Calico/VPP: All You Can Eat Networking

Bringing Kubernetes Goodness to your Hungriest Workloads

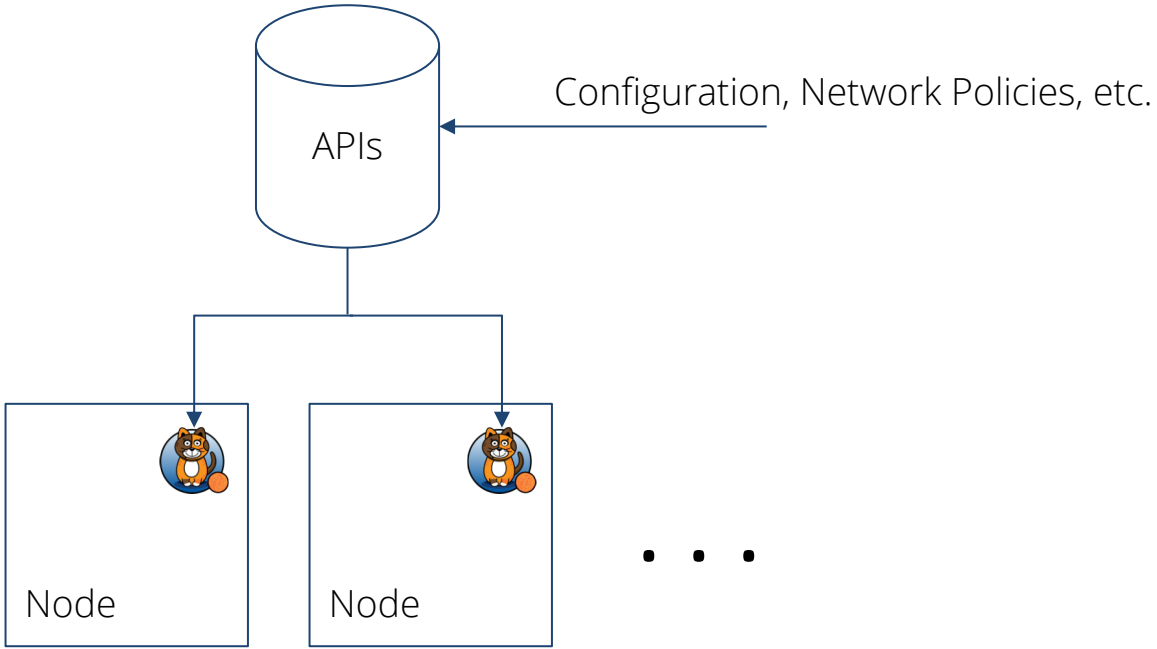
Aloÿs Augustin, Casey Davenport

FOSDEM 2021

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

What is Calico?



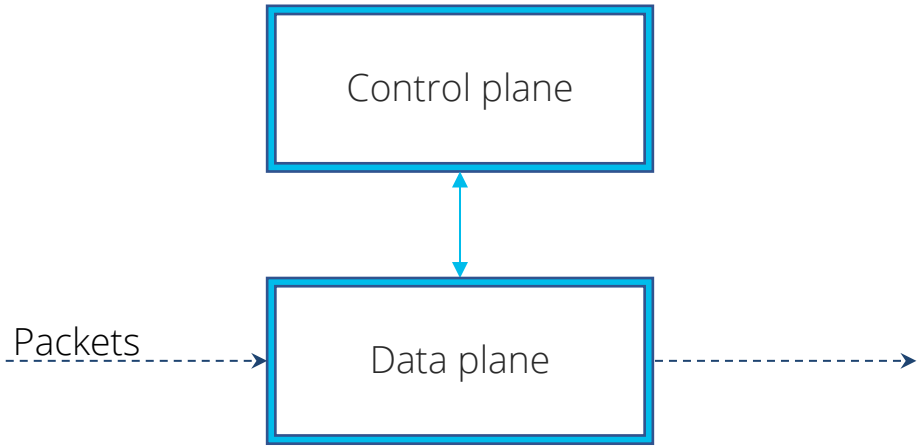
Under the hood

- CNI plugin / IPAM plugin:
  - Called by the container runtime on pod ADD / DEL on a per-pod basis
  - Configures pod network namespace with routes, devices, etc.
- calico/node:
  - Runs on every node as a DaemonSet
  - Makes routing and policy decisions, make sure they are enforced
  - two main subcomponents: **felix** and **BIRD**

