Kubernetes networking: Is there a cheetah within your Calico?

Even faster Kubernetes clusters with Calico, VPP & memif

Nathan Skrzypczak, Cisco Systems
Chris Tomkins, Tigera
FOSDEM 2022

Your Speakers Today:

- Nathan Skrzypczak - Software Engineer @ Cisco - Calico/VPP integration contributor.
- Biking and hiking enthusiast - even sea-kayaking at times
- And a french accent despite the name

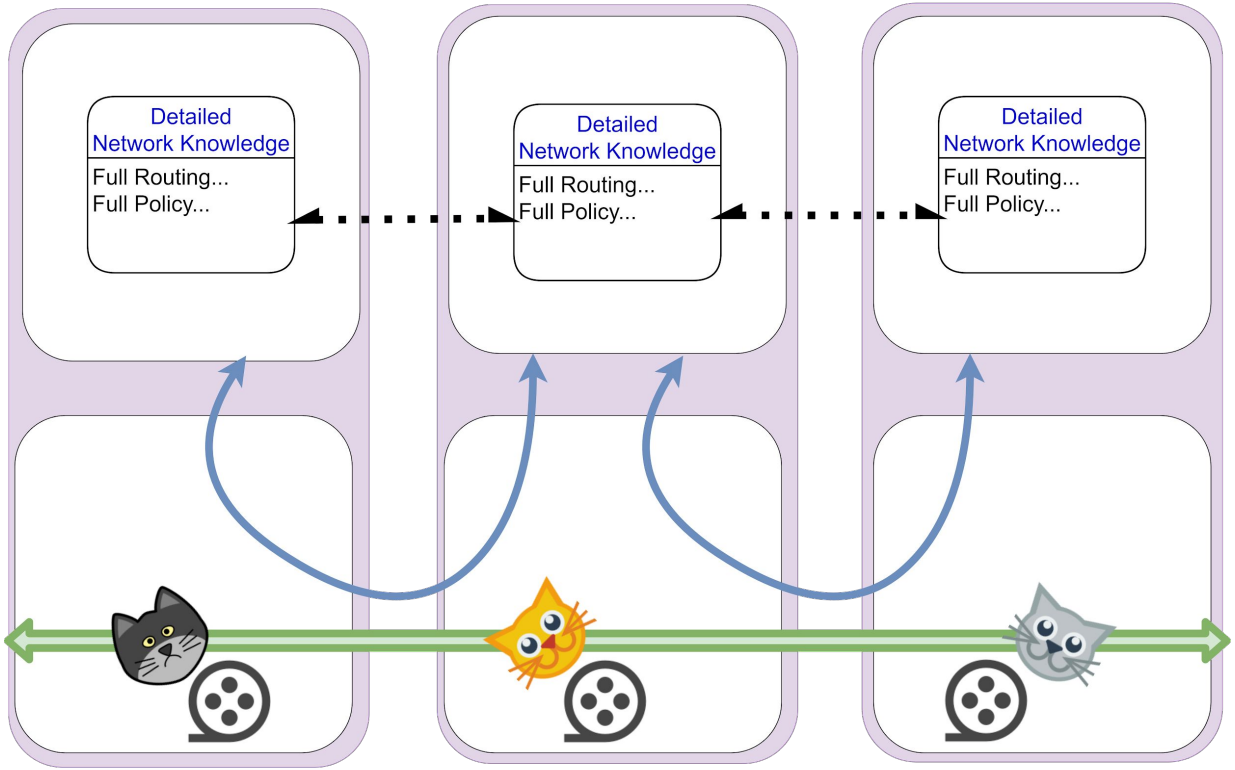
Your Speakers Today:

- Chris Tomkins - Lead Developer Advocate @ Tigera (Project Calico)
- Today's obsession: Japanese on Duolingo; but progress is not quick!
- I'm never without music; try Rustie.
- I'm always looking to learn and share. Let's connect!

What is Calico?

- Open-source Kubernetes networking and network policy
- Kubernetes pods, nodes, VMs, and legacy workloads
- Rich network policy APIs
- Battle-tested: deployed in production at scale
- Support for multiple data planes

Control plane (network knowledge)



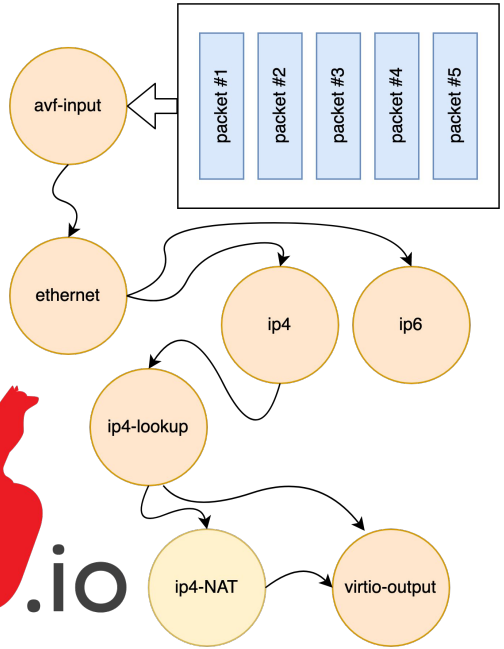
Data plane (cat videos)

Control Plane/Data Plane

- A data plane is the component of a networking device that transports user data. Calico offers:
  - **Linux iptables:**
    - Heavily battle tested
    - Good performance
    - Great compatibility and wide support
  - **Windows Host Networking Service:**
    - Windows containers can be deployed and secured
    - Works in any cloud computing provider or on-premises
  - **Linux eBPF:**
    - Scales to higher throughput/uses less CPU per Gigabit
    - Reduces first packet latency to services
    - Preserves external client source IP addresses all the way to the pod
    - Supports DSR (Direct Server Return) for better efficiency
- And... **VPP**!

What is VPP?

