QEMU in the Fast Lane: Accelerating KubeVirt Networking with eBPF

Daniel Borkmann, Anton Protopopov, Jussi Mäki

KubeCon North America 2025

ISOVALENT
now part of cisco

“In late 2023, a long-trusted virtualization staple became the biggest open question on the enterprise IT roadmap.” - MIT Technology Review

COMPUTING

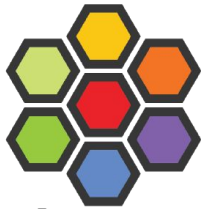
Turning migration into modernization

The VMware shake-up has led to an IT inflection point. Leaders are now weighing whether to renew, migrate, or redesign entirely for the cloud era.

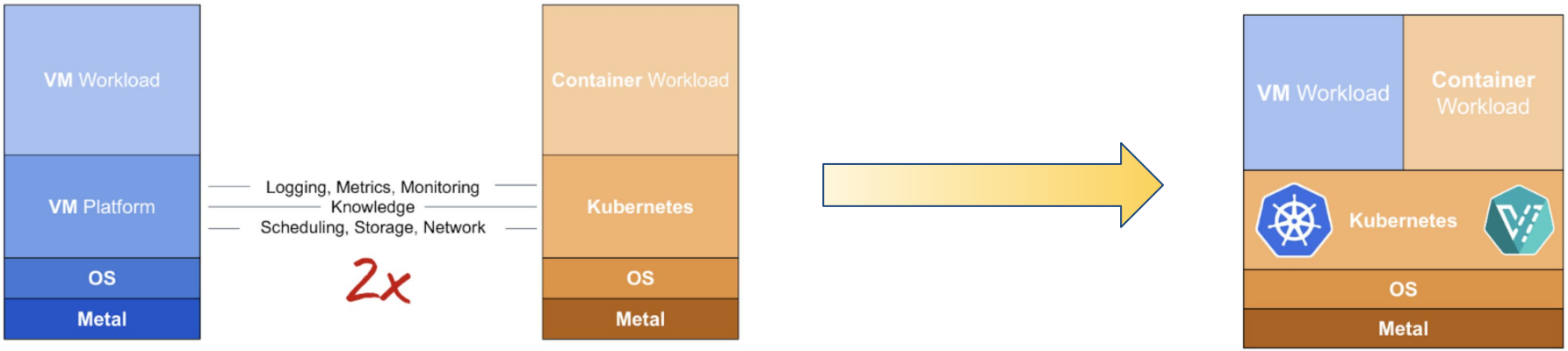
By MIT Technology Review Insights

October 2, 2025

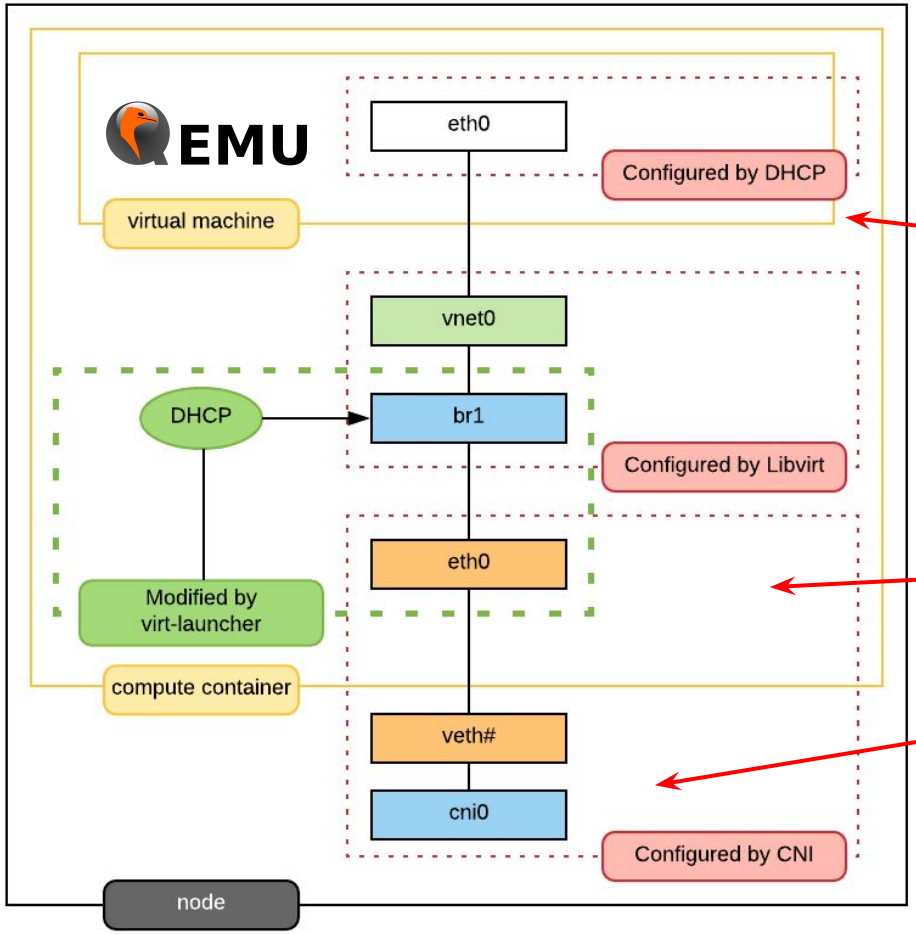
Are there viable OSS solutions which help with a strategic reset to modernize VM infrastructure into Kubernetes?

Yes! The answer is:  KubeVirt + 

Kubernetes & KubeVirt enables cost savings through infrastructure convergence:



[img: kubevirt.io]



veth pair

tap

bridge

(actual VM with workload inside)

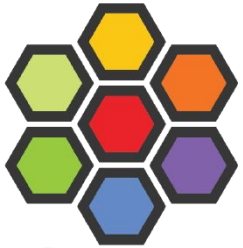
(k8s Pod with VM, own netns)

(init netns, e.g. Cilium in kube-system)

[img: kubevirt.io]

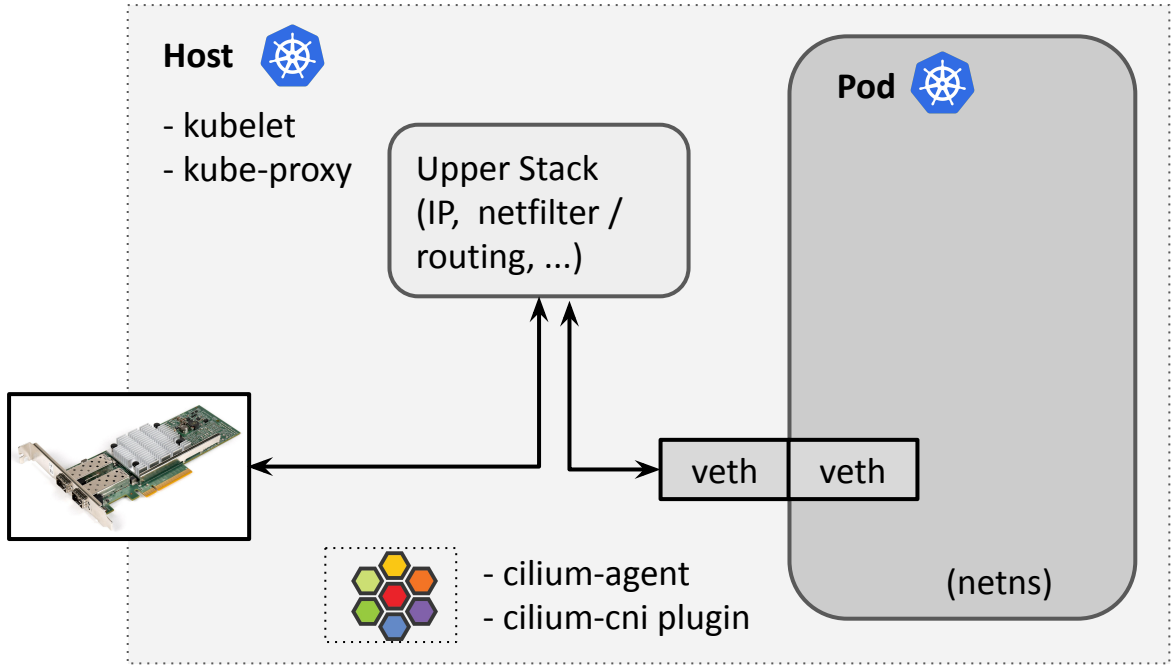
Moonshot idea for *this* talk:

Can we reimagine how we do VM networking in Kubernetes?



Lets first go back to fundamentals on how we removed the Pod (netns) overhead in Cilium ...

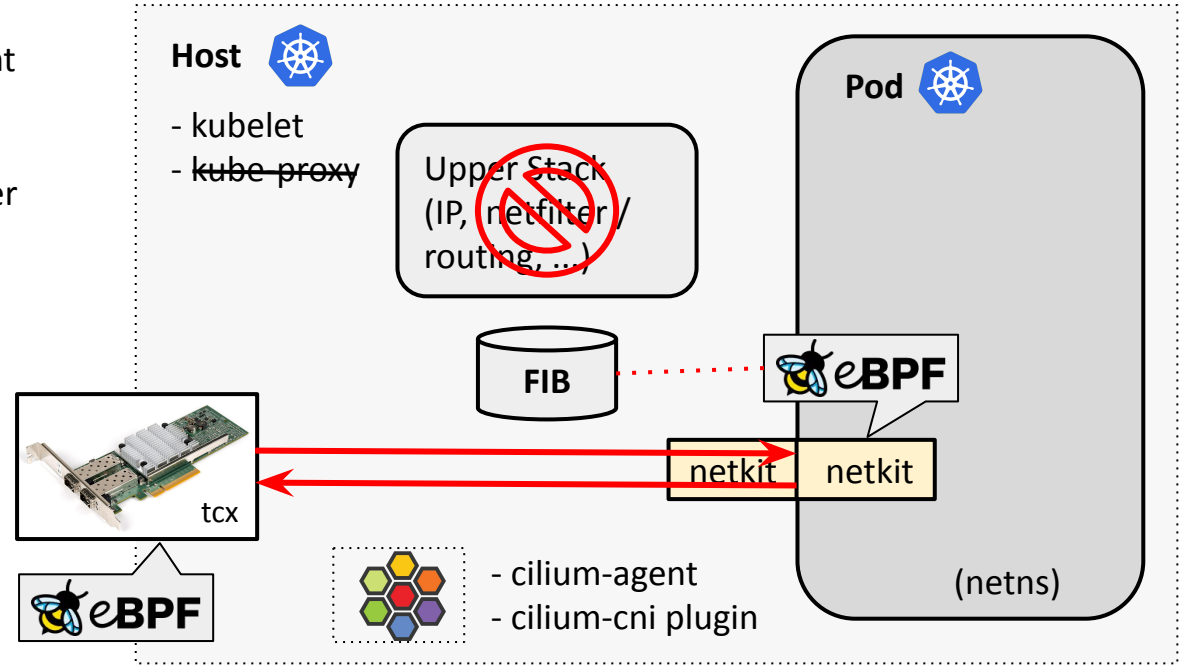
Standard Pod Datapath:



