容器网络性能优化之旅

刘梦馨@Kube-OVN

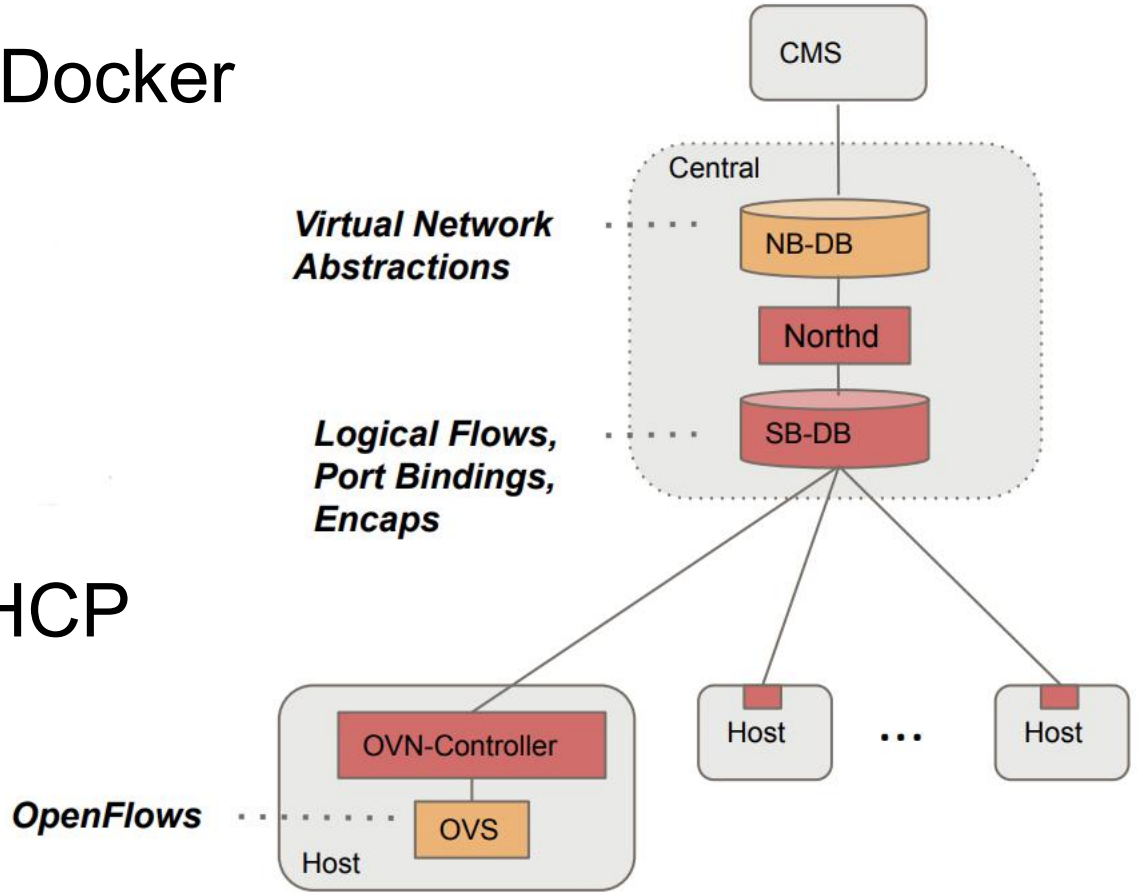
什么是Kube-OVN

- 一个 Kubernetes CNI 的实现
- 一个为 Kubernetes 设计的 SDN
- 一个 CNCF Sandbox 项目

什么是 OVN

- 一个 OVS 的控制平面
- OVN to OVS ≈ Kubernetes to Docker

- Overlay 网络，UDP进行封装
- 逻辑路由器，逻辑交换机
- ACL，QoS
- 负载均衡器，网关，DNS，DHCP



一年前的测试

K8S CNI Benchmark - Pod to P

CNI setup without any

Bare metal
Antrea
Calico MTU
Canal MTU
Cilium
Flannel
Kube-OVN
Kube-router
Weave Net MTU

