Beam Me Through the Datapath

VDUSE for OpenShift Virtualization

Jakob Meng
Telco Engineering

Maxime Coquelin
Fast Datapath Team

Red Hat

Agenda

- Project overview
- Network datapaths for containers
- Network datapaths for virtual machines
- Deep dive into userspace datapaths
- Enhanced workload partitioning
- Benchmark results and optimizations
- Conclusion

Project overview

Mission

Evaluate the OpenShift networking stack in userspace:

- Open vSwitch's **netdev** (~~**system**~~) bridges,
- **DPDK** (~~**kernel**~~) drivers when attaching physical NICs to OVS bridges,
- **VDUSE** (~~**VETH**~~) devices for containers (eth0),
- **VDUSE/vhost-vdpa** (~~**tap**~~ or ~~**SR-IOV**~~) devices for KubeVirt virtual machines,
- Enhanced **workload partitioning** to strictly isolate system, OVS, and user processes.

OpenShift Networking Transformed: Fully Embracing DPDK Datapaths in OVN-K8s!?
(Recording from OVS+OVN Conf 2024↗)

Value Proposition

Promises

- Deterministic datapath scheduling,
  esp. predictable packet latency
- Granular system partitioning & dimensioning
- Unified datapath for primary & secondary networks
- Enhanced performance while supporting
  VM live migration

Target Audience

- Telcos
- VMware customers considering migration
- Users requiring real-time networking

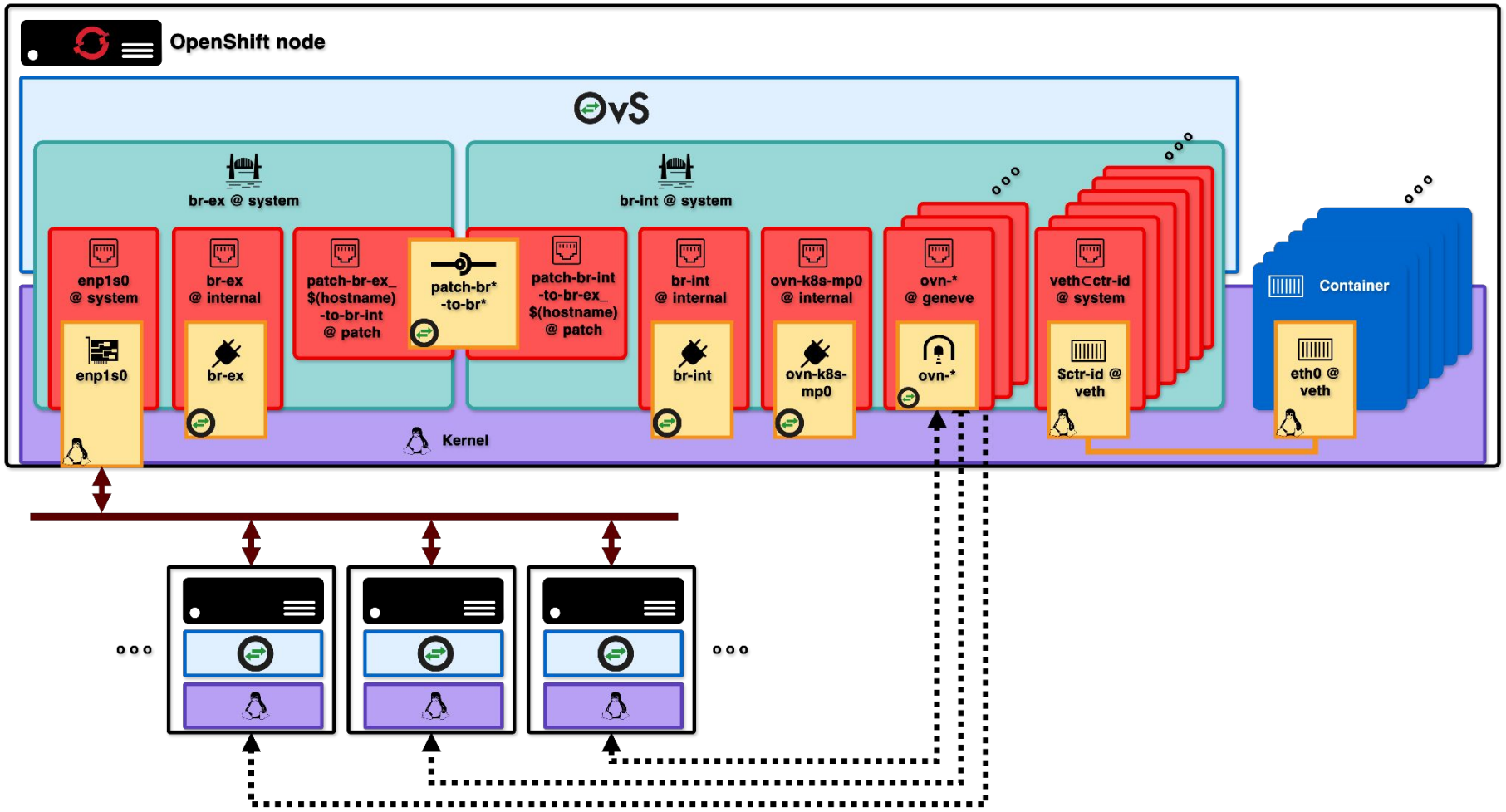
Network datapaths for containers

Recap

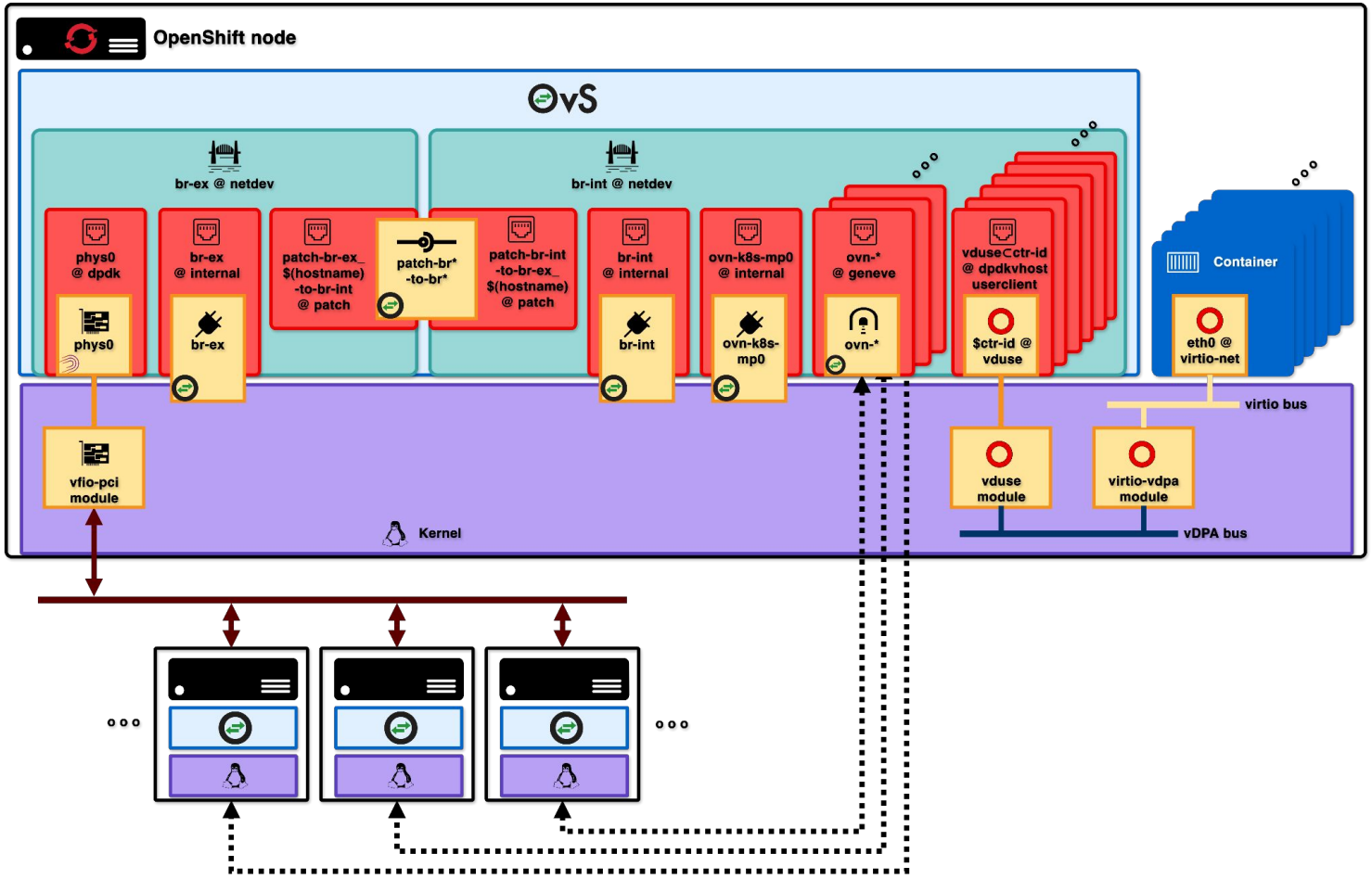
Previously on OVS+OVN Conf 2024...

- OpenShift Networking Transformed:
  Fully Embracing DPDK Datapaths in OVN-K8s!?
  Recording↗ / Slides↗