Cilium's Pod Datapath: Making Pods as fast as Host

Building Blocks:
- BPF kube-proxy replacement
- BPF Host Routing
- BIG TCP for GRO/GSO/TSO
- tcx-based BPF datapath layer
- netkit devices for Pods

Recommended:
- Cilium v1.18+
- Linux kernel v6.12+



Kernel 6.7: netkit devices (Daniel Borkmann, Nikolay Aleksandrov), **LWN coverage:** The BPF-programmable network device

Deep Dive: veth-replacement for Pods

netkit programmable virtual devices for BPF:

- Realizing multiple core ideas tailored for Cilium:
  - Better application scheduler accounting by removing heavy reliance on ksoftirqd-deferral under pressure

Deep Dive: veth-replacement for Pods

netkit programmable virtual devices for BPF:

- Realizing multiple core ideas tailored for Cilium:
  - Better application scheduler accounting by removing heavy reliance on ksoftirqd-deferral under pressure
  - No driver internal queueing for traffic in/out of the node, thus complete removal of netns overhead

Deep Dive: veth-replacement for Pods

netkit programmable virtual devices for BPF:

- Realizing multiple core ideas tailored for Cilium:
  - Better application scheduler accounting by removing heavy reliance on ksoftirqd-deferral under pressure
  - No driver internal queueing for traffic in/out of the node, thus complete removal of netns overhead
  - veth is L2, provide users choice between L3 (noarp) or L2 device option

Deep Dive: veth-replacement for Pods

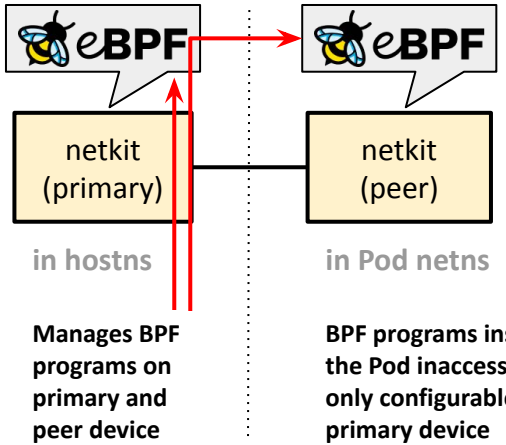
netkit programmable virtual devices for BPF:

- Realizing multiple core ideas tailored for Cilium:
  - Better application scheduler accounting by removing heavy reliance on ksoftirqd-deferral under pressure
  - No driver internal queueing for traffic in/out of the node, thus complete removal of netns overhead
  - veth is L2, provide users choice between L3 (noarp) or L2 device option
  - Device pair is partitioned into primary (hostns) and peer (Pod-netns) device for BPF management

Deep Dive: veth-replacement for Pods

netkit programmable virtual devices for BPF:

- Realizing multiple core ideas tailored for Cilium:
  - Better application scheduler accounting by removing heavy reliance on ksoftirqd-deferral under pressure
  - No driver internal queueing for traffic in/out of the node, thus complete removal of netns overhead
  - veth is L2, provide users choice between L3 (noarp) or L2 device option
  - Device pair is partitioned into primary (hostns) and peer (Pod-netns) device for BPF management

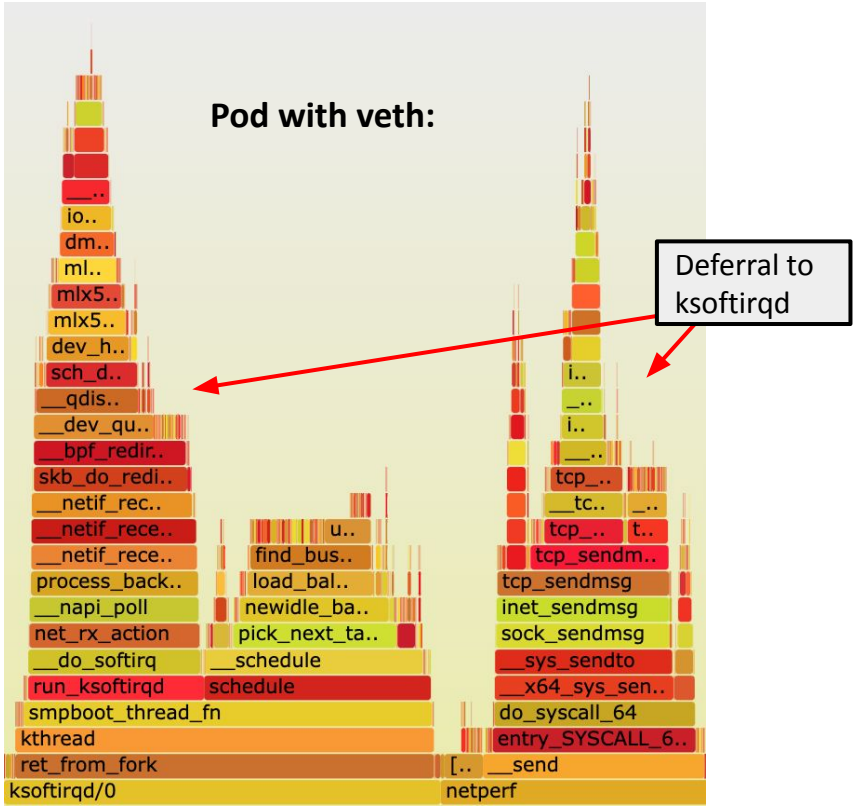


author	Daniel Borkmann <daniel@iogearbox.net> 2023-10-24 23:48:58 +0200
committer	Martin KaFai Lau <martin.lau@kernel.org> 2023-10-24 16:06:03 -0700
commit	35dfaad7188cdc043fde31709c796f5a692ba2bd (patch)
tree	53a88f1799ac38892434318a47278de4a255bc19
parent	42d31dd601fa43b9afdf069d1ba410b2306a4c76 (diff)
download	linux-35dfaad7188cdc043fde31709c796f5a692ba2bd.tar.gz

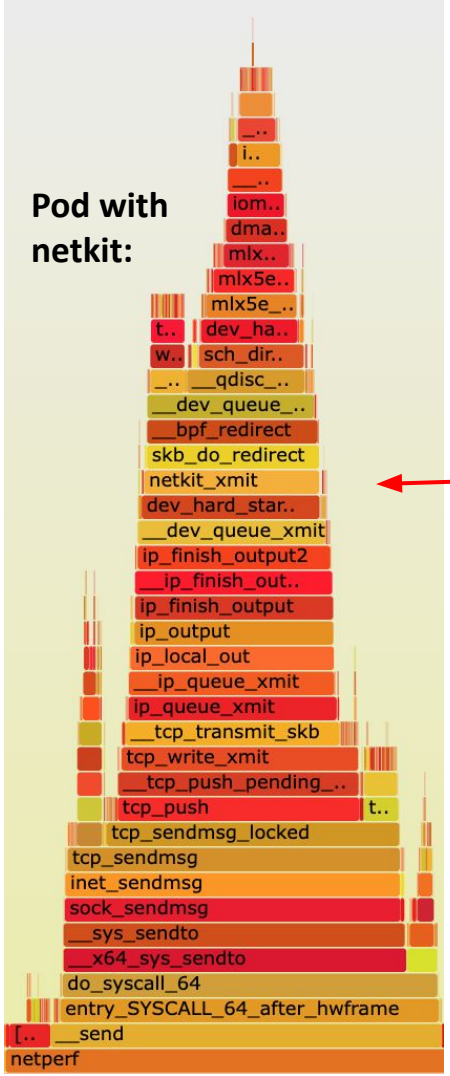
netkit, bpf: Add bpf programmable net device

```
This work adds a new, minimal BPF-programmable device called "netkit"
(former PoC code-name "meta") we recently presented at LSF/MM/BPF. The
core idea is that BPF programs are executed within the drivers xmit routine
and therefore e.g. in case of containers/Pods moving BPF processing closer
to the source.
```

veth vs netkit: Backlog Queue



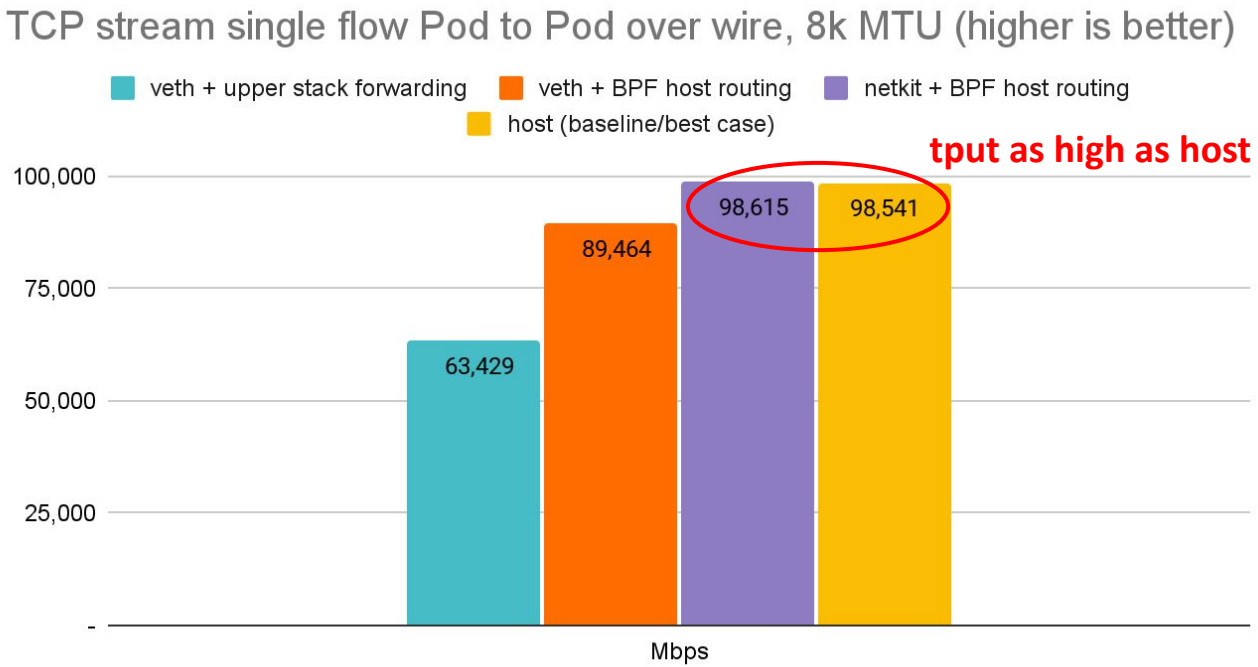
Deferral to ksoftirqd



Remains in process context all the way, leading to better process scheduler decisions.