2020-08-27 - Alexis Ducastel - infrabuilder - Source : https://github.com/

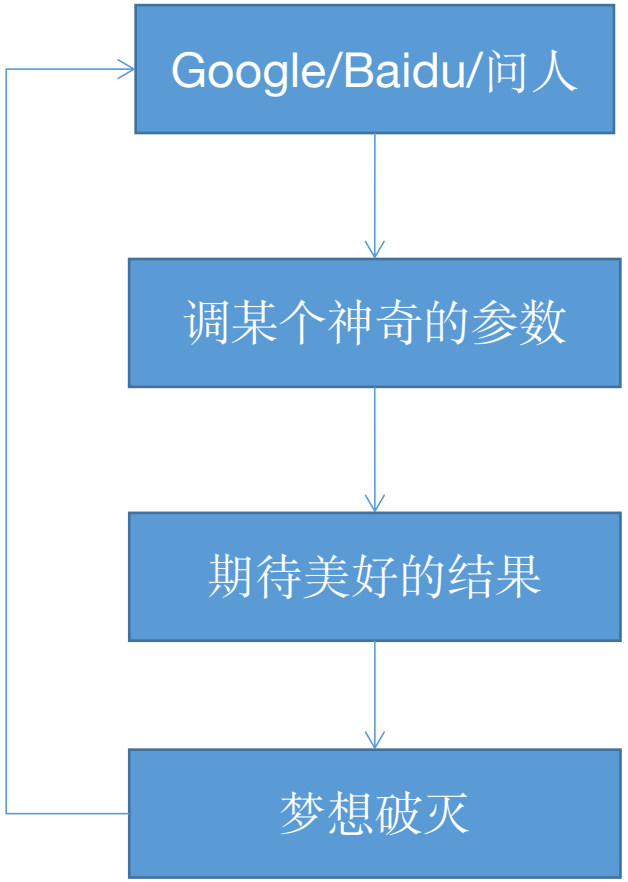
P - Bandwidth

9885
9796
9757
9742
9726
9846
9593

chmark-k8s-cni-2020-08

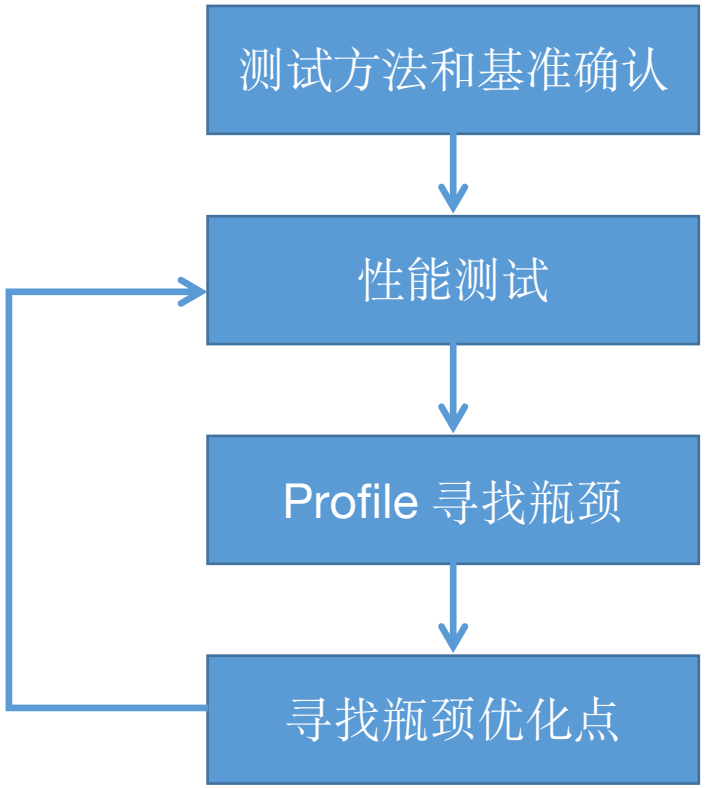
												Security features	
												Encryption	
												activation	Performance
												at deploy time	Slow
												anytime	Very fast
												no	n/a
												at deploy time	Slow
												no	n/a
												no	n/a
												no	n/a
Weave Net	manual	very fast	very fast	very fast	Fast	very low	Low	Low	Low	Low	yes	yes	at deploy time	Slow