Network of an OpenShift node

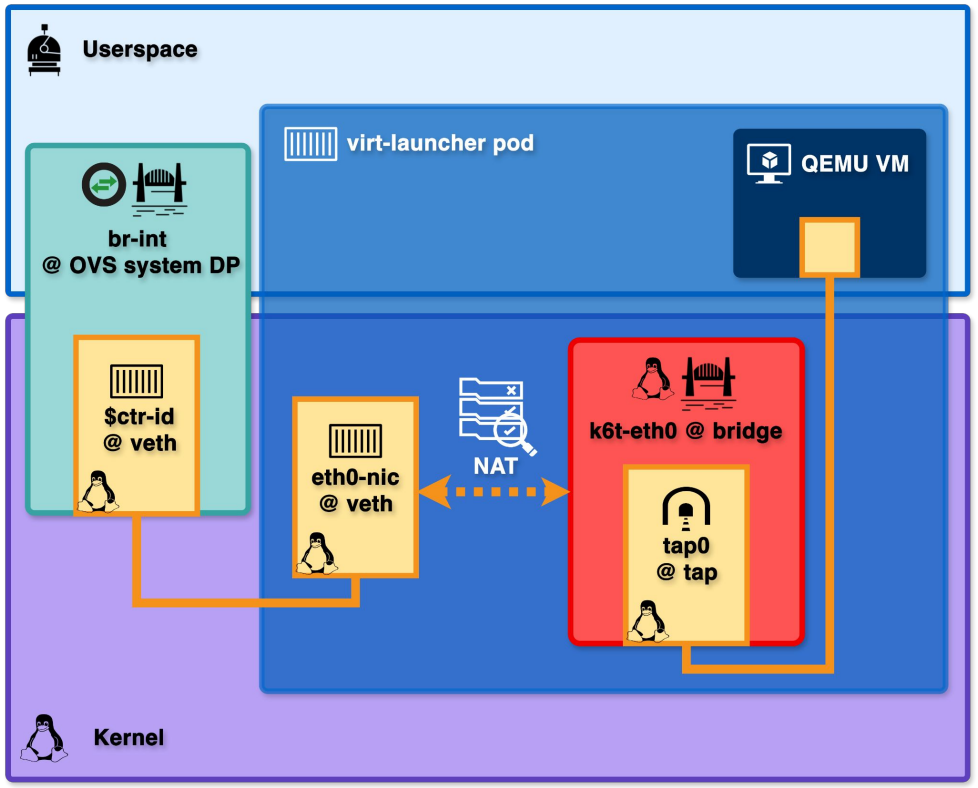


Network of an OpenShift node with DPDK and VDUSE

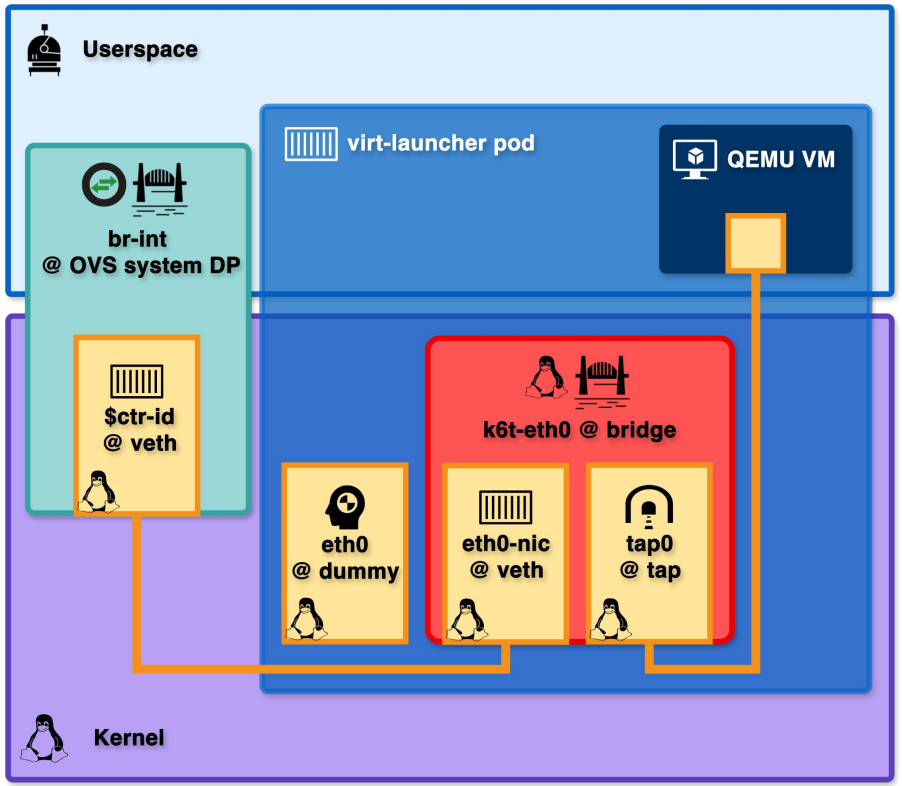


Network datapaths for virtual machines

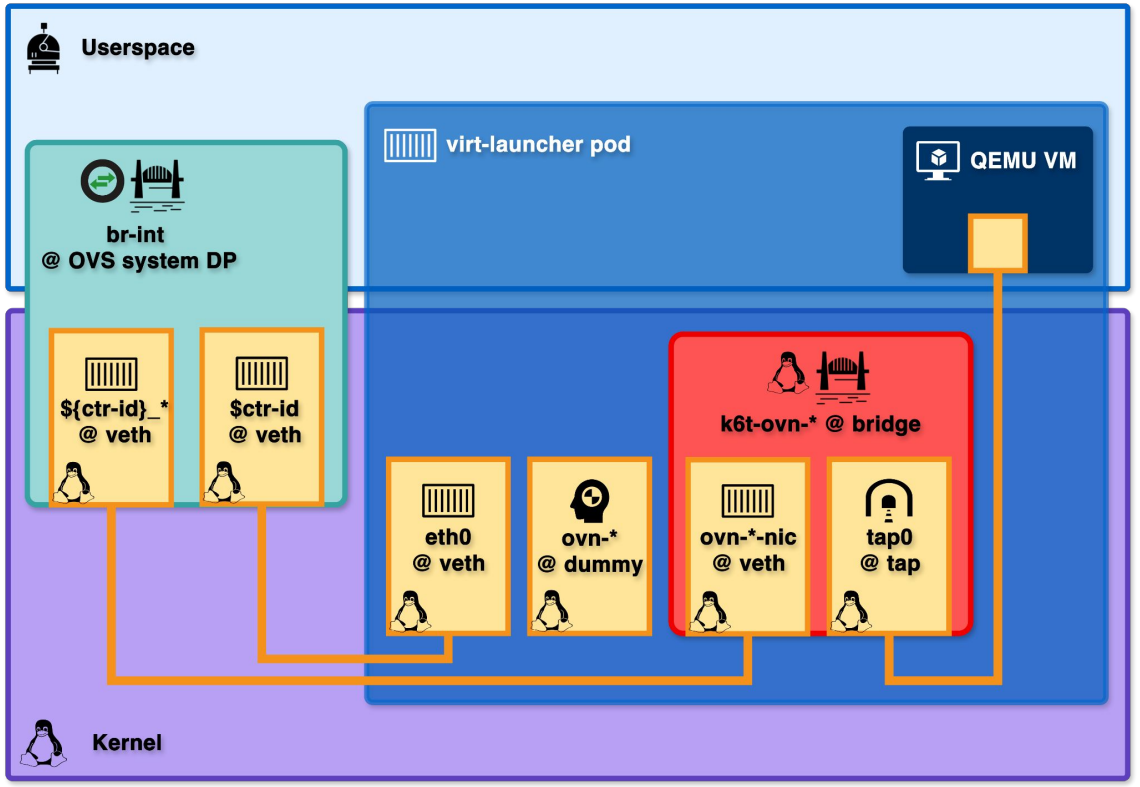
KubeVirt VM with default pod networking (masqerade)



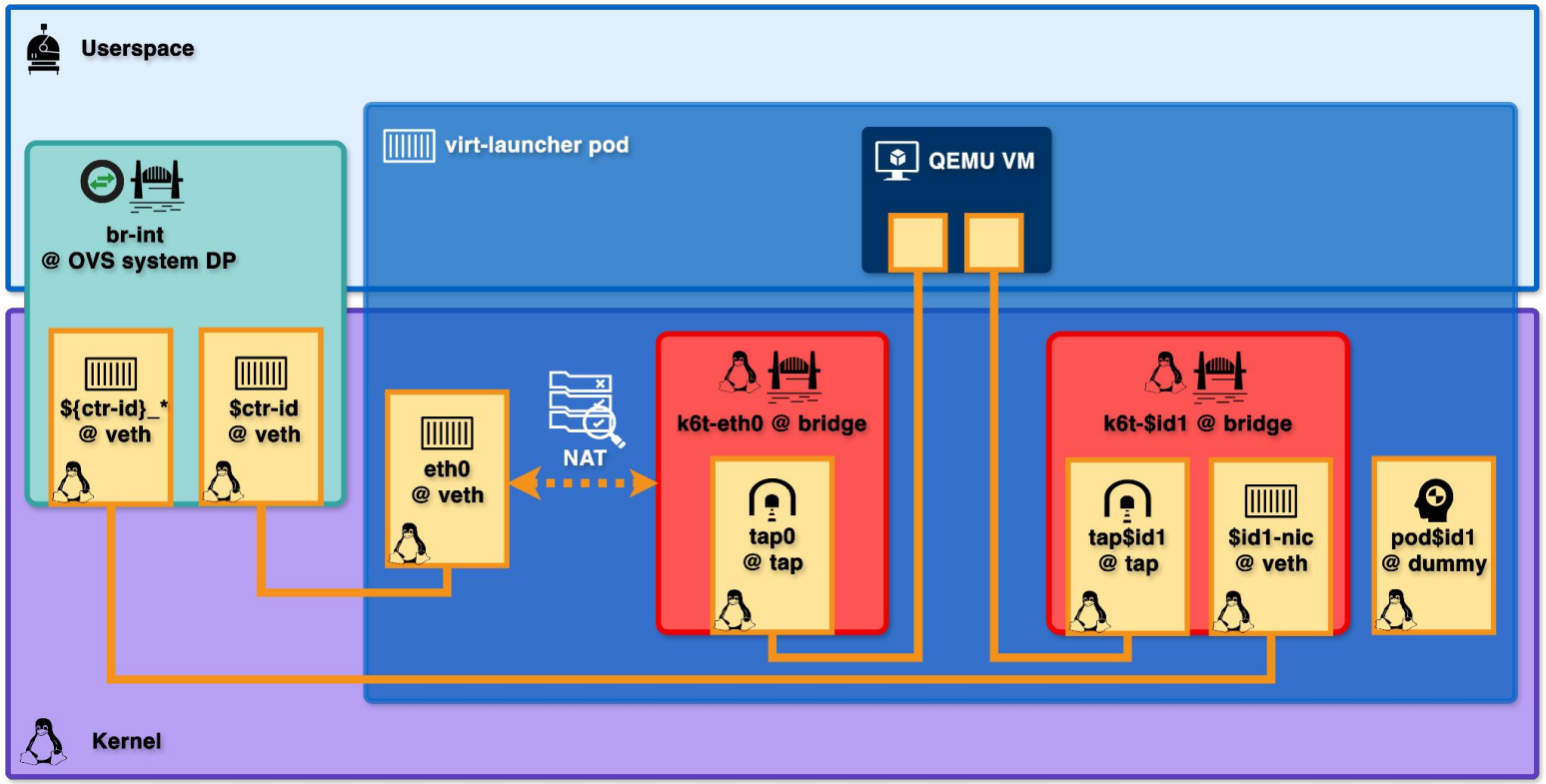
KubeVirt VM with default pod networking (bridge)



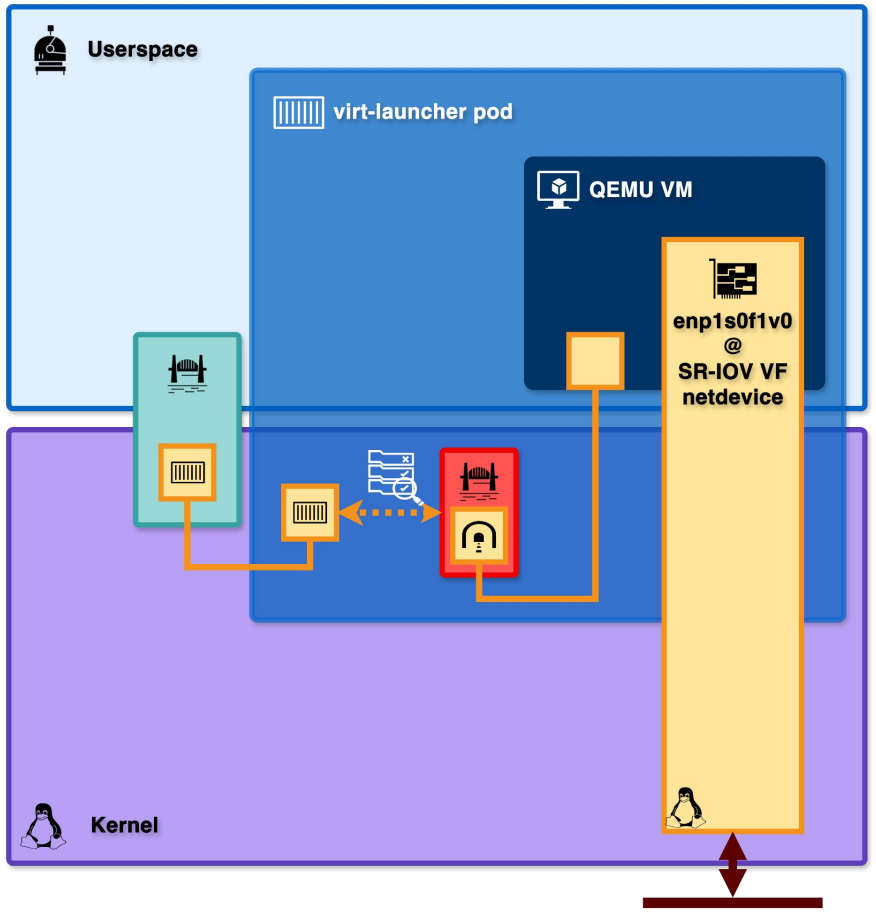
KubeVirt VM with a User-Defined Network (UDN)



