TKE使用eBPF优化k8s service

Jianmingfan

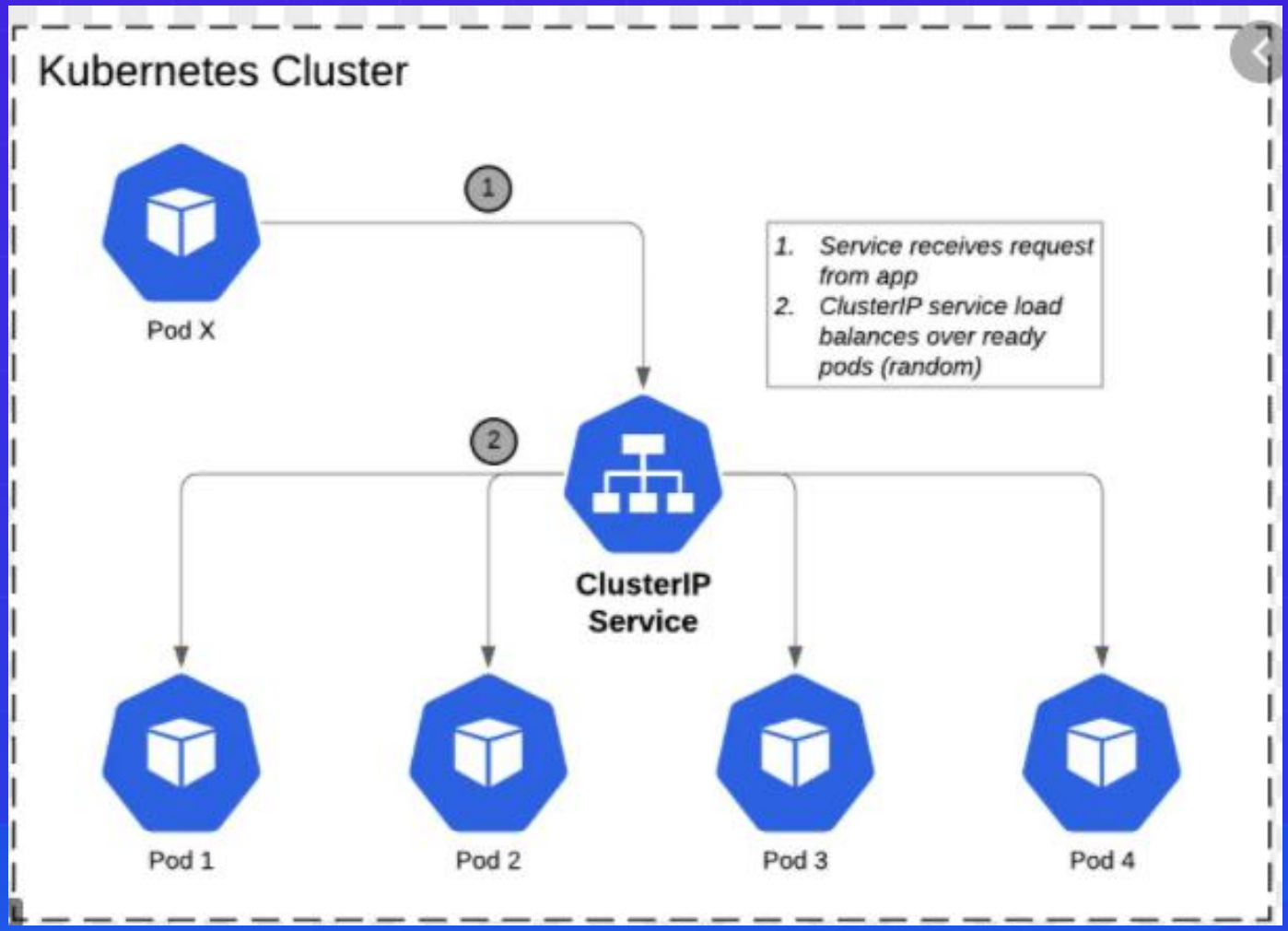
腾讯云

目录

01 Service的现状及问题

什么是k8s Service

- 应用通过固定的VIP访问一组pod，应用对Pod ip变化无感知
- 本质是一个负载均衡器
- ClusterIP提供集群内的访问
- NodePort 提供集群外部的访问