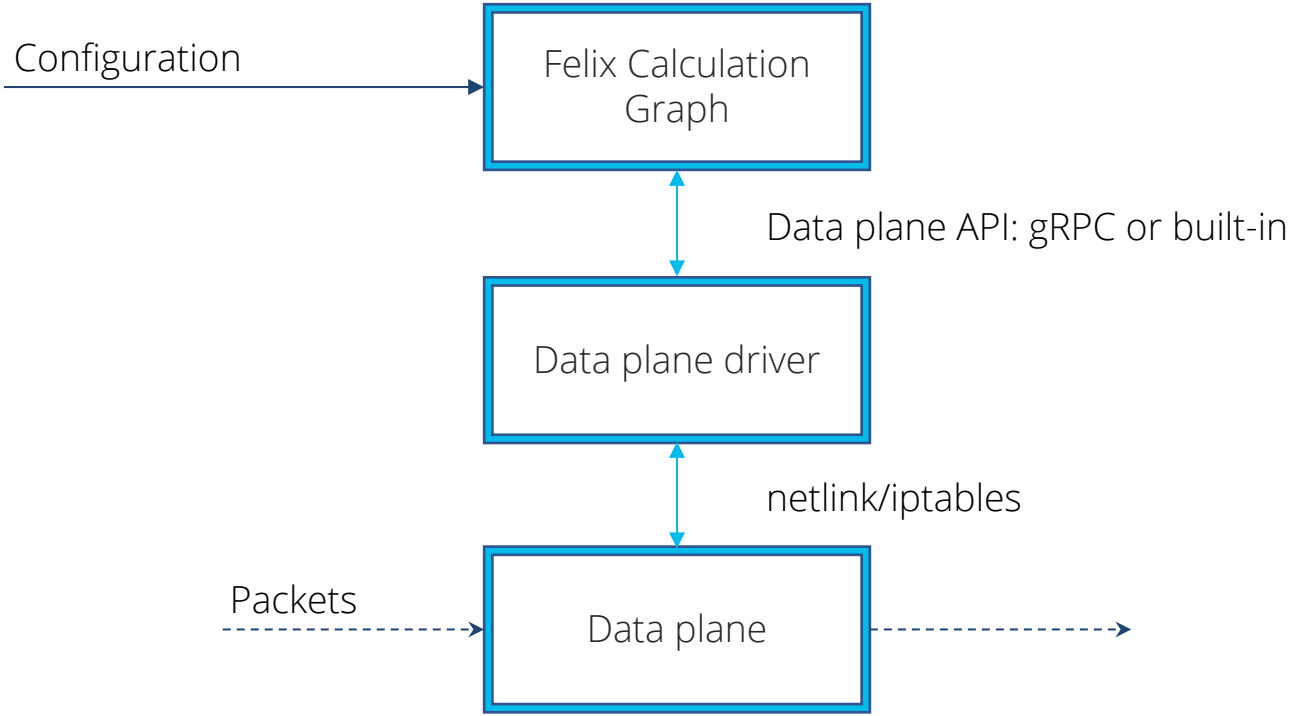
Calico design philosophy

Use the right tool for the job

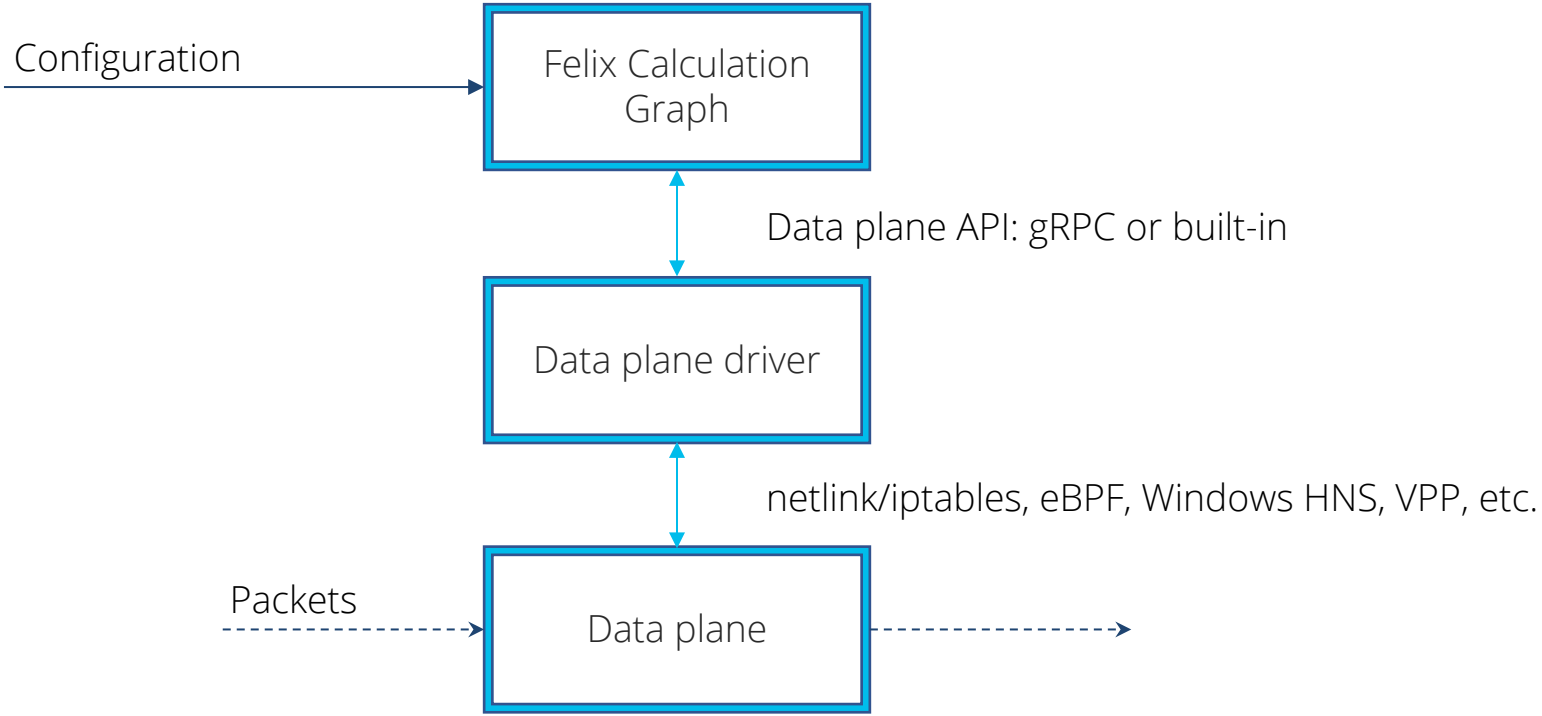
Calico design philosophy



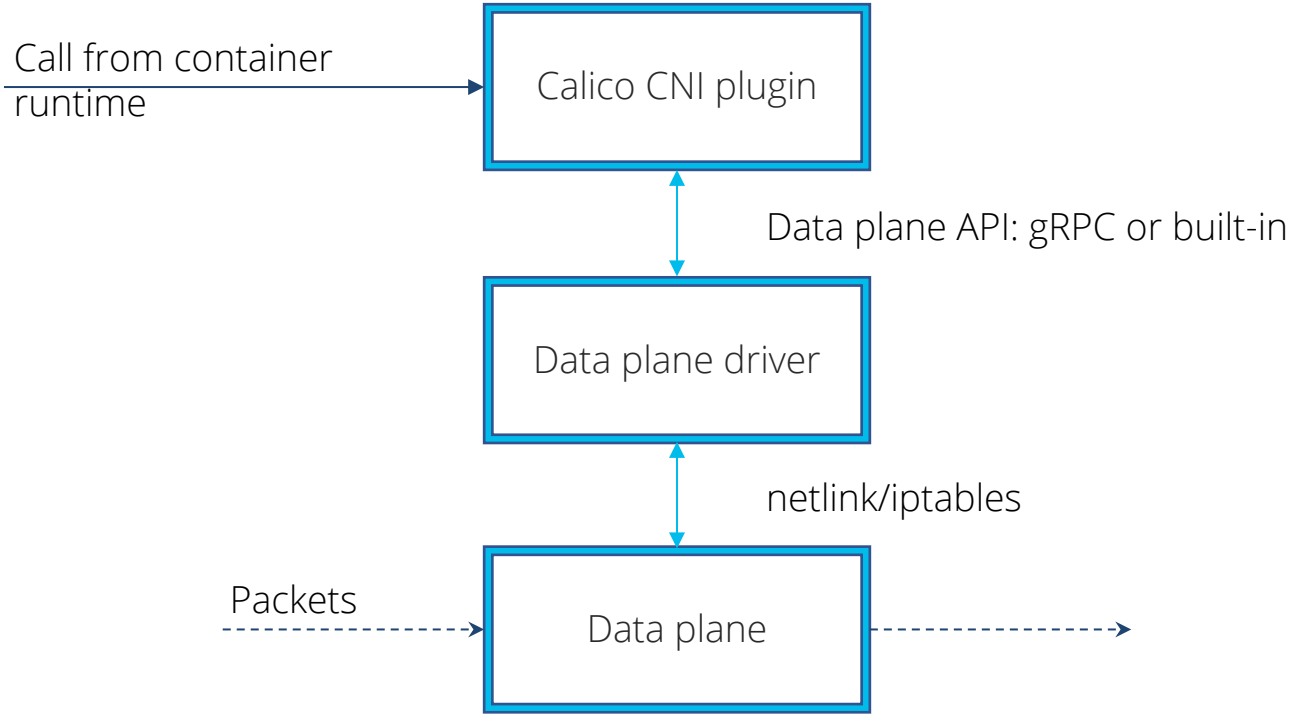
Calico design philosophy



Calico design philosophy



Calico design philosophy



Active community

- 200+ contributors on GitHub
- Regular quarterly releases
- Active slack community of users and developers

Calico/VPP integration

What is VPP?