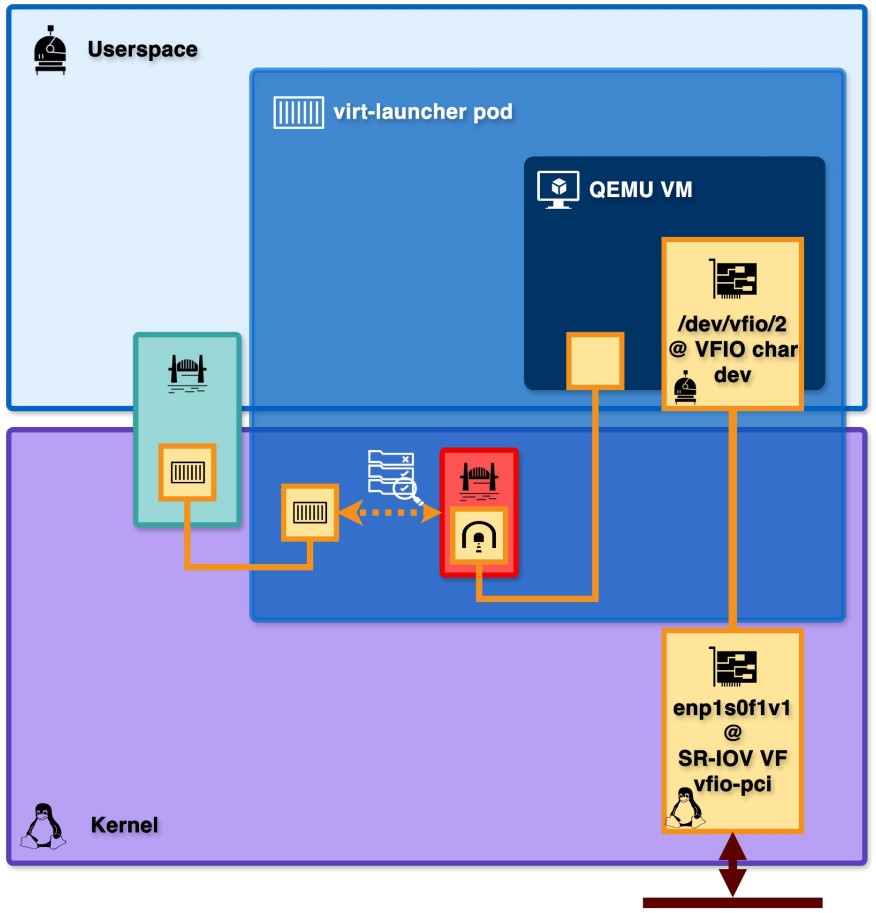
KubeVirt VM with a OVN-Kubernetes Secondary Network



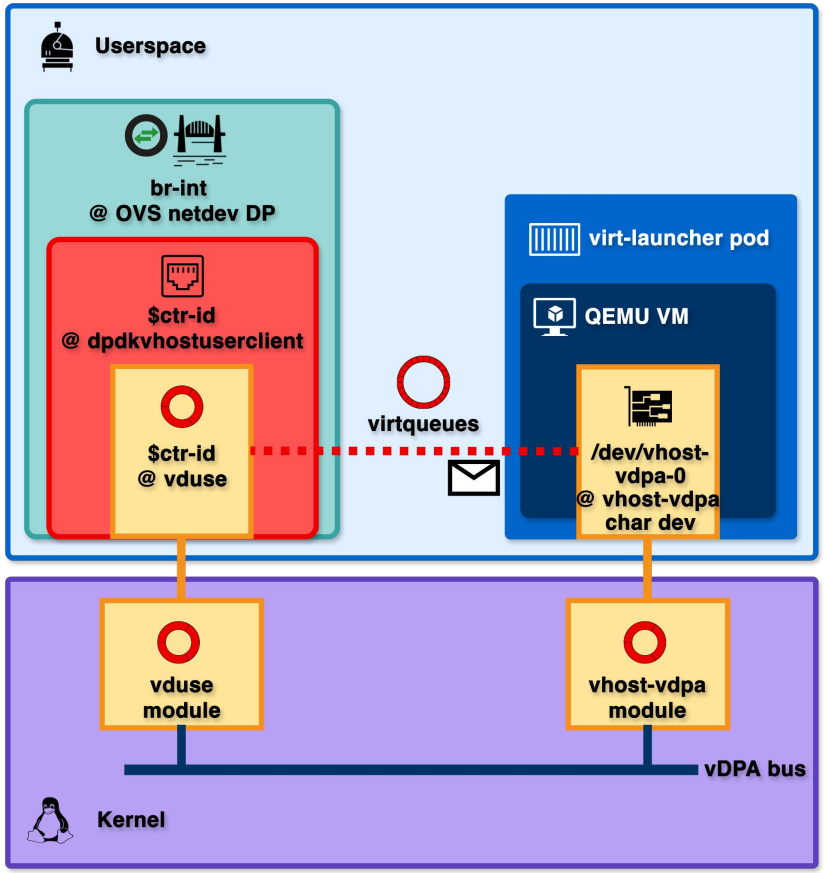
KubeVirt VM with default pod network and a SR-IOV VF (netdev)



KubeVirt VM with default pod network and a SR-IOV VF (vfio-pci)



KubeVirt VM with a primary VDUSE/vhost-vDPA network



Deep dive into userspace datapaths

Why would we need DPDK in OCP/K8s?

The OVS kernel datapath is already used in production and handles most of the use cases. But we believe there are several reasons why a userspace datapath could bring benefits in some cases:

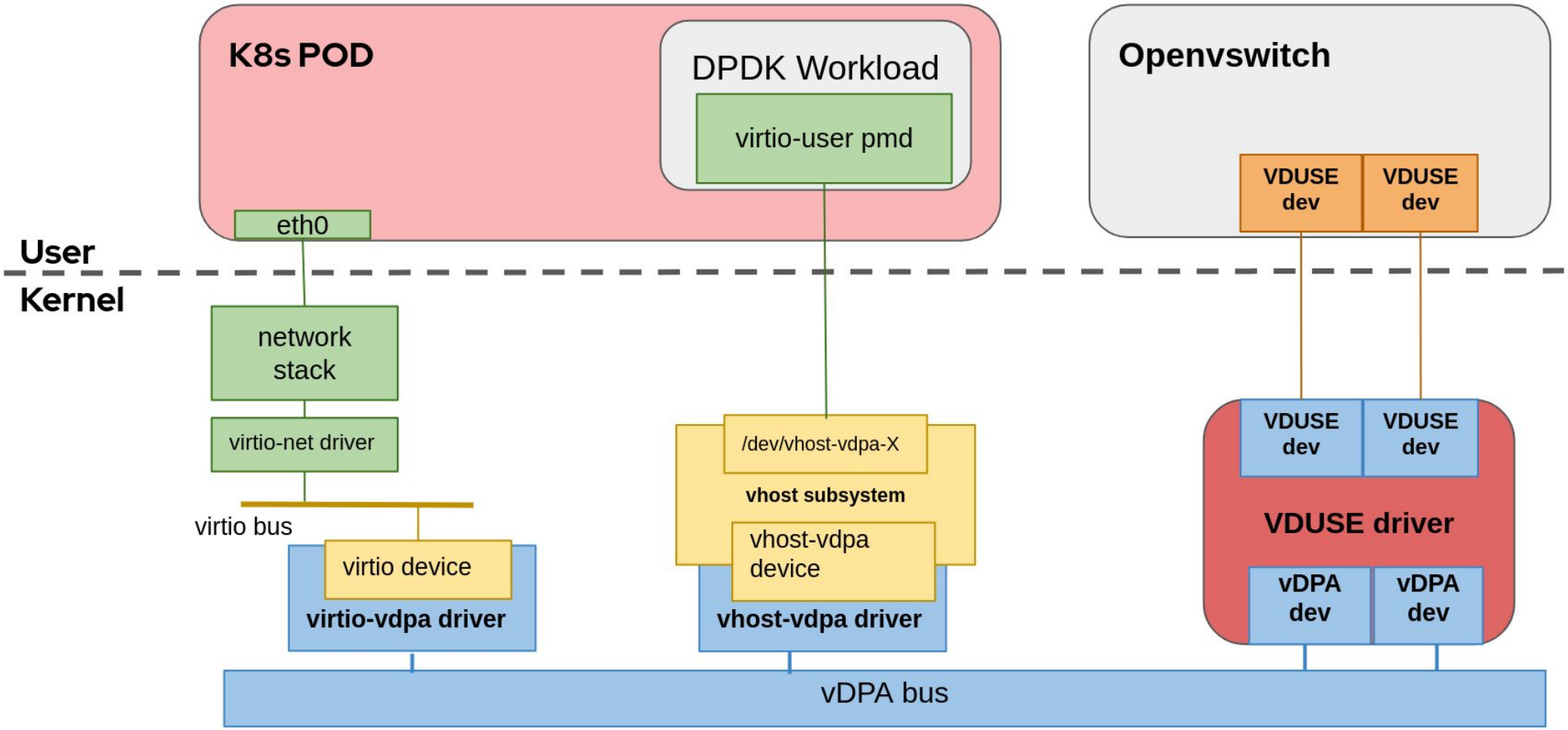
- Partitioning: Isolation of the networking infra from the workloads
- Determinism: Provide better predictability of the packets latency
- Uniformity: Same datapath for primary and secondary networks

VDUSE

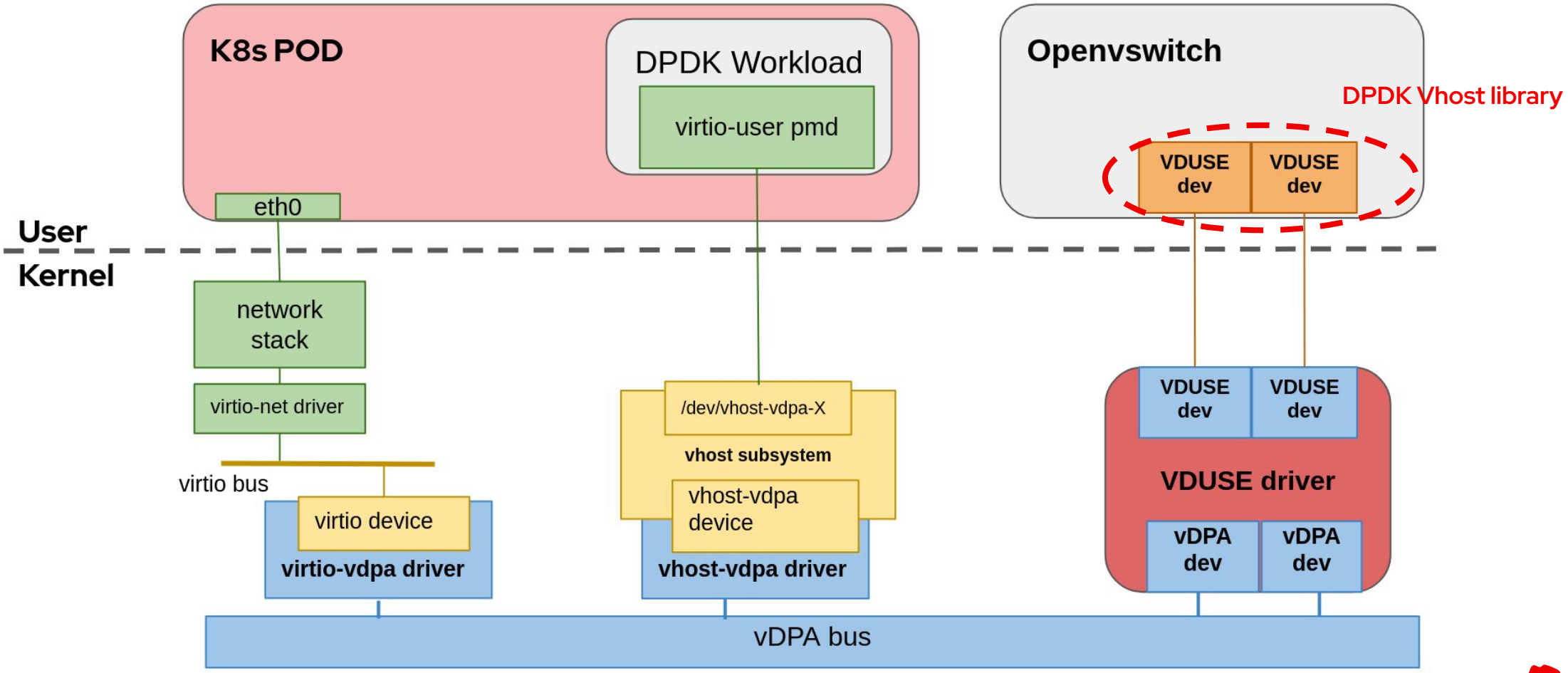
VDUSE stands for **v**DPA **d**evice in **use**rspace