iptables mode

- 在netfilter pre-routing阶段做DNAT
- 在netfilter post-routing阶段做SNAT
- 每个service 添加一条或多条rules。使用数组管理rules。
- 仅支持随机的调度算法

iptables mode

优势

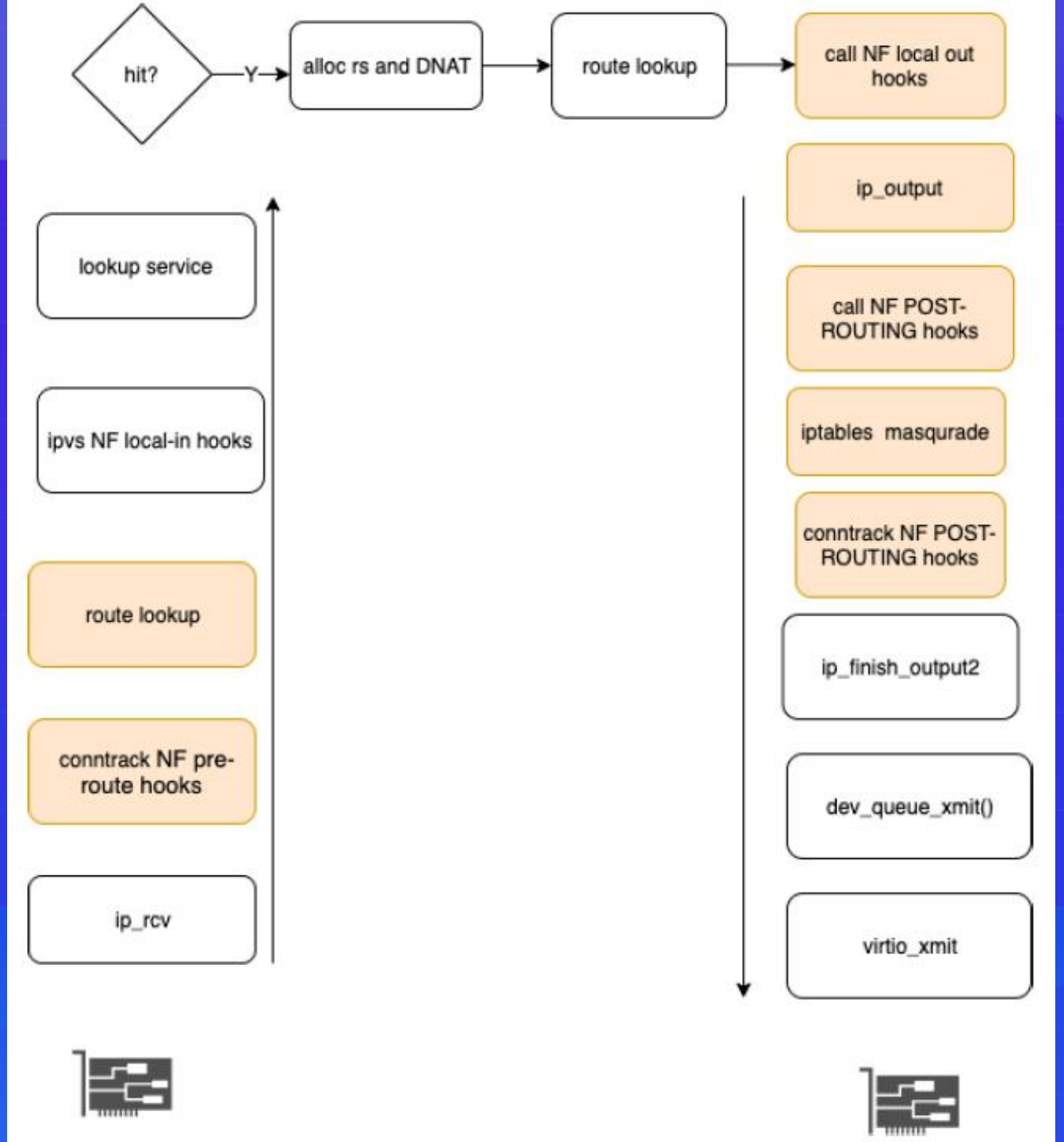
- kube-proxy代码实现比较简单
- iptables 在linux 已经广泛部署