- Fast, open-source userspace networking dataplane - https://fd.io/
- Feature-rich L2-3-4 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
  - Transparent for users except for basic initial interface configuration
- Custom VPP plugins for K8s networking:
  - Optimized NAT plugin for service load balancing
  - Specific plugin for efficient Calico policies enforcement
- VPP configuration optimized for container environments:
  - Interrupt mode, SCHED_RR scheduling
  - No hugepages required
  - GRO / GSO support for container interfaces

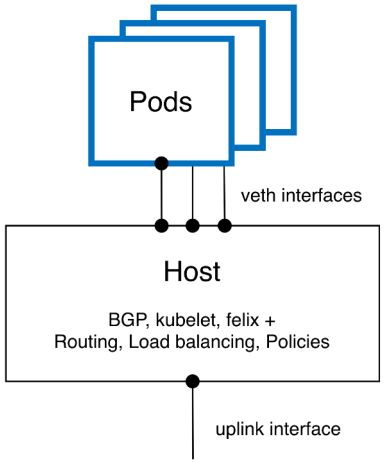
Benefits

- Performance
  - World-class encryption performance: IPsec / Wireguard
  - Reduced overall CPU consumption
- Operational simplicity
  - Network stack decoupled from OS - easier to upgrade
  - VPP is packaged as a regular container
  - Very limited kernel dependencies
- Better control over resources dedicated to container networking
- Extensibility through VPP plugins

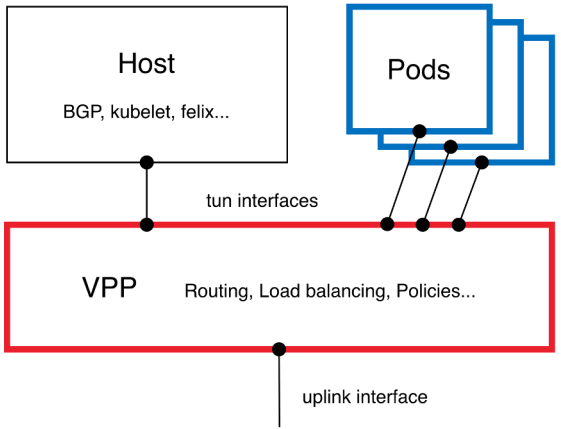
Logical network topology

- VPP inserts itself between the host and the network
- Pure layer 3 network model (no ARP/mac address in the pods)

