Kubernetes IPVS vs Iptables

陈绥 UCloud研发工程师

UCLOUD 优刻得

自我介绍

2016年加入UCloud

主要工作方向

容器 微服务 ServiceMesh

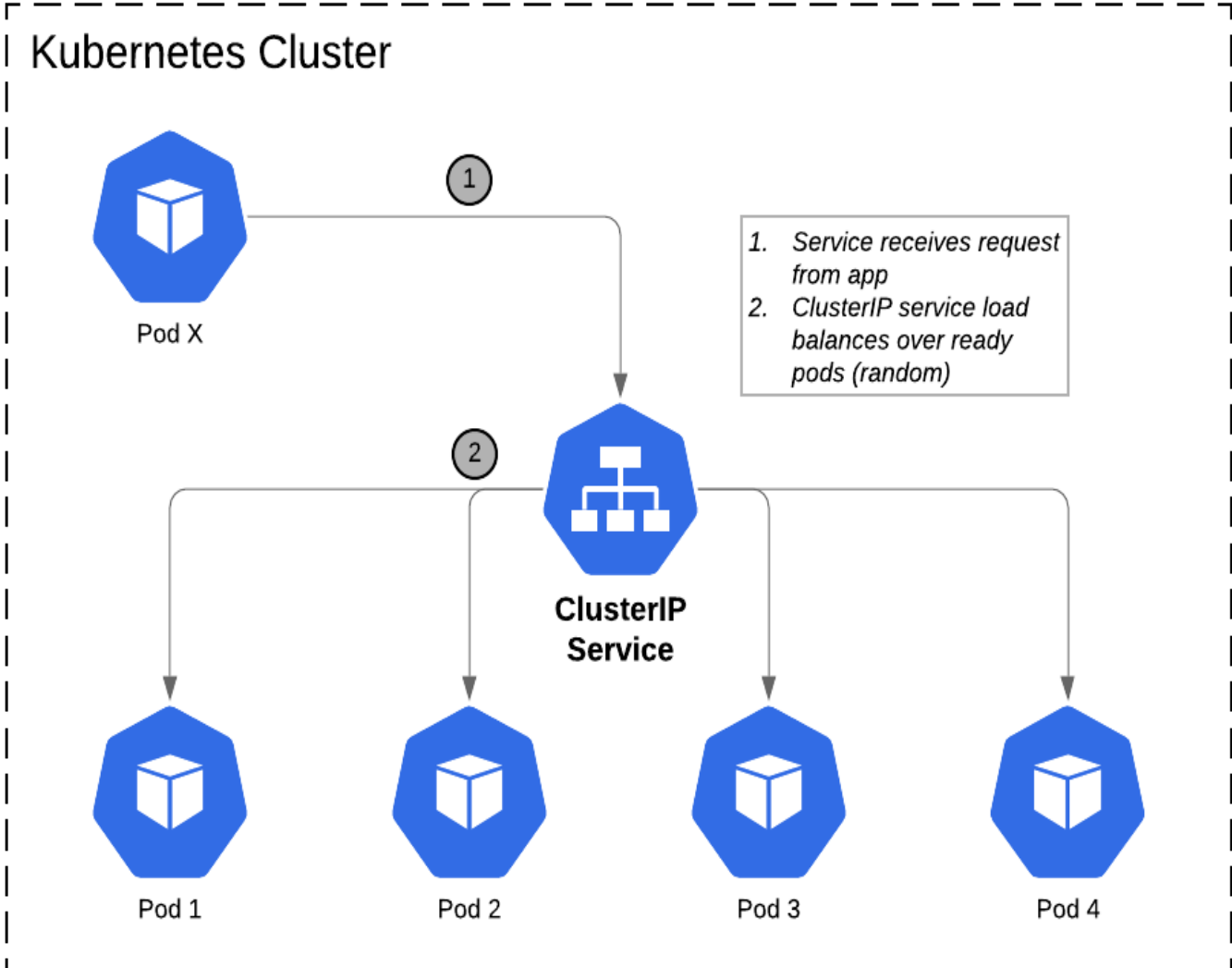
UCLOUD 优刻得

K8S Service的实现原理

IPVS和Iptables性能对比

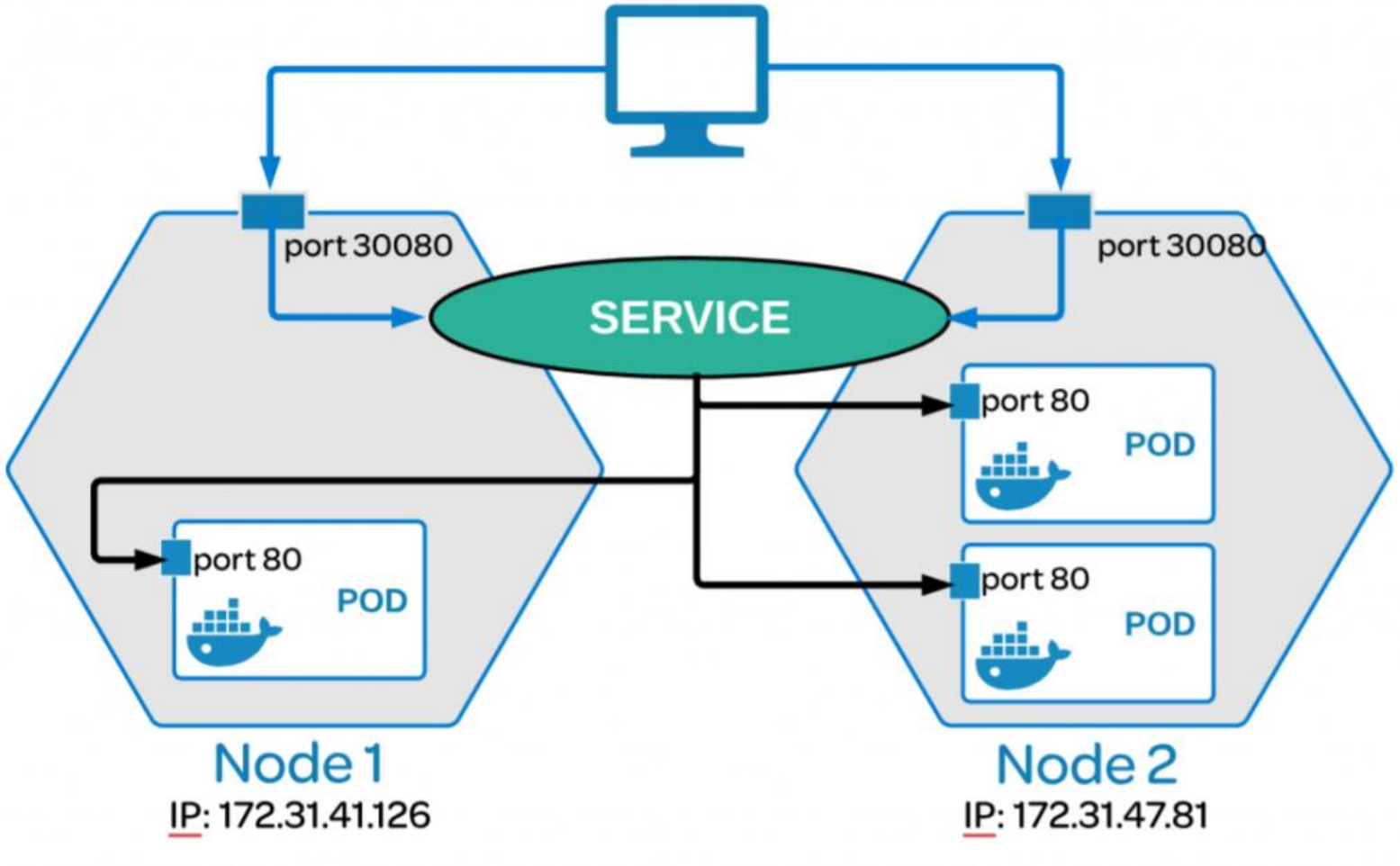
Iptables的那些坑除了SNAT还有啥?