Cilium's Pod Datapath: Making Pods as fast as Host

Building Blocks:
- BPF kube-proxy replacement
- BPF Host Routing
- BIG TCP for GRO/GSO/TSO
- tcx-based BPF datapath layer
- netkit devices for Pods

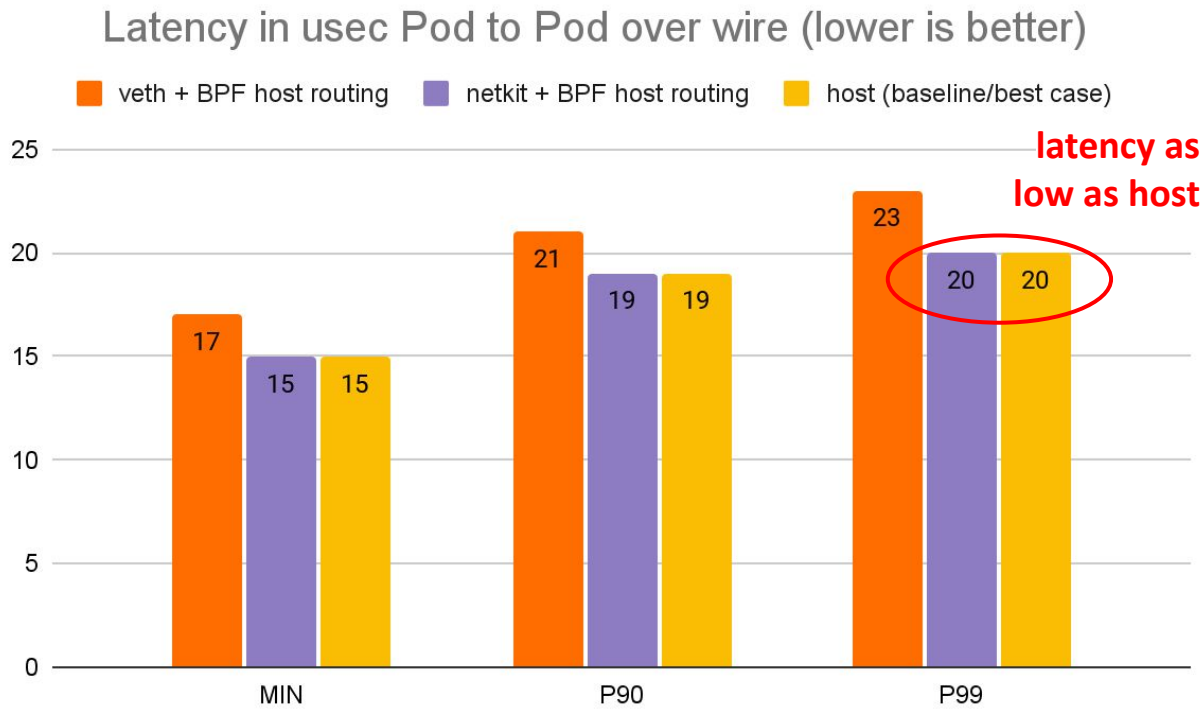


* 8264 MTU for data page alignment in GRO

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 8264 MTU
Receiver: taskset -a -c <core> tcp_mmap -s (non-zerocopy mode), Sender: taskset -a -c <core> tcp_mmap -H <dst host>

Cilium's Pod Datapath: Making Pods as fast as Host

Building Blocks:
- BPF kube-proxy replacement
- BPF Host Routing
- BIG TCP for GRO/GSO/TSO
- tcx-based BPF datapath layer
- netkit devices for Pods



* 8264 MTU for data page alignment in GRO

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off
netperf -t TCP_RR -H <remote pod> -- -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

netkit adoption outside of Cilium

ByteDance

CONTAINERS / EBPF / LINUX / NETWORKING

Netkit to Network a Million Containers for ByteDance

Built on eBPF, netkit offers a swifter alternative to Virtual Ethernet for container networking, ByteDance engineers have concluded.

Jan 29th, 2025 6:00am by Joab Jackson

netkit adoption outside of Cilium

CONTAINERS / EBPF / LINUX / NETWORKING

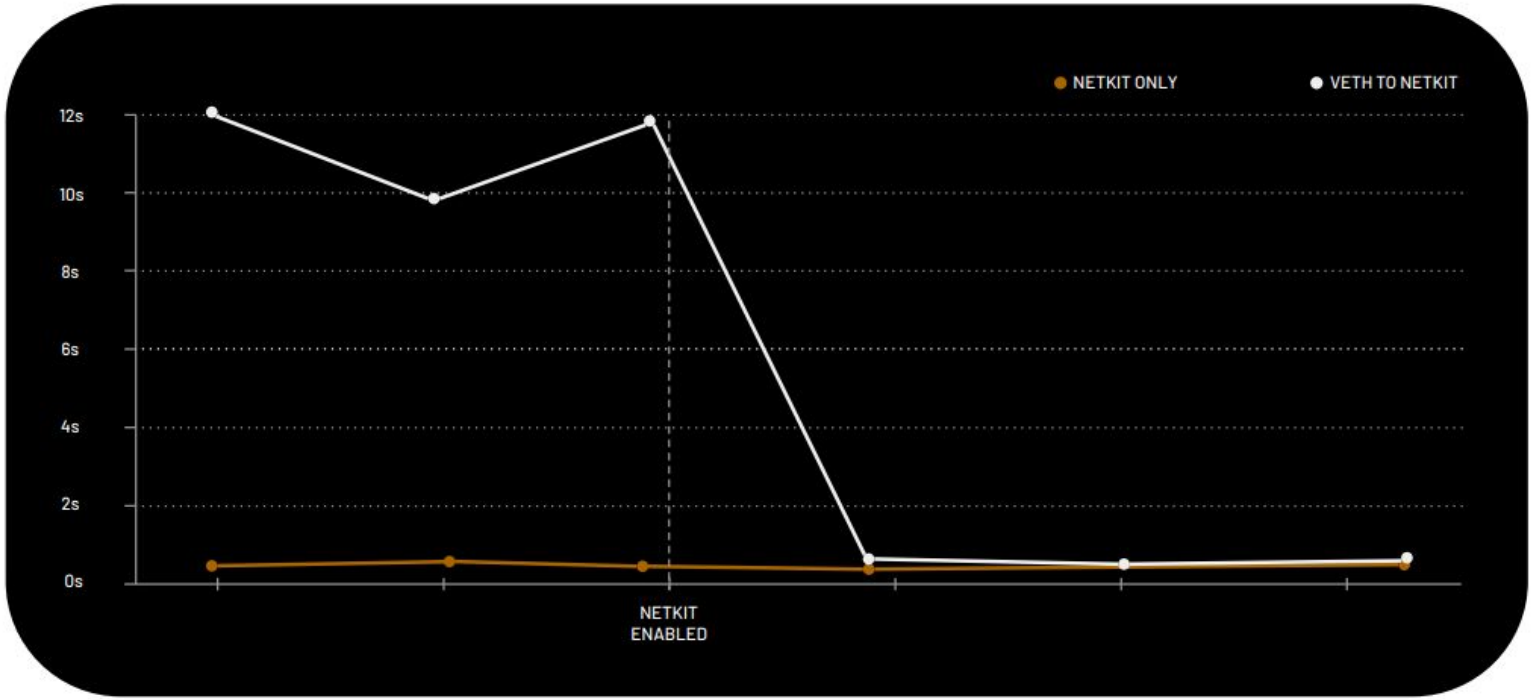
Netkit to Network a Million Containers for ByteDance

Built on eBPF, netkit offers a swifter alternative to Virtual Ethernet for container networking, ByteDance engineers have concluded.

Jan 29th, 2025 6:00am by Joab Jackson

"Good news, we have finished the Proof of Concept (PoC) and netkit provided a 12% increase in CPS (Connections Per Seconds) compared with veth (incredible!). And now we plan to use netkit in our DC as much as possible." - Chen Tang, ByteDance