Regular Calico

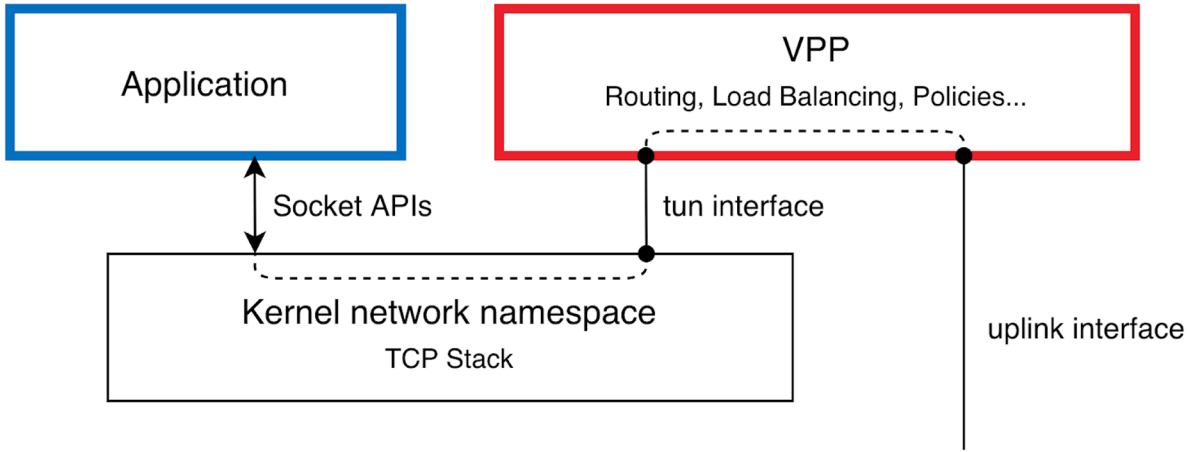


Calico/VPP

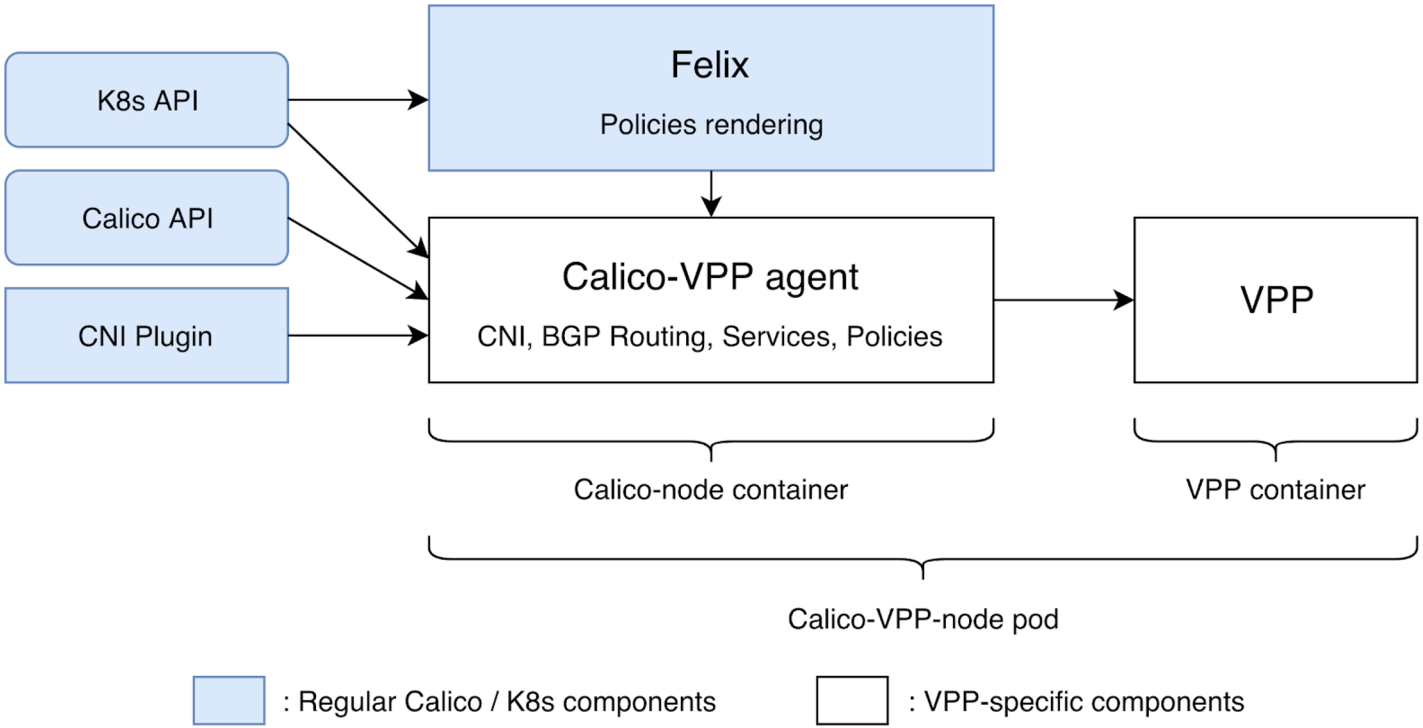


Packet flow

- One tun interface per pod
- No changes required to the applications
- Kernel provides pod isolation / namespacing



Software architecture



Project status

- Open-source on Github
  - https://github.com/projectcalico/vpp-dataplane
- Alpha status
- Calico incubation project
  - Most Calico features are now supported