IPVS就完美么？来看大型翻车现场



Kubernetes Service

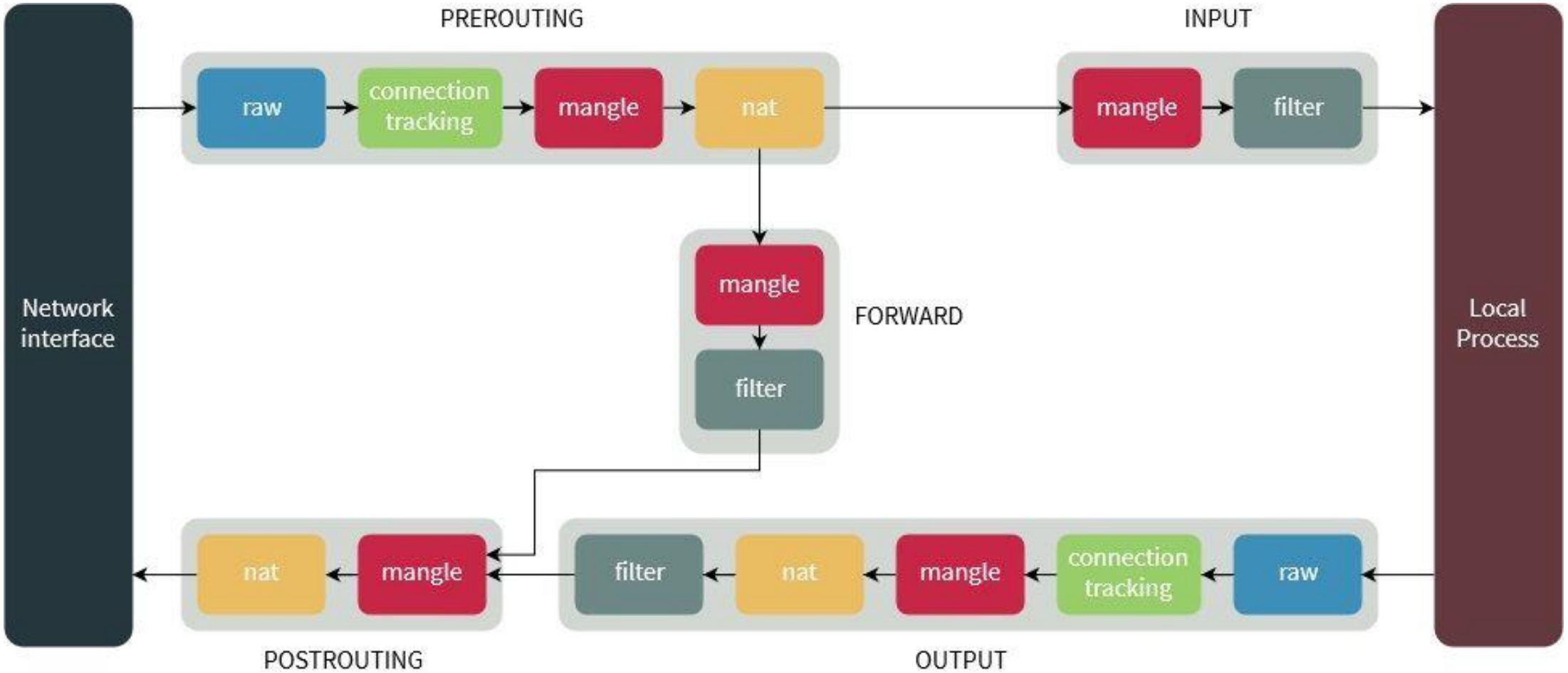
A service allows you to dynamically access a group of replica pods.