- Fast, open-source userspace networking dataplane https://fd.io/
- Feature-rich L2/L3/L4 networking Tunneling, NAT, ACL, crypto, TCP, Quic,...
- Easily extensible through plugins
- Supports virtual and physical interfaces
- Fast API >200k updates/second
- Highly optimized for performance: vectorization, cache efficiency
- Multi-architecture: x86, ARM



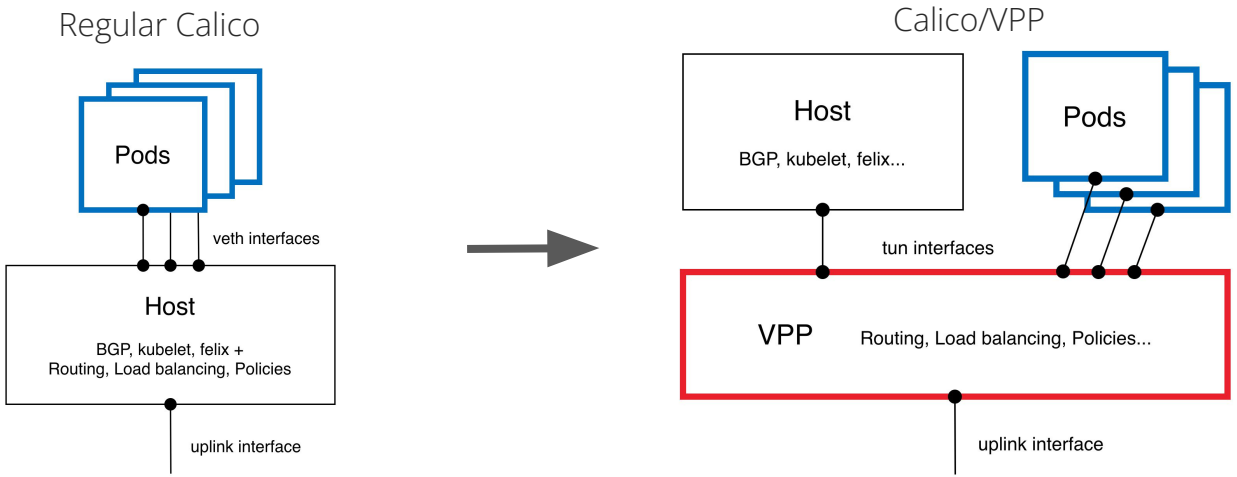
Calico/VPP integration

- VPP dataplane option for Calico
  - Deployed on all nodes as a DaemonSet
  - Transparent for users (e.g. operator)
- Calico control plane configured to drive VPP
  - Optimized NAT plugin for service load balancing
  - Specific plugin for efficient Calico policies
- VPP optimized for container environments:
  - Interrupt mode, SCHED_RR scheduling
  - Lightweight (no hugepages, no dpdk, ...)
  - GRO / GSO support for container interfaces

```
# dedicated configmap for VPP settings
kind: ConfigMap
apiVersion: v1
metadata:
  name: calico-vpp-config
  namespace: calico-vpp-dataplane
data:
  # K8s service prefix. We currently cannot retrieve this from the API,
  # so it must be manually configured
  service_prefix: 10.96.0.0/12

  # Configure the name of VPP's physical interface
  vpp_dataplane_interface: eth1

  # Configures how VPP grabs the physical interface
  # available values are :
  # - ""         : will select the fastest driver among those supported for this interface
  # - avf        : use the native AVF driver
  # - virtio     : use the native virtio driver (requires hugepages)
  # - af_xdp     : use AF_XDP sock family (require at least kernel 5.4)
  # - af_packet  : use AF_PACKET sock family (slow but failsafe)
  # - none       : dont configure connectivity
  vpp_uplink_driver: ""

  # Configuration template for VPP.
  vpp_config_template: |-
    unix {
      nodaemon
      full-coredump
      cli-listen /var/run/vpp/cli.sock
      pidfile /run/vpp/vpp.pid
      exec /etc/vpp/startup.exec
    }
```

How does it work ?

- VPP inserts itself between the host and the network
  - Uplink consumed with optimised drivers :
    DPDK / native drivers / AF_XDP
  - Pure layer 3 network model (no ARP/mac address in the pods)



Why do this ?

- Adding dataplane functionalities

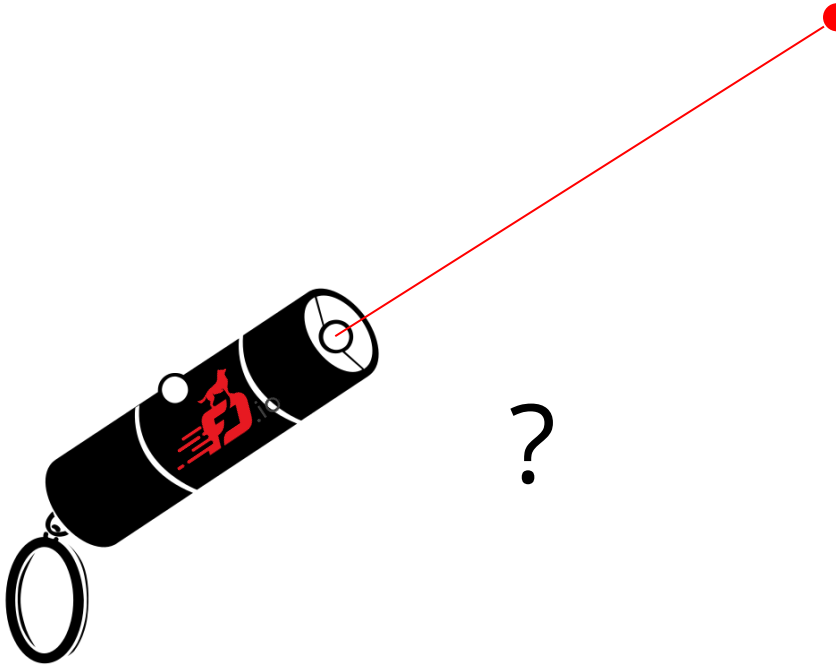
  (Maglev LB, srv6, ...)
- Extending the network (e.g. multi-net)

Image by Kyaw Thet / the nounproject

Why do this ?

- Adding dataplane functionalities

  (Maglev LB, srv6, ...)
- Extending the network (e.g. multi-net)
- Go Faster !