基本方法



- 结果：什么都没调出来

基本方法



- 目标：不能和其他网络插件差太多
- 结果：某些场景下性能比宿主机网络还要好点

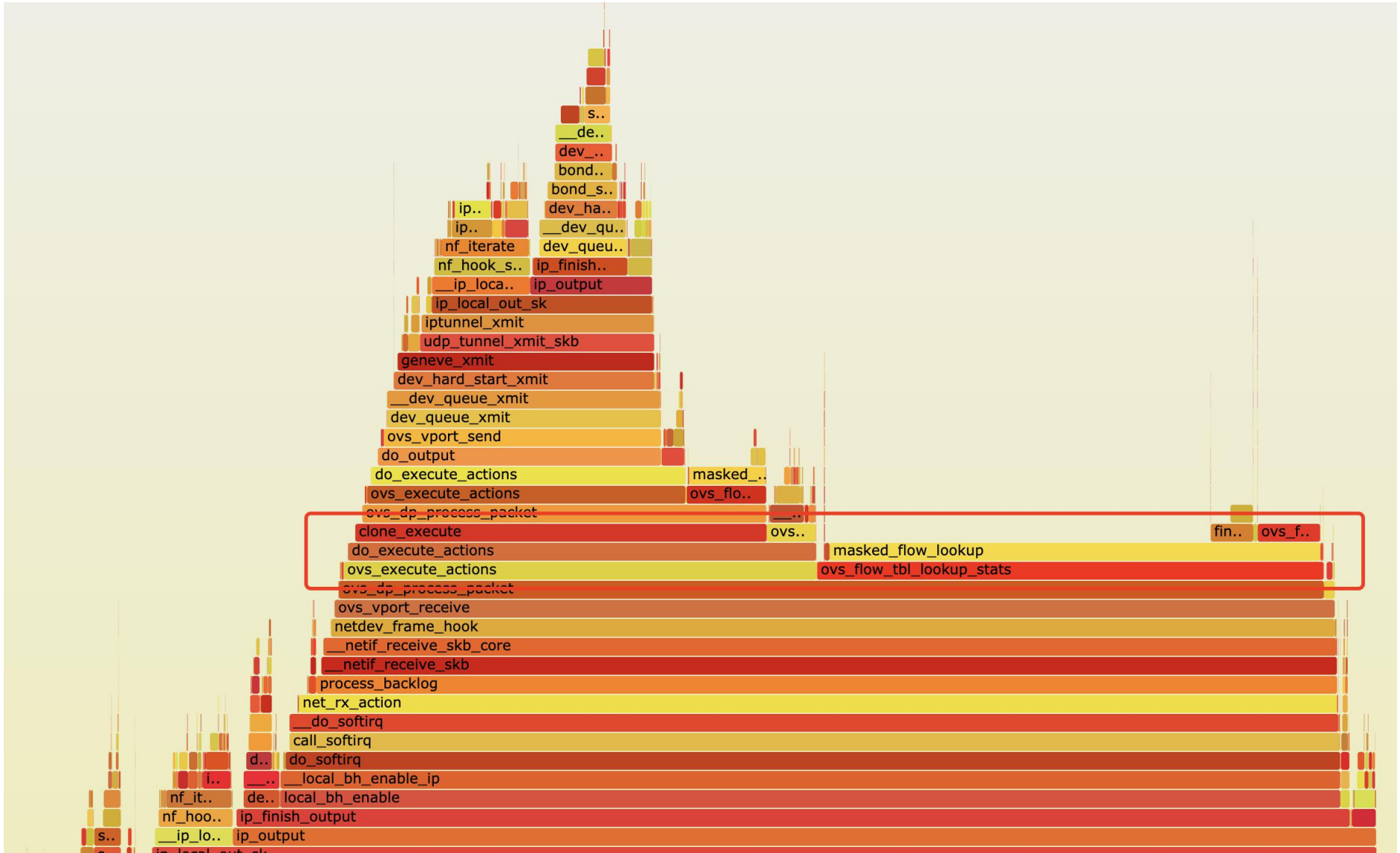
- 一些结论和方法可适用于其他网络插件

测试基准和方法

- 使用固定~~物理机~~物理机和虚拟机，只运行性能测试容器避免感干扰
- cpu 性能模式，避免延迟测试浮动
  - cpupower frequency-set -g performance
- qperf 收集~~小包~~不同大小包的 ~~udp_lat~~ tcp_bw/udp_bw/tcp_lat/udp_lat
  - qperf 10.16.0.4 -ub -oo msg_size:1 -vu tcp_lat tcp_bw udp_lat udp_bw
- 火焰图收集 CPU Profile 信息
  - perf record -F 999 -a -g -p $(pidof qperf)

1byte	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
host	13.1	12.4	28.2	6.02
container	25.7	22.9	27.1	1.59

第一轮Profile



OVS NAT action花费30%CPU

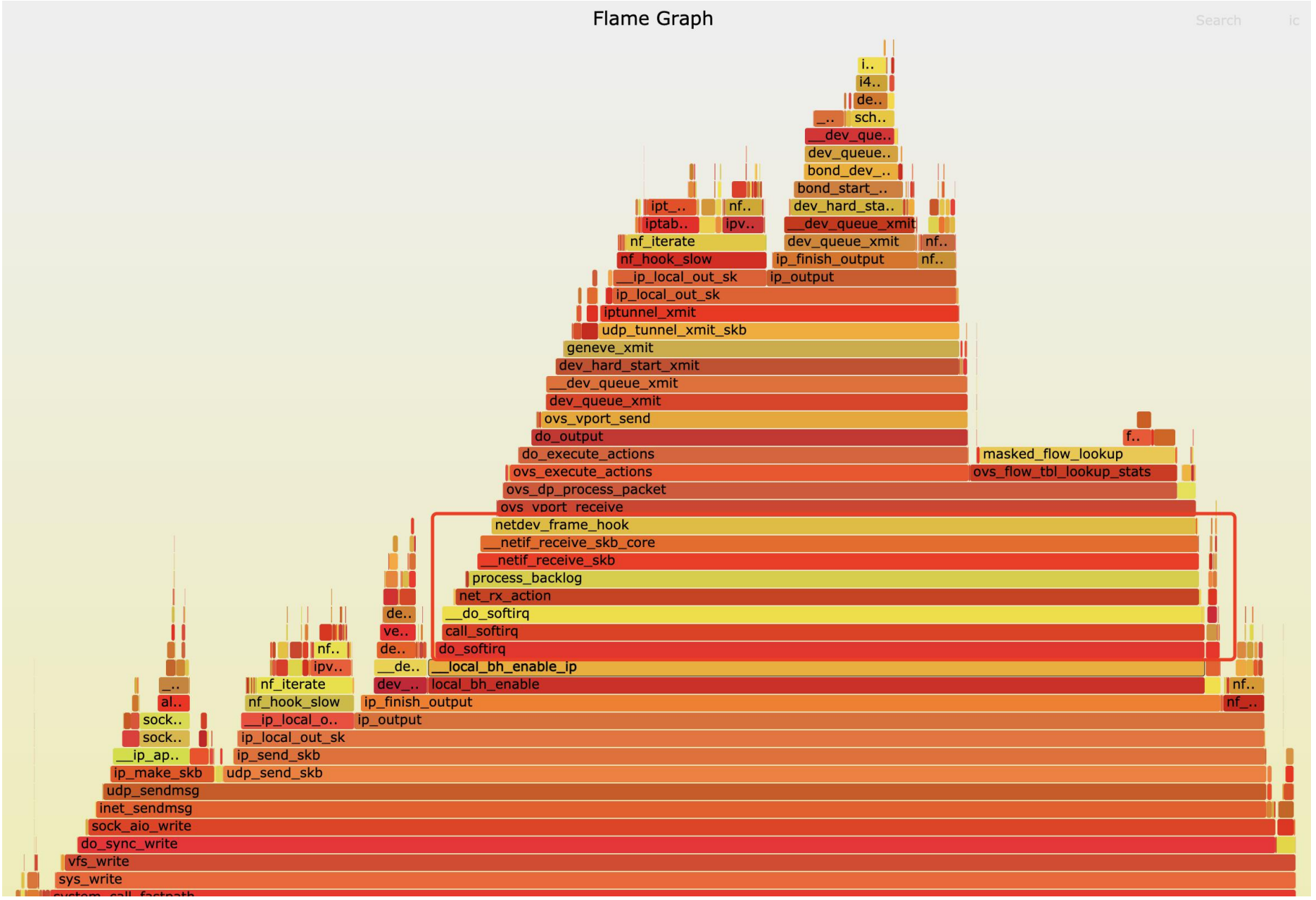
第一轮优化

- OVN 内部逻辑只要用 LB 所有流量都会过 ct 和 ovs_clone
- 删除 OVN LB 逻辑，改回用 kube-proxy 实现服务发现
  - ovn-nbctl ls-lb-del ovn-defaul
  - 更改ovn 流表编排非 svc 流量不过 ct 和 ovs_clone