Demo

- Calico/VPP deployment
- VPP restart

Performance optimizations

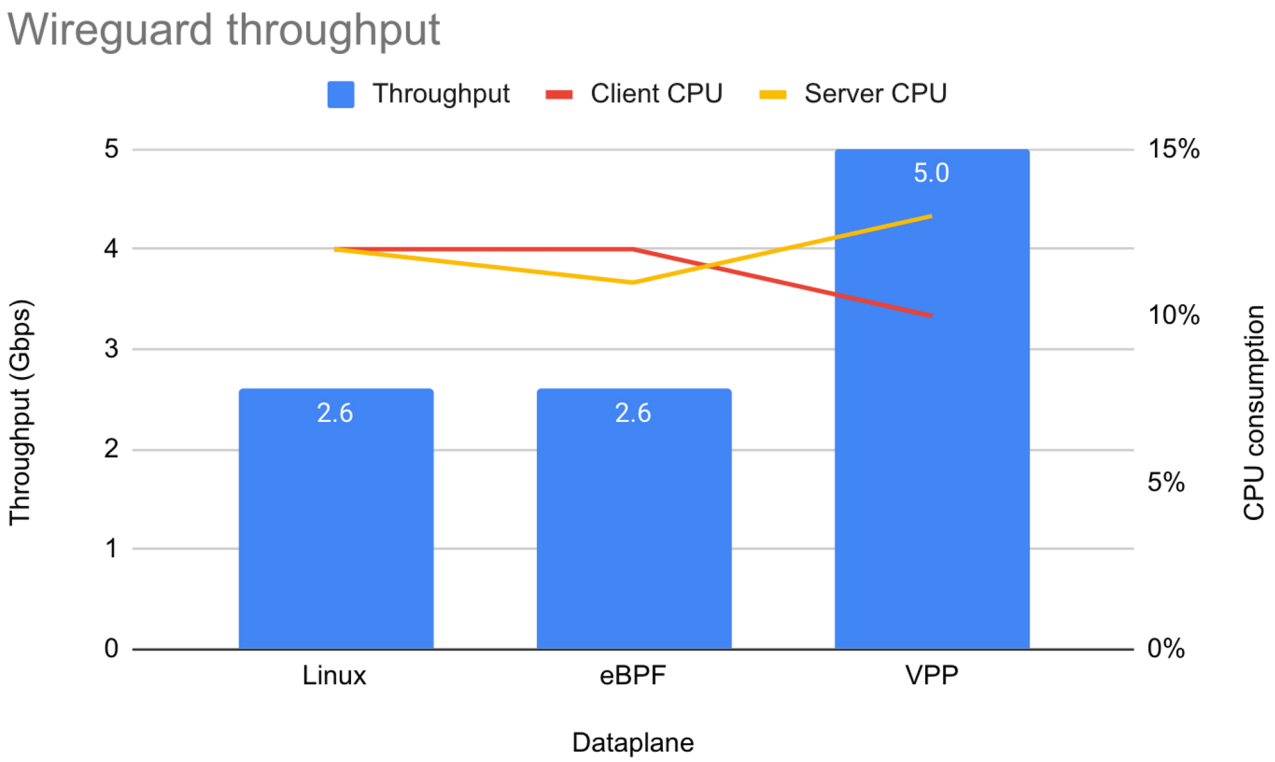
Testbed configuration

- **Hardware:** 2x Cisco C240-M5 UCS with
  - Intel Xeon Platinum 8168 CPU (24c, 48t @ 2.7GHz)
  - 384GB 2666MHz DDR4
  - Intel XL710 40G NIC - configured with 1500 bytes MTU
- Software
  - Ubuntu 18.04, kernel 5.4.0-51
  - Kubernetes 1.18, Calico 3.17.1
  - nginx, iperf from Ubuntu packages
  - wrk master from https://github.com/wg/wrk
- Methodology
  - Results averaged over 3 runs