- Usually, vDPA devices are physical (ConnectX6, Octeon, ...)
  - Devices implements the Virtio datapath
  - Vendor specific control path
  - vDPA drivers implement vDPA callbacks to control the device
- VDUSE is purely software, with two components
  - A kernel driver that connects to the vDPA bus
  - A userspace application that implements the actual device

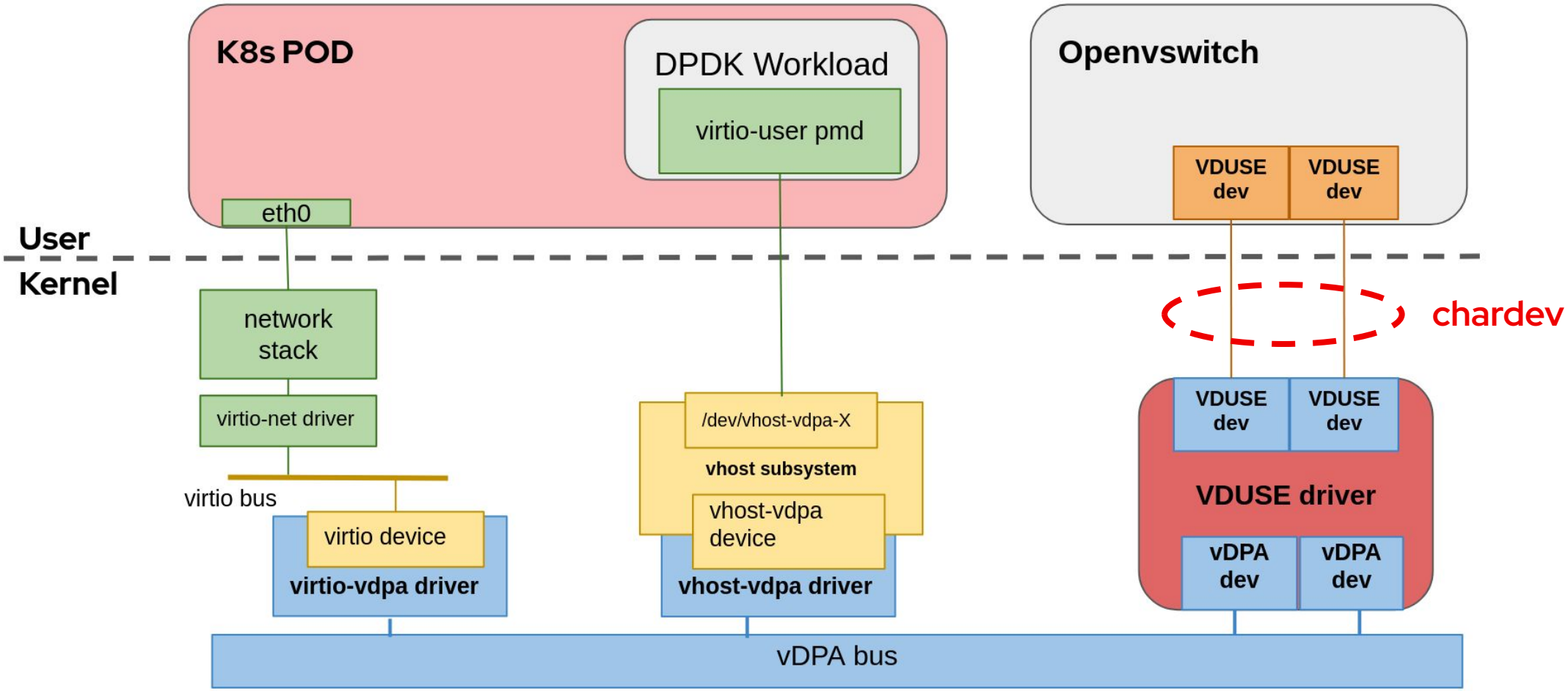
VDUSE for networking architecture



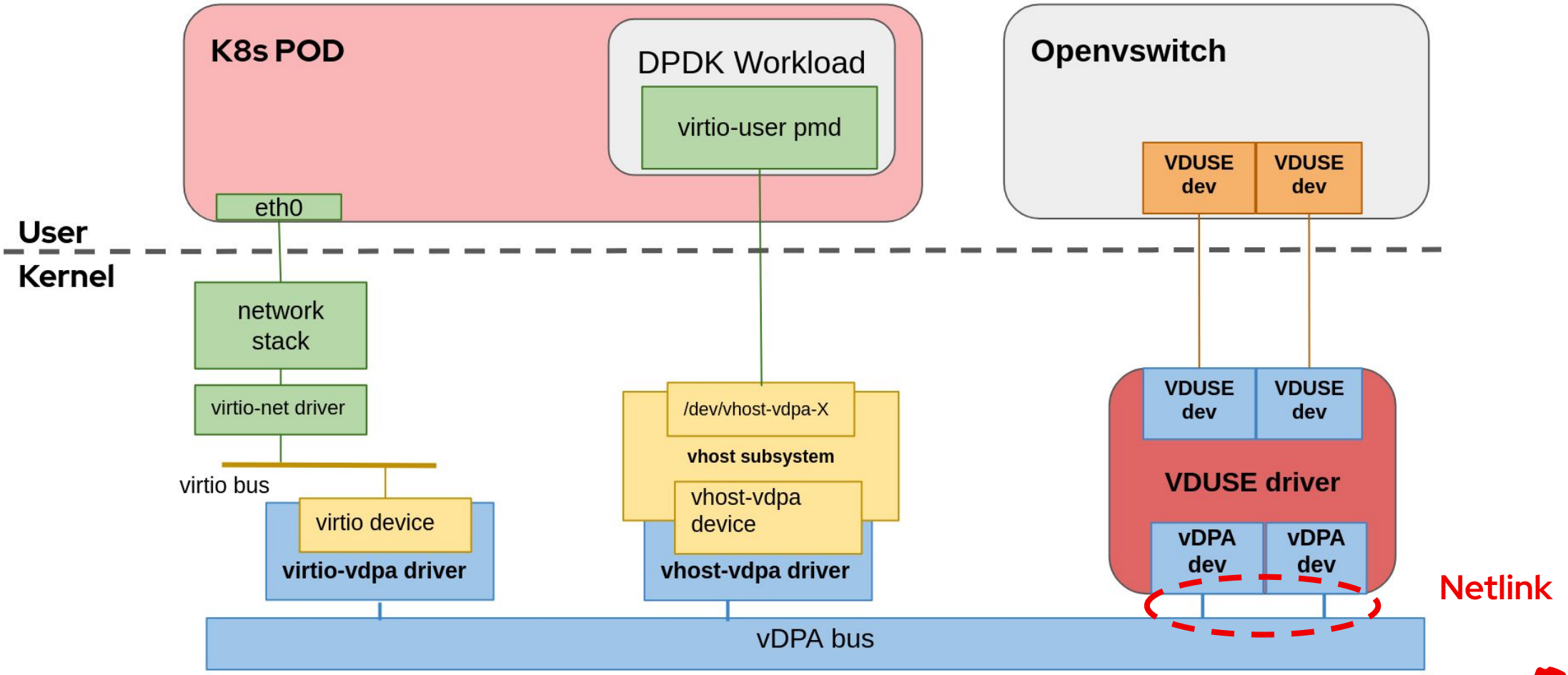
VDUSE for networking architecture



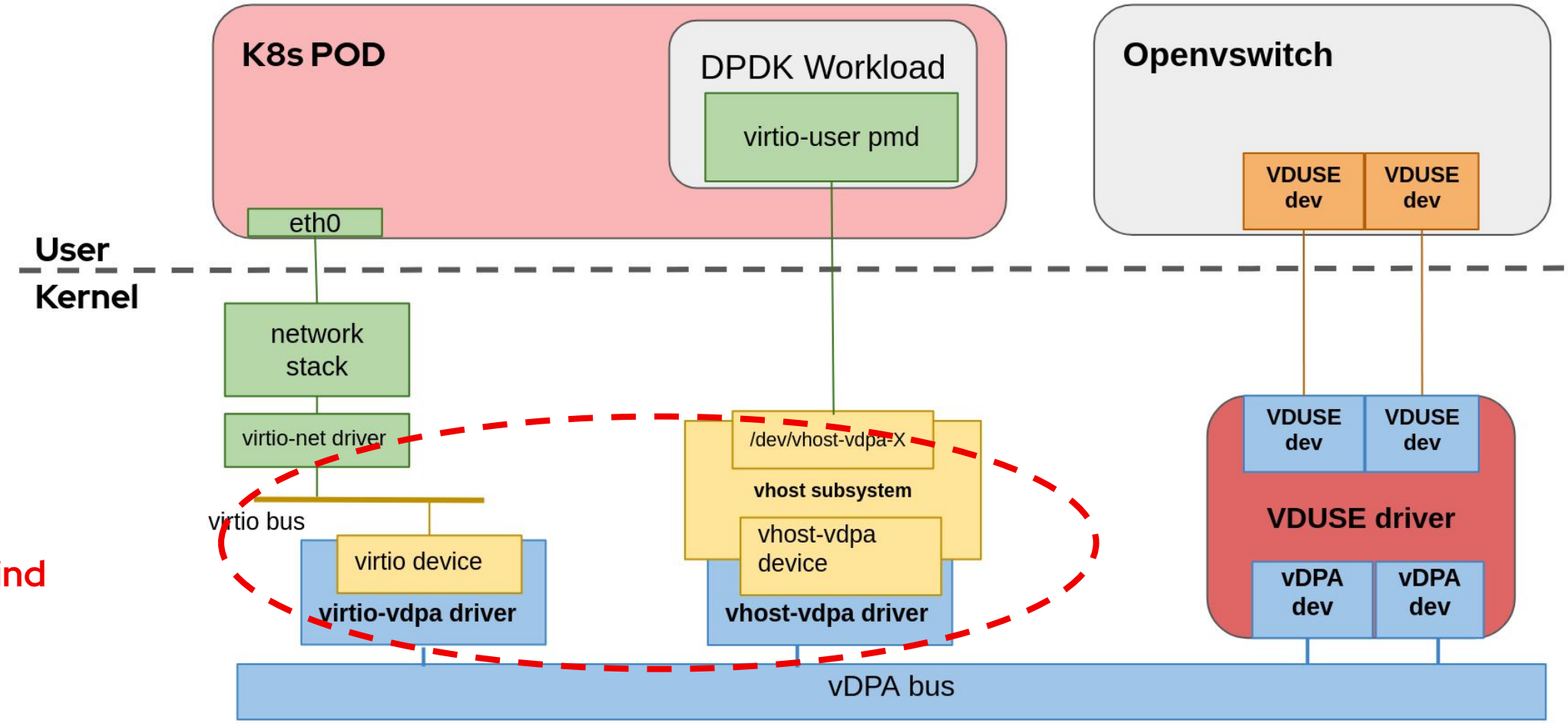
VDUSE for networking architecture



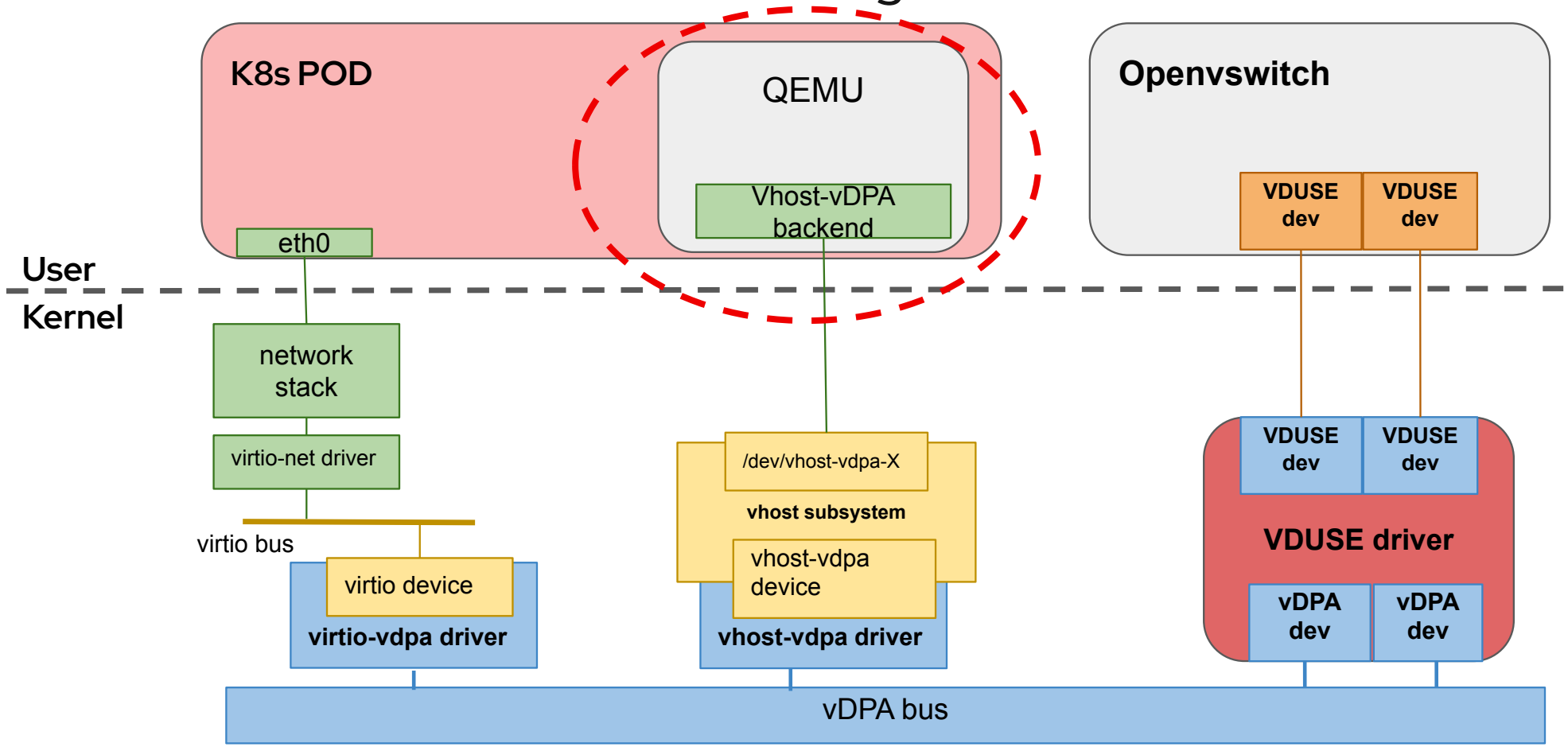
VDUSE for networking architecture



VDUSE for networking architecture



VDUSE for networking architecture



Enhanced workload partitioning

CPU affinity with Workload Partitioning (stock OpenShift 4.14+)

CPU affinity with Workload Partitioning (stock OpenShift 4.14+)