	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
host	13.1	12.4	28.2	6.02
container	25.7	22.9	27.1	1.59
without lb	18.5	16.5	27.5	2.8

- 延迟降低 30%

第二轮 Profile



发包过程中为什么会有软中断和recieve 操作?

veth处理消耗 5% CPU

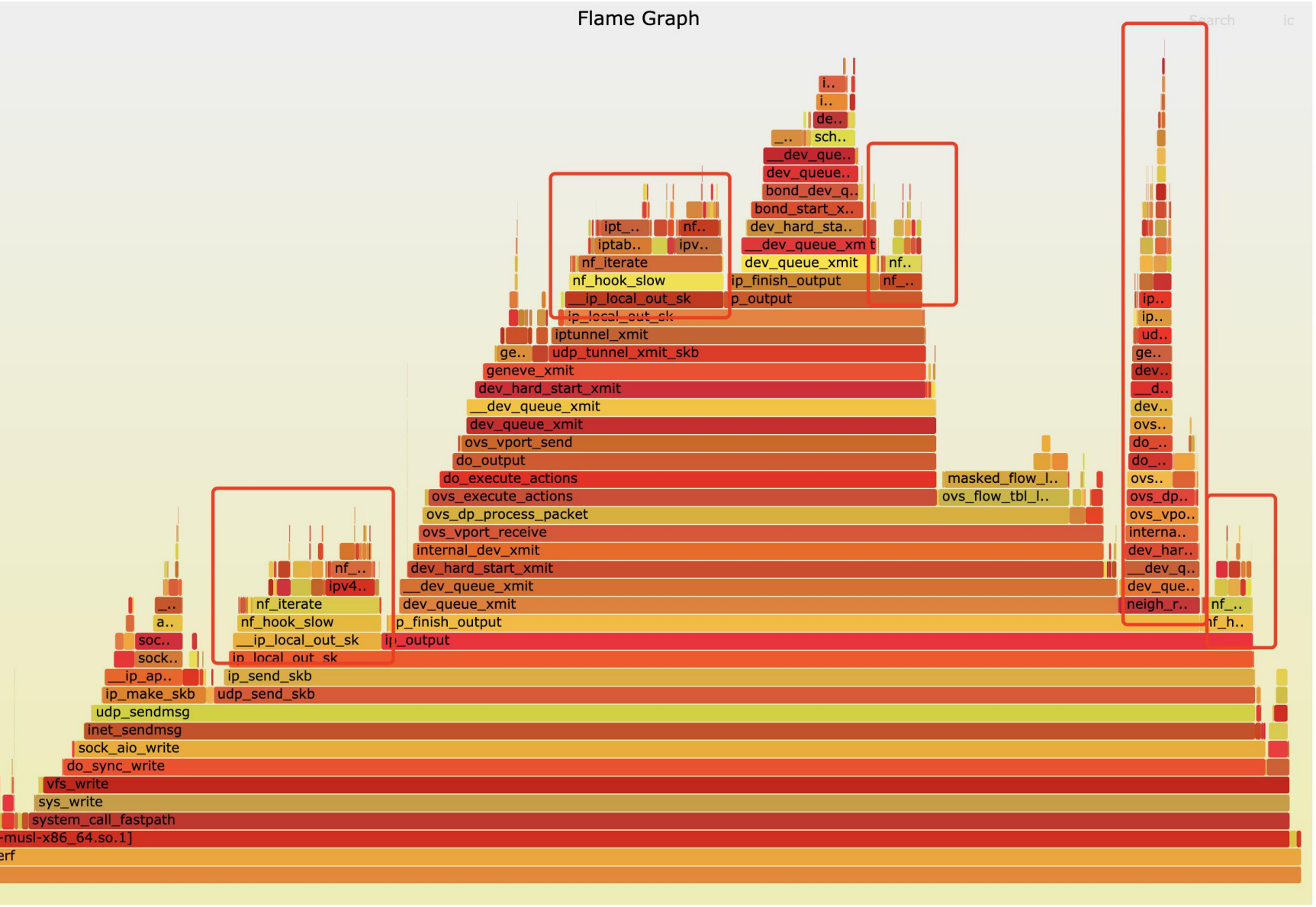
第二轮优化

- 使用 internal-port 代替 veth

	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
host	13.1	12.4	28.2	6.02
container	25.7	22.9	27.1	1.59
without lb	18.5	16.5	27.5	2.8
internal-port	18	16.7	27.4	3.13