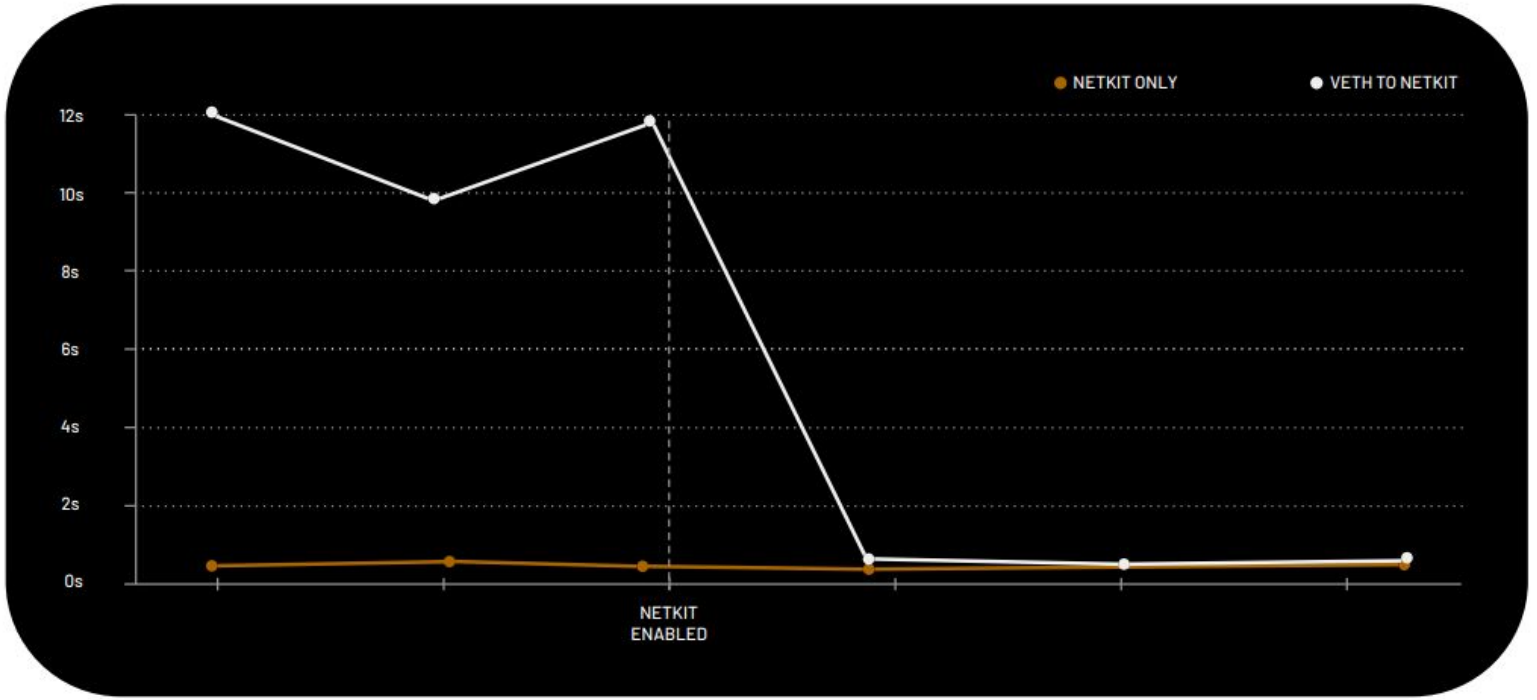
netkit adoption outside of Cilium



100GiB single stream TCP @ 8K MTU

Meta (fleet-wide rollout): P99 reduction from 12s to 0.1s

netkit adoption outside of Cilium



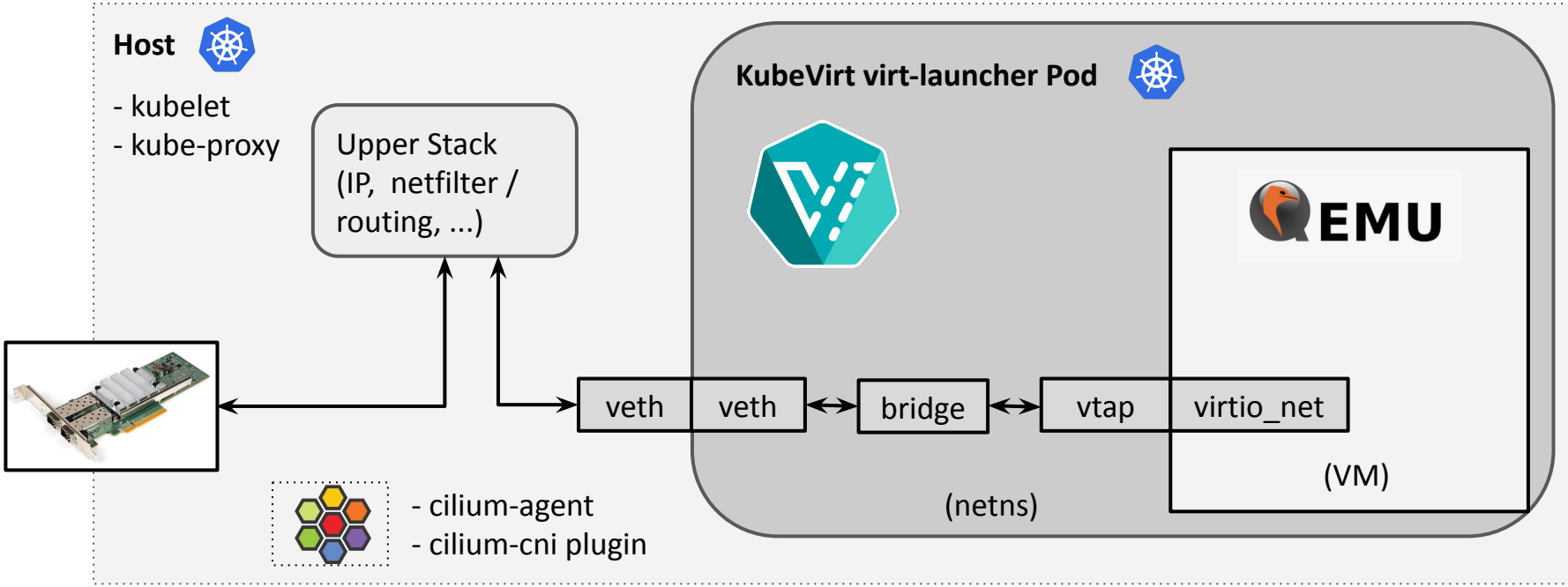
100GiB single stream TCP @ 8K MTU

Meta (fleet-wide rollout): P99 reduction from 12s to 0.1s

"With netkit, network namespaces can be used without incurring a performance penalty." - Mike Willard, Meta

Could netkit also help in terms of VM networking?

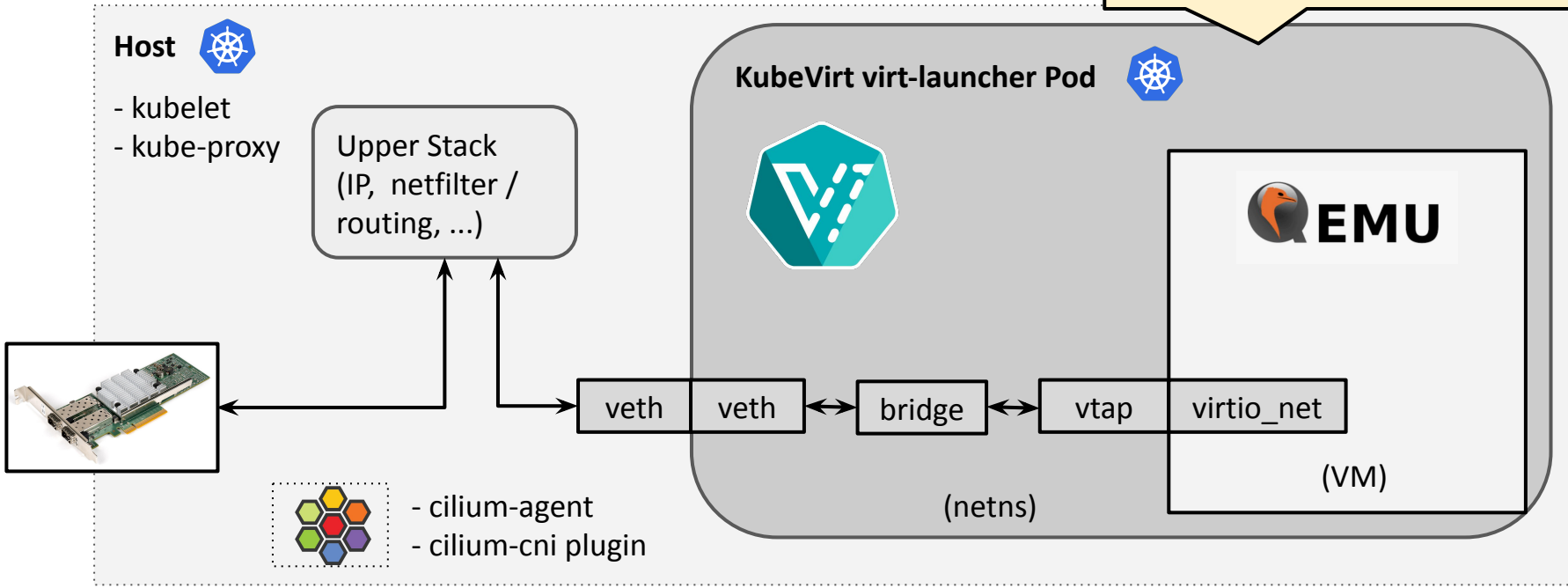
Standard KubeVirt Pod Datapath:



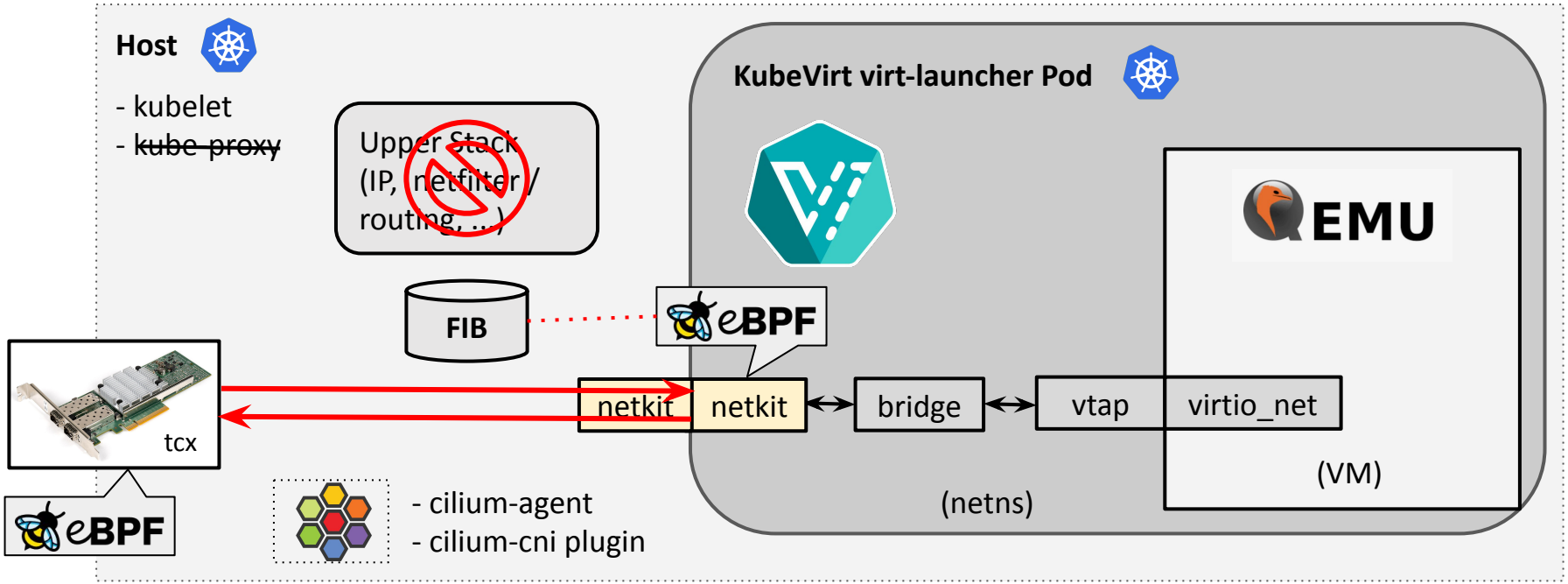
Standard KubeVirt Pod Datapath:

Fyi: virt-launcher Pod provides ...
- cgroups
- namespaces

... for hosting the VM process



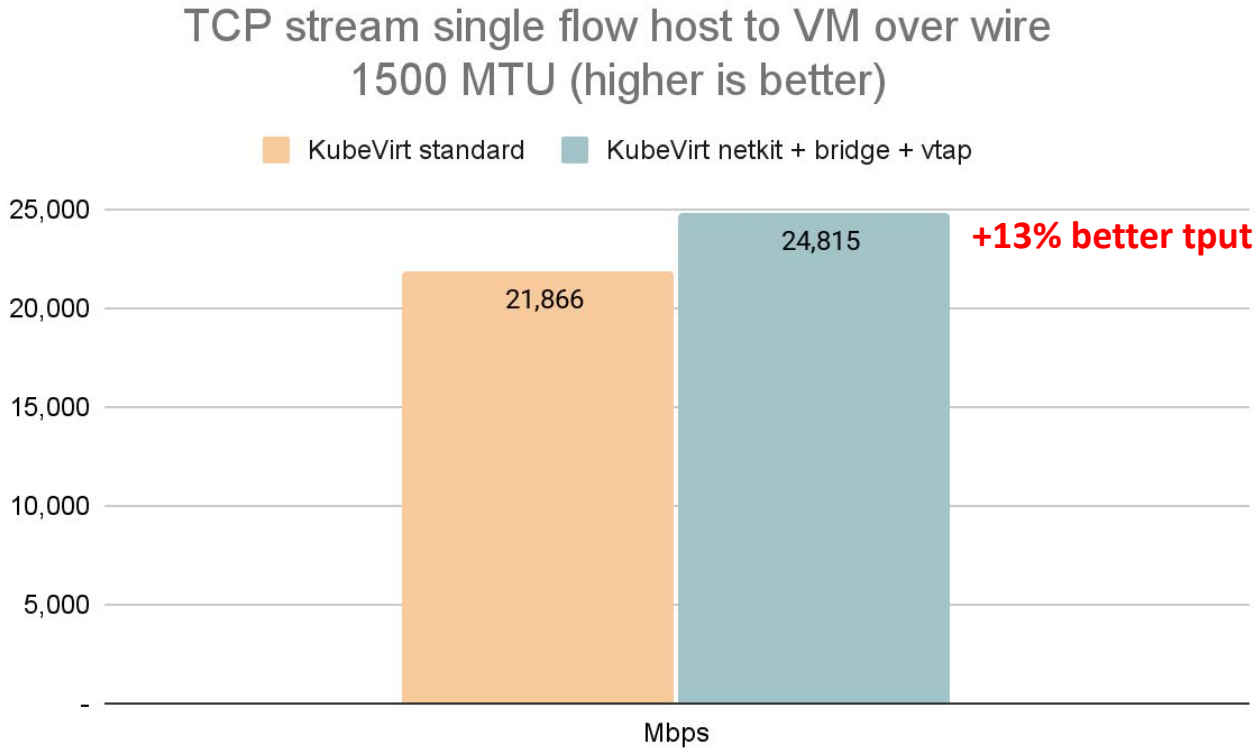
Cilium's KubeVirt Pod Datapath:



Cilium's KubeVirt Pod Datapath:

Building Blocks:
- BPF kube-proxy replacement
- BPF Host Routing
- BIG TCP for GRO/GSO/TSO
- tcx-based BPF datapath layer
- netkit devices for Pods

(... but still KubeVirt with bridge + vtap)

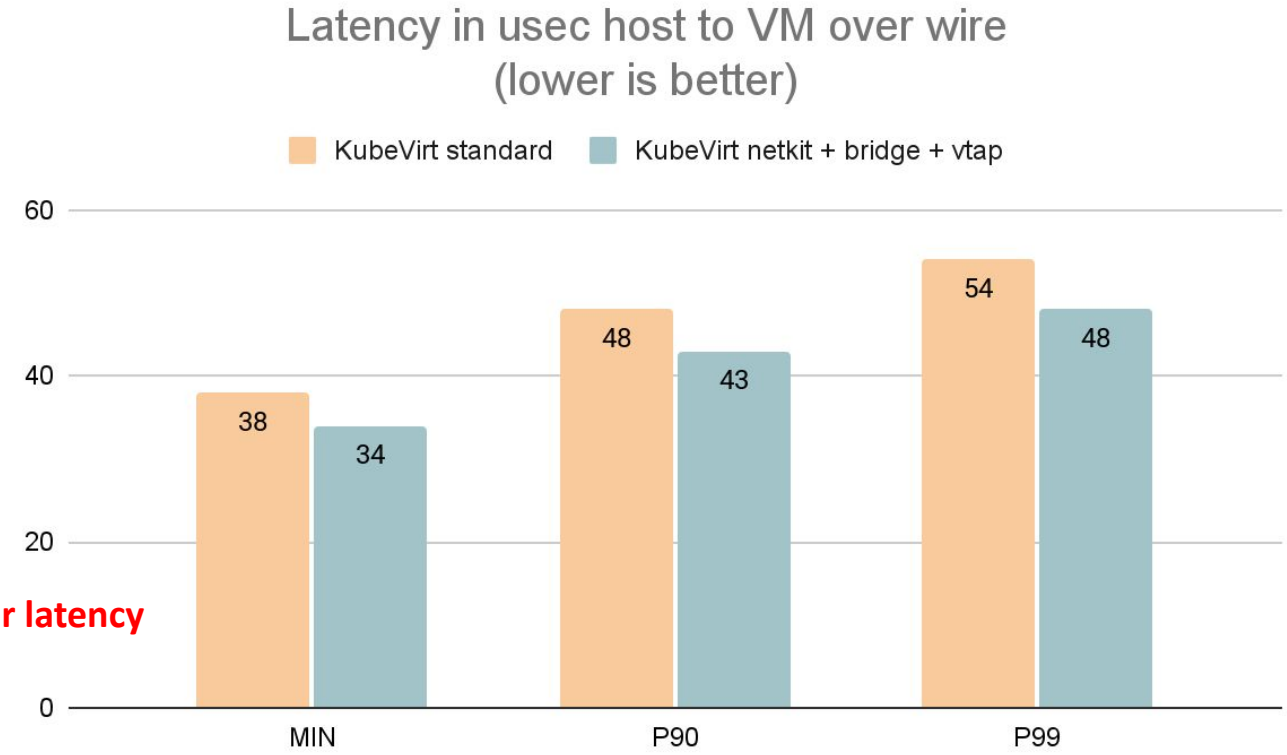


Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 1500 MTU
netperf -t TCP_RR -H <VM IP> -- -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

Cilium's KubeVirt Pod Datapath:

Building Blocks:
- BPF kube-proxy replacement
- BPF Host Routing
- BIG TCP for GRO/GSO/TSO
- tcx-based BPF datapath layer
- netkit devices for Pods

(... but still KubeVirt with bridge + vtap)



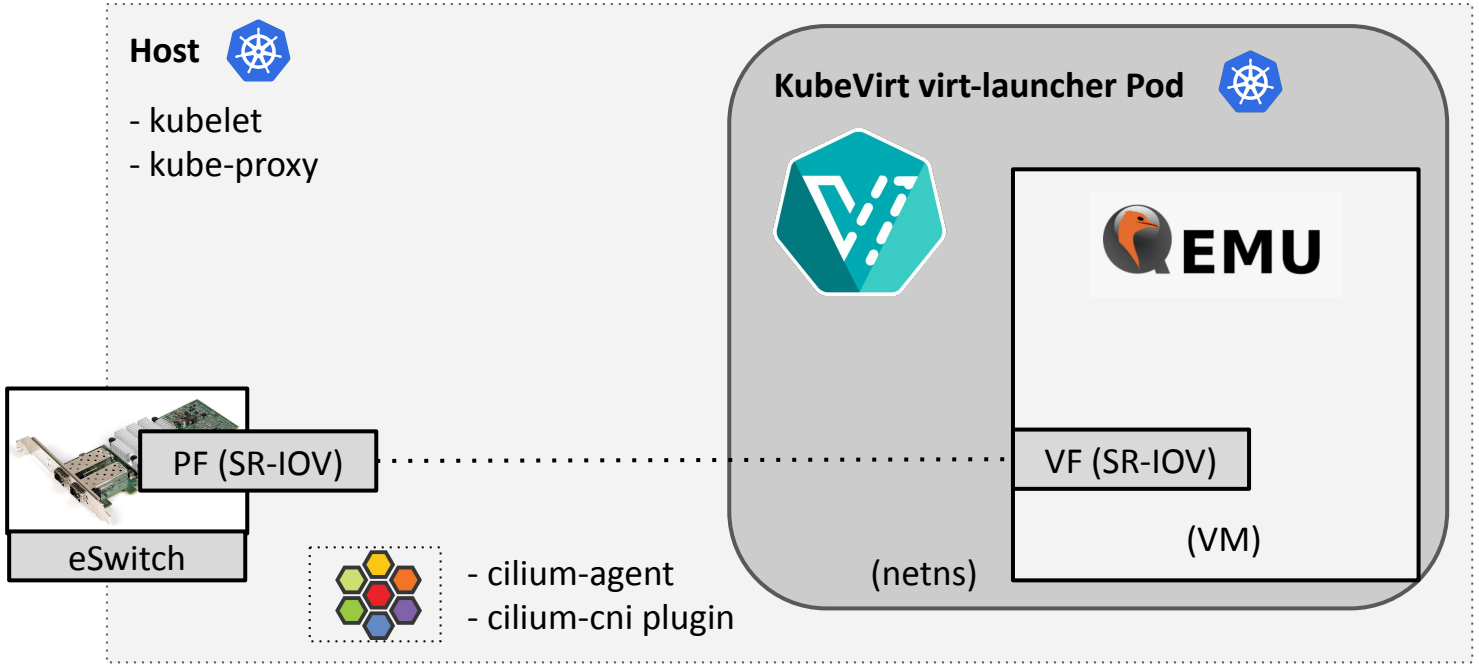
12% lower latency

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 1500 MTU
netperf -t TCP_RR -H <VM IP> -- -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

Removal of the netns overhead helps a bit, but the main overhead is still on the VM side.

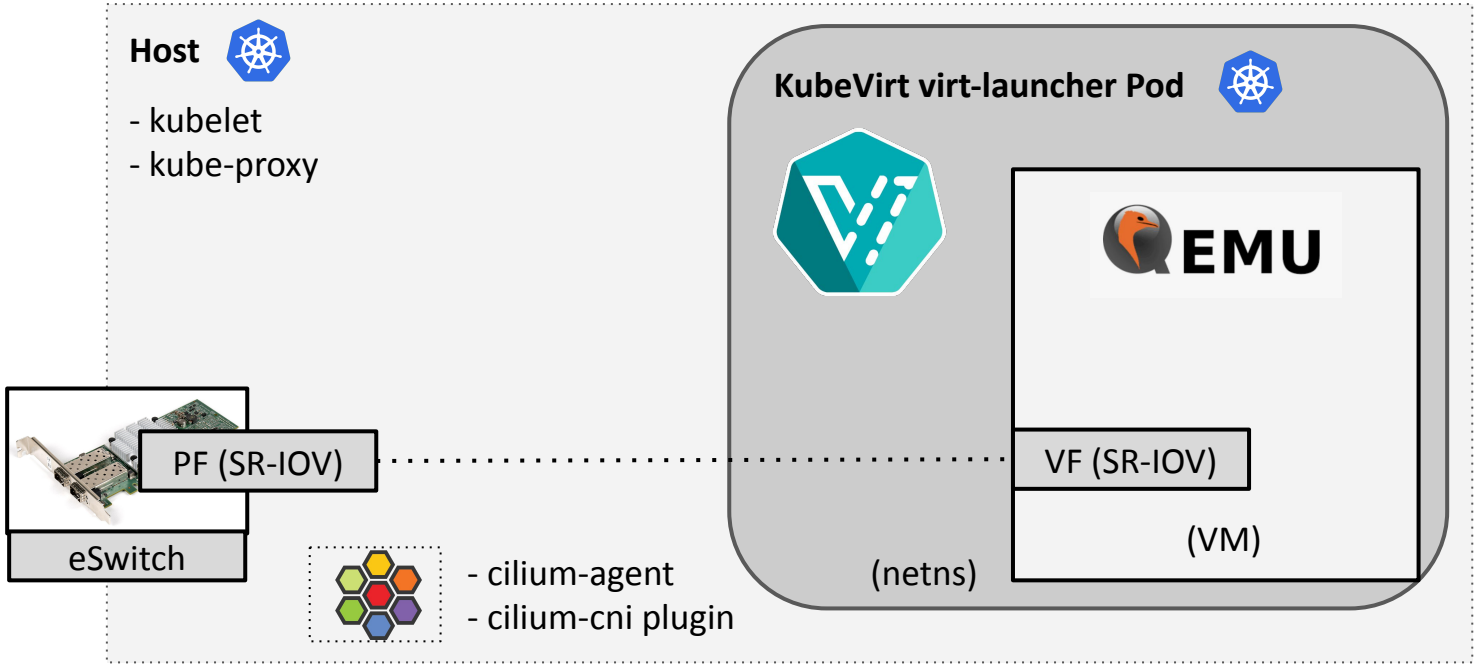
What about fundamentally challenging the bridge & vtap status quo?

What about SR-IOV?