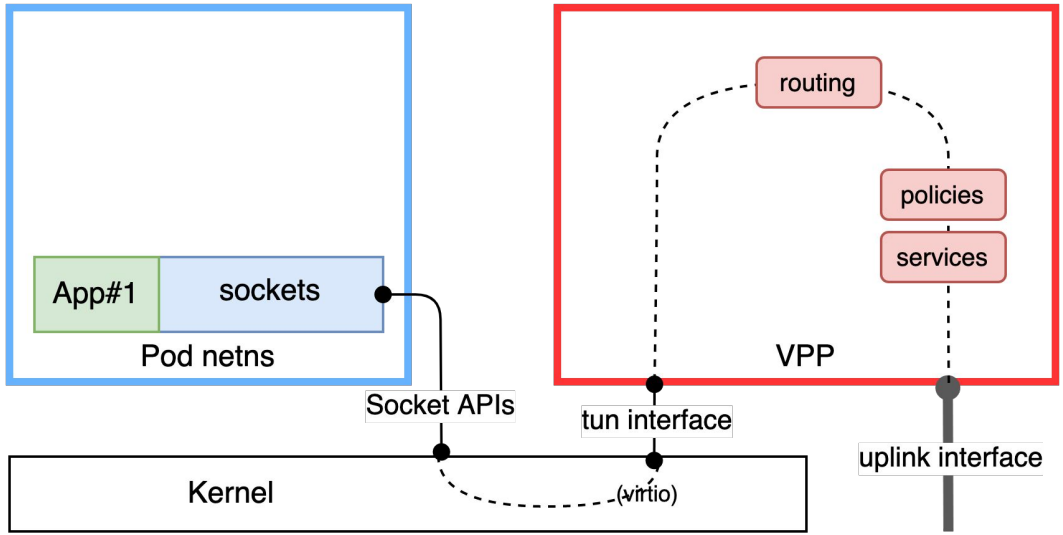
?

Image by Kyaw Thet / the nounproject

Optimizing the data path

Applications usually consume packets from the kernel with Socket APIs.

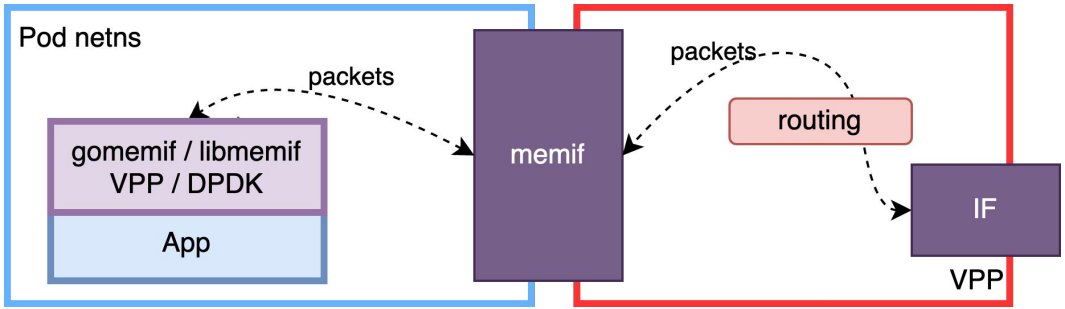
- Standard for apps
- But goes through the kernel
- Socket APIs were not designed for performance levels of modern apps
- Slower network (TCP, pps…) & crypto stack (hence GSO)
- Does two copies (VPP & socket)



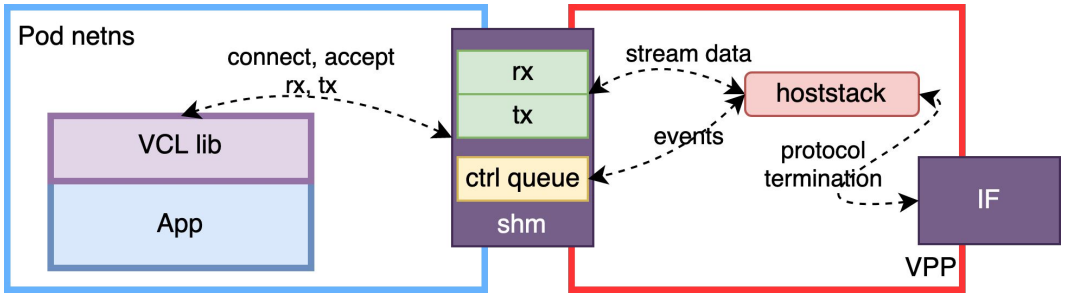
Optimizing the data path

Going straight from VPP to the application ?

- If the application handles packets : **memif interfaces**



- If the application terminates L4+ protocols : **VPP host stack**

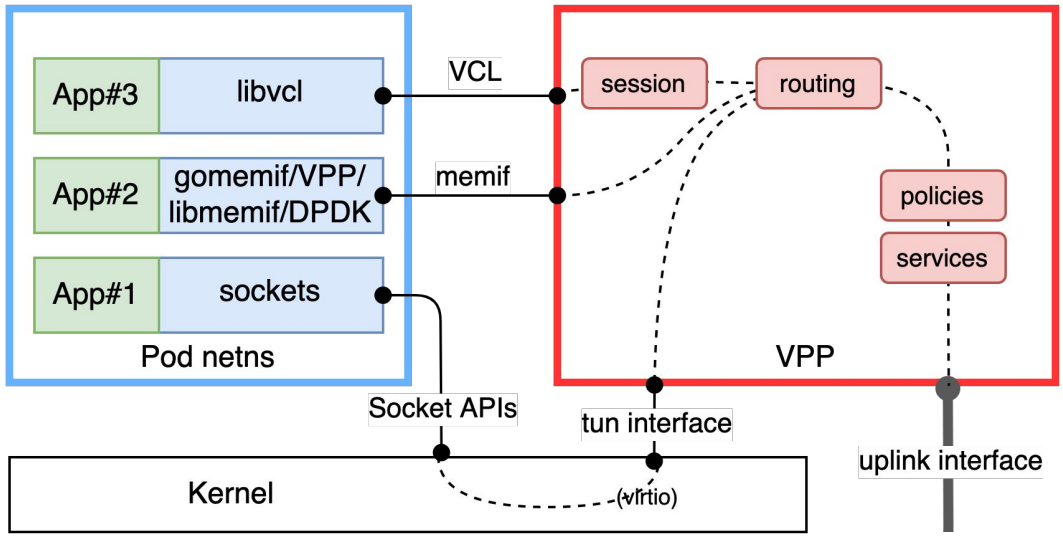


Optimizing the data path

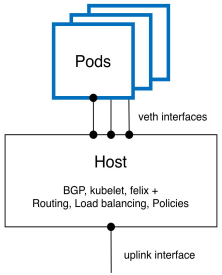
Going straight from VPP to the application ?