- 延迟无明显变化

第三轮 Profile



新增 neigh_resolve_output 占据 6% CPU 消耗

nf_hook_slow 27% CPU 消耗

iptables/nftables/ipvs/conntrack 均会注册 hook

容器/宿主机 * 2 nf_hook_slow in/out *2 nf_hook_slow

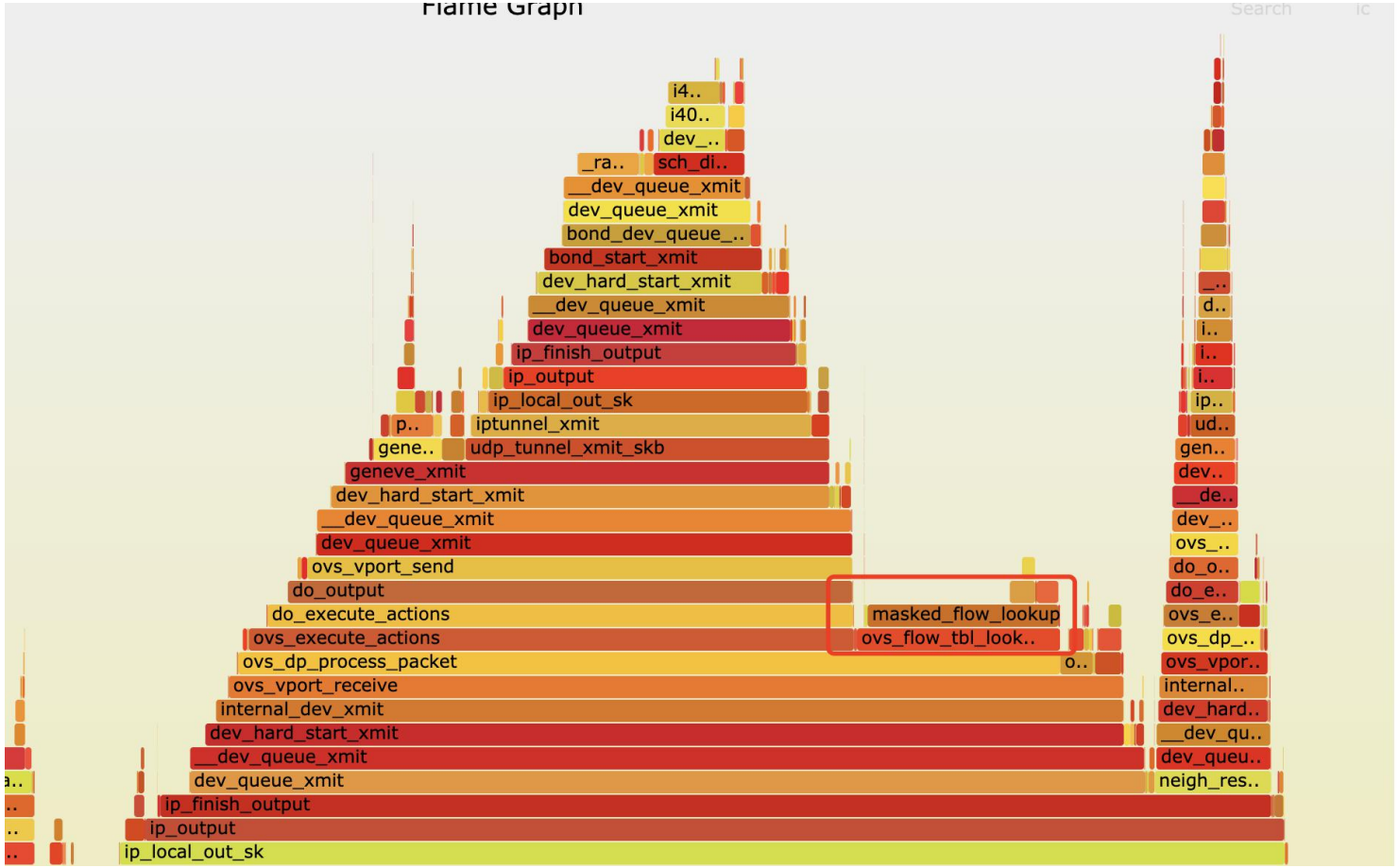
第三轮优化

- 编写 netfilter 模块提前绕行容器网络流量和隧道流量
  - NF_STOP 提前终止 netfilter 执行
  - https://github.com/kubeovn/kube-ovn/tree/master/fastpath

	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
host	13.1	12.4	28.2	6.02
container	25.7	22.9	27.1	1.59
without lb	18.5	16.5	27.5	2.8
internal-port	18	16.7	27.4	3.13
netfilter	15.1	13.5	27.5	4.82