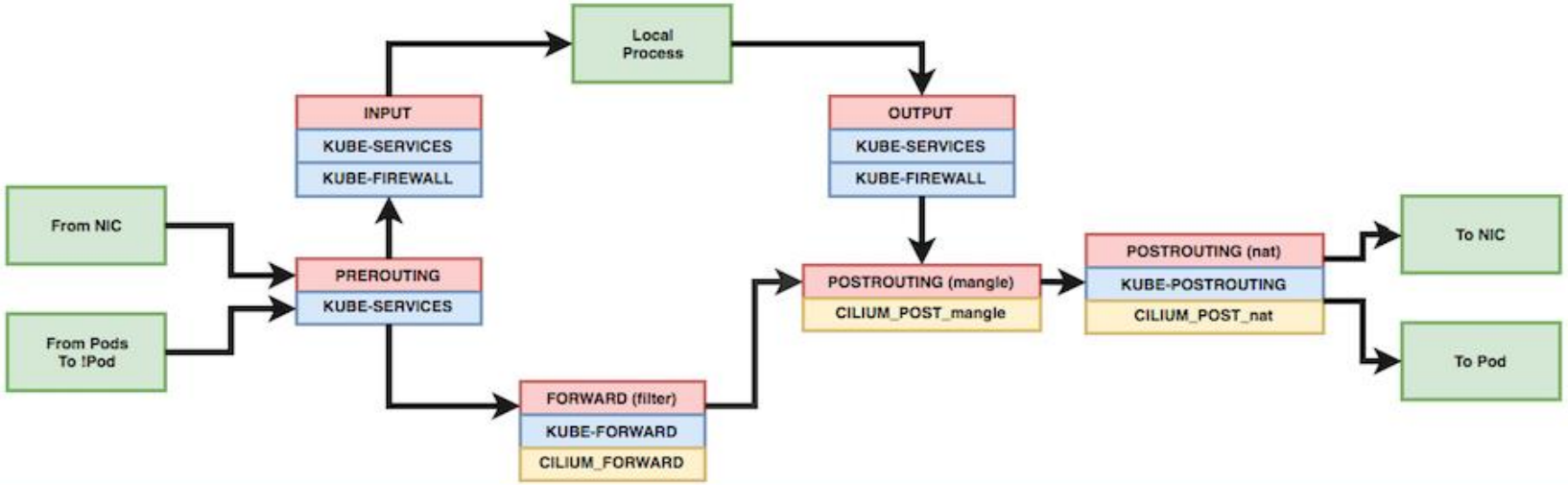


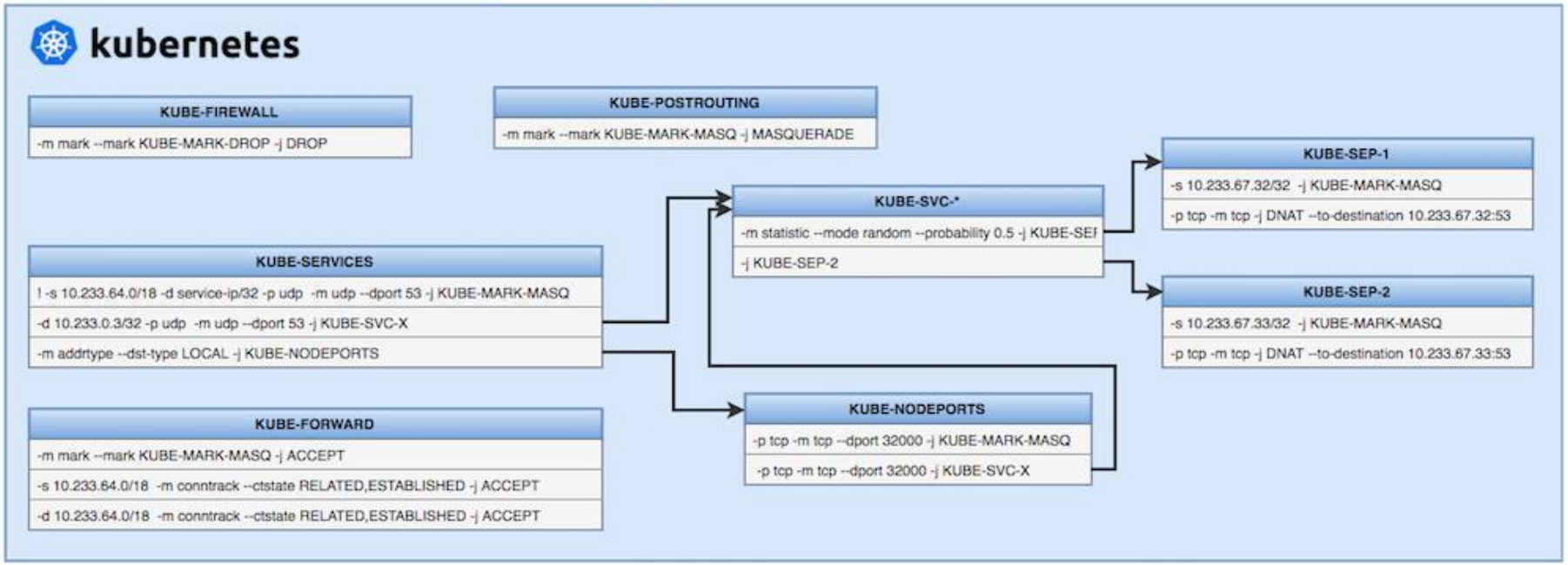
USER SPACE

```yaml
apiVersion: v1
kind: Service
metadata:
  name: http1
spec:
  type: NodePort
  ports:
  - name: http1
    port: 80
    protocol: TCP
    targetPort: 8080
  selector:
    app: http
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: p1
  labels:
    app: http
spec:
  containers:
  - name: http
    image: uhub.service.ucloud.cn/wxyz/http:1.0
    imagePullPolicy: Always
    ports:
    - containerPort: 8080
---
apiVersion: v1
kind: Pod
metadata:
  name: p2
  labels:
    app: http
spec:
  containers:
  - name: http
    image: uhub.service.ucloud.cn/wxyz/http:1.0
    imagePullPolicy: Always
    ports:
    - containerPort: 8080
```


PREROUTING
INPUT
raw
connection tracking
mangle
nat
mangle
filter
Network interface
mangle
filter
FORWARD
Local Process
nat
mangle
filter
nat
mangle
connection tracking
raw
POSTROUTING
OUTPUT


Local Process
INPUT
KUBE-SERVICES
KUBE-FIREWALL
OUTPUT
KUBE-SERVICES
KUBE-FIREWALL
From NIC
From Pods To !Pod
PREROUTING
KUBE-SERVICES
FORWARD (filter)
KUBE-FORWARD
CILIUM_FORWARD
POSTROUTING (mangle)
CILIUM_POST_mangle
POSTROUTING (nat)
KUBE-POSTROUTING
CILIUM_POST_nat
To NIC
To Pod


kubernetes
KUBE-FIREWALL
-m mark --mark KUBE-MARK-DROP -j DROP
KUBE-POSTROUTING
-m mark --mark KUBE-MARK-MASQ -j MASQUERADE
KUBE-SERVICES
! -s 10.233.64.0/18 -d service-ip/32 -p udp -m udp --dport 53 -j KUBE-MARK-MASQ
-d 10.233.0.3/32 -p udp -m udp --dport 53 -j KUBE-SVC-X
-m addrtype --dst-type LOCAL -j KUBE-NODEPORTS
KUBE-SVC-*
-m statistic --mode random --probability 0.5 -j KUBE-SEP
-j KUBE-SEP-2
KUBE-SEP-1
-s 10.233.67.32/32 -j KUBE-MARK-MASQ
-p tcp -m tcp -j DNAT --to-destination 10.233.67.32:53
KUBE-SEP-2
-s 10.233.67.33/32 -j KUBE-MARK-MASQ
-p tcp -m tcp -j DNAT --to-destination 10.233.67.33:53
KUBE-NODEPORTS
-p tcp -m tcp --dport 32000 -j KUBE-MARK-MASQ
-p tcp -m tcp --dport 32000 -j KUBE-SVC-X
KUBE-FORWARD
-m mark --mark KUBE-MARK-MASQ -j ACCEPT
-s 10.233.64.0/18 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-d 10.233.64.0/18 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT

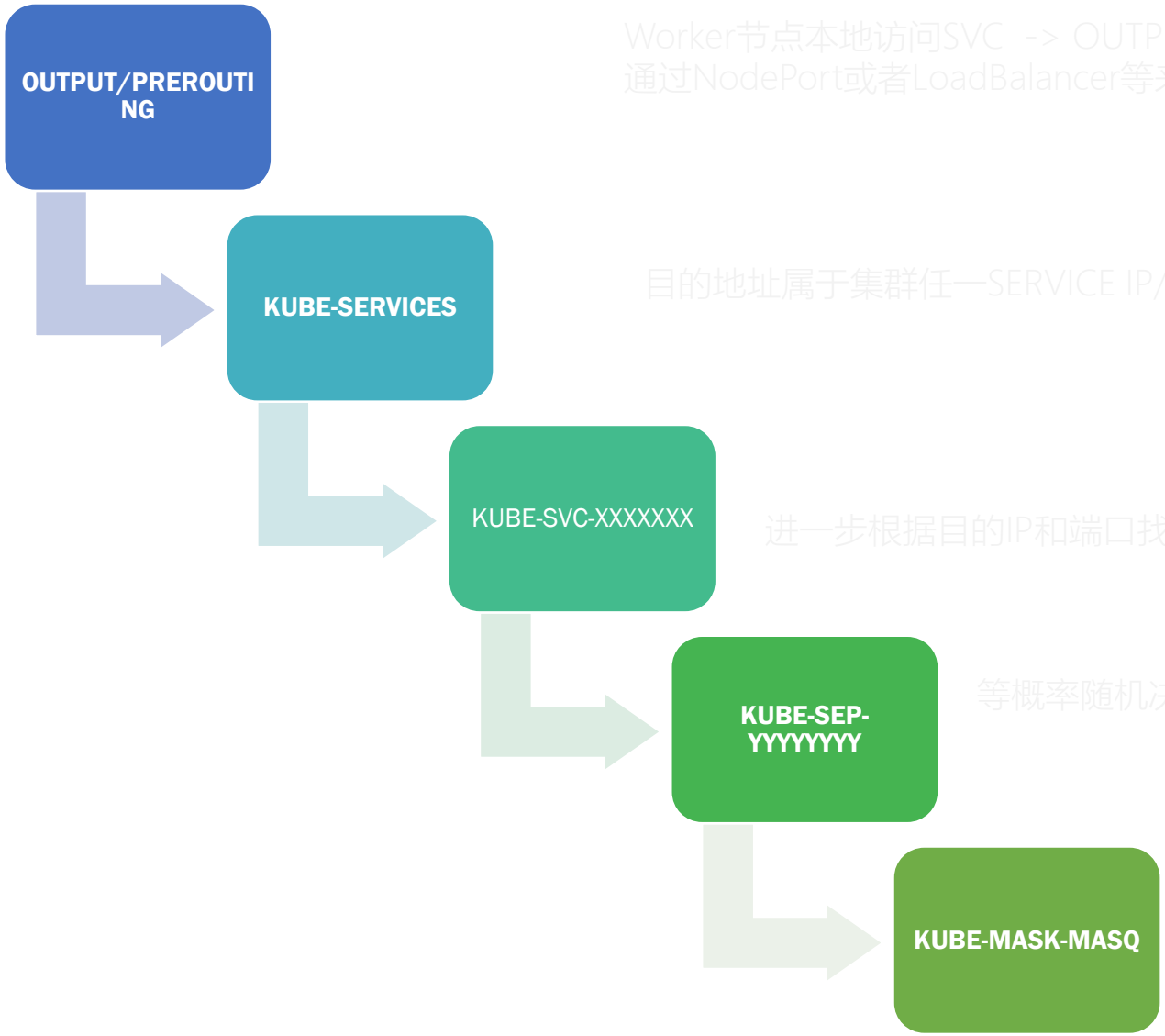
-A PREROUTING

-A OUTPUT

-A KUBE-SERVICES	CLUSTER-IP + EXTERNAL-IP NUMBER
-A KUBE-SVC-XXXXXXXX	SVC NUMBER
-A KUBE-NODEPORTS	SVC(Type=NodePort) NUMBER
-j KUBE-SEP-YYYYYYYY	ENDPOINTS NUMBER
-j KUBE-MASK-MASQ	1

集群内客户端（Node or Pod inside Node）通过CLUSTER-IP访问SVC的链路顺序

1. 请求流量出自本机->进入OUTPUT
2. 目标为CLUSTER IP和SVC端口，进入KUBE-SERVICES
3. KUBE-SERVICES根据目标地址，找到对应KUBE-SVC-XXXXXXXX
4. KUBE-SVC-XXXXXXXX根据等概率随机原则，进入KUBE-SEP-YYYYYYYY
5. KUBE-SEP-YYYYYYYY完成最终的DNAT，将目标IP地址改成PodIP，源地址改成Node IP或者CNI网桥的IP，源端口变换后续分析


OUTPUT/PREROUTING
KUBE-SERVICES
KUBE-SVC-XXXXXXX
KUBE-SEP-YYYYYYY
KUBE-MASK-MASQ

发生SNAT的场景：

1. 有来自集群外部的流量通过NodePort或者LoadBalancer IP/Port进入集群

2. 集群内Pod通过Service IP(包括Cluster IP)访问自身。

SNAT的特点：

- 源IP变换为Node节点的IP或者CNI网桥的IP；
- 源端口是否在允许的范围内(默认1024-64512)，并检查是否可用；
- 如源端口不可用，调用nf_net_l4proto-unique_tuple()找到未使用的TCP端口替换；

NodePort为什么需要SNAT?

如何获取请求源IP?

spec.externalTrafficPolicy=Local

LoadBalancer如何获取client真实IP? externalTrafficPolicy=Local + Health Check

SNAT的问题：高并发短连接下(5W以上)，SYN丢包导致业务相应慢，甚至超时，失败（issue#78547）

Node节点不存在SNAT IP池，为了保持连接五元组的唯一性，每次SNAT都需要消耗Node的一个端口。

Netfilter模块需要为每个进行SNAT的连接寻找可用的端口， nf_net_l4proto-unique_tuple()， 这个函数流程如下：

1. 一个初始位置开始搜索并复制最近一次分配的端口；
2. 递增++；
3. 用nf_nat_used_tuple()检查端口是否已被使用。如果已被使用，重复上一步；
4. 用刚分配的端口更新最近一次分配的端口并返回。

nf_nat_used_tuple()的问题

端口分配和把连接插入conntrack有延迟，当多个新TCP连接同时寻找SNAT替换端口时，可能存在nf_nat_used_tuple()返回成功，但conntrack表插入失败。