- If the application handles packets: **memif interfaces**
- If the application terminates L4+ connections: **VPP host stack**
- Exposed via pod annotations

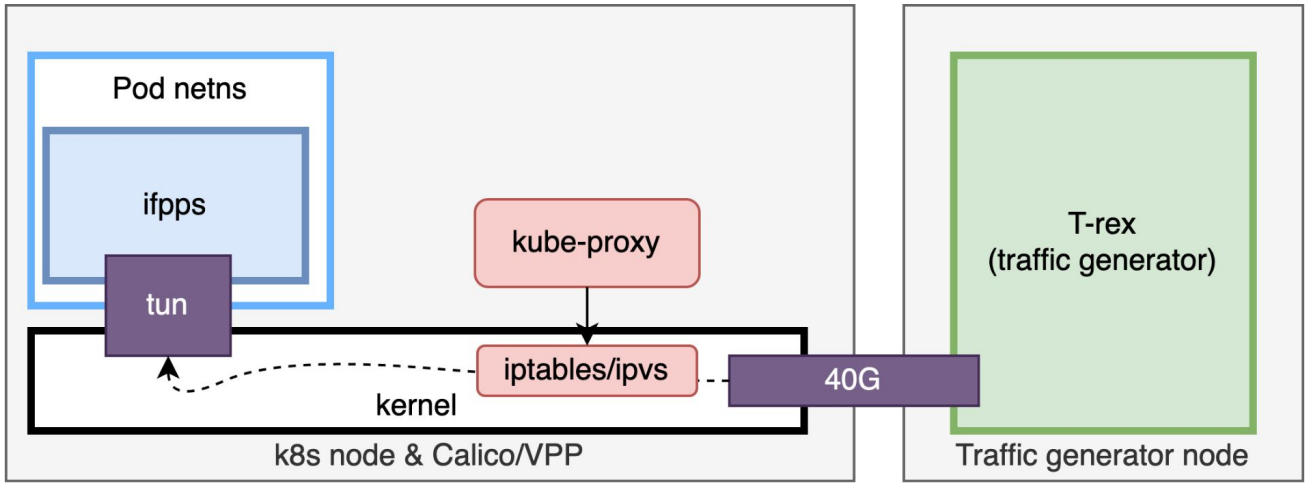
- Full userspace networking
- Zero copy APIs
- Regular sockets still work (e.g. DNS)



Small packets - Calico/linux

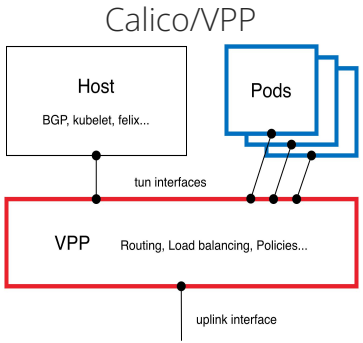


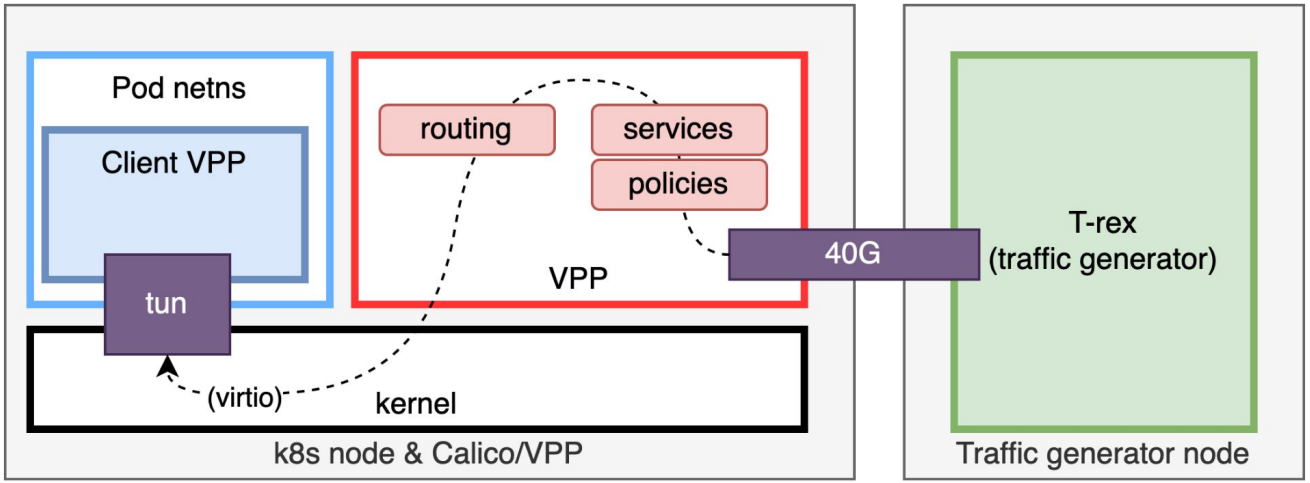
Regular Calico



2-node Skylake@3.2Ghz (baremetal) - ubuntu 20.04 - XL710, 40G NICs

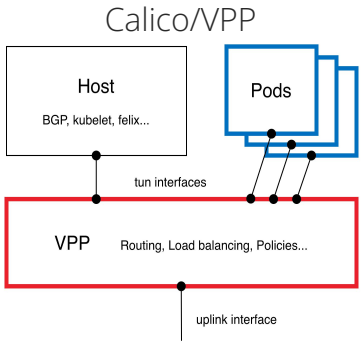
Small packets - Calico/VPP [virtio]