- 延迟降低 20%

第四轮 Profile



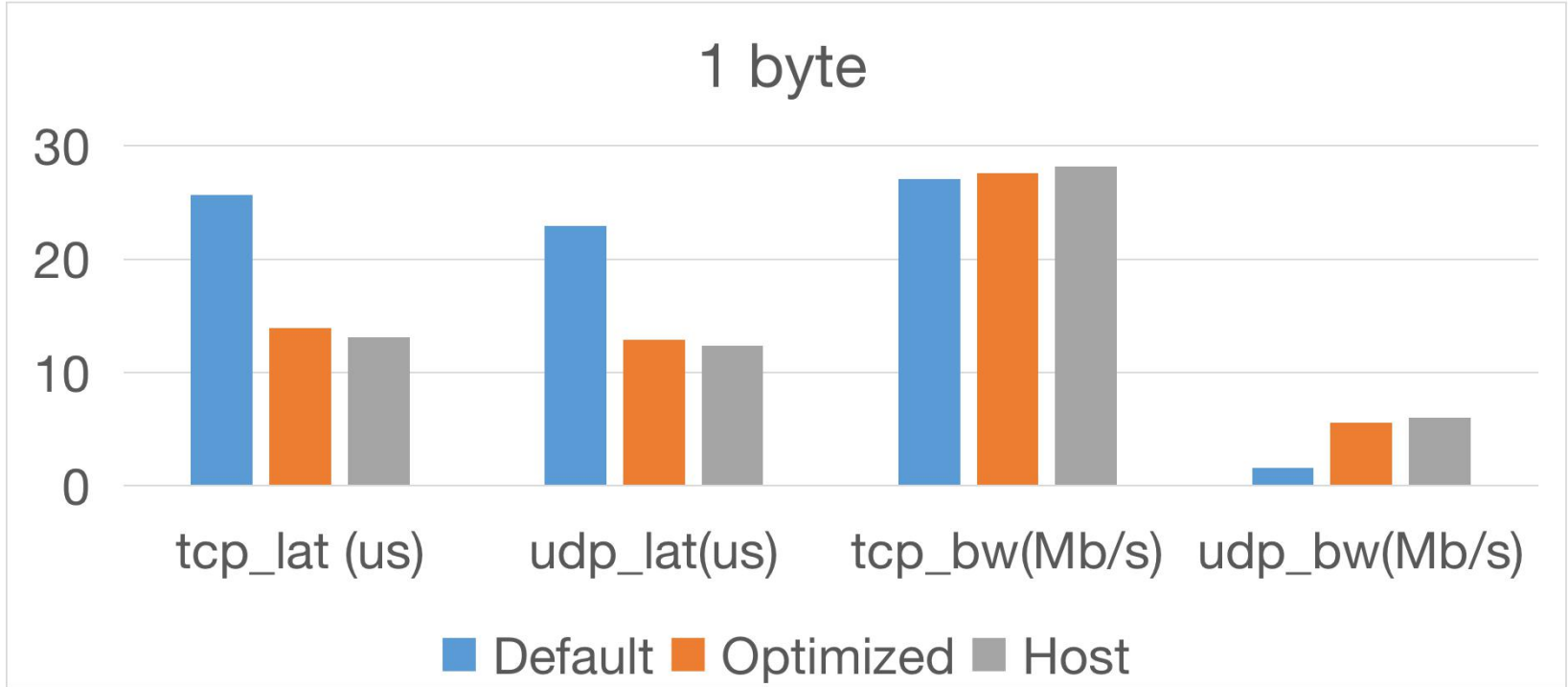
ovs_flow_tbl_lookup 占据 10% CPU 消耗

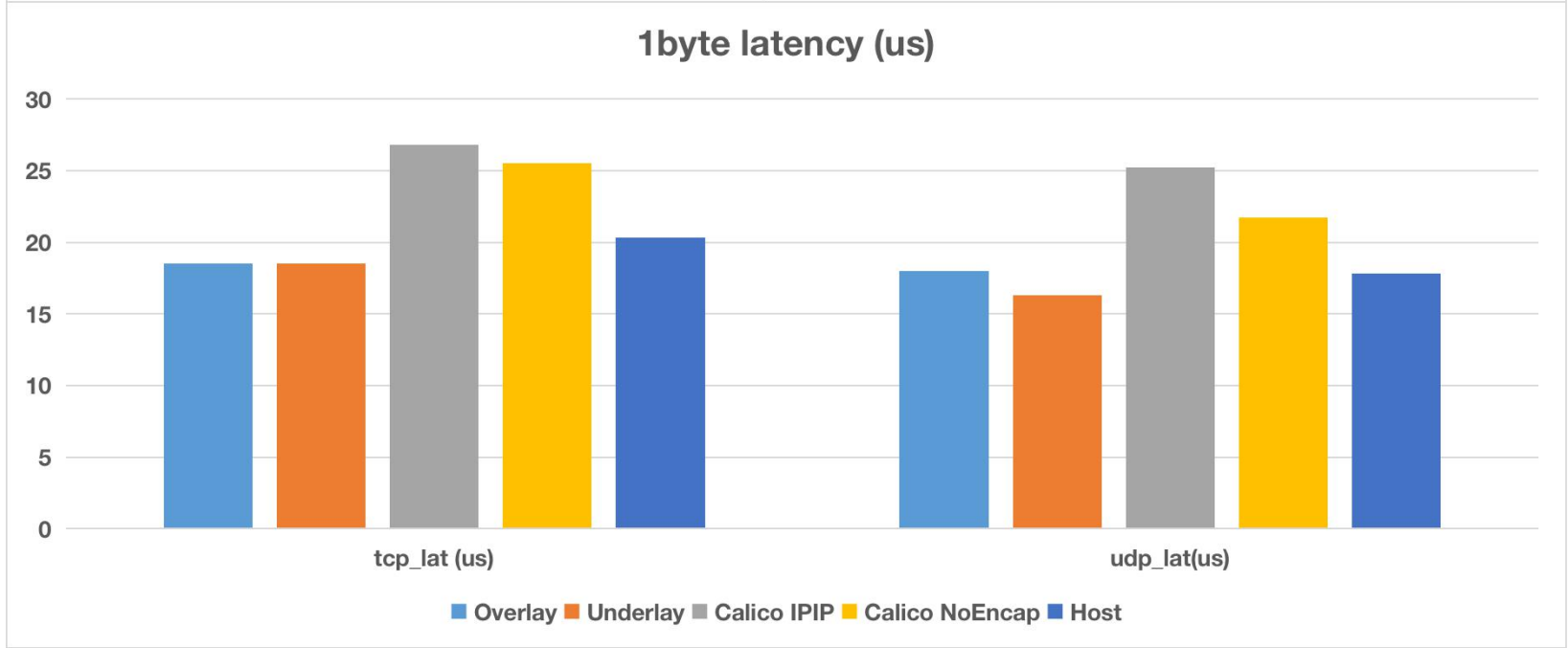
第四轮优化

- 重新编译 ovs 内核模块，增加 x86 指令优化
  - ./configure --with-linux=/lib/modules/$(uname -r)/build CFLAGS="-g -O2 -mpopcnt -msse4.2"

	tcp_lat (us)	udp_lat (us)	tcp_bw (Mb/s)	udp_bw(Mb/s)
host	13.1	12.4	28.2	6.02
container	25.7	22.9	27.1	1.59
without lb	18.5	16.5	27.5	2.8
internal-port	18	16.7	27.4	3.13
netfilter	15.1	13.5	27.5	4.82
intel optimize	13.9	12.9	27.6	5.57

- 延迟降低 5%


1 byte
30
20
10
0
tcp_lat (us)
udp_lat(us)
tcp_bw(Mb/s)
udp_bw(Mb/s)
Default
Optimized
Host


1byte latency (us)
30
25
20
15
10
5
0
tcp_lat (us)
udp_lat(us)
Overlay
Underlay
Calico IPIP
Calico NoEncap
Host