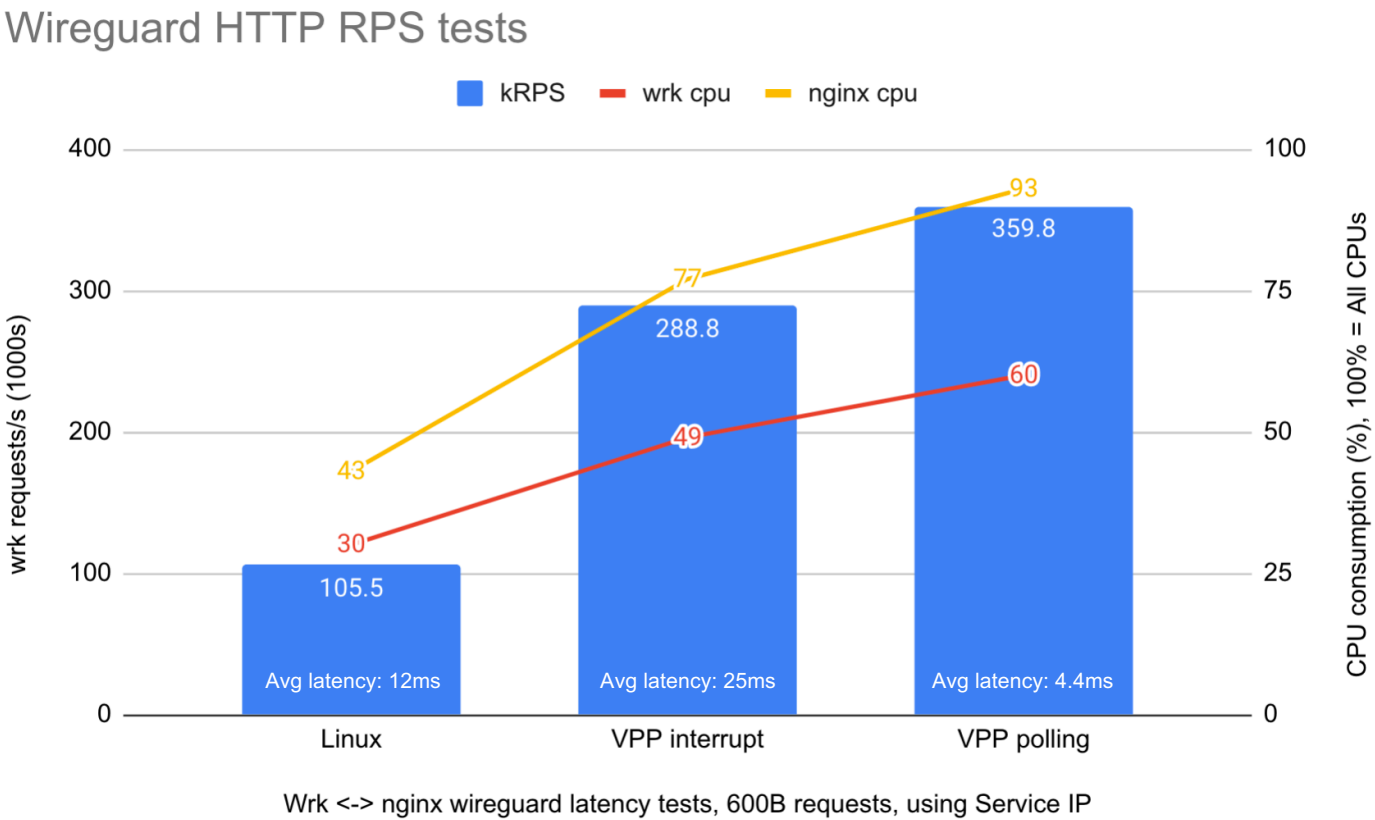
VPP Wireguard implementation

- Wireguard implementation in VPP contributed by Artem Glazychev (Xored)
- Used in Calico/VPP to provide encryption compatible with other dataplanes
- Benchmark:
  - Wireguard encryption between nodes
  - Comparison of 3 dataplanes: Linux, eBPF, VPP
  - 1) Single flow iperf tests between two pods
  - 2) wrk test, nginx server, 600B requests

Wireguard benchmarks



Wireguard benchmarks

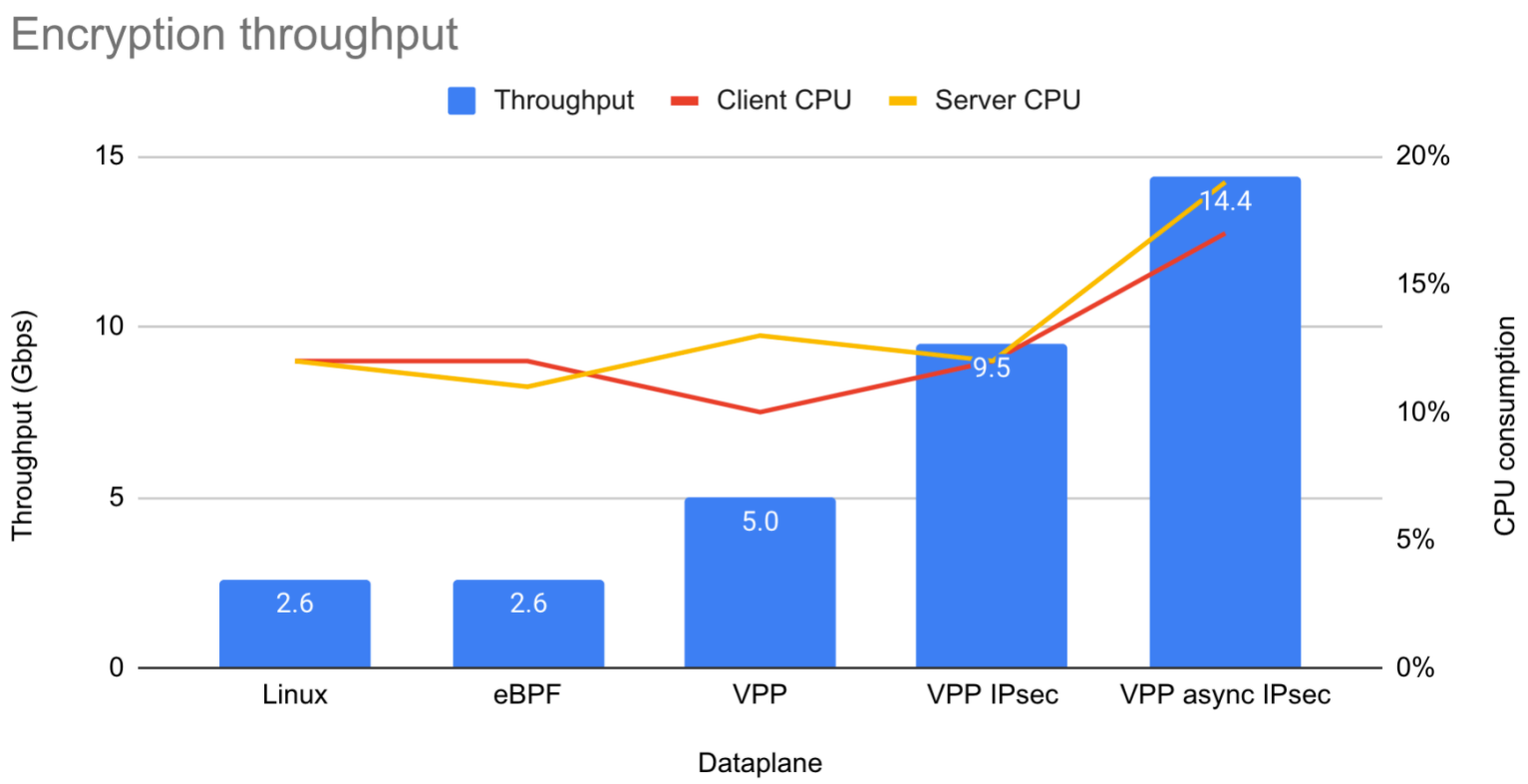


Asynchronous IPsec encryption

- VPP IPsec improvements contributed by Intel
  - Crypto operations are processed asynchronously by workers independently of packet I/O
  - More details in Fan Zhang talk later