不足之处

- 控制平面的时间复杂度是O(N^2)，当service达到上千时，修改rule耗时超过半小时。
- 数据平面的时间复杂度是O(N)
- 调度算法比较少，仅仅支持random的
- iptables rule 不容易调试

IPVS mode

- 使用hashtable 管理service
- IPVS 仅仅提供了DNAT，还需要借用iptables+conntrack 做SNAT



IPVS mode

优势

- 控制面和数据面算法复杂度都是O(1)
- 经历了二十多年的运行，比较稳定成熟
- 支持多种调度算法

不足之处

- 没有绕过conntrack，由此带来了性能开销
- 在k8s的实际使用中还有一些Bug

02 优化的方法

指导思路

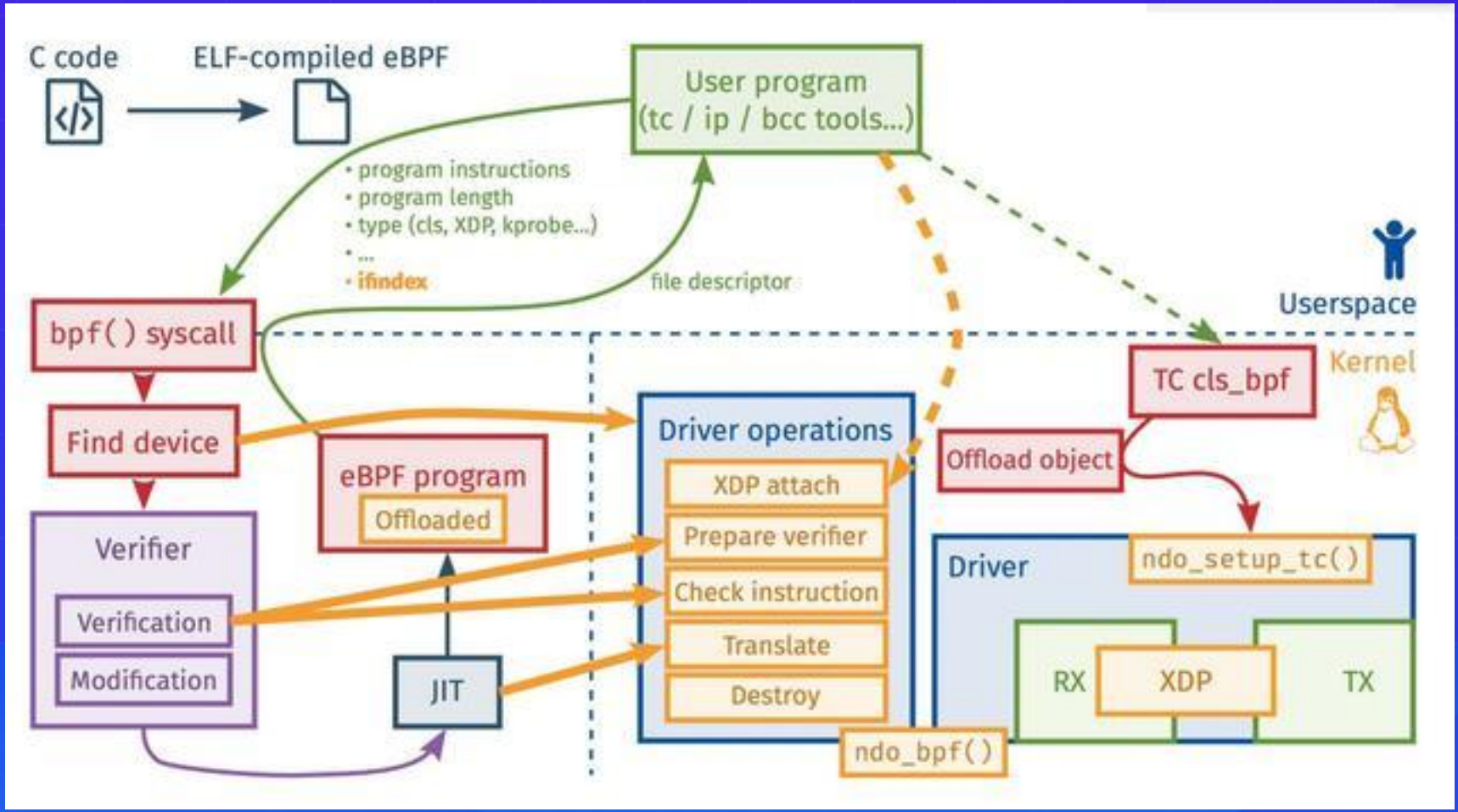
- 用尽量少的cpu指令处理每一个报文
  - 不能独占cpu
- 兼顾产品的稳定性，功能足够丰富

