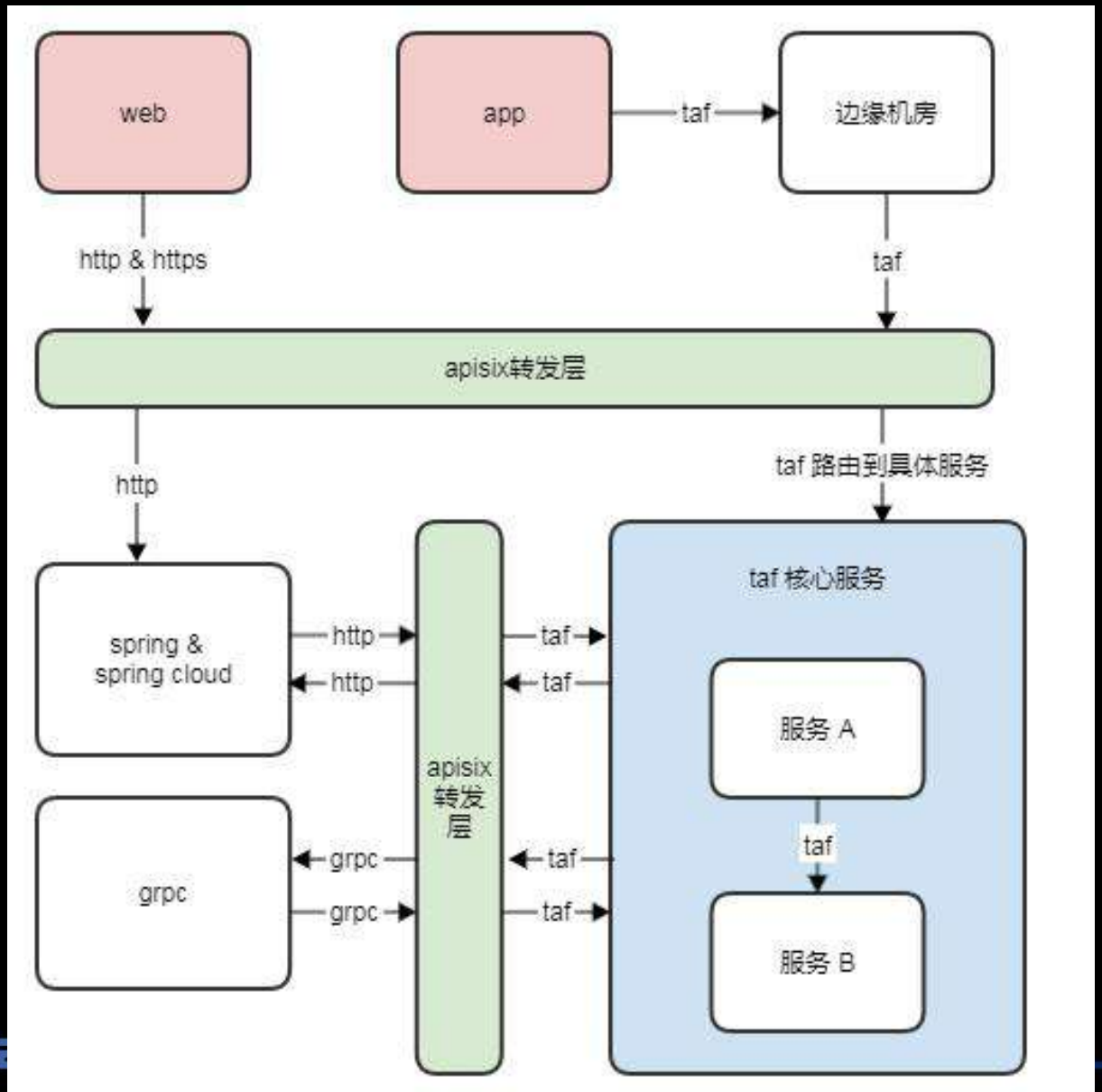
- http 转 taf，　主要针对公司的微服务场景
- http 转 grpc，　主要针对 AI 场景

2、入口统一

- 接入包括 App 内部 rpc 协议转发入口
- 统一流量控制策略

APACHE APISIX

融入微服务



APACHE APISIX

感谢聆听

THANKS