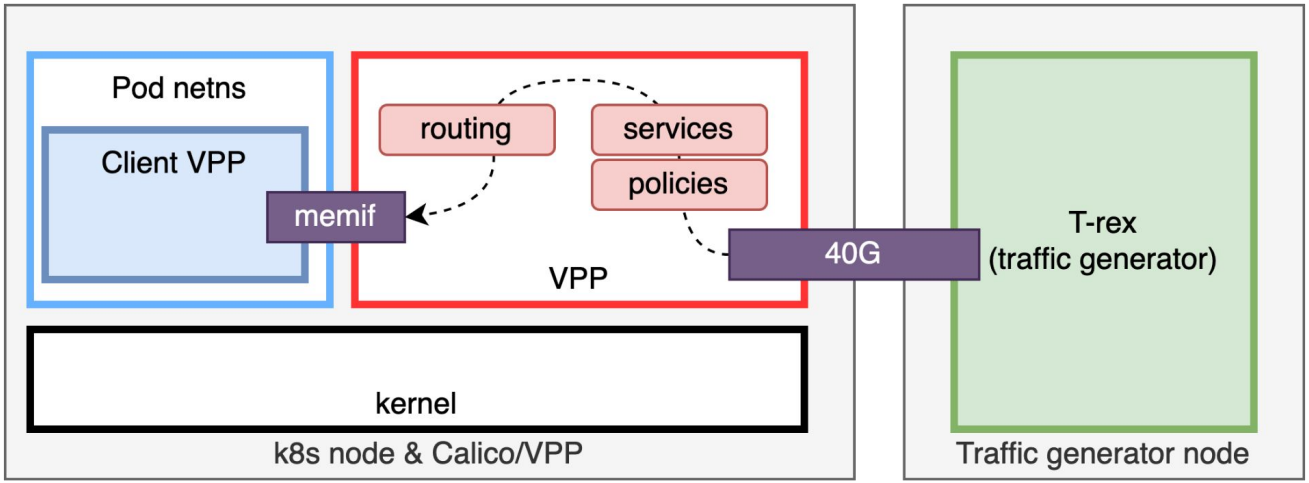




2-node Skylake@3.2Ghz (baremetal) - ubuntu 20.04 - XL710, 40G NICs

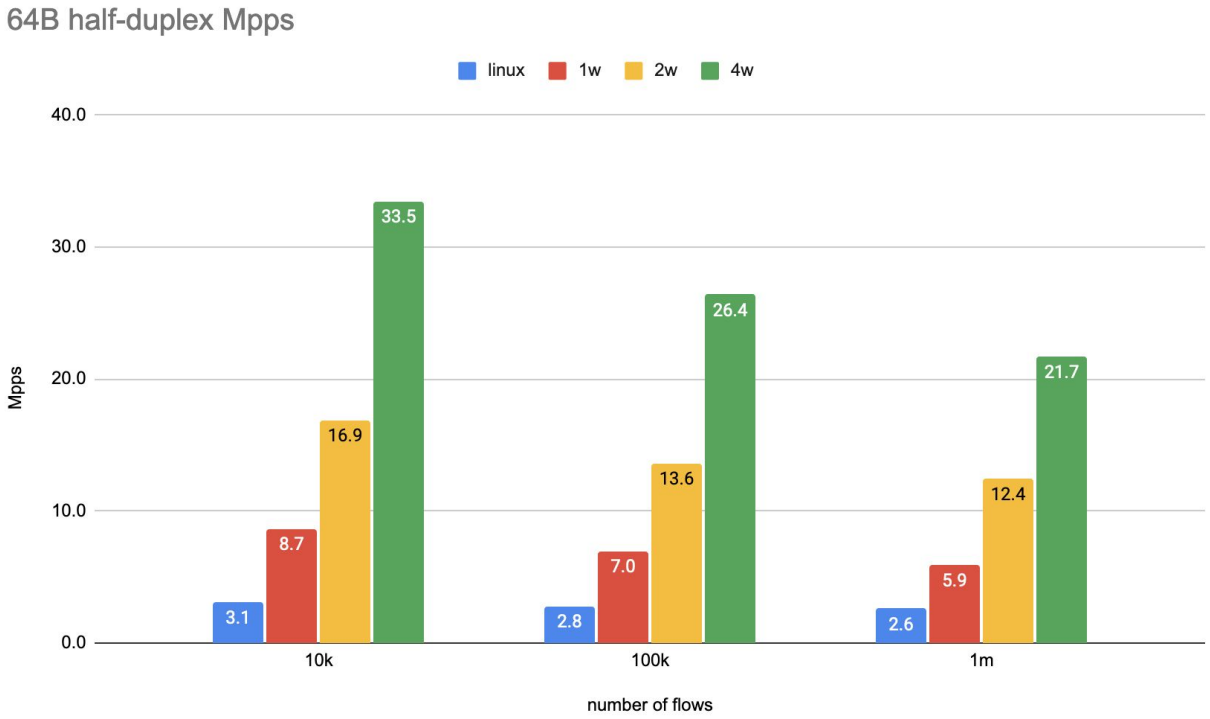
Small packets - Calico/VPP [memif]





2-node Skylake@3.2Ghz (baremetal) - ubuntu 20.04 - XL710, 40G NICs

Small, but fast packets !

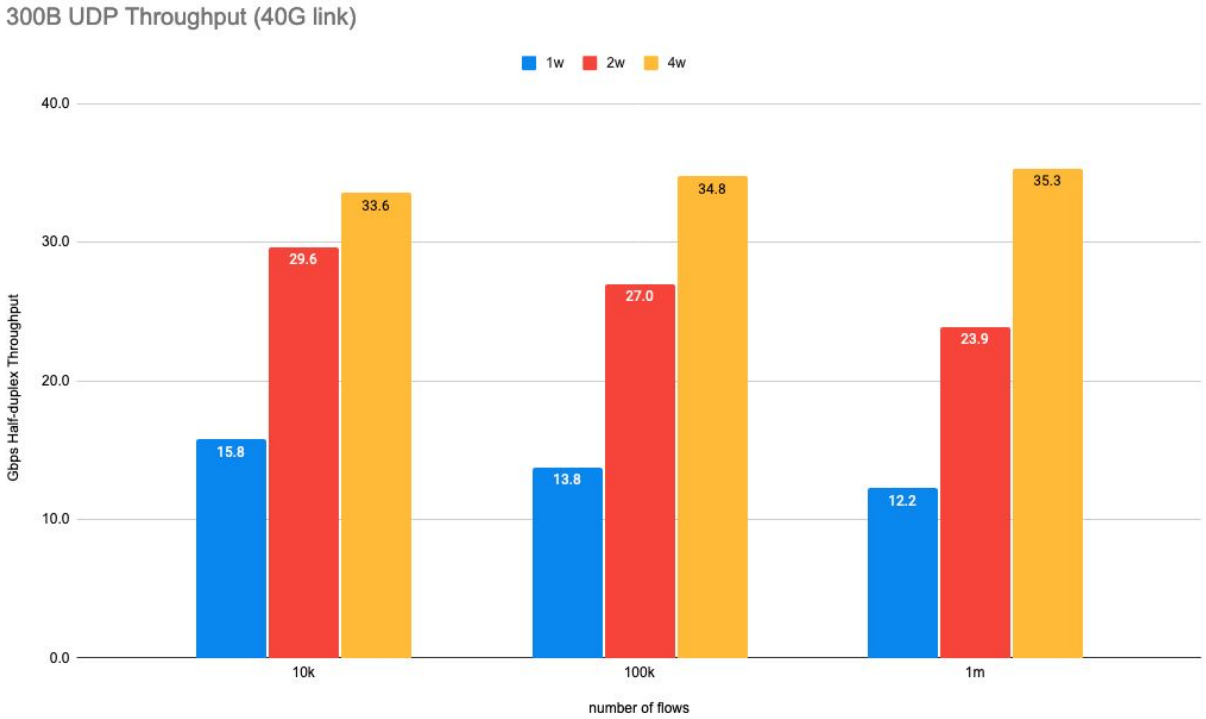


- UDP Packets (64B)
- Half duplex, measuring rx packets per second
- VPP runs with one main thread and the given number of workers

Scales linearly with VPP workers

ServiceIP cost ~5%

And throughput ?