Fast:
- HW virtualization
- VFs are lightweight PCIe functions

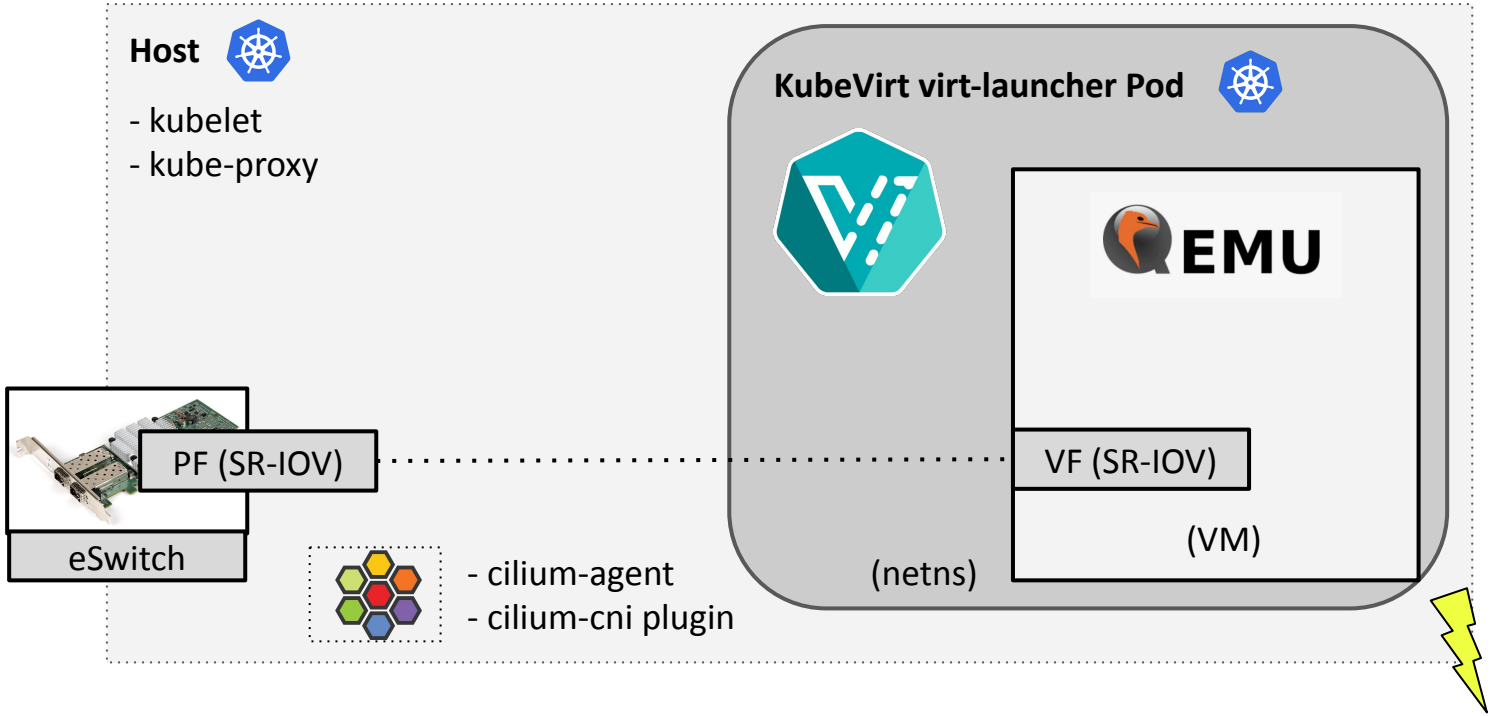
What about SR-IOV?



Fast:
- HW virtualization
- VFs are lightweight PCIe functions

Issues:
- Completely hidden from host networking stack
- No observability
- No policy enforcement
- E/W K8s service access?
- Capabilities depend on underlying hardware

What about SR-IOV?



Fast:
- HW virtualization
- VFs are lightweight PCIe functions

Issues:
- Completely hidden from host networking stack
- No observability
- No policy enforcement
- E/W K8s service access?
- Capabilities depend on underlying hardware

No unified container / VM Kubernetes stack.

Other alternatives?

QEMU / QEMU / Commits / **cb039ef3**

Commit `cb039ef3` authored 1 year ago by **Ilya Maximets** Committed by **Jason Wang** 1 year ago

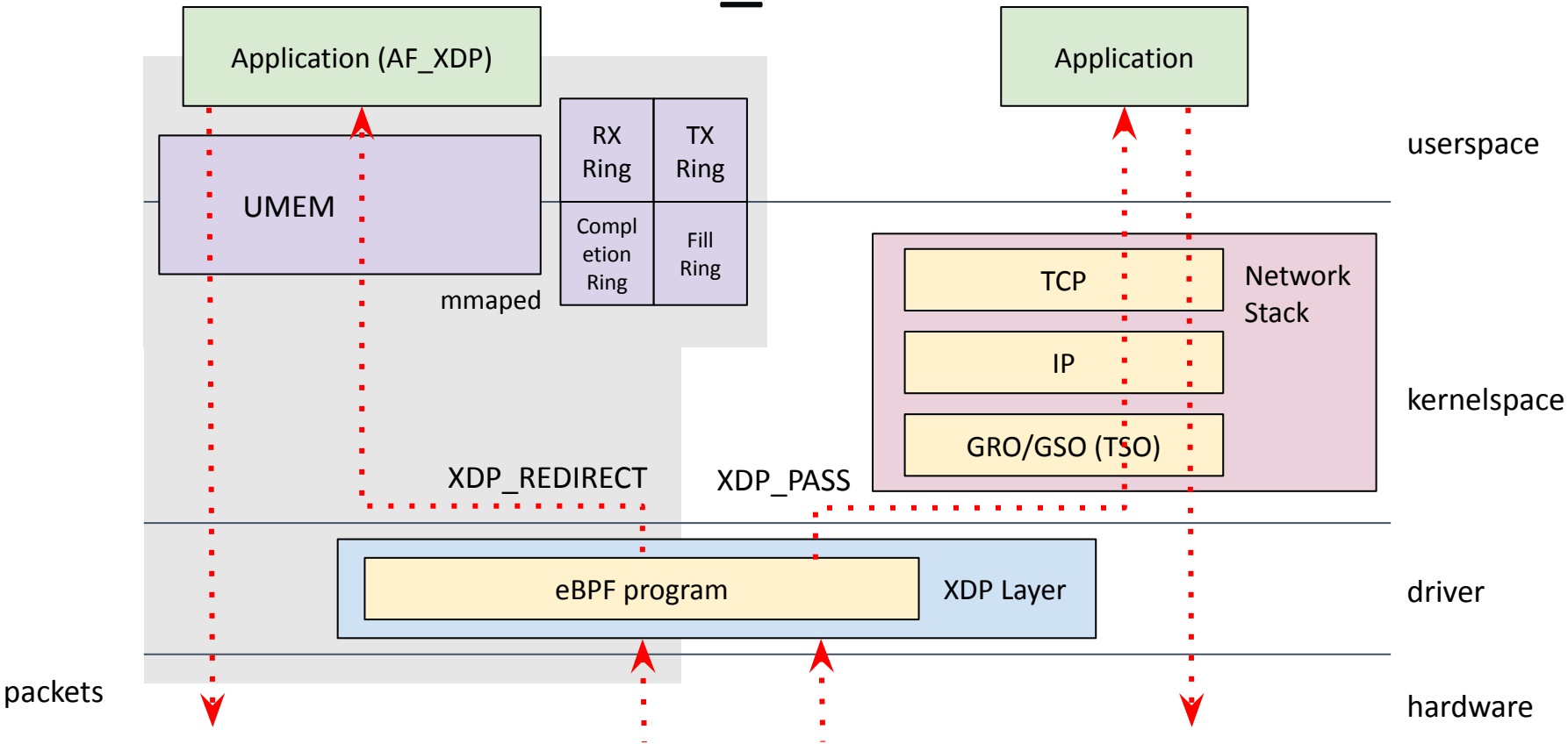
net: add initial support for AF_XDP network backend

```
AF_XDP is a network socket family that allows communication directly
with the network device driver in the kernel, bypassing most or all
of the kernel networking stack.  In the essence, the technology is
pretty similar to netmap.  But, unlike netmap, AF_XDP is Linux-native
and works with any network interfaces without driver modifications.
Unlike vhost-based backends (kernel, user, vdpa), AF_XDP doesn't
require access to character devices or unix sockets.  Only access to
the network interface itself is necessary.
```

AF_XDP

- Zero-copy mechanism for high-performance networking
- Takes advantage of XDP datapath
- Native kernel integration, no out-of-tree drivers/vendor lock-in
- Original PoC (queue bifurcation proposal) which then led to the development of AF_XDP was actually targeted at QEMU

AF_XDP


Application (AF_XDP)
Application
UMEM
RX Ring
TX Ring
Completion Ring
Fill Ring
mmaped
TCP
IP
GRO/GSO (TSO)
Network Stack
XDP_REDIRECT
XDP_PASS
eBPF program
XDP Layer
userspace
kernelspace
driver
hardware
packets

There’s a catch: KubeVirt launches QEMU in its own Pod (netns). No AF_XDP support in that netns given it needs to attach to a phys device.

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to

Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)

Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue

Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue
- The netkit device is moved into the target Pod (netns) and applications can bind against the netkit device + queue

Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

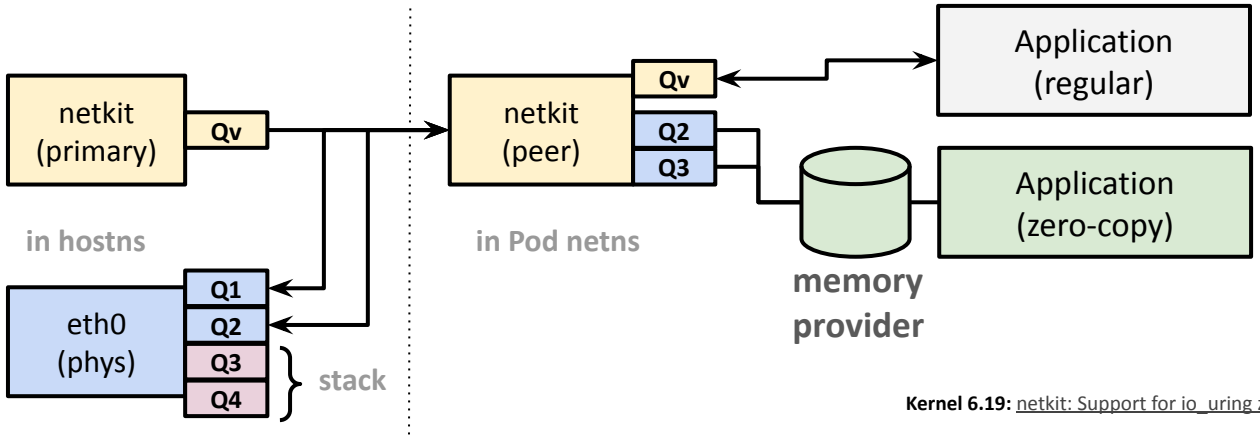
- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue
- The netkit device is moved into the target Pod (netns) and applications can bind against the netkit device + queue
- This opens up native speed for AF_XDP zero-copy, io_uring & devmem zero-copy

Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue
- The netkit device is moved into the target Pod (netns) and applications can bind against the netkit device + queue
- This opens up native speed for AF_XDP zero-copy, io_uring & devmem zero-copy

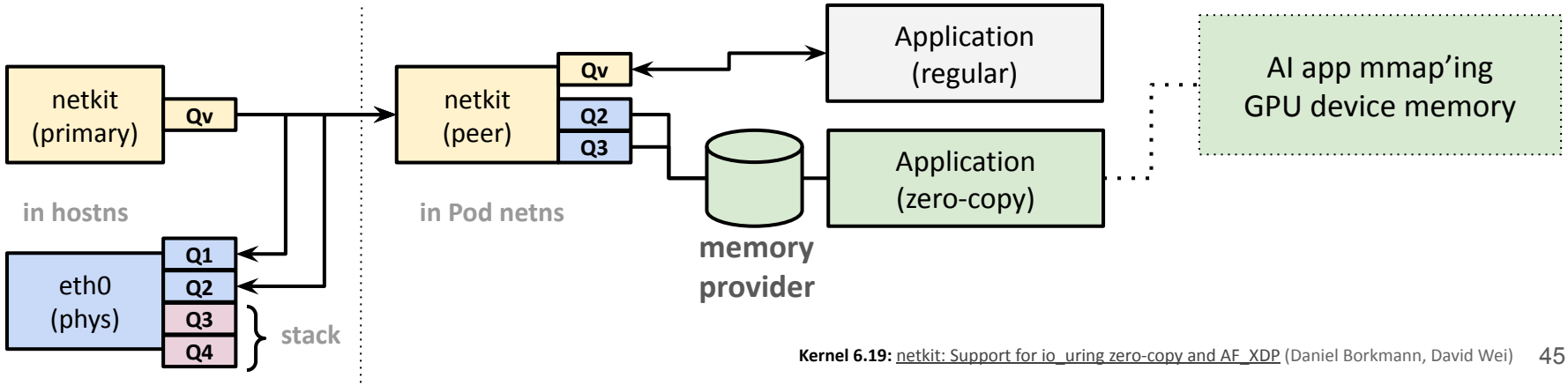


Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue
- The netkit device is moved into the target Pod (netns) and applications can bind against the netkit device + queue
- This opens up native speed for AF_XDP zero-copy, io_uring & devmem zero-copy

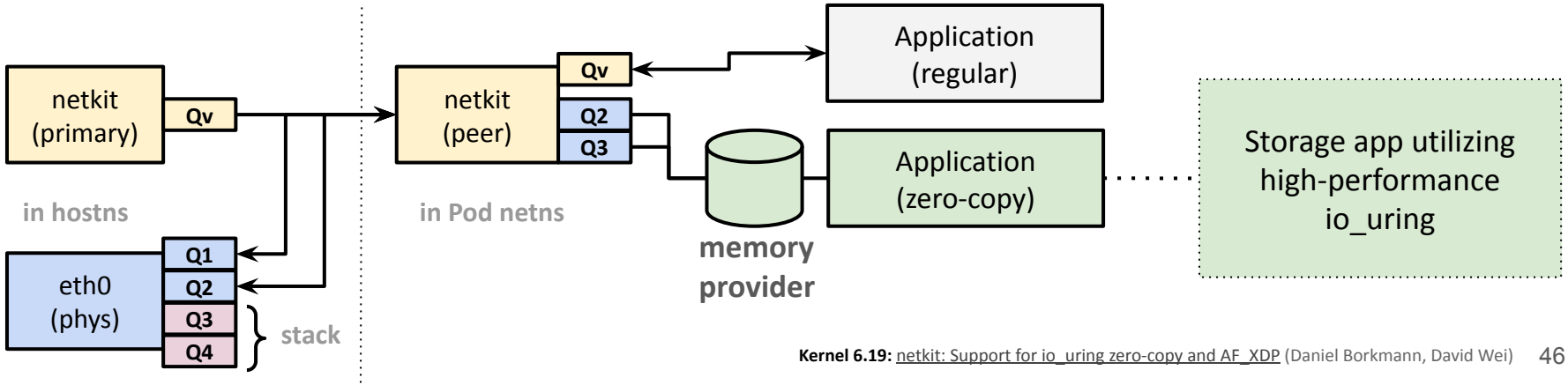


Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue
- The netkit device is moved into the target Pod (netns) and applications can bind against the netkit device + queue
- This opens up native speed for AF_XDP zero-copy, io_uring & devmem zero-copy

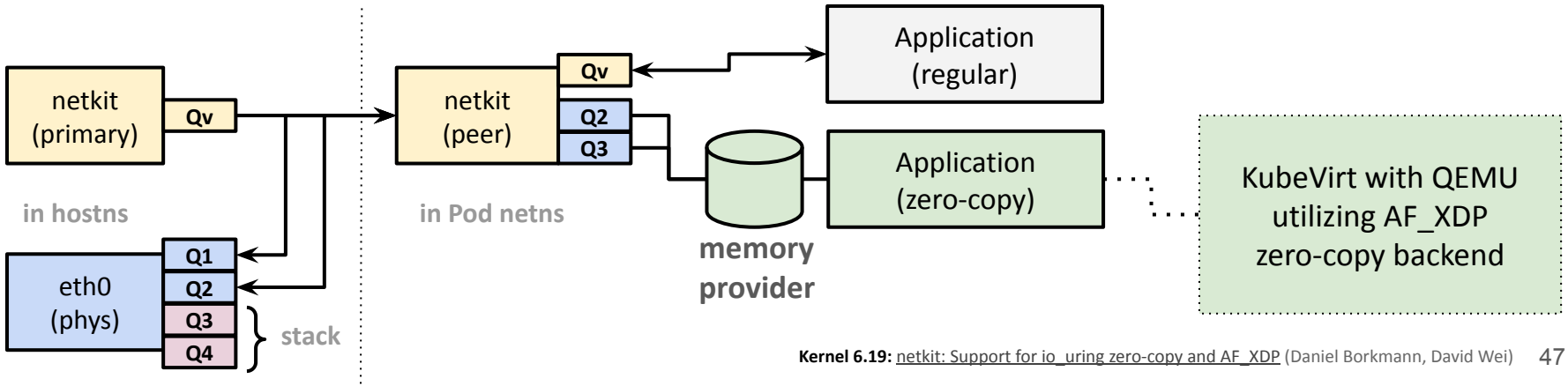


Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Deep Dive: netkit & Hardware Queue Binding for Pods

netkit extension for *native* zero-copy performance inside Pods:

- Physical NICs have multiple RX/TX queues, they can have multiple RSS contexts where traffic can be steered to
- The default RSS context can be reduced and a new one sliced to steer traffic based on rules (e.g. MAC)
- netkit can then be created with numrxqueues > 1 and bind a physical to a virtual queue
- The netkit device is moved into the target Pod (netns) and applications can bind against the netkit device + queue
- This opens up native speed for AF_XDP zero-copy, io_uring & devmem zero-copy



Kernel 6.19: netkit: Support for io_uring zero-copy and AF_XDP (Daniel Borkmann, David Wei)

Same Queue Binding, but Different Zero-Copy ‘Beasts’...

AF_XDP zero-copy

- Full control of everything, e.g. useful for DPDK-like applications in user space
- Build-your-own load balancer, gateway, firewall, *QEMU* has a backend
- Upper kernel stack is bypassed, we reuse the driver, netdevice, XDP

io_uring with TCP zero-copy support

- The industry is now shifting towards header/data split for high-end NICs (bnxt/mlx5/gve)
- Data (payload) pages mmap’ed with user space application or GPU device memory
- Upper kernel stack is *reused*, not bypassed - packet traverses up the normal stack

Results: 4096 MTU

[kernel-recipies]

MTU	memcmp	Engine	BW	Gain	Net CPU busy%	Net CPU softirq%
4096	✅	epoll	66.9 Gbps		24.8%	23.7%
4096	✅	io_uring ZC	92.2 Gbps	+37.8%	36.6%	35.0%
4096	❌	epoll	82.2 Gbps		33.2%	32.6%
4096	❌	io_uring ZC	116.2 Gbps	+41.4%	48.2%	46.6%

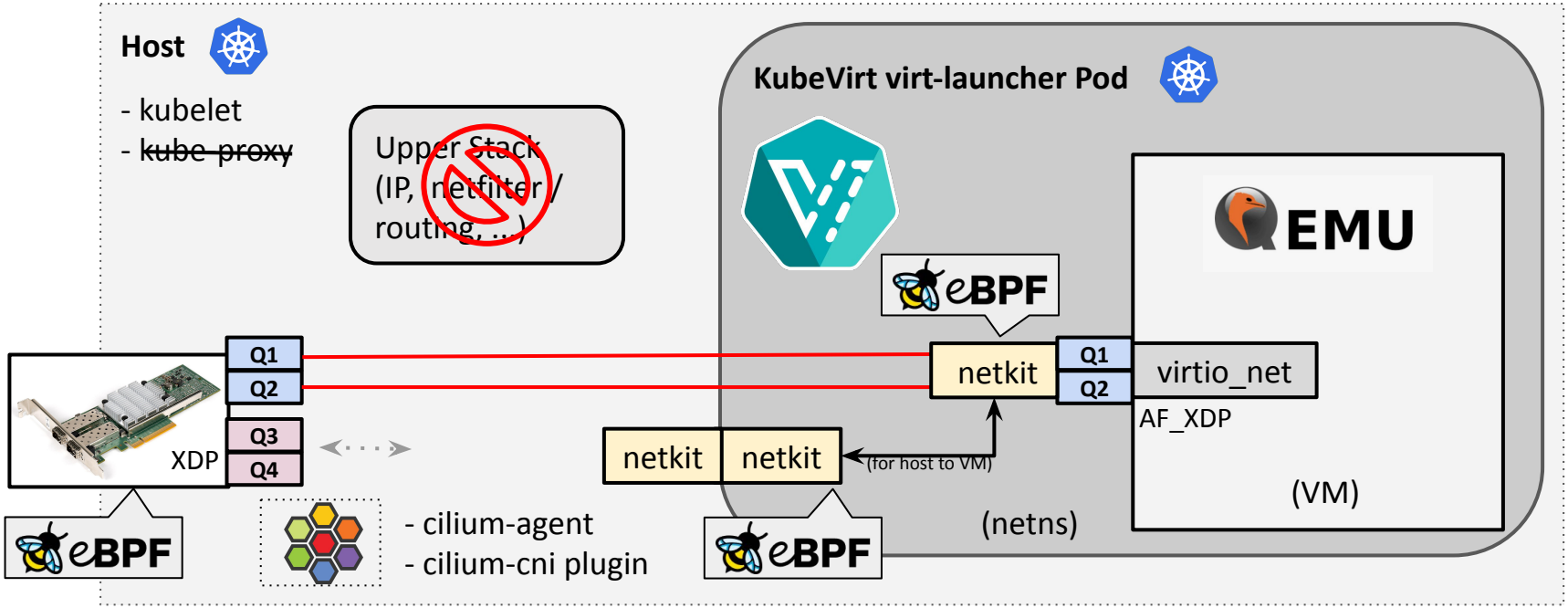
(beyond 100 Gbit/s for a single TCP stream requires zero-copy)

io_uring with TCP zero-copy support

- The industry is now shifting towards header/data split for high-end NICs (bnxt/mlx5/gve)
- Data (payload) pages mmap'ed with user space application or GPU device memory
- Upper kernel stack is *reused*, not bypassed - packet traverses up the normal stack

How can the netkit queue-binding & AF_XDP be applied for KubeVirt Pods?