- Each core shown implicitly includes its HT sibling
- Number of reserved and isolated cores is choosen solely for illustrative purposes
- Core split is applied uniformly across all NUMA nodes
- Dynamic CPU affinity (pinning) of OVS is available and force-enabled on 4.14+
- €-pod with tuning for real-time processing (esp. irq-load-balancing.crio.io disabled)
- $-pod without tuning for real-time proc

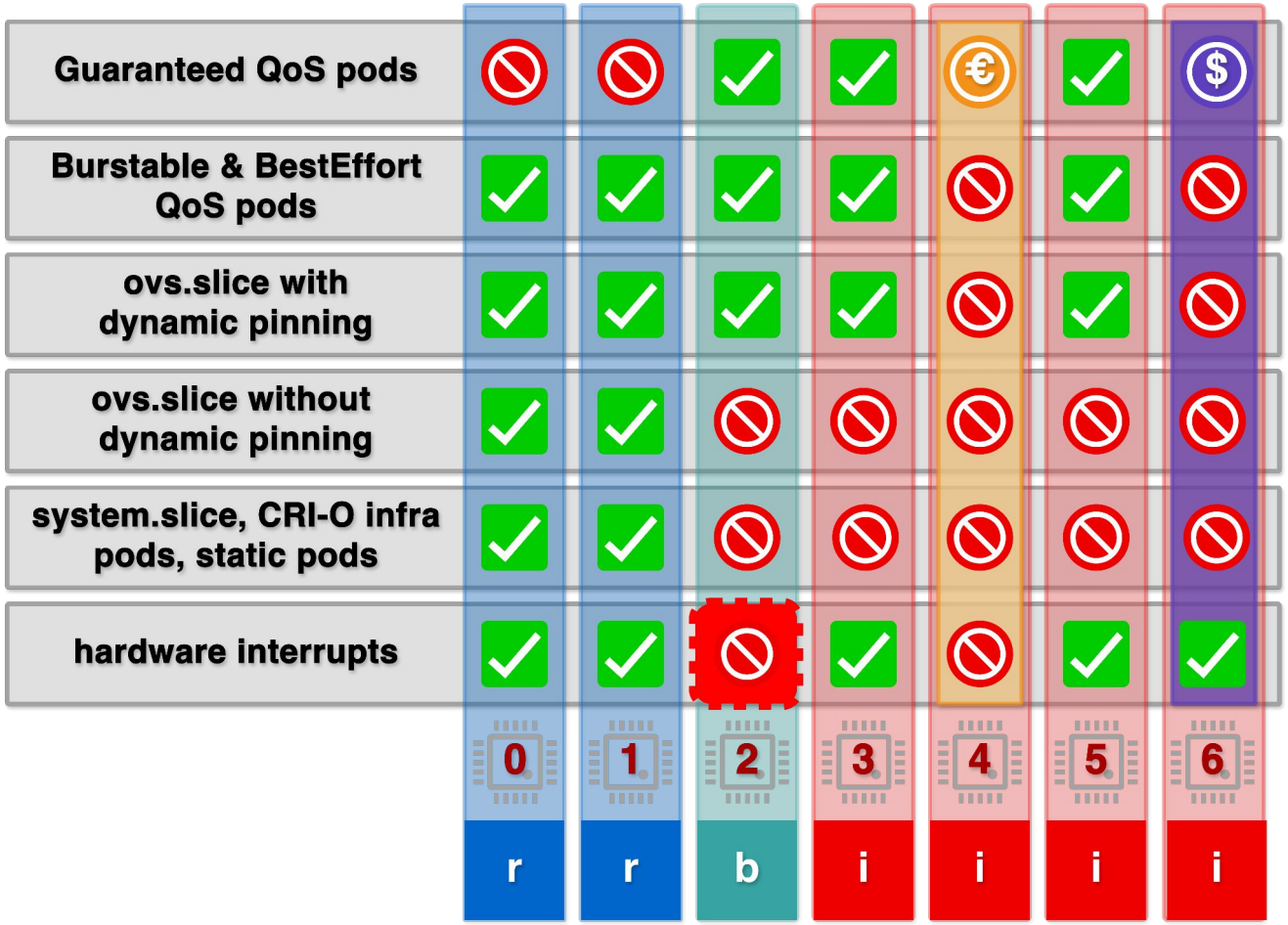
CPU affinity with Workload Partitioning (strict-cpu-reservation)

CPU affinity with Workload Partitioning (strict-cpu-reservation)

CPU Manager static policy option:

- *strict-cpu-reservation=true*

CPU affinity with Workload Partitioning (irqbalanceBanned)



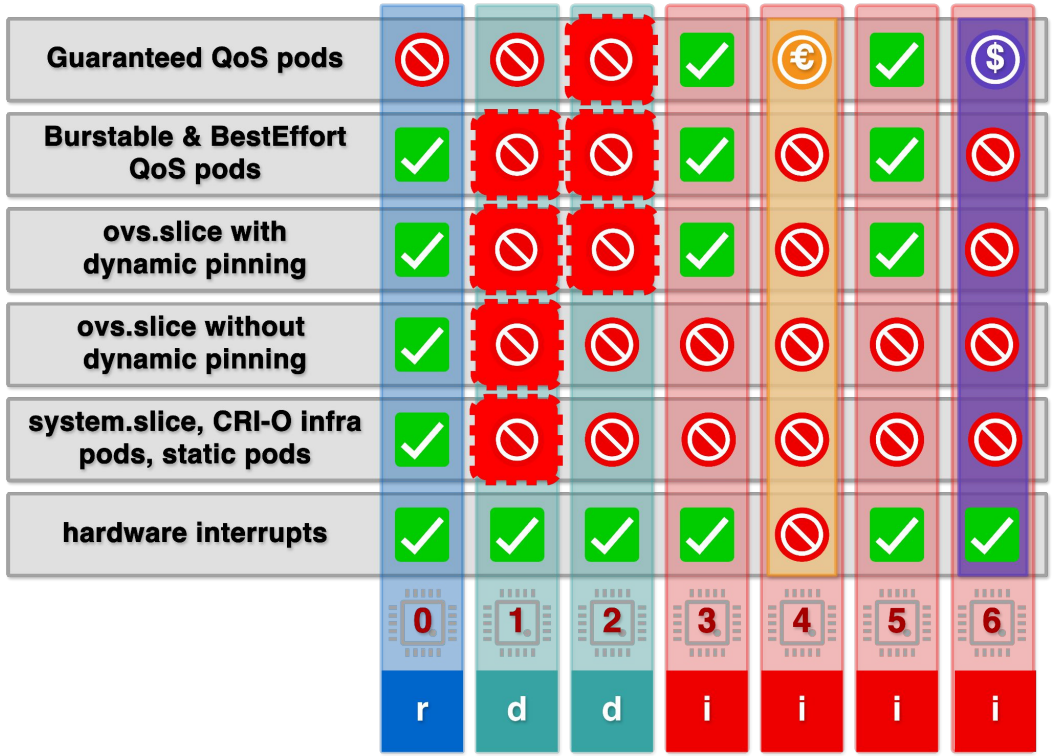
CPU affinity with Workload Partitioning (irqbalanceBanned)

New option for the PerformanceProfile API:

- *irqbalanceBanned *CPUSet*

CPU affinity with Workload Partitioning (dedicated)