- UDP Packets (300B)
- Half duplex, measuring rx throughput
- VPP runs with one main thread and the given number of workers

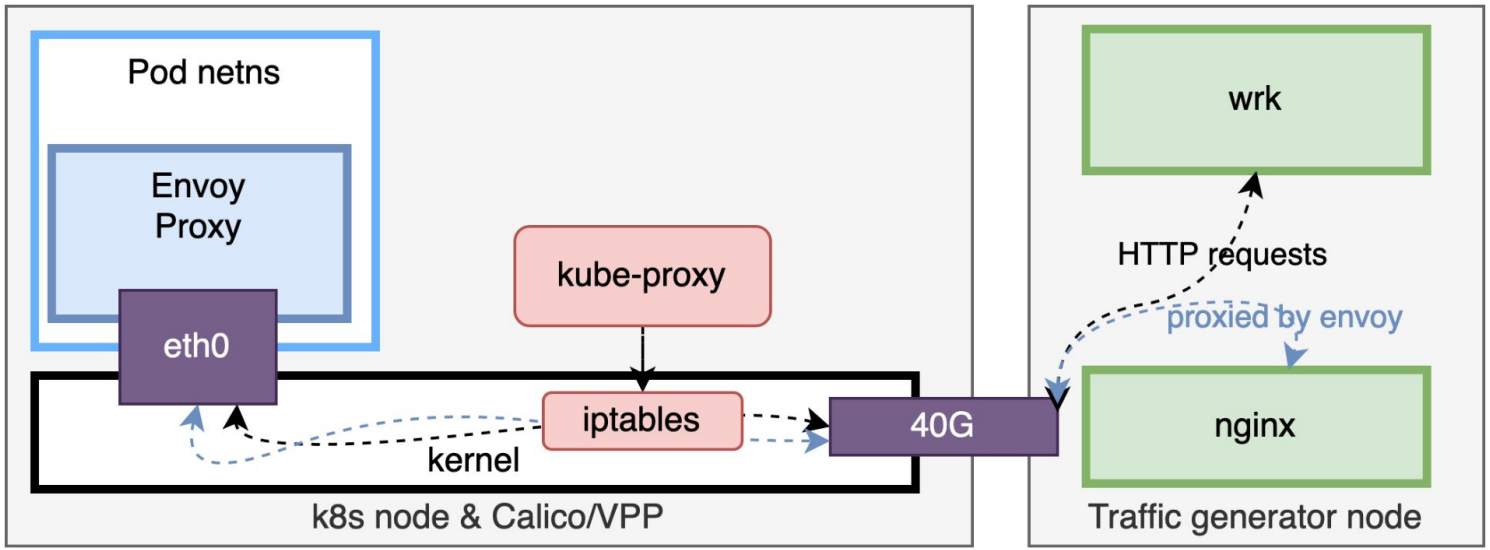
- With bigger packets, the link quickly becomes the bottleneck
- Linux around 300Mbps

And TCP ?

An envoy+VCL story

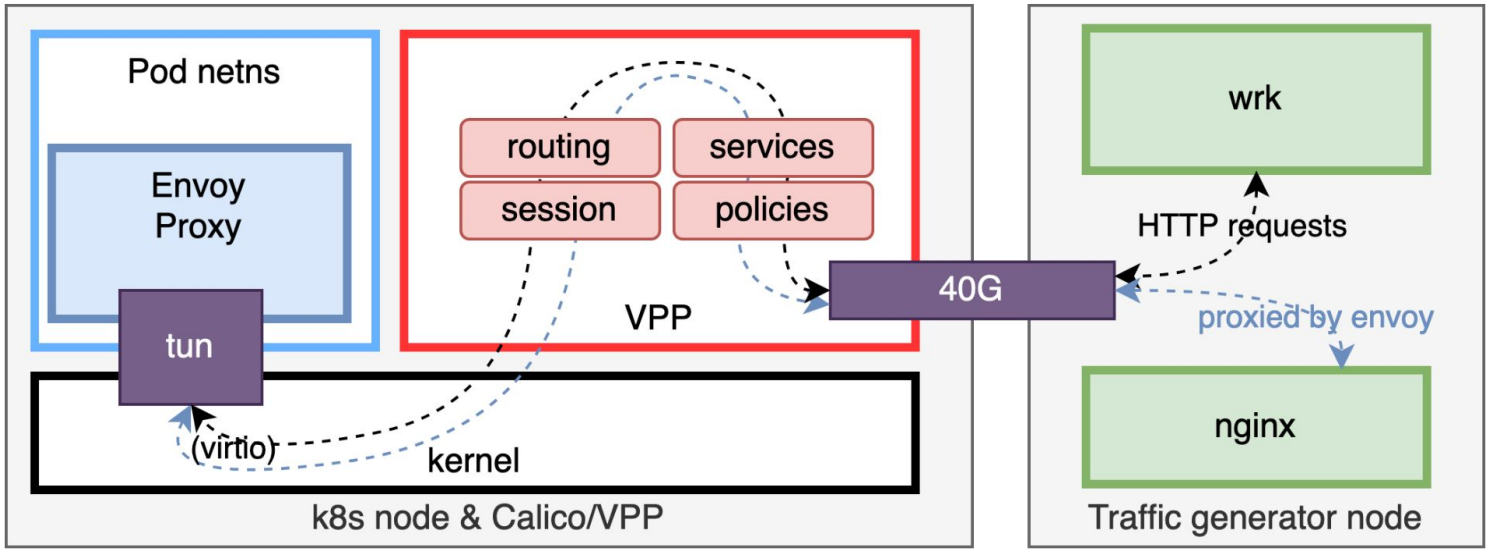
Endpoint - Envoy & Calico/linux

Benchmarking the performance of envoy running in a k8s Cluster
- wrk (traffic generator) to nginx
- 64B requests, measuring RPS