Cilium's Future KubeVirt Pod Datapath:

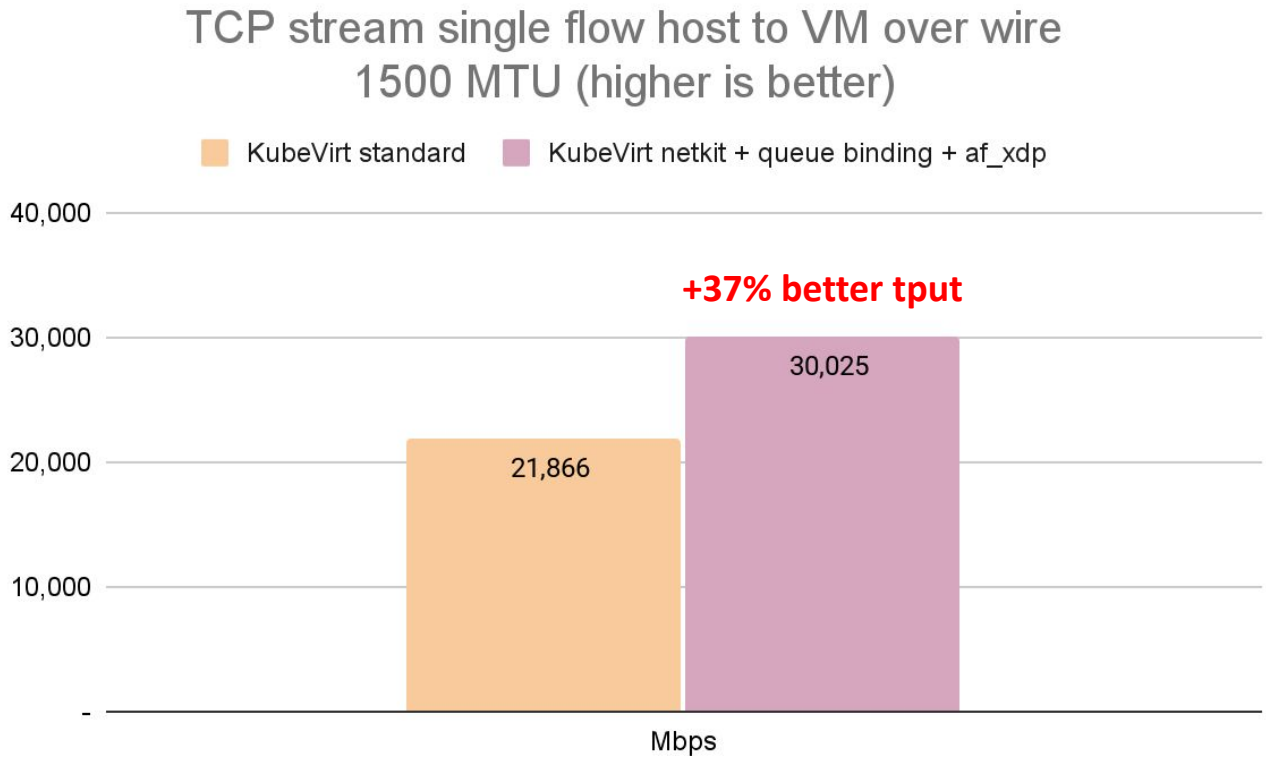


Preliminary benchmarks with netkit + AF_XDP for KubeVirt look promising!

Cilium's Future K8s KubeVirt Pod Datapath:

Building Blocks:

- Cilium sets up new RSS context based on VM MAC
- Creates netkit device and binds netkit to phys queue
- Cilium KubeVirt plugin to launch QEMU with af-xdp bound to netkit

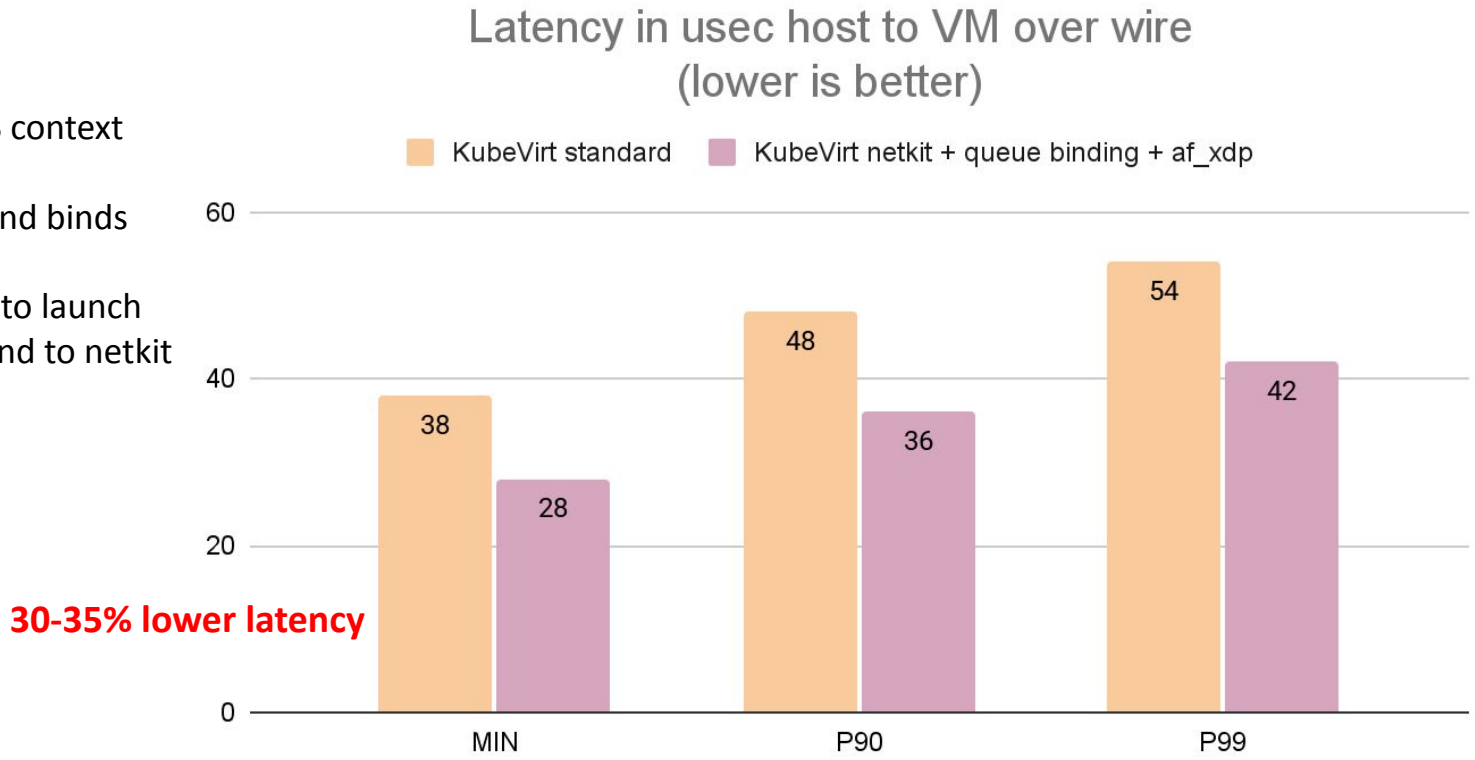


Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 1500 MTU
netperf -t TCP_RR -H <VM IP> -- -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

Cilium's Future K8s KubeVirt Pod Datapath:

Building Blocks:

- Cilium sets up new RSS context based on VM MAC
- Creates netkit device and binds netkit to phys queue
- Cilium KubeVirt plugin to launch QEMU with af-xdp bound to netkit



30-35% lower latency

Back to back: AMD Ryzen 9 3950X @ 3.5 GHz, 128G RAM @ 3.2 GHz, PCIe 4.0, ConnectX-6 Dx, mlx5 driver, striding mode, LRO off, 1500 MTU
netperf -t TCP_RR -H <VM IP> -- -O MIN_LATENCY,P90_LATENCY,P99_LATENCY,THROUGHPUT

Ongoing & future work for KubeVirt acceleration

- XDP API overhaul in the Linux kernel:
  - Support for native multi-attach via bpf_mprog infrastructure
  - Support for attaching BPF only to specific queues

Ongoing & future work for KubeVirt acceleration

- XDP API overhaul in the Linux kernel:
  - Support for native multi-attach via bpf_mprog infrastructure
  - Support for attaching BPF only to specific queues
- AF_XDP extensions in the Linux kernel:
  - New AF_XDP TX hook for BPF attachments
  - BPF xsk-map-less redirect to the target socket
  - AF_XDP TSO offload support

Ongoing & future work for KubeVirt acceleration

- XDP API overhaul in the Linux kernel:
  - Support for native multi-attach via bpf_mprog infrastructure
  - Support for attaching BPF only to specific queues
- AF_XDP extensions in the Linux kernel:
  - New AF_XDP TX hook for BPF attachments
  - BPF xsk-map-less redirect to the target socket
  - AF_XDP TSO offload support
- Further optimize QEMU's AF_XDP integration:
  - Various low-hanging fruit: utilizing scatter/gather IO in backend
  - Bigger items further helping performance: full zero-copy into VM

Small demo: Cilium and KubeVirt with netkit’s queue binding and AF_XDP QEMU backend ...

```
+ yq -C .spec.template < vm.yaml | head -20
{
  "metadata": {
    "labels": {
      "kubevirt.io/vm": "vm-net-binding-netkit"
    },
    "annotations": {
      "cilium.io/bind-queue": "enp2s0f1np1/15",
      "cilium.io/mac": "18:c0:4d:77:88:99"
    }
  },
  "spec": {
    "domain": {
      "devices": {
        "interfaces": [
          {
            "name": "netkit",
            "binding": {
              "name": "netkit"
            }
          }

.................
```

Acknowledgements

Ilya Maximets (Red Hat; QEMU's initial AF_XDP backend integration)

Nikolay Aleksandrov (Nvidia; netkit co-creator)

Jordan Rife, Alasdair McWilliam (Google & Isovalent; Cilium netkit contributions)

David Wei, Jakub Kicinski, Stanislav Fomichev (Meta; io_uring & kernel networking)

Martin Lau (Meta; netkit production rollout)

Chen Tang (Bytedance; netkit production rollout)

Björn Töpel, Magnus Karlsson, Maciej Fijalkowski (Rivos & Intel, AF_XDP)

Cilium, netdev, BPF & QEMU communities

ISOVALENT
now part of CISCO

Thank you! Questions?

github.com/cilium/cilium

cilium.io

ebpf.io

kubevirt.io

Cilium & KubeVirt Instruqt Intro Lab

Cilium & netkit Tuning Guide

netkit Linux kernel upstream driver