这时候执行conntrack –S可用发现大量insert_failed，调用方的感受： SYN进入丢包->重传，甚至超时，服务调用失败。

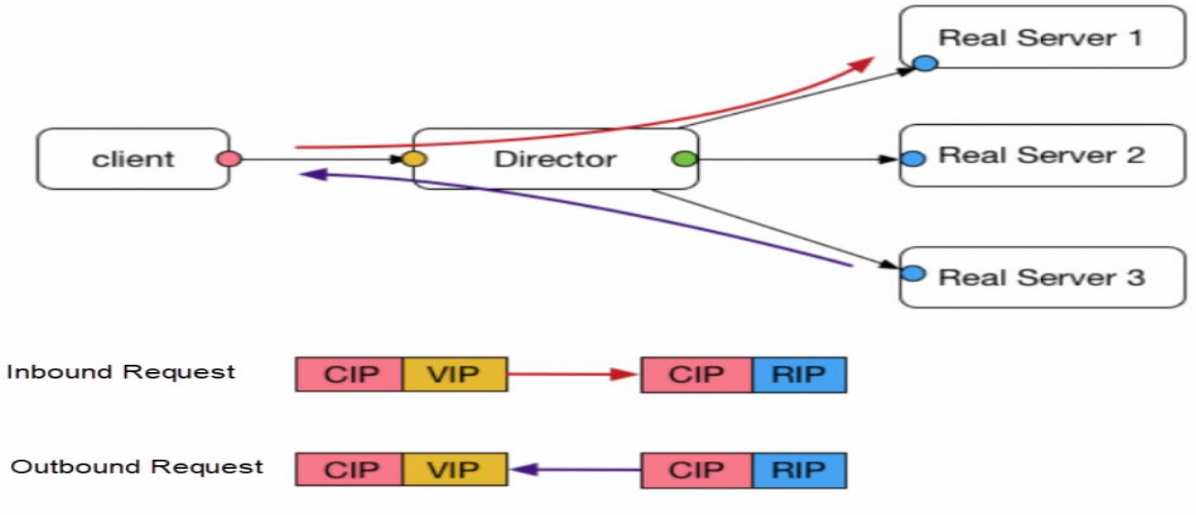
优化思路

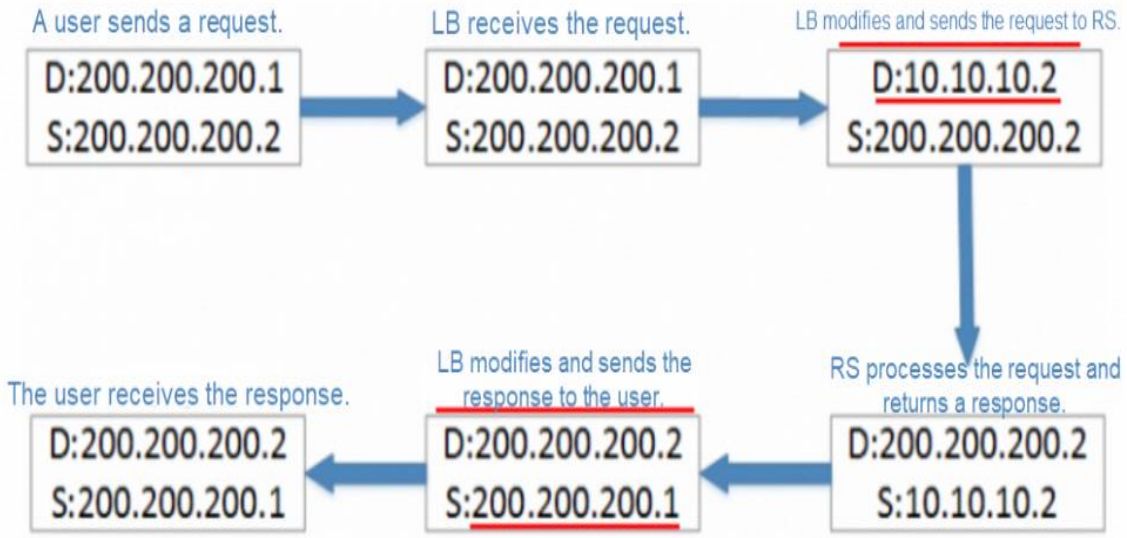
IPVS的NAT模式

来去都必须经过Director

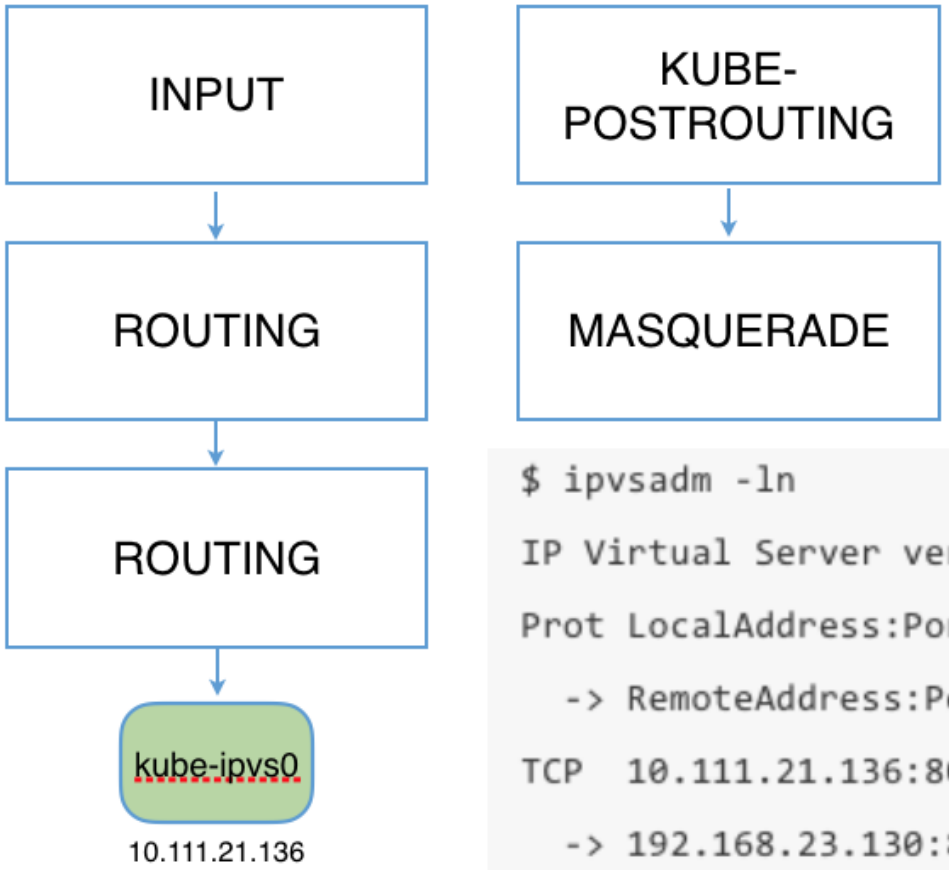
需要对报文做SNAT和DNAT

支持端口映射





Kubeproxy的IPVS工作模式




```
$ ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.111.21.136:80 rr persistent 10800
  -> 192.168.23.130:80            Masq    1      0          0
  -> 192.168.23.134:80            Masq    1      0          0
```

Kubeproxy的IPVS工作模式

- 使用ipset替换了iptables部分功能，简化管理
- 创建kube-ipvs0网卡，将所有ServiceIP绑定该网卡，从而使访问流量进入INPUT链
- 为Service和Endpoints创建ipvs virtual server和real server

IPVS模式下的保留的iptables规则：

-A KUBE-MARK-MASQ 用于标记需要**SNAT**的包

-A KUBE-POSTROUTING 用于**SNAT**

-A KUBE-NODE-PORT 用于**NodePort**模式下集群外部访问流量导入**KUBE-NODE-PORT-TCP/UDP ipset**规则

-A KUBE-SERVICES 将访问**Cluster IP**的流量导入**KUBE-CLUSTER-IP Ipset**规则

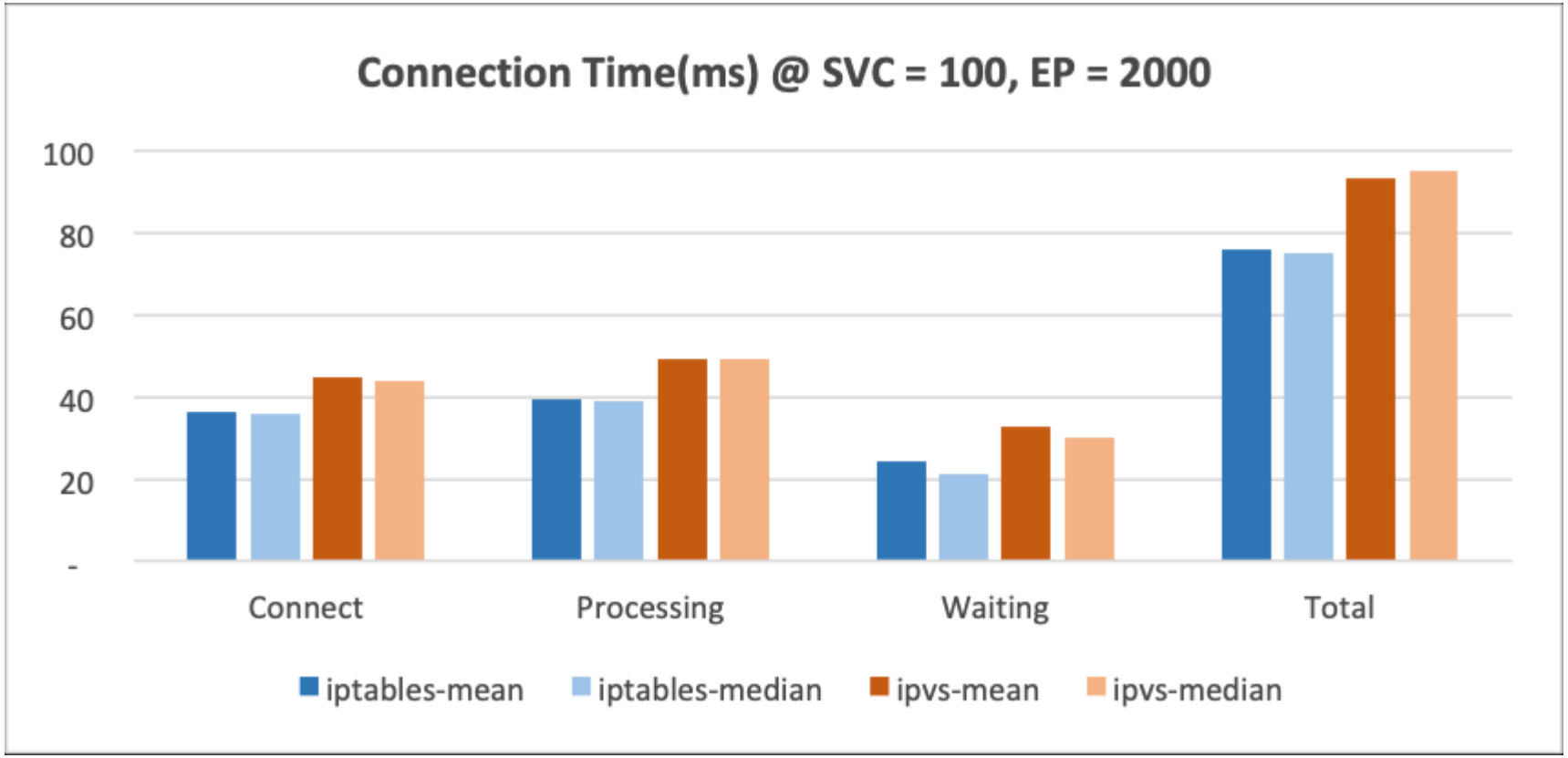
无论Service和Endpoints数目有多少，kube-proxy总保持恒定且少量的iptables规则。