Endpoint - Envoy & Calico/VPP

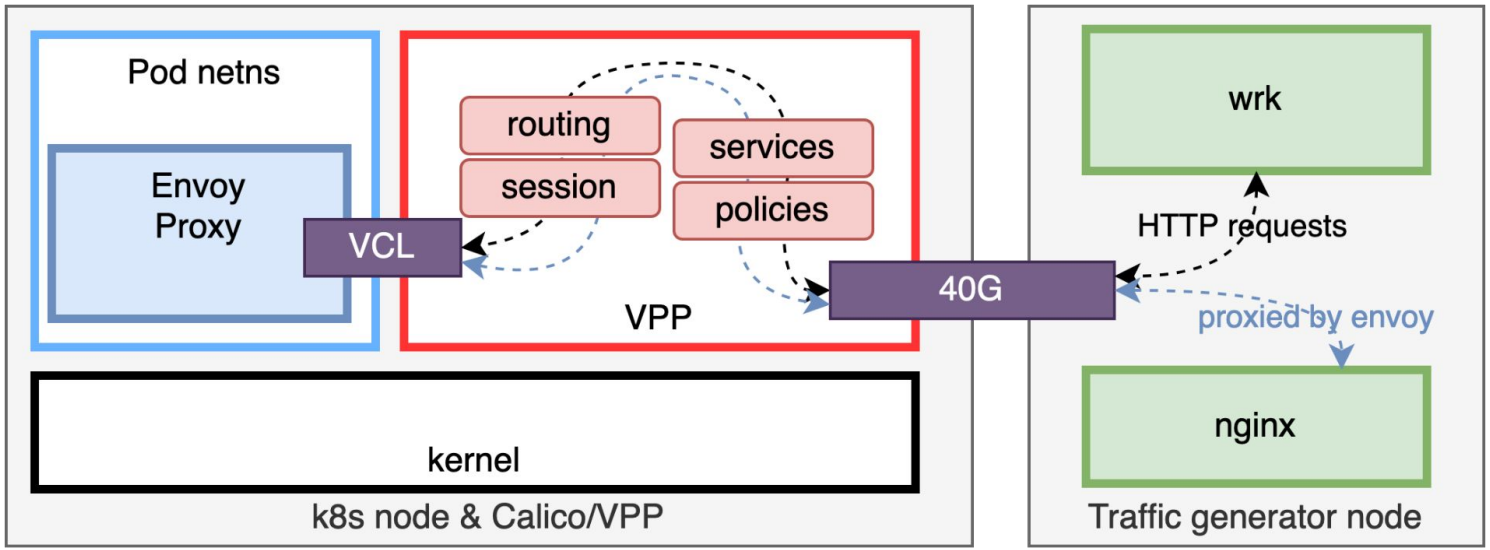
We can use VPP as dataplane, but still run envoy unmodified



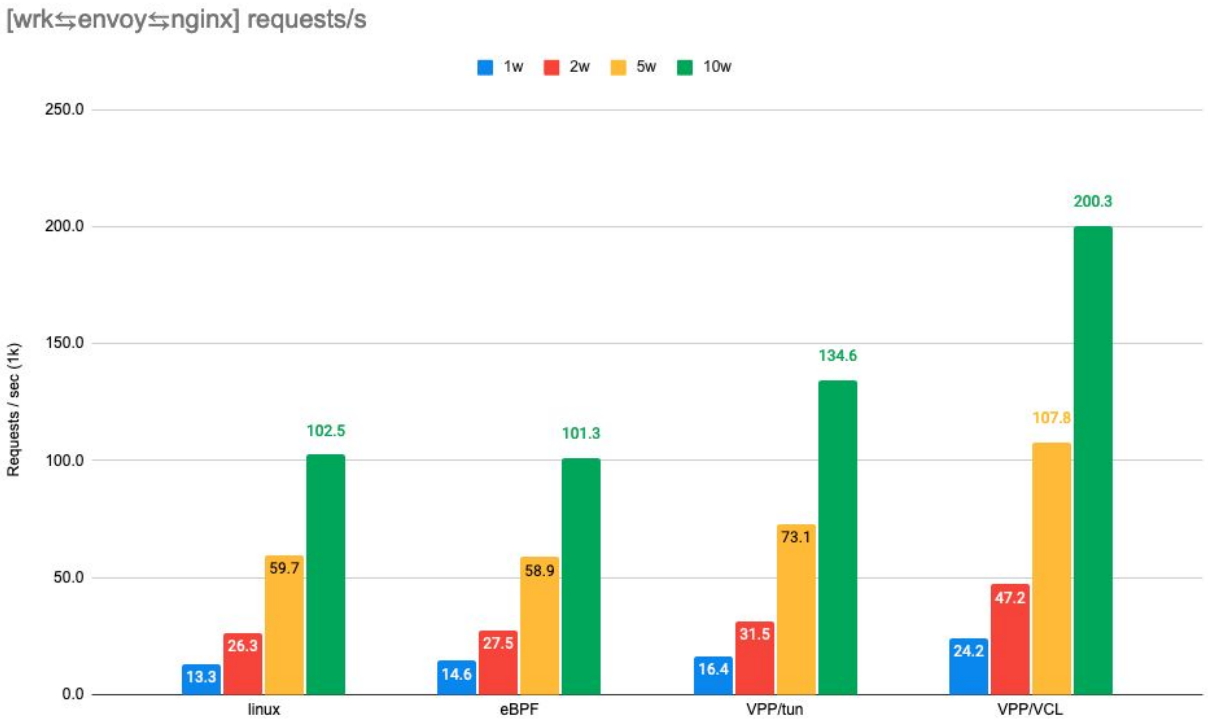
Endpoint - Envoy & Calico/VPP

Running Envoy with VCL support

Images `envoyproxy/envoy-contrib:v1.21.0`



Run kitty, run !