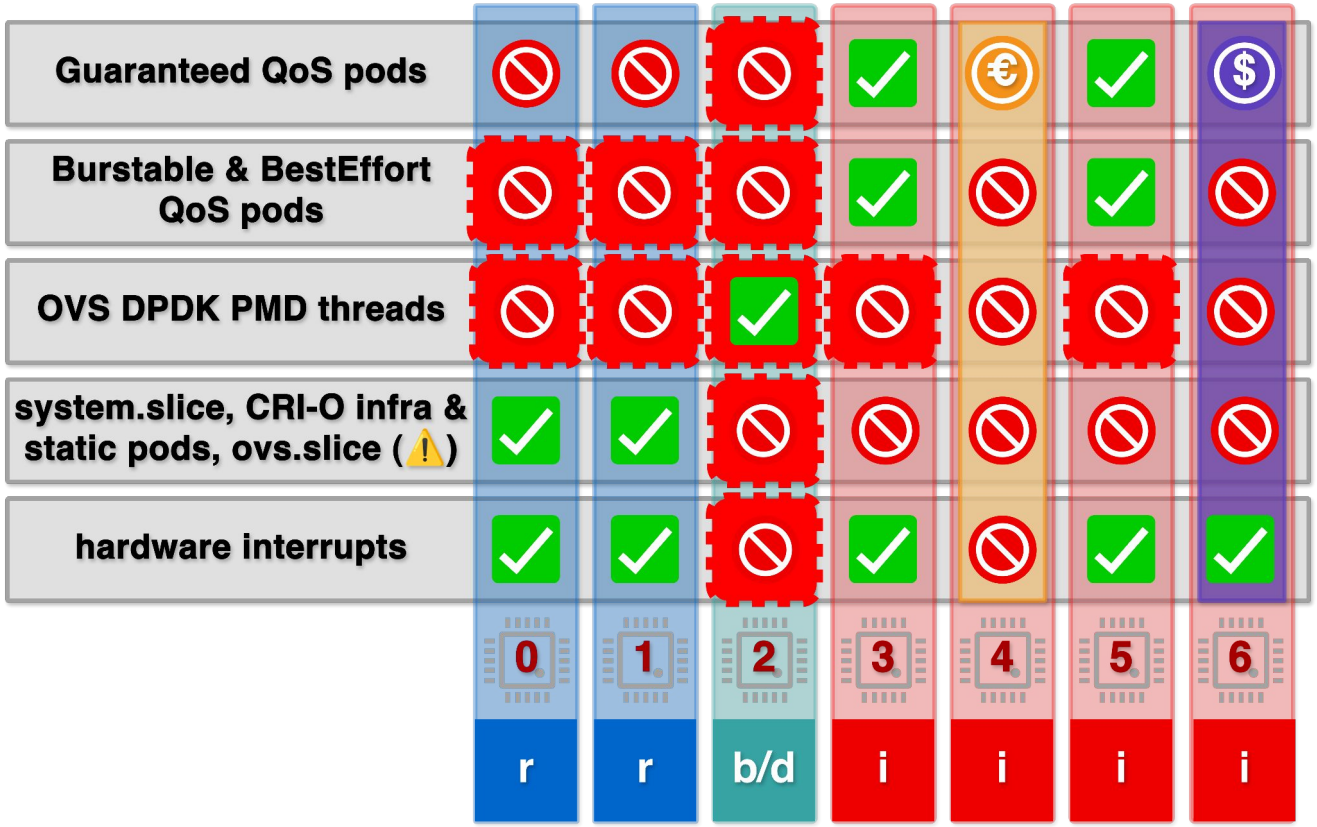
CPU affinity with Workload Partitioning (dedicated)



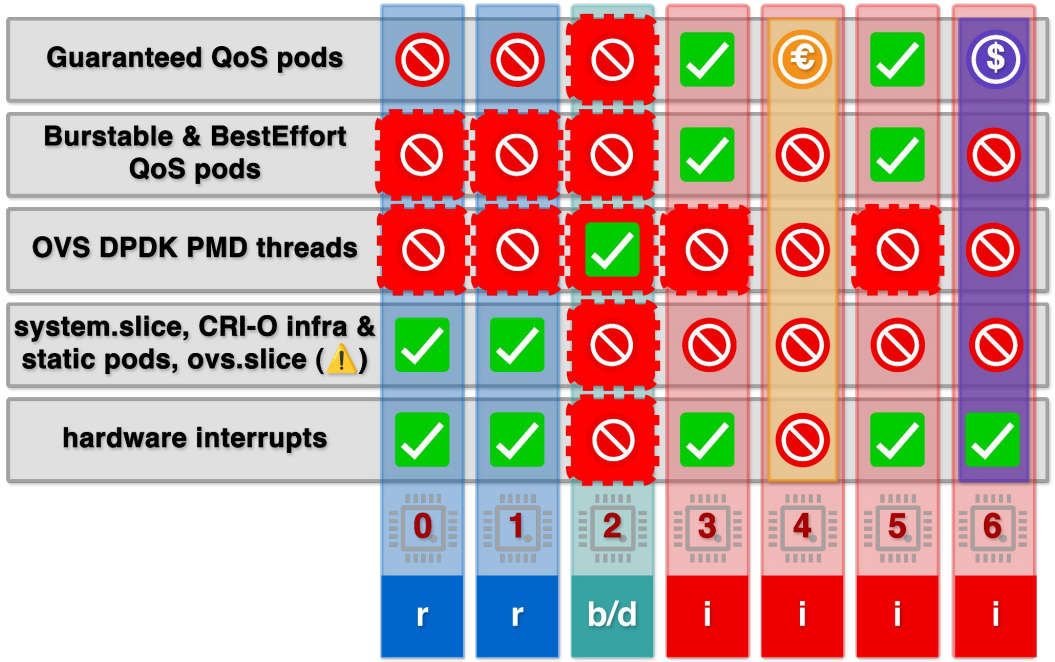
New option for the PerformanceProfile API:

- *dedicated *CPUSet*

CPU affinity with Workload Partitioning (PoC v3)



CPU affinity with Workload Partitioning (PoC v3)



CPU Manager static policy option:

- **strict-cpu-reservation=true**

New options for PerformanceProfile API:

- ***dedicated *CPUSet***
- ***irqbalanceBanned *CPUSet***
- ***disableOvsDynamicPinning: true***

MachineConfig:

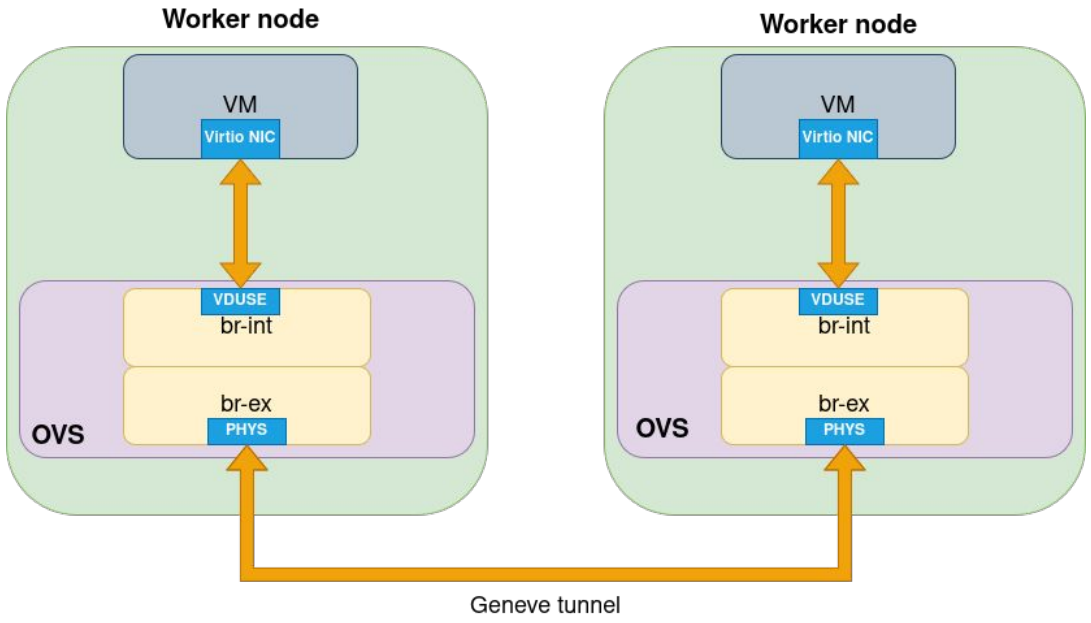
- ovs-vsctl set … other_config:pmd-cpu-mask=0x4
- Drop-ins for ovs-*.service define ***CPUAffinity=***

Benchmark results and optimizations

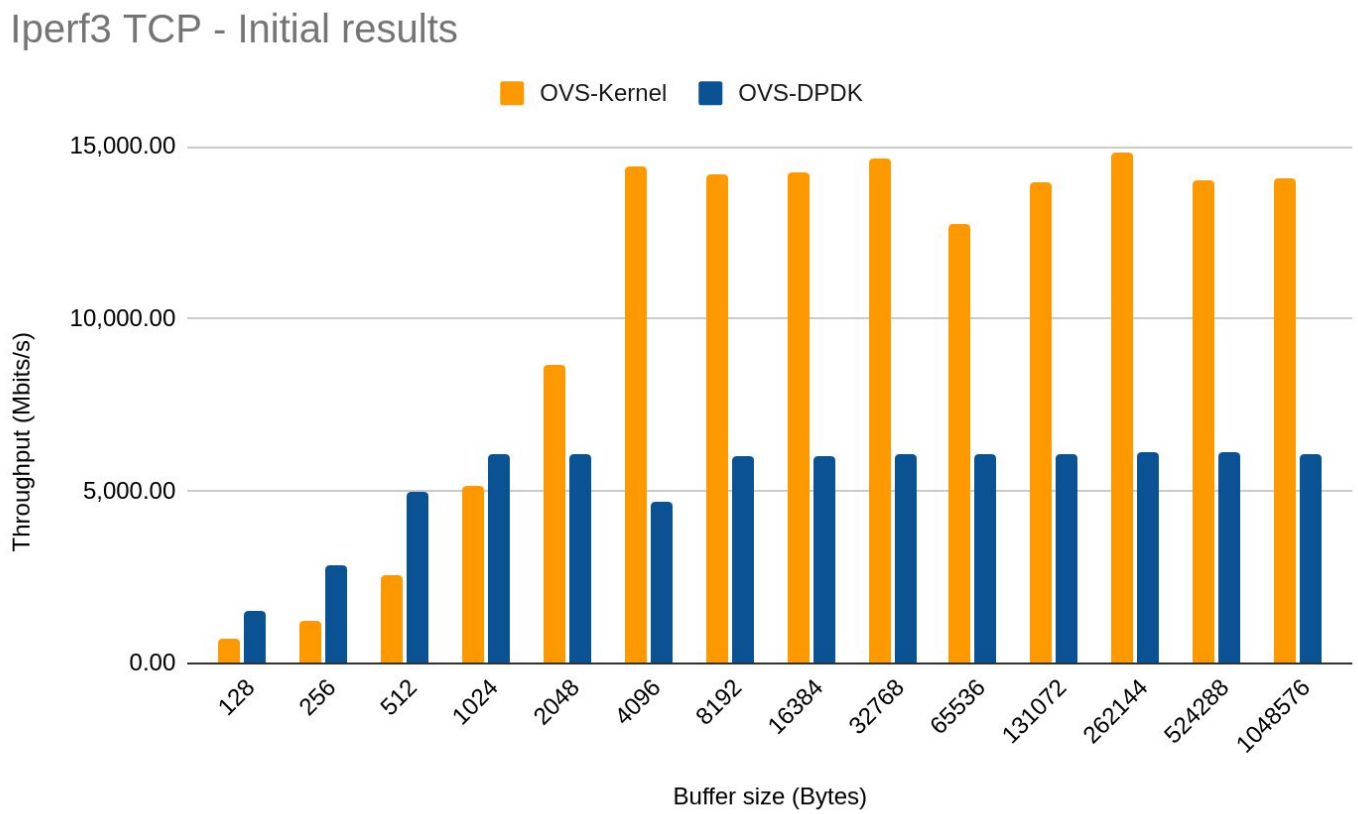
Inter-nodes VM to VM benchmarking

VM to VM performance comparison between two worker nodes:

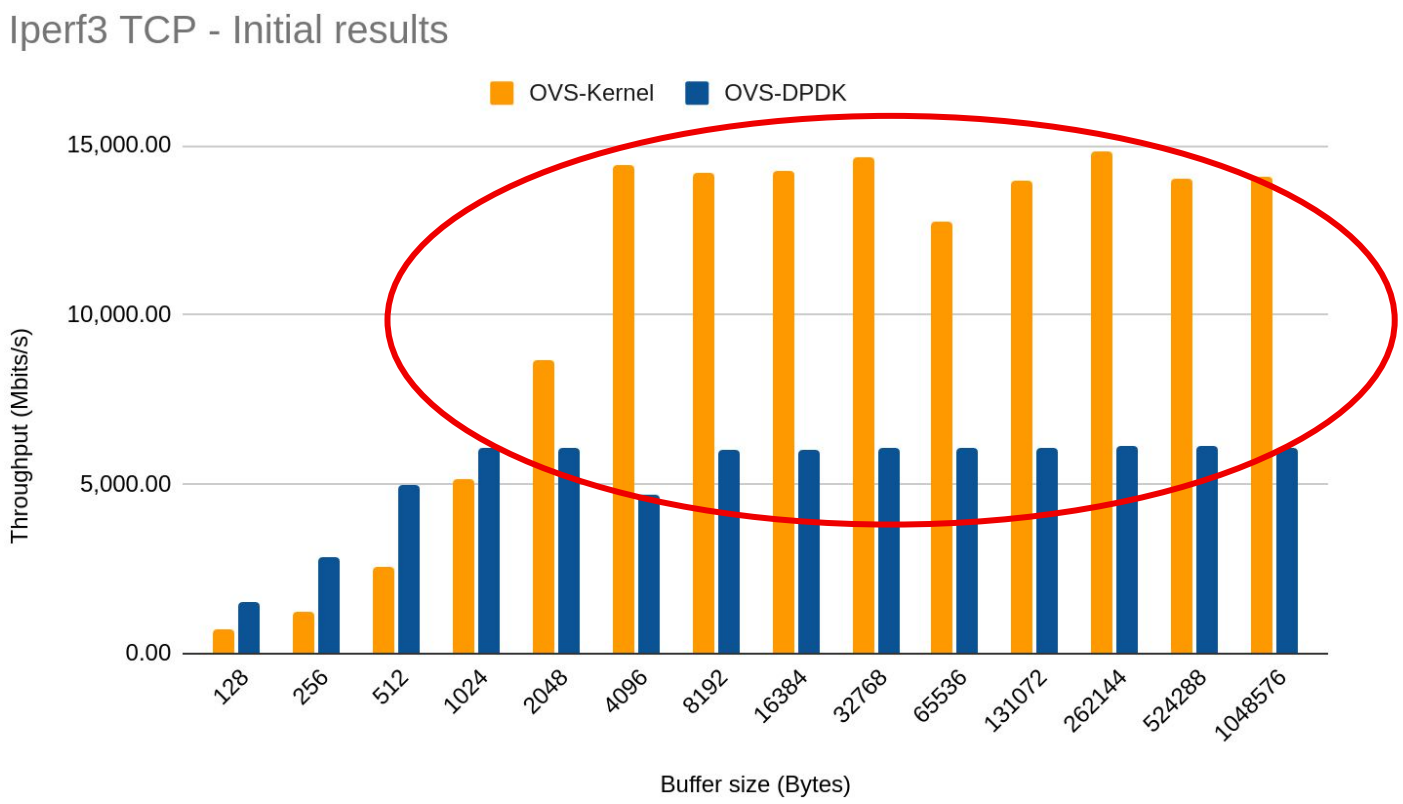
- Hardware
  - Intel(R) Xeon(R) Silver 4310
  - Nvidia ConnectX-6 Dx
- Software
  - Openshift Virtualization v4.19+
  - OVS v3.5+ (using 2MB hugepages)
- VMs
  - 2 vCPUs, 2GB memory
  - No hugepages, no vCPU pinning



Inter-node TCP benchmark



Inter-node TCP benchmark - large buffers



OVS-Kernel outperforms OVS-DPDK on larger buffers

Inter-node TCP benchmark - large buffers

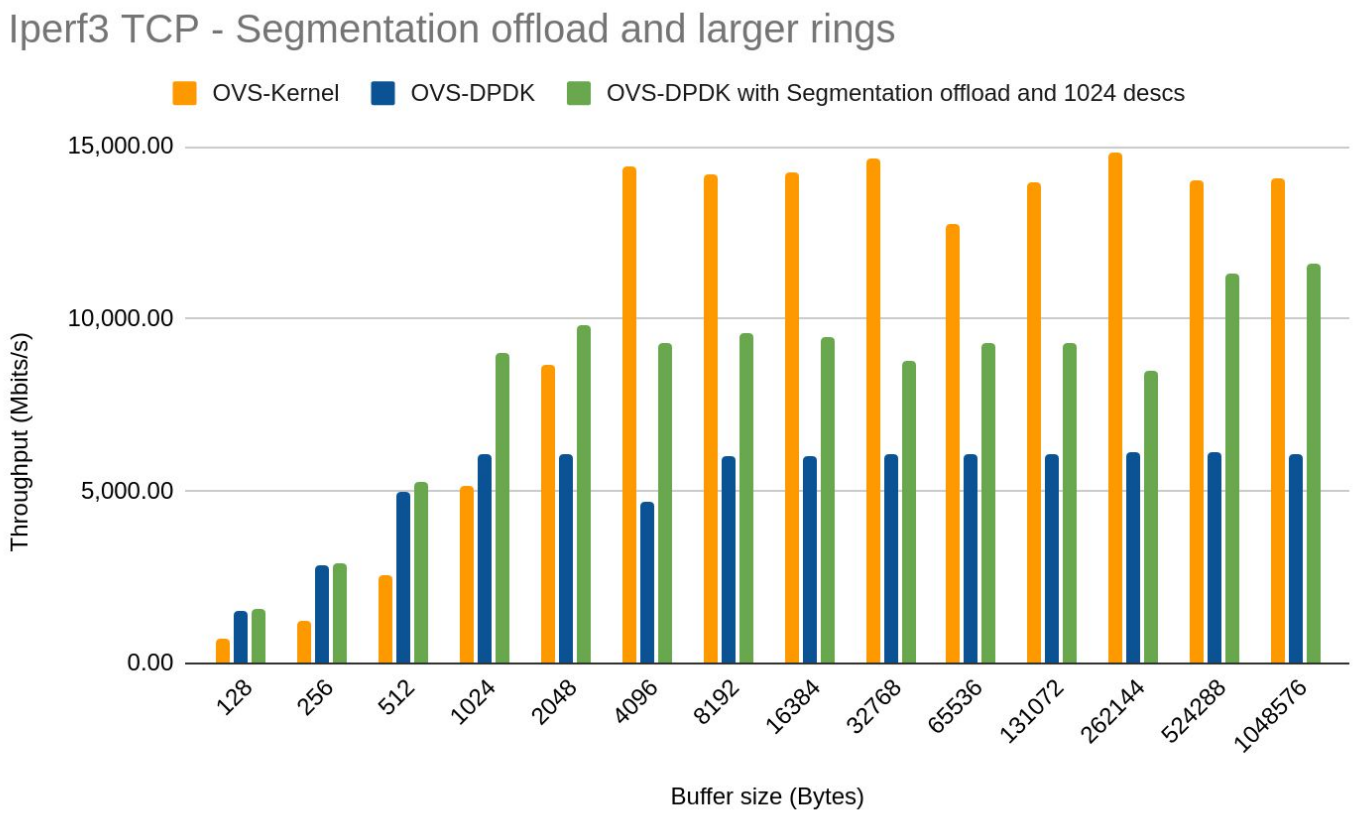
Dumping OVS coverage counters, we noticed segmentation was done in SW (***netdev_soft_seg_good***)

- **CX-6 Dx with mlx5 PMD does not support outer UDP checksum offload**
- Outer UDP optimizations by David Marchand
  - See ***Revisiting checksum offloads in OVS***

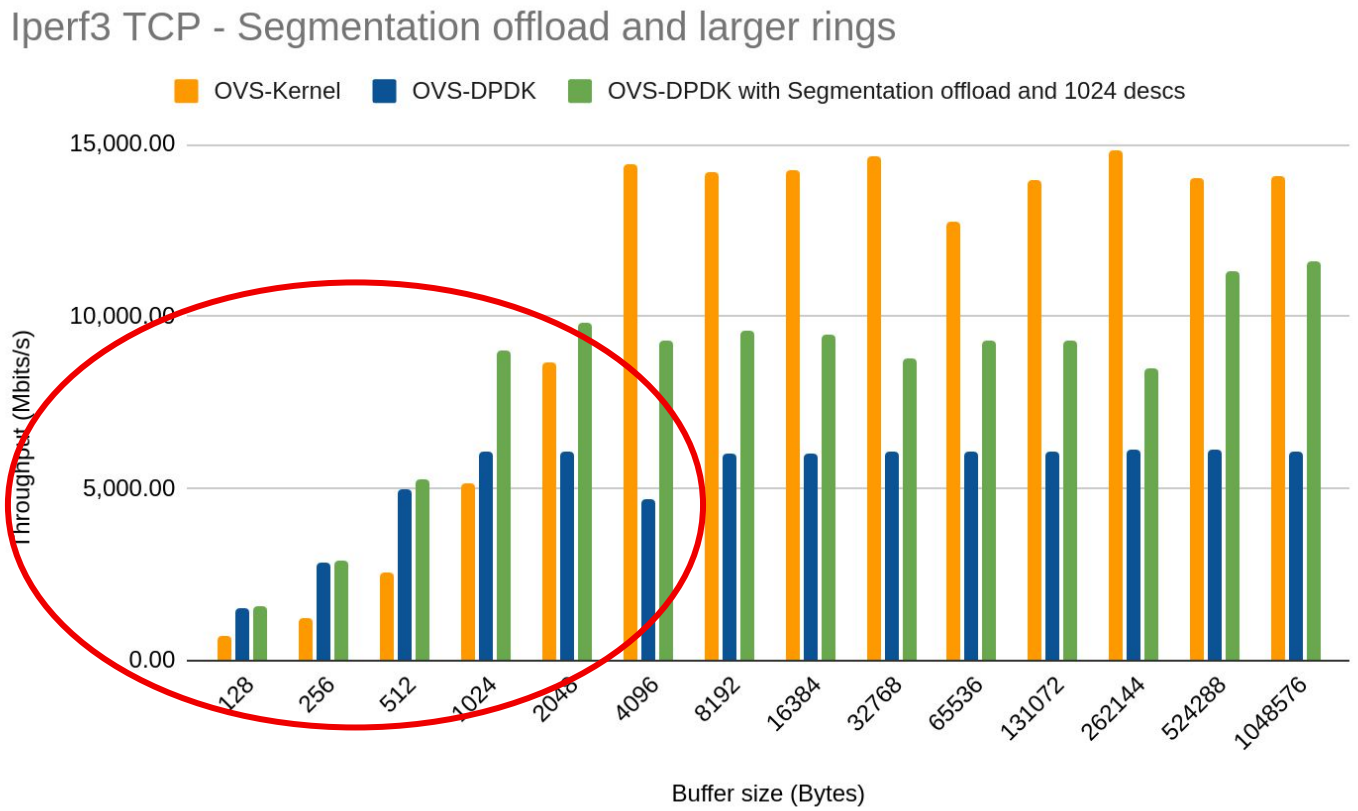
On the receiver node, we notice some Tx drops on the VDUSE port:

- The Virtio devices rings are fixed to **256** descriptors => **1024** is advised on Openstack when using OVS-DPDK
- Cannot be changed in Kubevirt for now, modifying QEMU defaults for testing purpose