IPVS模式的性能优势主要体现在

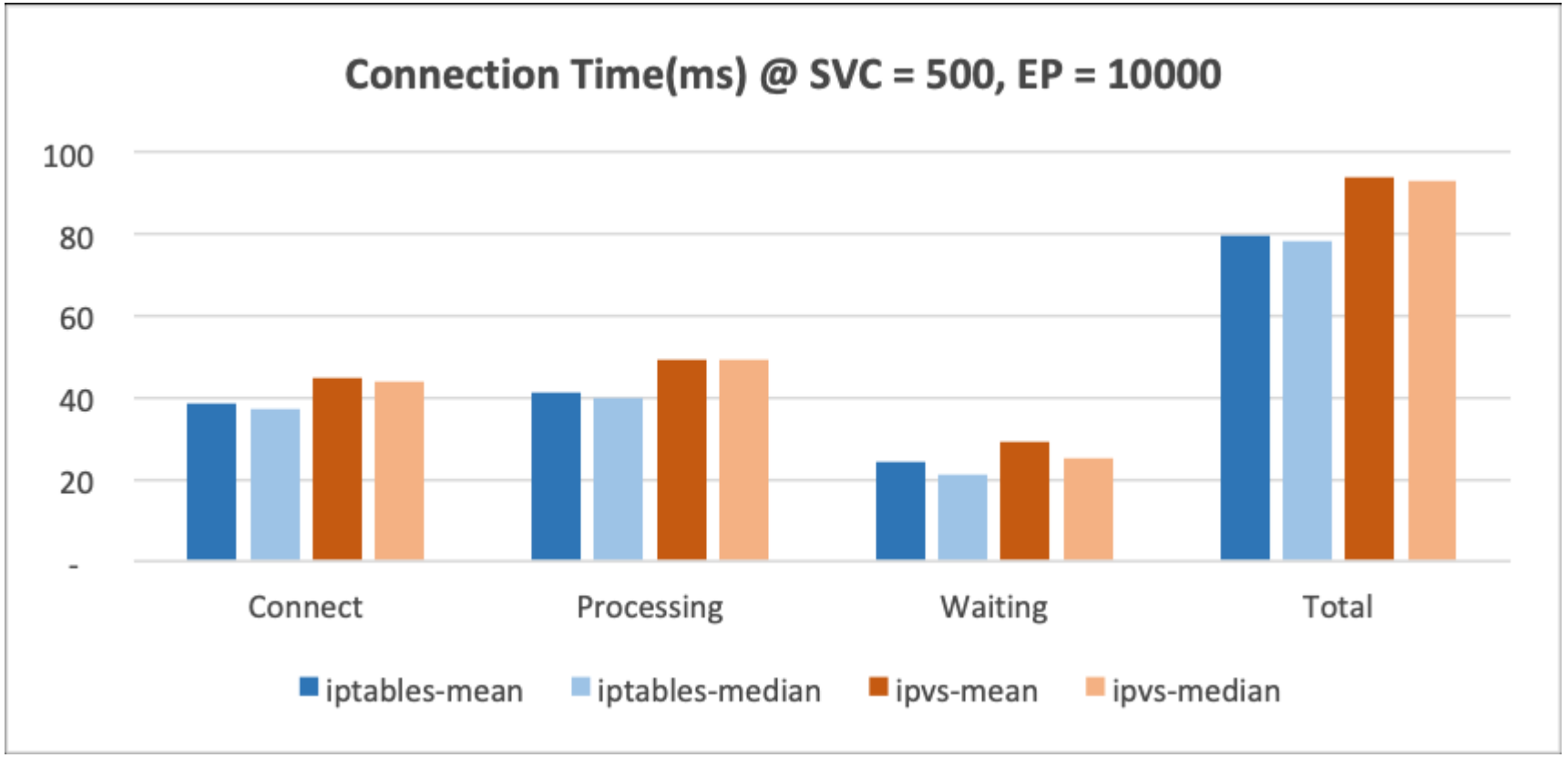
1） Kube-Proxy在更新Service-Endpoint转发规则耗时更低，cpu，mem消耗更低。