性能问题解决了

1K数据包吞吐量优化

- qperf -t 30 10.16.0.4 -ub -oo msg_size:1K -vu tcp_lat tcp_bw udp_lat udp_bw

- perf record -F 999 -a -g -p $(pidof qperf)

	tcp_lat (us)	udp_lat (us)	tcp_bw (Gb/s)	udp_bw(Gb/s)
host	28.4	27.2	9.41	5.81
container	28.2	28.2	4.94	5.13

1K数据包吞吐量优化

```
     port.tx_dropped_link_down: 0
[root@da-c3-small-x86-02 165]# ethtool -S eno2 |grep droppe
     rx_dropped: 5732843
     tx_dropped: 0
     port.rx_dropped: 0
     port.tx_dropped_link_down: 0
[root@da-c3-small-x86-02 165]#
```

- ethtool -G eno2 rx 4096

	tcp_lat (us)	udp_lat (us)	tcp_bw (Gb/s)	udp_bw(Gb/s)
host	28.4	27.2	9.41	5.81
container	28.2	28.2	4.94	5.13
4096 rx	28.2	28.3	8.39	5.28

性能问题又解决了

虚拟机吞吐量优化

```
[root@liumengxin-ovn1-192 ~]# iperf3 -c 192.168.16.45
Connecting to host 192.168.16.45, port 5201
[  4] local 192.168.16.44 port 54198 connected to 192.168.16.45 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  1.43 GBytes  12.2 Gbits/sec    1   3.01 MBytes
[  4]   1.00-2.00   sec  2.04 GBytes  17.6 Gbits/sec    0   3.01 MBytes
[  4]   2.00-3.00   sec  2.34 GBytes  20.1 Gbits/sec    0   3.01 MBytes
[  4]   3.00-4.00   sec  1.96 GBytes  16.9 Gbits/sec    0   3.01 MBytes
[  4]   4.00-5.00   sec  2.06 GBytes  17.7 Gbits/sec    0   3.01 MBytes
[  4]   5.00-6.00   sec  1.75 GBytes  15.1 Gbits/sec    0   3.01 MBytes
[  4]   6.00-7.00   sec  1.81 GBytes  15.5 Gbits/sec    0   3.01 MBytes
[  4]   7.00-8.00   sec  1.91 GBytes  16.4 Gbits/sec    0   3.01 MBytes
[  4]   8.00-9.00   sec  1.96 GBytes  16.8 Gbits/sec    0   3.01 MBytes
[  4]   9.00-10.00  sec  2.01 GBytes  17.3 Gbits/sec    0   3.01 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  19.3 GBytes  16.5 Gbits/sec    1             sender
[  4]   0.00-10.00  sec  19.3 GBytes  16.5 Gbits/sec                  receiver

iperf Done.
[root@liumengxin-ovn1-192 ~]# docker exec -it d1c4dc1858ed sh
/ # iperf3 -c 10.16.0.14
Connecting to host 10.16.0.14, port 5201
[  4] local 10.16.0.8 port 52462 connected to 10.16.0.14 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   160 MBytes  1.34 Gbits/sec    0   1.55 MBytes
[  4]   1.00-2.00   sec   200 MBytes  1.68 Gbits/sec    0   1.55 MBytes
[  4]   2.00-3.00   sec   216 MBytes  1.81 Gbits/sec    0   1.55 MBytes
[  4]   3.00-4.00   sec   181 MBytes  1.52 Gbits/sec    0   1.55 MBytes
[  4]   4.00-5.00   sec   176 MBytes  1.48 Gbits/sec    0   1.55 MBytes
[  4]   5.00-6.00   sec   202 MBytes  1.70 Gbits/sec    0   1.55 MBytes
[  4]   6.00-7.00   sec   185 MBytes  1.55 Gbits/sec    0   1.55 MBytes
[  4]   7.00-8.00   sec   200 MBytes  1.68 Gbits/sec    0   1.55 MBytes
[  4]   8.00-9.00   sec   209 MBytes  1.75 Gbits/sec    0   1.55 MBytes
[  4]   9.00-10.00  sec   181 MBytes  1.52 Gbits/sec    0   1.55 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  1.87 GBytes  1.60 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  1.86 GBytes  1.60 Gbits/sec                  receiver

iperf Done.
```