弯路

- 为什么DPDK不行?
  - 独占cpu，不适合分布式的lb
- 为什么纯粹的eBPF方法不行
  - 不够成熟

eBPF 简介

- 编写eBPF程序
- 编译成eBPF中间代码
- 注入内核
- 挂载到network traffic control
- 报文激发eBPF代码



技术创新点一

IPVS 绕过conntrack

- IPVS 对conntrack的功能依赖
  - Iptables SNAT
- 具体如何绕过conntrack?
  - 进报文
    - 将处理请求的钩子从nf local-in 前移到nf pre-routing
    - skb的路由指针是NULL
    - 处理分片
  - 出报文
    - 本来的逻辑：
      - Nf local out -> ip_output -> NF postrouting -> ip_finish_output
    - 修改成：
      - 对kenel 做了hack，直接访问ip_finish_output

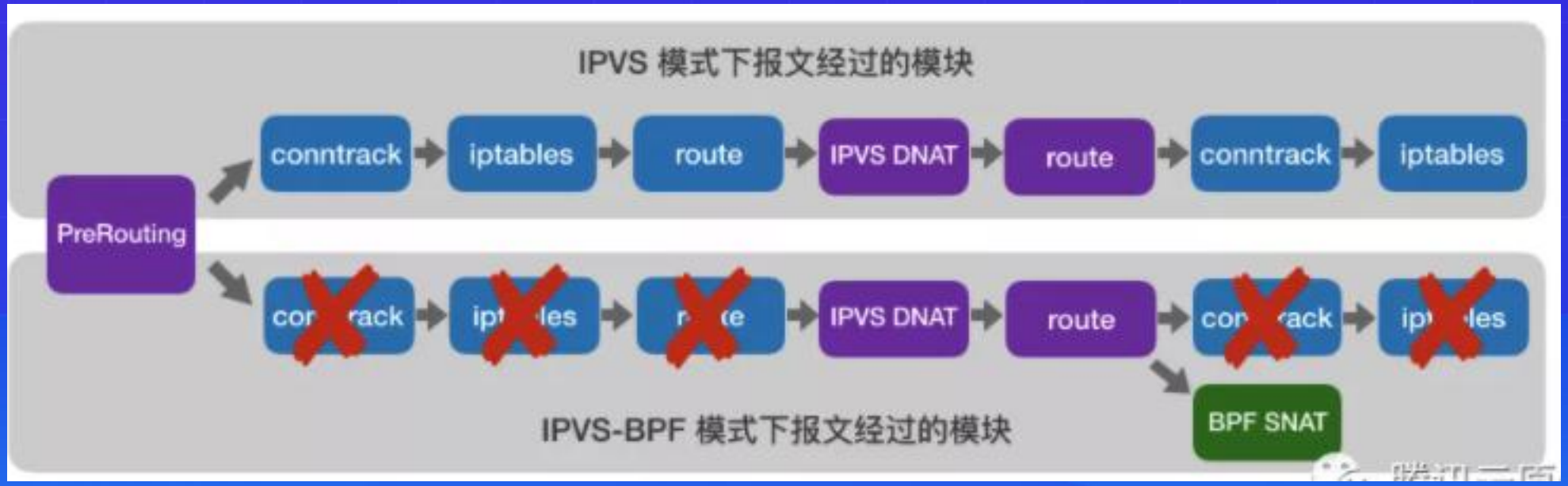
技术创新点二

IPVS 如何做SNAT?