2）转发性能上，当svc超过1000时，ipvs有明显性能优势。但svc和ep总数低于500时，iptables转发性能更优。

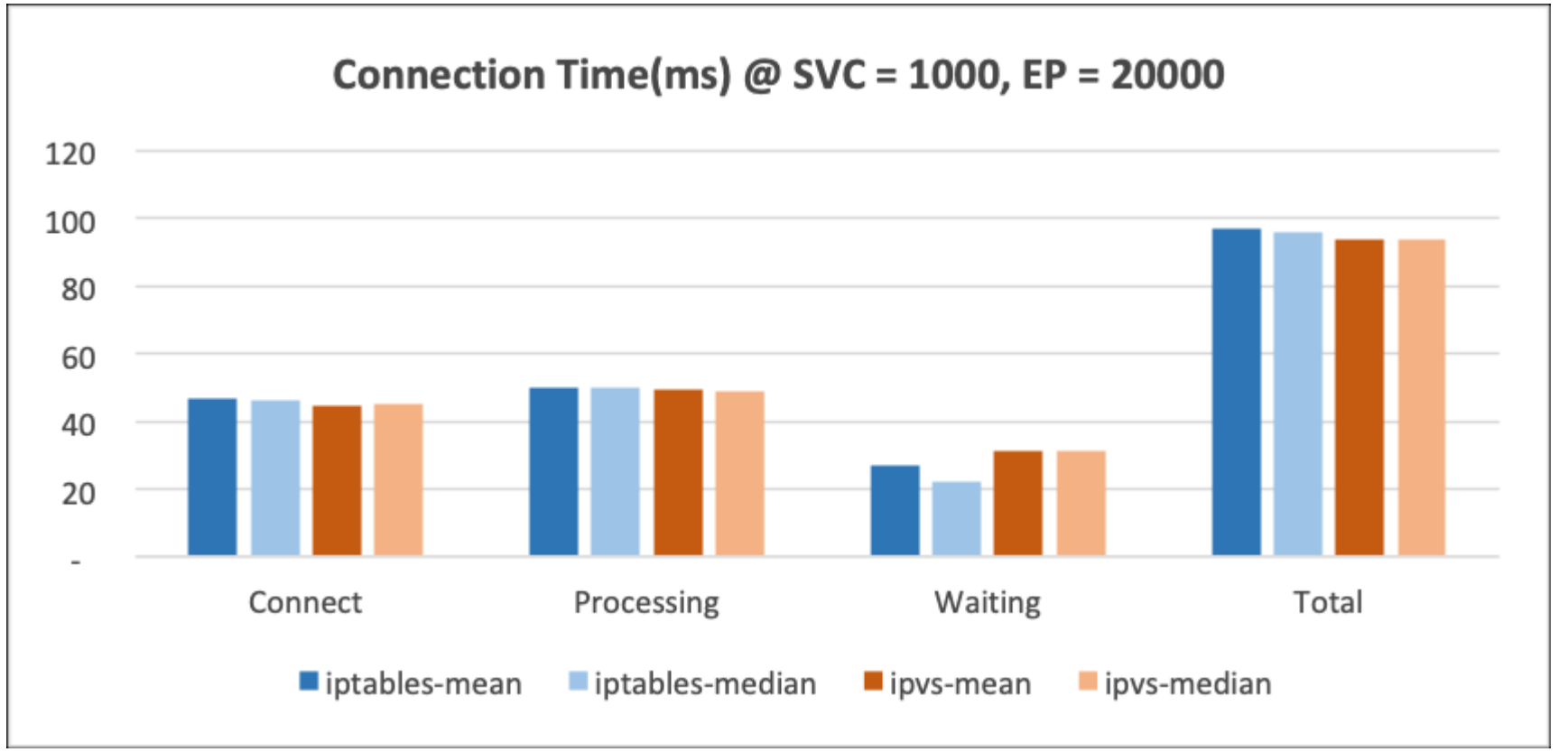
SVC=100， EP=2000



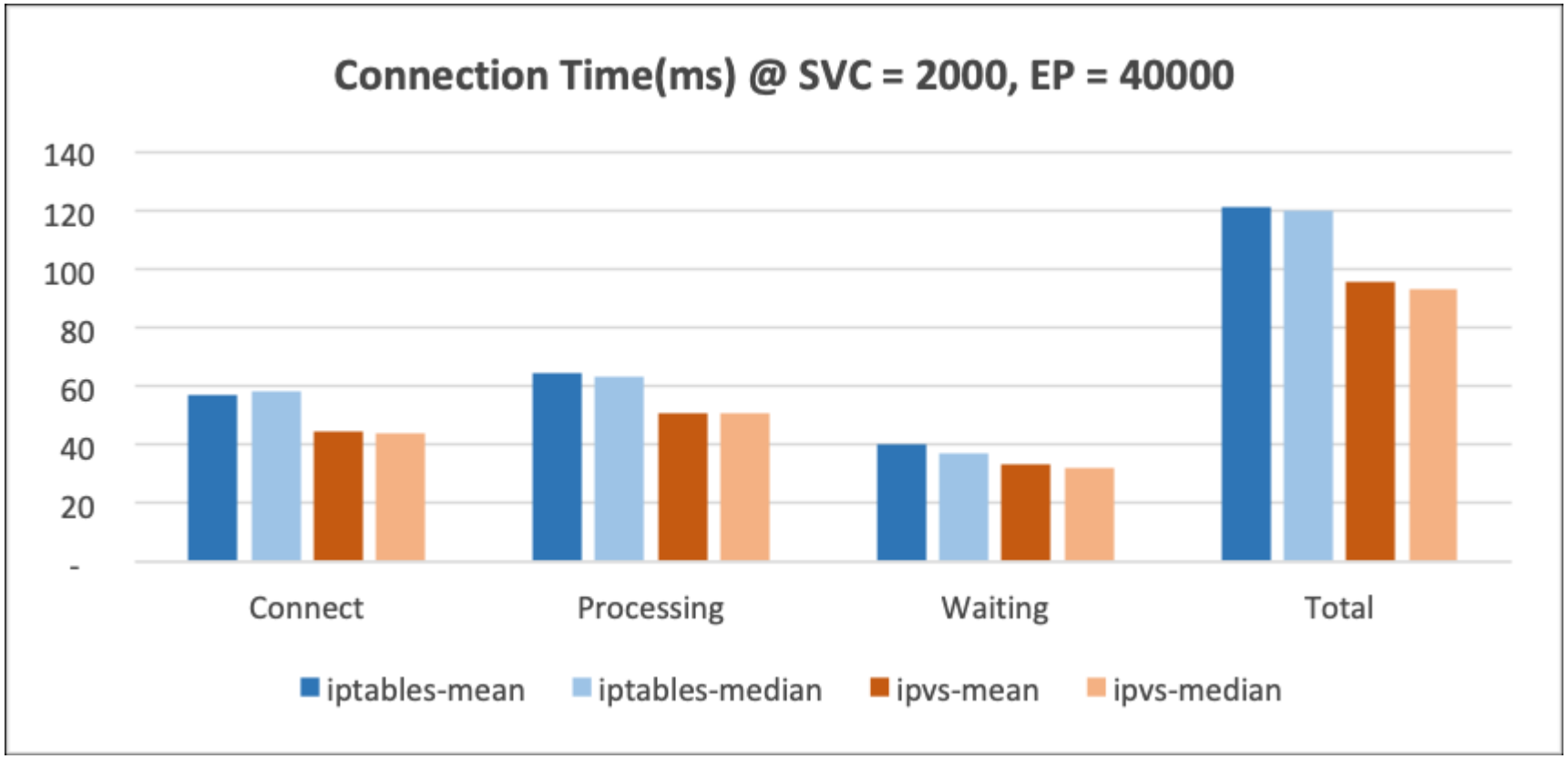
SVC=500， EP=10000



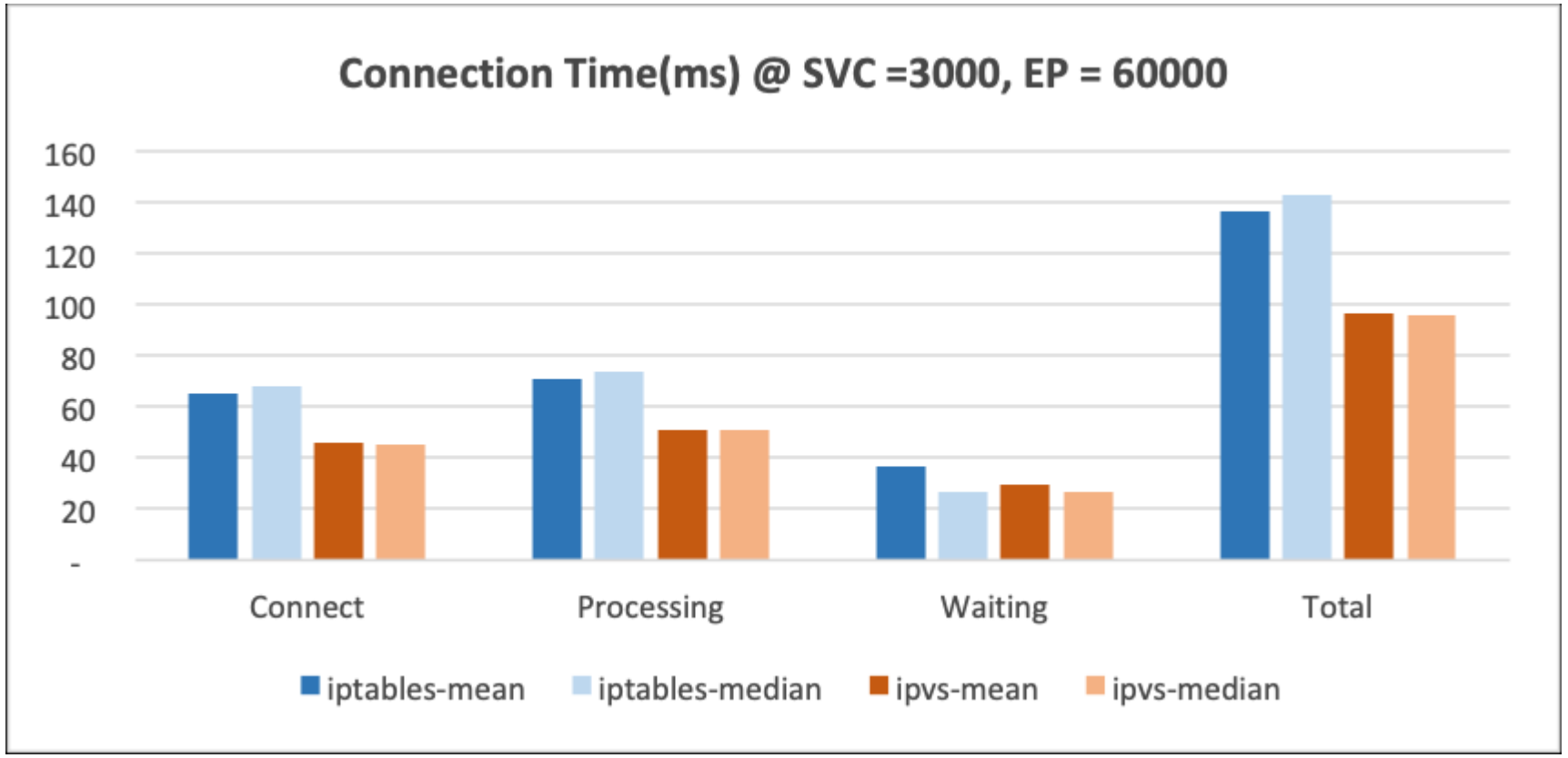
SVC=1000， EP=20000



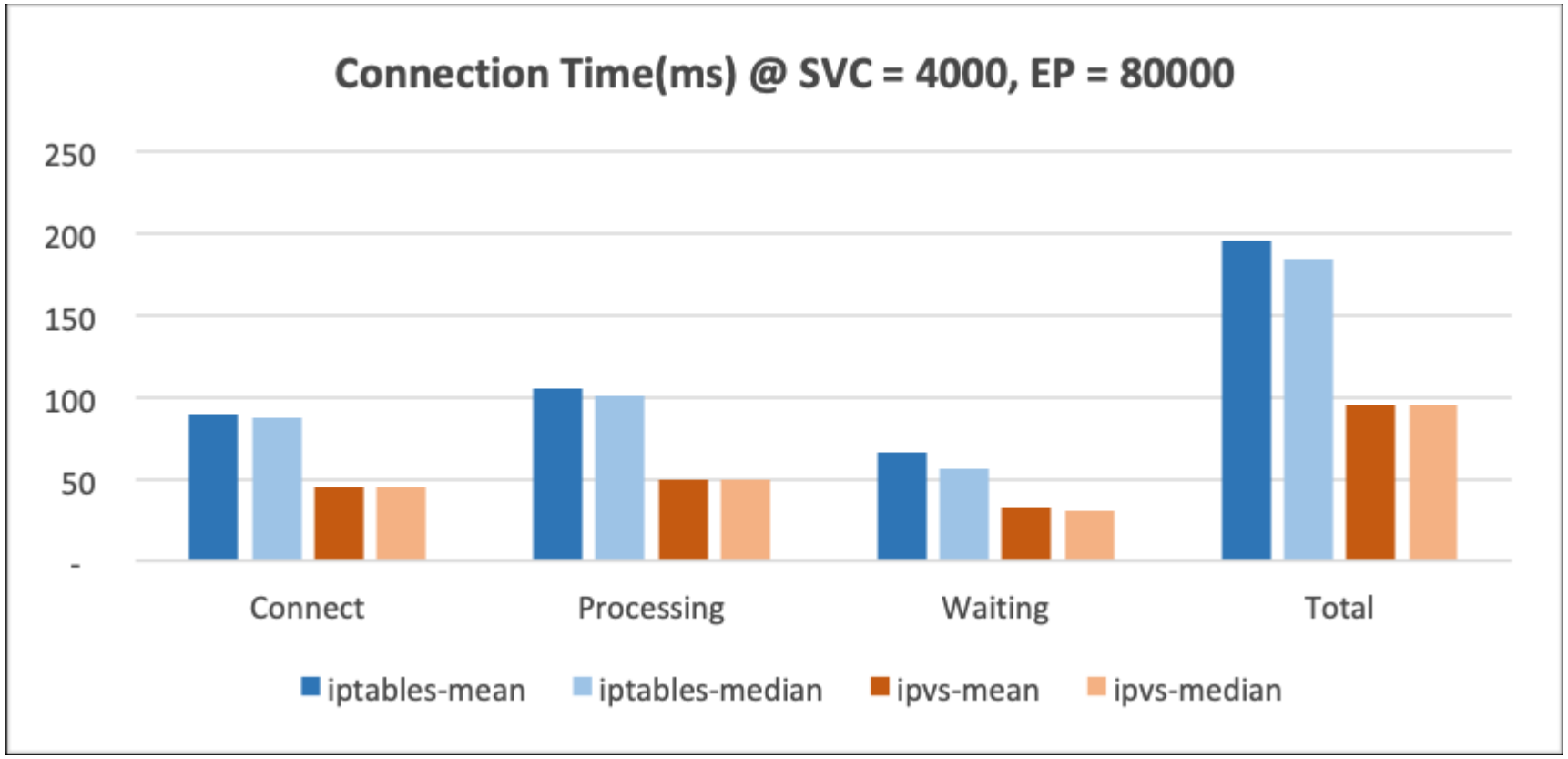
SVC=2000， EP=40000



SVC=3000， EP=60000



SVC=4000， EP=80000



现象：IPVS模式集群，服务完成滚动更新后，外部请求通过ServiceIP调用持续失败，报错信息No route to host。 而滚动更新后的Pod服务稳定，直连Pod IP调用正常。

	iptables	ipvs
转发性能	SVC和EP数量小时性能好，随着数量增大急剧下降	性能稳定，与SVC和EP数量无关
更新规则性能	较低，随着SVC和EP数目增大急剧下降	较高，且不随着SVC和EP数量增大而下降
是否存在SNAT端口冲突问题	存在	存在

UCLOUD 优刻得

THANKS