virtio_net 类型虚拟网卡无法调整 rx queue

Profile 发现 client 端存在瓶颈大量 tcp_push 怀疑无法使用 tso 等卸载能力

Geneve 封装为 UDP 虚拟机 CPU 能力不足情况下差距明显

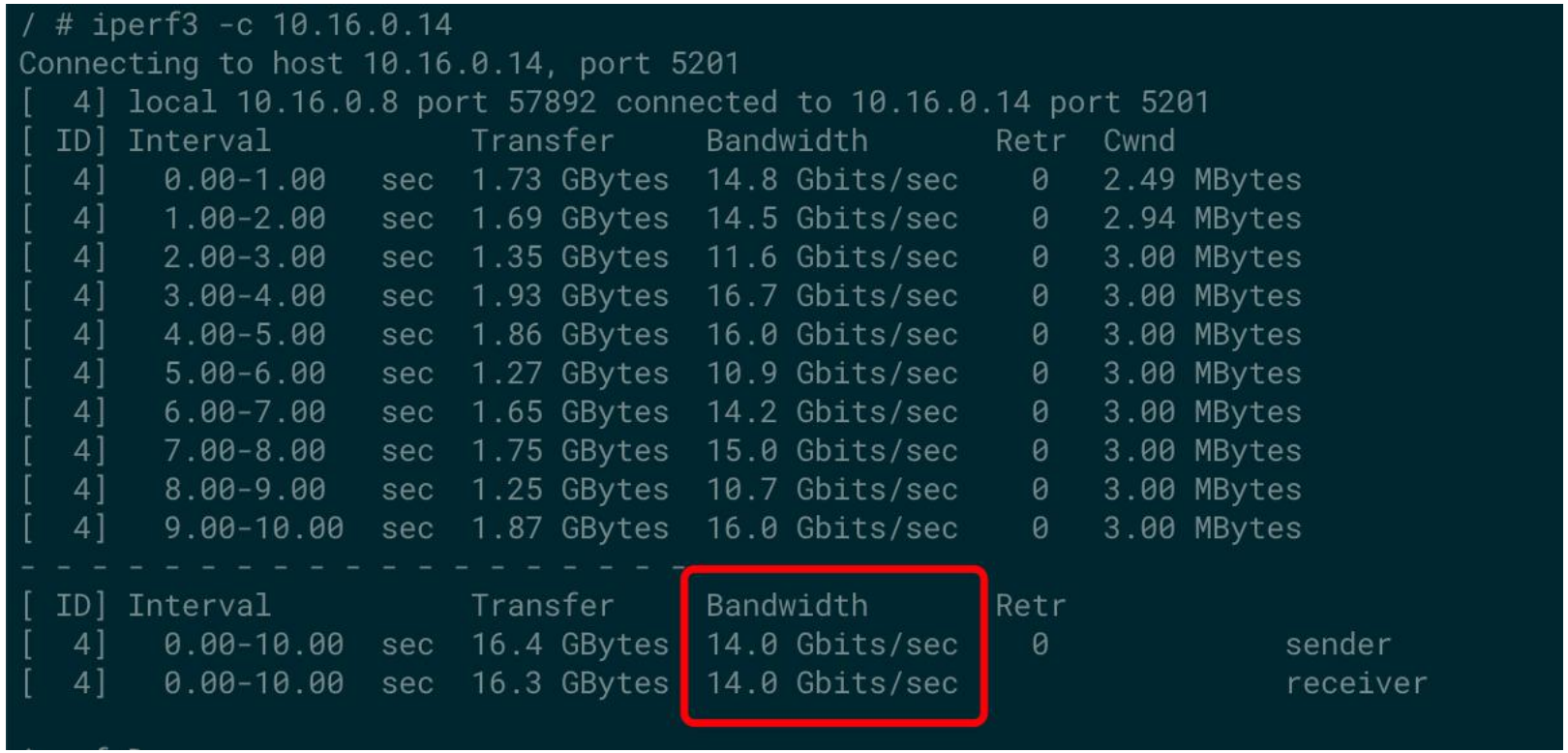
虚拟机吞吐量优化

- 使用 STT 封装代替 Geneve 封装
  - ovs-vsctl set open . external_ids:ovn-encap-type=stt
  - stt 使用 tcp 格式 header 可利用网卡 tso 能力
  - 1.6Gbit/s -> 11.7 Gbit/s
  - profile 发现 STT 依赖 netfilter 实现，带回 20% cpu 开销

```
/ # iperf3 -c 10.16.0.14
Connecting to host 10.16.0.14, port 5201
[  4] local 10.16.0.8 port 55140 connected to 10.16.0.14 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  1.26 GBytes  10.8 Gbits/sec    0   2.51 MBytes
[  4]   1.00-2.00   sec  1.26 GBytes  10.9 Gbits/sec    0   3.02 MBytes
[  4]   2.00-3.00   sec  1.60 GBytes  13.7 Gbits/sec    0   3.02 MBytes
[  4]   3.00-4.00   sec  1.52 GBytes  13.1 Gbits/sec    0   3.02 MBytes
[  4]   4.00-5.00   sec  1.36 GBytes  11.6 Gbits/sec    0   3.02 MBytes
[  4]   5.00-6.00   sec  1.32 GBytes  11.4 Gbits/sec    0   3.02 MBytes
[  4]   6.00-7.00   sec  1.16 GBytes  9.97 Gbits/sec    0   3.02 MBytes
[  4]   7.00-8.00   sec  1.17 GBytes  10.1 Gbits/sec    0   3.02 MBytes
[  4]   8.00-9.00   sec  1.47 GBytes  12.7 Gbits/sec    0   3.02 MBytes
[  4]   9.00-10.00  sec  1.47 GBytes  12.6 Gbits/sec    0   3.02 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  13.6 GBytes  11.7 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  13.6 GBytes  11.7 Gbits/sec                  receiver
```

虚拟机吞吐量优化

- STT 模块优化
  - STT netfilter hook 挂载点前移
  - Netfilter OUTPUT 跳过 STT 端口流量处理
  - 11.7Gbit/s -> 14.0 Gbit/s



经验总结

- 经过优化 Kube-OVN 可以达到接近宿主机水平性能
- 其他容器网络可以遵循类似的方法进行优化
- 优化建议
  - 慎用 OVN LB，避免延迟开销
  - 使用 STT 封装代替 Geneve/Vxlan 获取更好吞吐量
  - 性能关键链路避开 Netfilter
  - 利用指令集相关优化进行编译
  - 调整 rx queue 长度
  - 先 profile 后优化
  - 不要畏惧改内核（20%+ 性能提升）

未来的方向

- OVS-DPDK? eBPF?
- 智能网卡
  - ct，openflow，tunnel，qos offload
  - arm core: ovs/ovn offload
- UDP offload
  - https://developers.redhat.com/articles/2021/11/05/improve-udp-performance-rhel-85#
- New CT Solution
  - https://www.youtube.com/watch?v=LIxyZaDwKAc&list=PLaJlRa-xItwARDGAUp7lXviOgOhcRxSU-&index=31
  - https://www.openvswitch.org/support/ovscon2021/slides/a_new_solution_conntrack.pdf

Thanks