- Benchmark:
  - One worker dedicated to I/O, one dedicated to crypto
  - Iperf single flow throughput measurement
  - Current bottleneck is I/O worker

Asynchronous IPsec benchmarks

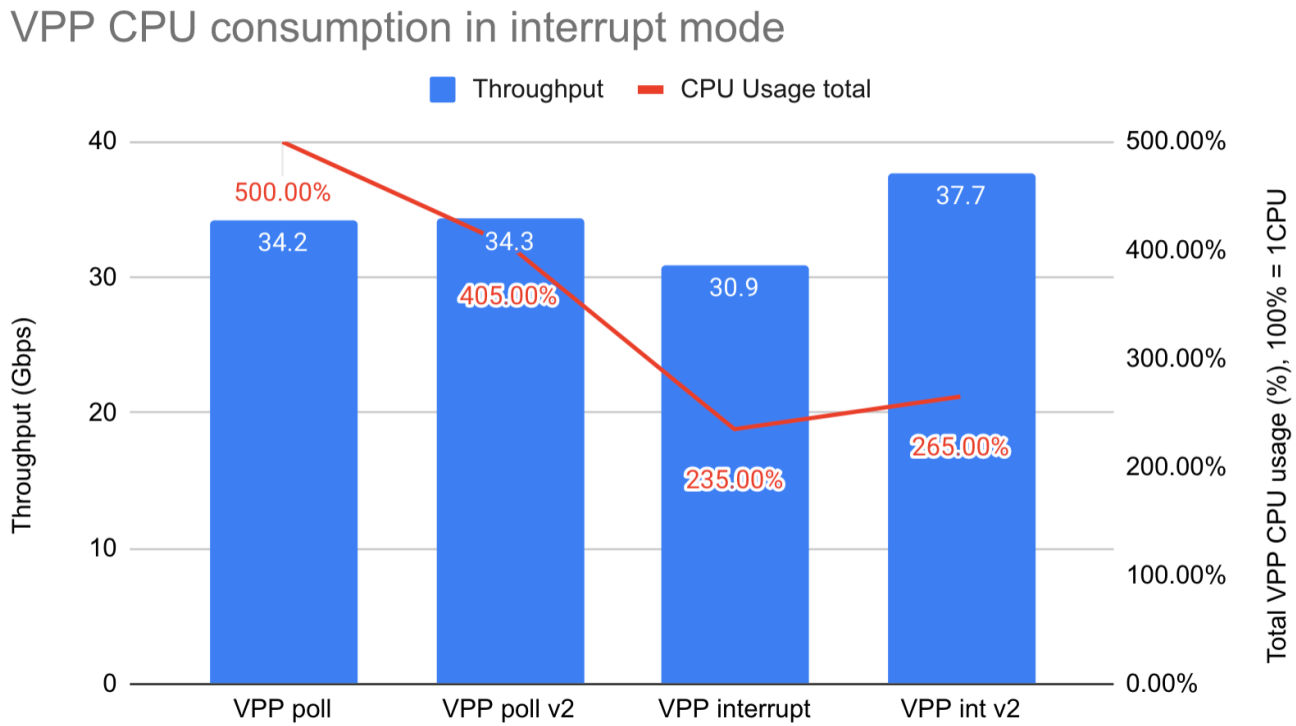


Improved interrupt handling

- Before:
  - Interrupts delivered to main thread
  - Workers check triggered interrupts every 100us
  - sleep if there is nothing to do
- Now:
  - Interrupts delivered directly to the workers
- Expectations:
  - Reduced latency
  - Reduced CPU consumption

Interrupt mode benchmarks

Setup: iperf client ← tun → vpp ← avf – avf → vpp ← tun → iperf server



Acknowledgements

Many thanks to:

- The FD.io VPP community for their continued support and contributions
- The Calico team for their great support and feedback

Wrapping up

- Upcoming features:
  - Maglev load balancing
  - Operator integration
  - Calico GA :)

- Join the Calico/VPP slack to stay up to date!
  - https://calicousers.slack.com/archives/C017220EXU1
- Check out our docs if you'd like to learn more or try it out:
  - https://github.com/projectcalico/vpp-dataplane/wiki

References

- Calico: https://www.projectcalico.org/
- FD.io/VPP: https://fd.io/
  - Continuous performance testing: https://docs.fd.io/csit/master/trending/introduction/dashboard.html
- Calico dataplane driver for VPP:
  - Code: https://github.com/projectcalico/vpp-dataplane
  - Doc: https://github.com/projectcalico/vpp-dataplane/wiki
  - Slack channel: https://calicousers.slack.com/archives/C017220EXU1
- 40Gbps pod-to-pod IPSec for Calico with VPP:
  - https://medium.com/fd-io-vpp/getting-to-40g-encrypted-container-networking-with-calico-vpp-on-commodity-hardware-d7144e52659a