- 在linux traffic control上挂一段eBPF 代码，在网卡出报文之前做SNAT
  - 尽量将大部分代码放在eBPF中，方便升级和维护。
- eBPF loader 创建eBPF map时，将map的id 传给IPVS内核模块
- 在ip_vs_new_conn 时，插入eBPF map
  - (protocol, clientip:cport -> targetip:dport)
- 在ip_vs_conn_unlink时，删除eBPF map
  - 由于eBPF中没有timer机制

优化方法评价

- 优势
  - 大大缩短了数据通路，完全绕过了conntrack/iptables
- 不足
  - 对内核模块做了一定的修改，部署更困难



03 和业界方法比较

和其他的优化方法对比

V.S. 纯粹的eBPF service

- 复用了IPVS timer来回收eBPF map。避开了eBPF map没有timer的问题
- 继承了IPVS丰富的功能，稳定性。例如调度算法丰富。

V.S. Taobao IPVS SNAT patch

- 优势
  - 完全绕过了conntrack/iptables
  - 对内核修改更小

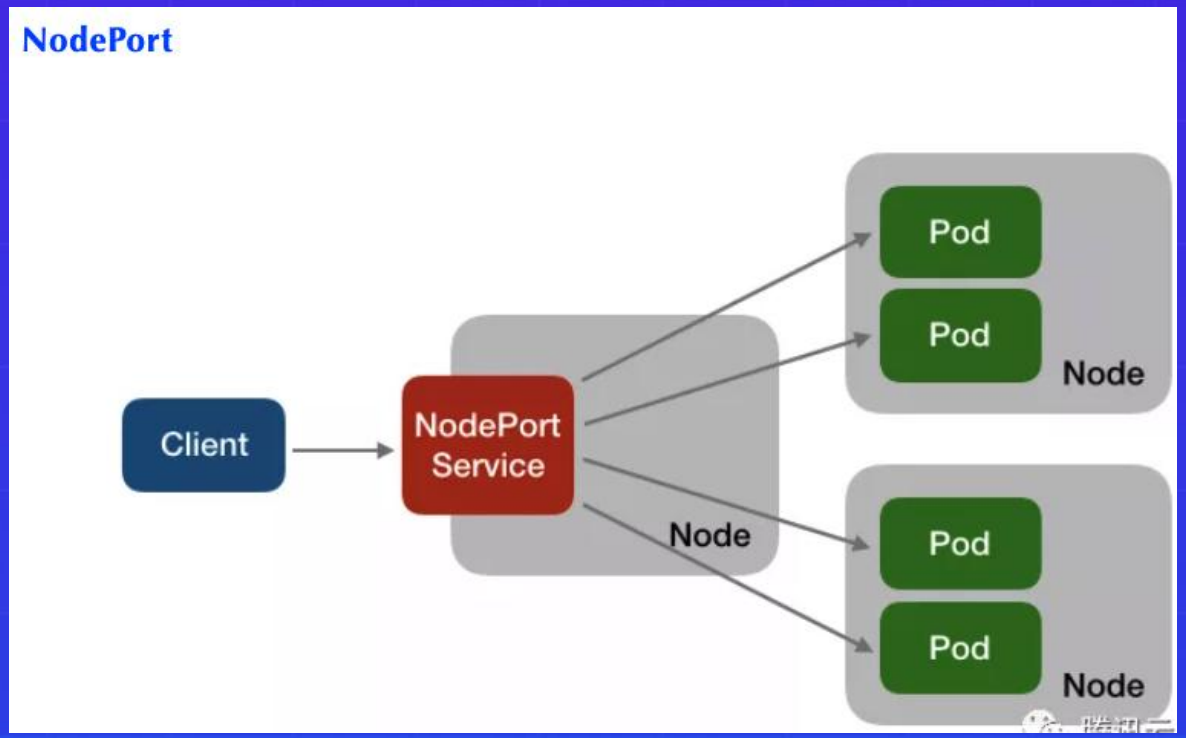
04 性能测试

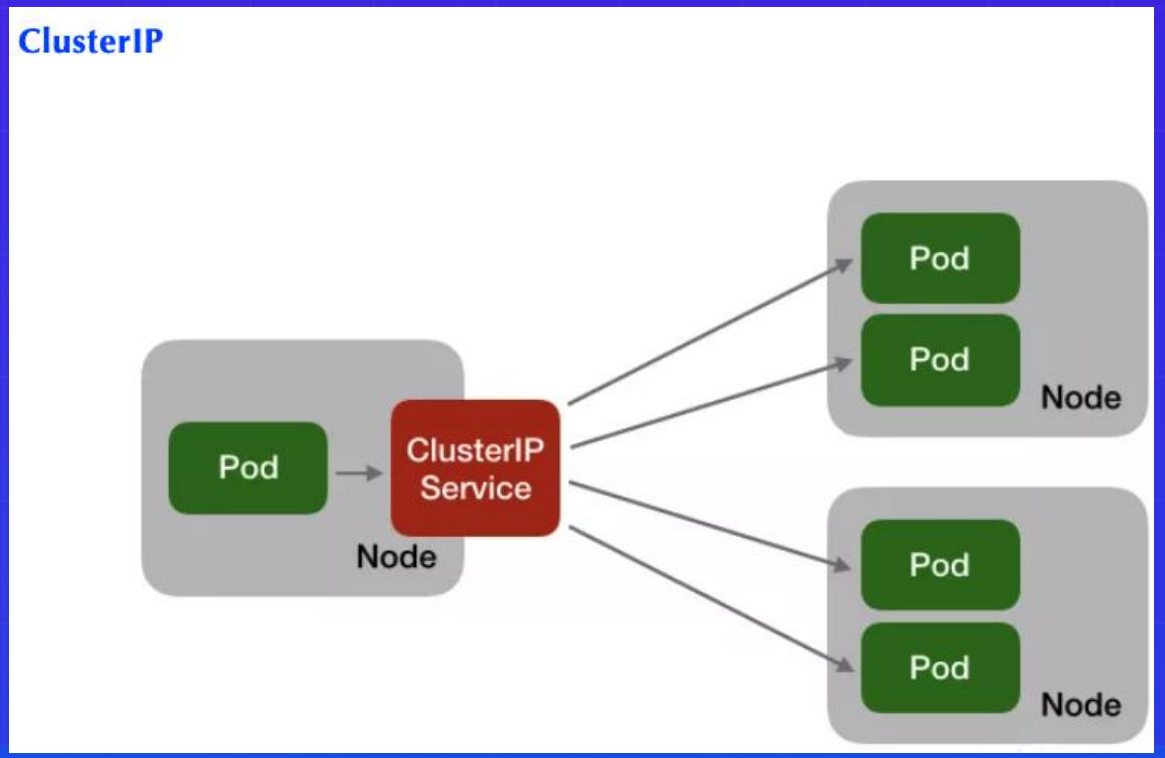
设置测试环境

性能测试踩过的坑

- 配置一样的cluster，性能可能不同。
  - 多个CVM分布在同一台物理主机
- 同一个cluster，在不同的时间段，性能可能不同
  - cpu 超卖
- 使用同一个cluster，在相近的时间段，比较两种mode
- 使得cpu成为瓶颈点
  - cpu和网卡pps的比例关系 < 1/50w pps
- Target server pool /client pool 的配置要足够强大。