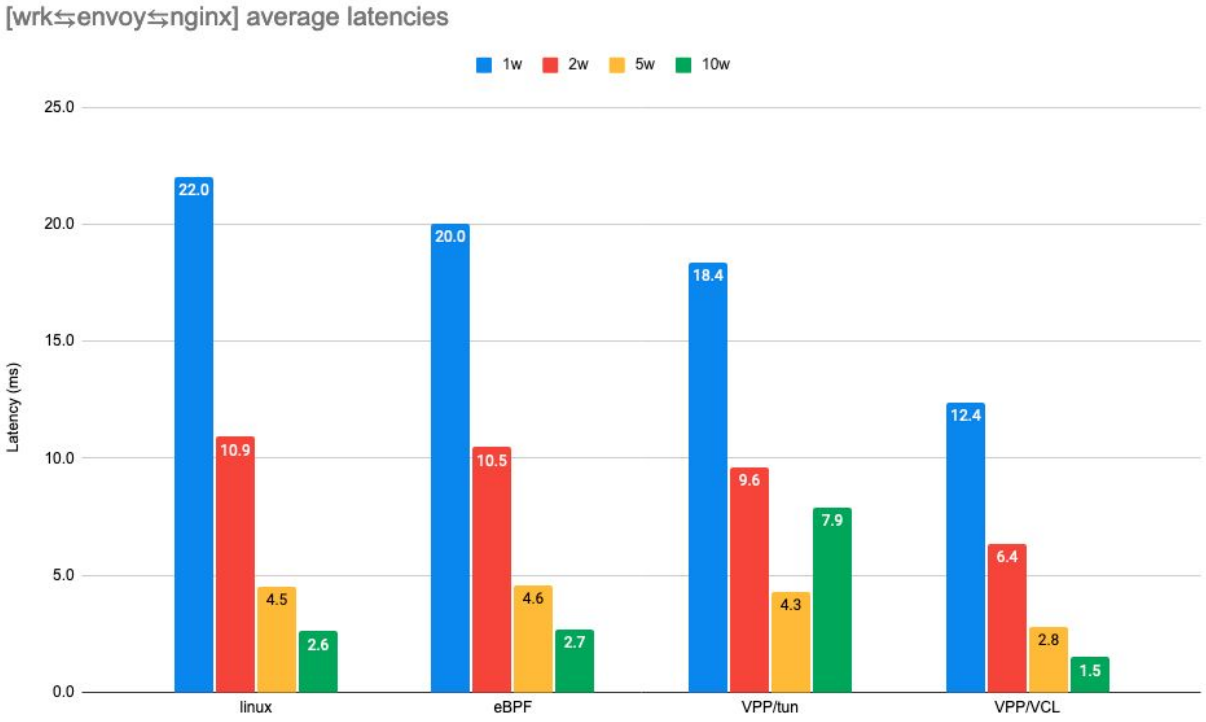


- Measuring requests per second
- TCP requests with wrk,
  30 threads, 300 connections
- 4 Nginx workers, 64B payload
- VPP 1 worker / 1 main
- Scaling number per Envoy
  `–concurrency=1..10`

Comparing RPS performance is tricky

Linux network processing is done on the same worker as envoy uses, VPP uses an extra worker for the dataplane work.

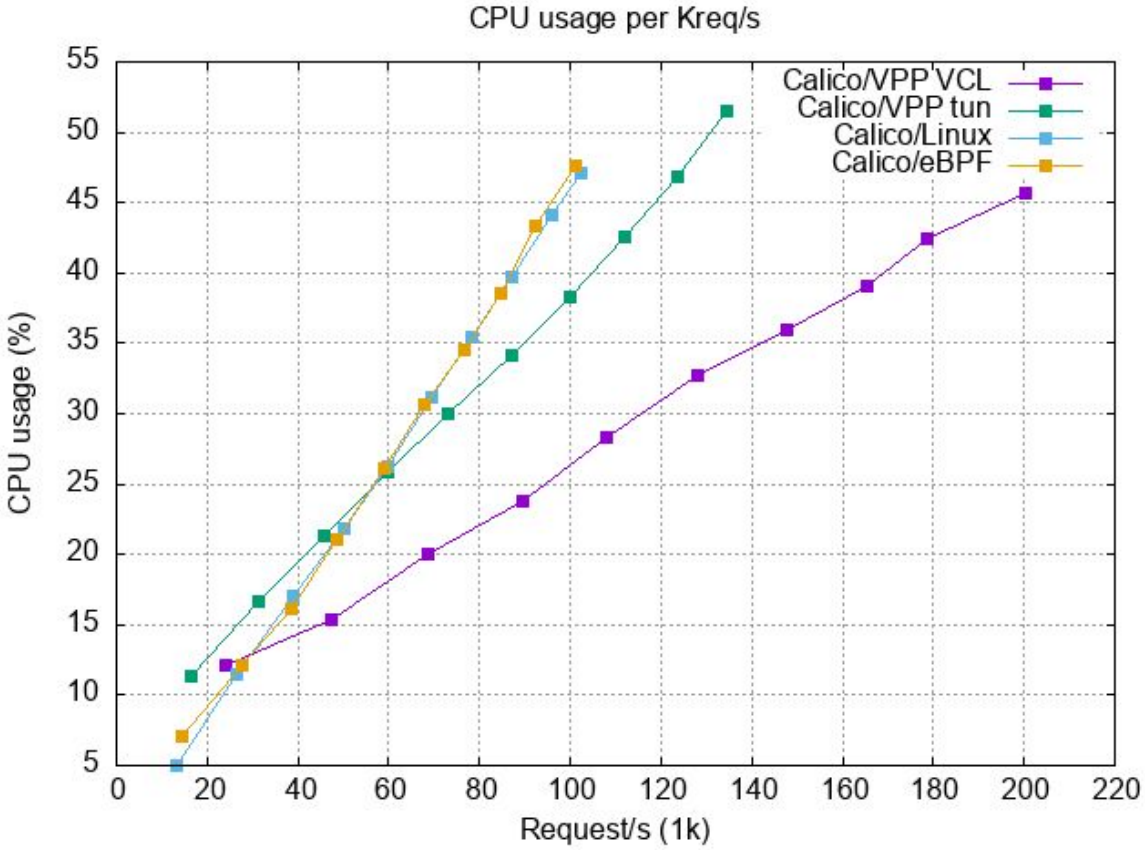
Run kitty, run !



- Measuring requests per second
- TCP requests with wrk, 30 threads, 300 connections
- 4 Nginx workers, 64B payload
- Scaling number per Envoy `–concurrency=1..10`

Latencies improves with more workers.

Run kitty, run !



- Measuring requests per second
- TCP requests with wrk, 30 threads, 300 connections
- 4 Nginx workers, 64B payload
- Scaling number per Envoy `–concurrency=1..10`

VPP/VCL with 5 envoy workers, and one VPP reaches 100k RPS, same as envoy/linux with 10 workers.

Wrapping up

- New VPP-based userspace dataplane option for Calico
- memif support offers a code path which can handle the incredible performance levels we all expect from modern apps.
- Complements Calico’s workload protection with incredible WireGuard performance to protect data-in-flight in edge environments
- Additional advanced experimental feature support such as VCL and QUIC allowing you to stay ahead of the curve

Wrapping up

- Beta status expected in Calico v3.22 - currently anticipated late January, so it could well be live as you see this!
- Contributions welcome!
  @ https://github.com/projectcalico/vpp-dataplane
- Join us on the Calico Users Slack #VPP channel
  @ https://calicousers.slack.com/archives/C017220EXU1