测试拓扑

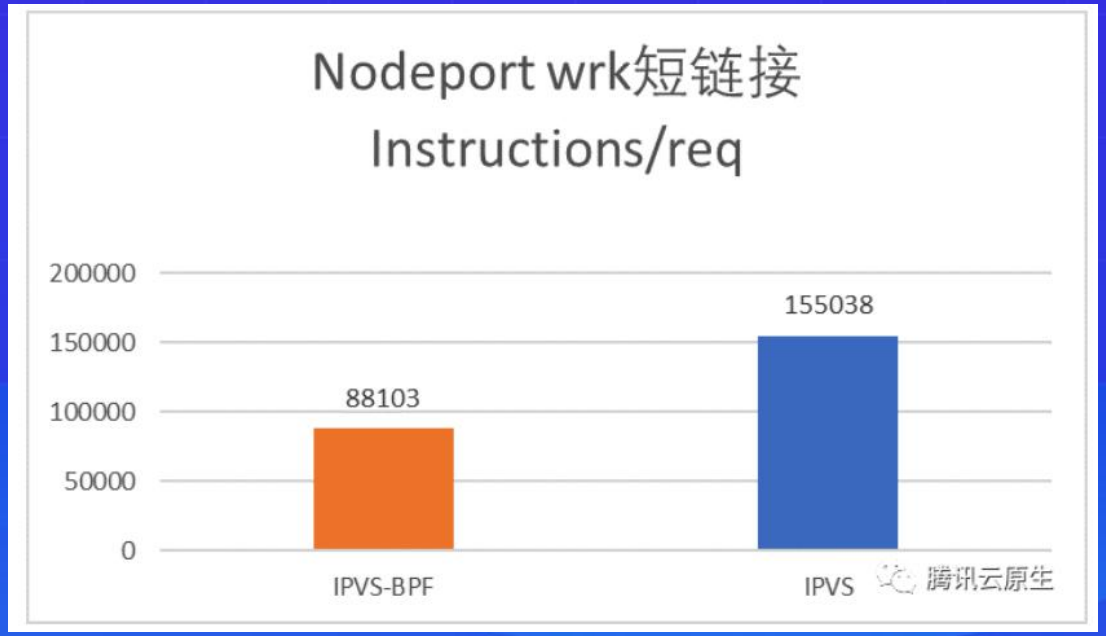

NodePort
Client
NodePort Service
Node
Pod
Pod
Node
Pod
Pod
Node


ClusterIP
Pod
ClusterIP Service
Node
Pod
Pod
Node
Pod
Pod
Node

测试数据

Service类型	短连接cps	短连接p99延迟	长连接吞吐
clusterIP	+40%	-31%	无，见下文
nodePort	+64%	-47%	+22%

- 处理每一个req耗费的指令数目降低了38%



05 解决的BUG

IPVS conn_reuse_mode = 1性能低

- 原因
  - conn_reuse_mode的本意是当ip_vs_conn出现复用时，重新调度，创建新的ip_vs_conn.
  - 新旧ip_vs_conn引用了同一个conntrack，但是没有引用计数。
  - 当old ip_vs_conn 释放conntrack后，导致新ip_vs_conn没有conntrack的引用。
- 解决方式
  - 通过增加引用计数的方法解决了这个问题。

DNS 解析偶尔5s延迟

- 原因
  - DNS 同时发了2个UDP请求，分别请求A和AAAA。源端口是cport。
  - Iptables SNAT 为请求1分配了lport=cport
  - 很快Iptables SNAT 为请求2分配了同样的lport
  - Conntrack Post routing 函数中，将请求1的lport插入conntrack，成功！
  - Conntrack Post routing 函数中，将请求2的lport插入conntrack，失败，丢包，导致延迟。
- 解决方法
  - eBPF代码在分配lport和插入hash的外围，加了一个大循环，插入失败会重新分配lport。

06 未来的工作

未来的工作

- 适配更多的linux 发行版本
  - Tencent linux， ubuntu, centos
- 独立开源
  - 内核修改在github.com/Tencent/TencentOS-kernel/

THANK YOU

感谢聆听

Jianmingfan
腾讯云

了解更多云原生技术和动态，请关注腾讯云原生公众号