Inter-node TCP benchmark

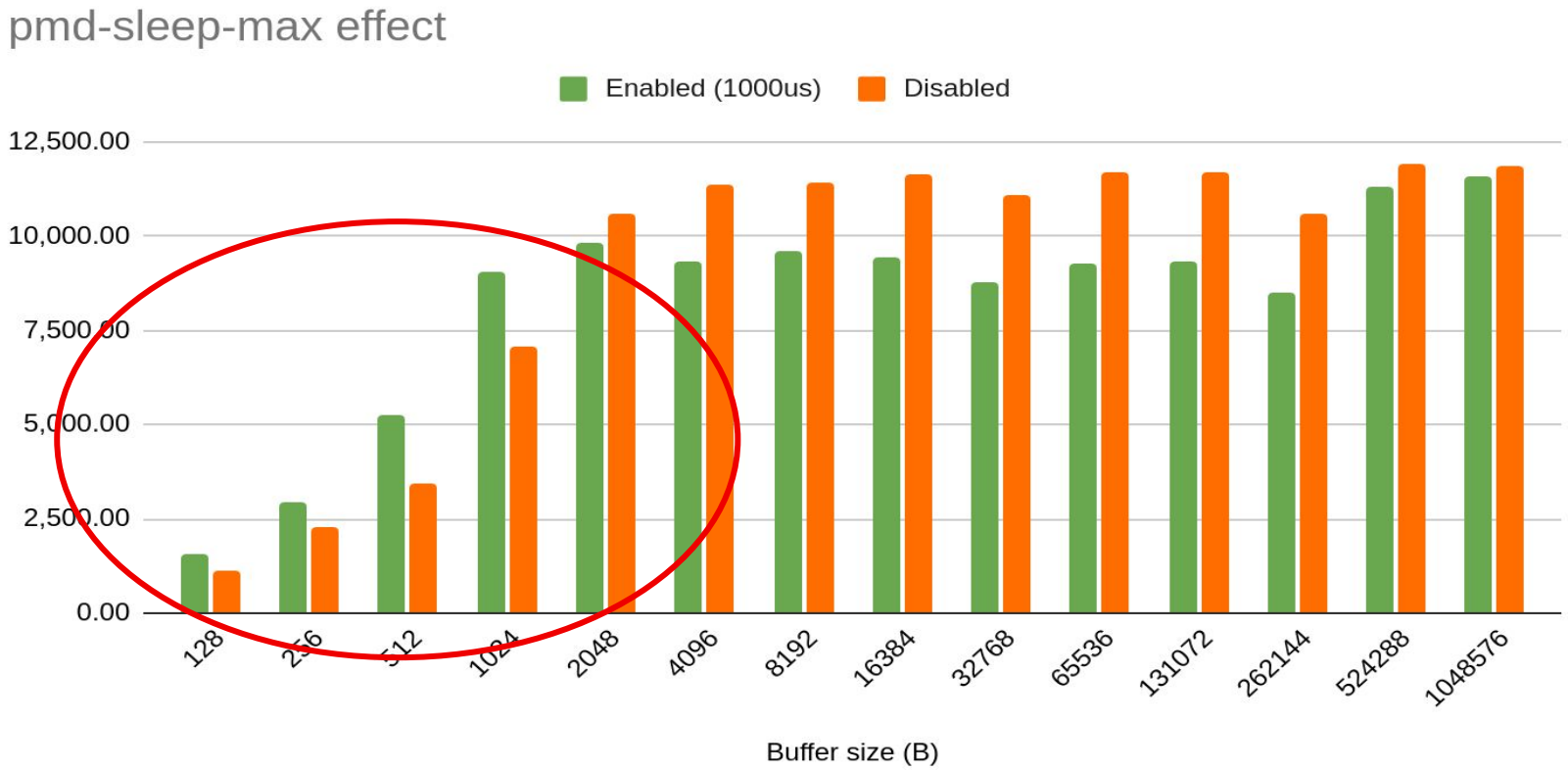


Inter-node TCP benchmark - small buffers



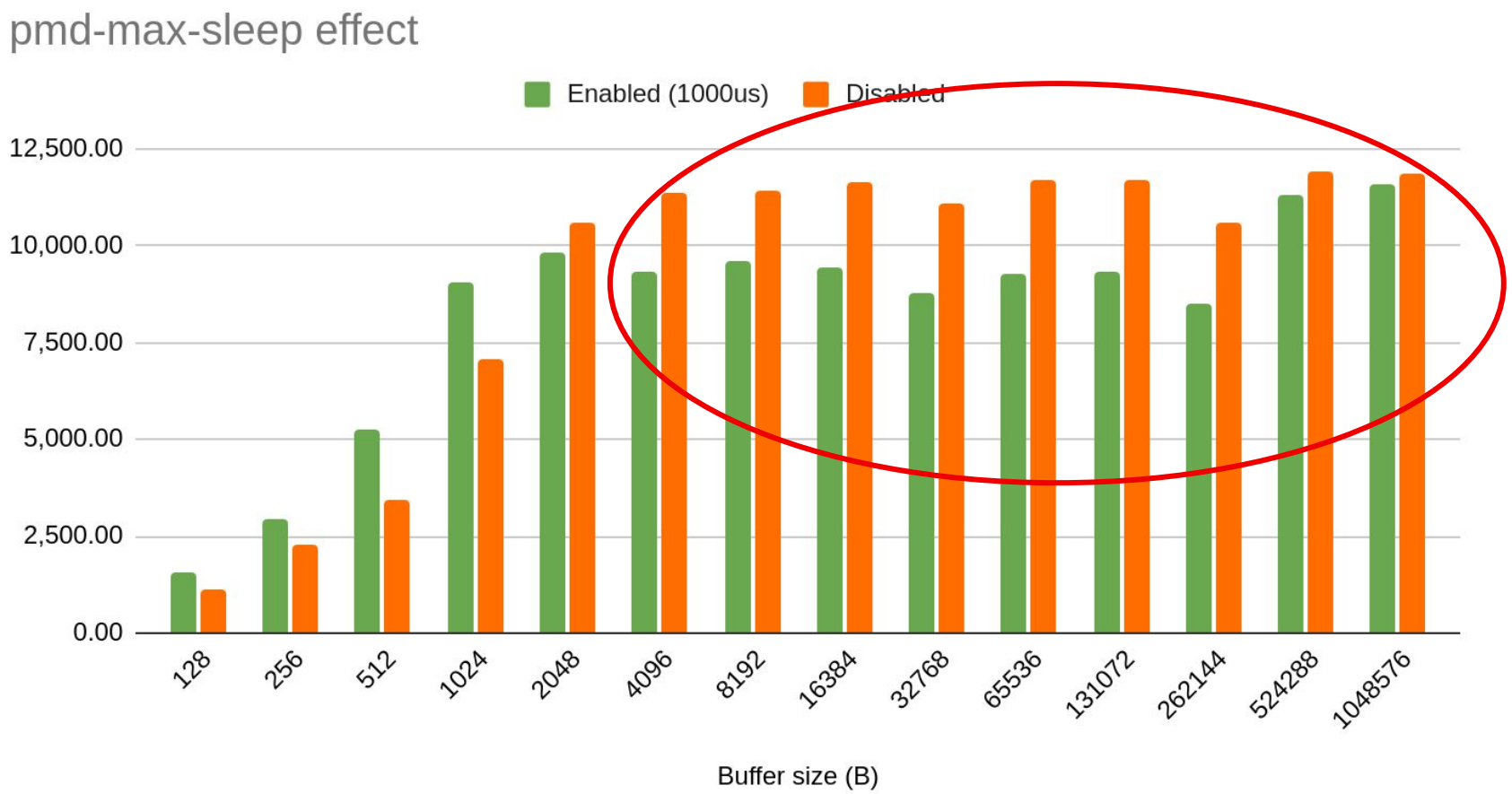
VDUSE is better, but... pmd-sleep-max option was enabled!

Inter-node TCP benchmark - pmd-sleep-max



Enabling the PMD thread to sleep improves performance on small buffers

Inter-node TCP benchmark - pmd-sleep-max



But has a negative impact on larger buffer sizes

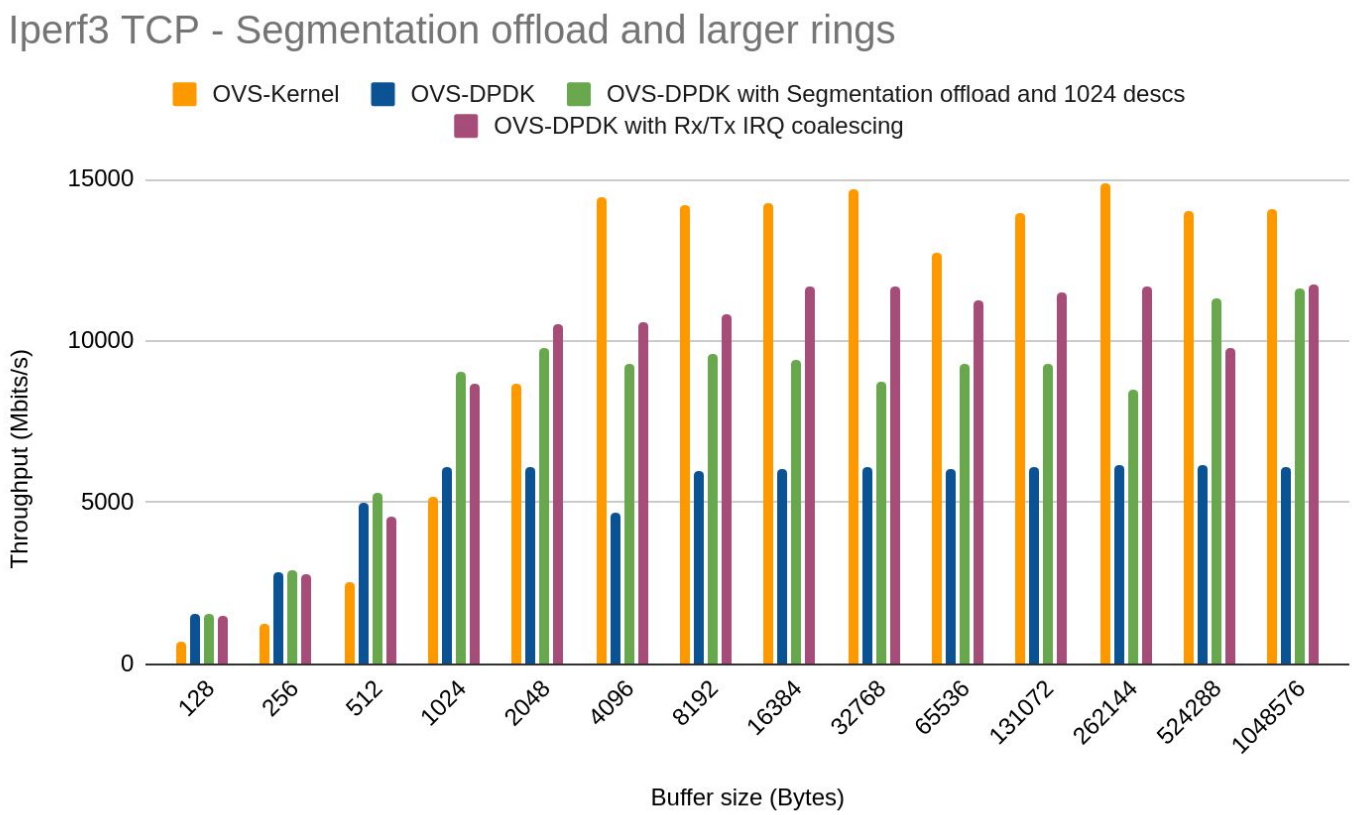
Inter-node TCP benchmark - pmd-sleep-max

- Introducing some sleeps reduces number of interrupts injected into the guest
  - **Reduces the vCPU load**
  - **Improves batching**
- For example, for iperf3 TCP with 128B buffers
  - pmd-sleep-max disabled : ~**90000 IRQs/sec**, QEMU **~200% CPU** load
  - pmd-sleep-max enabled : **~3000 IRQs/sec**, QEMU **~115% CPU** load
- **How to reduce the number of interrupts injected without introducing sleeps in the datapath?**

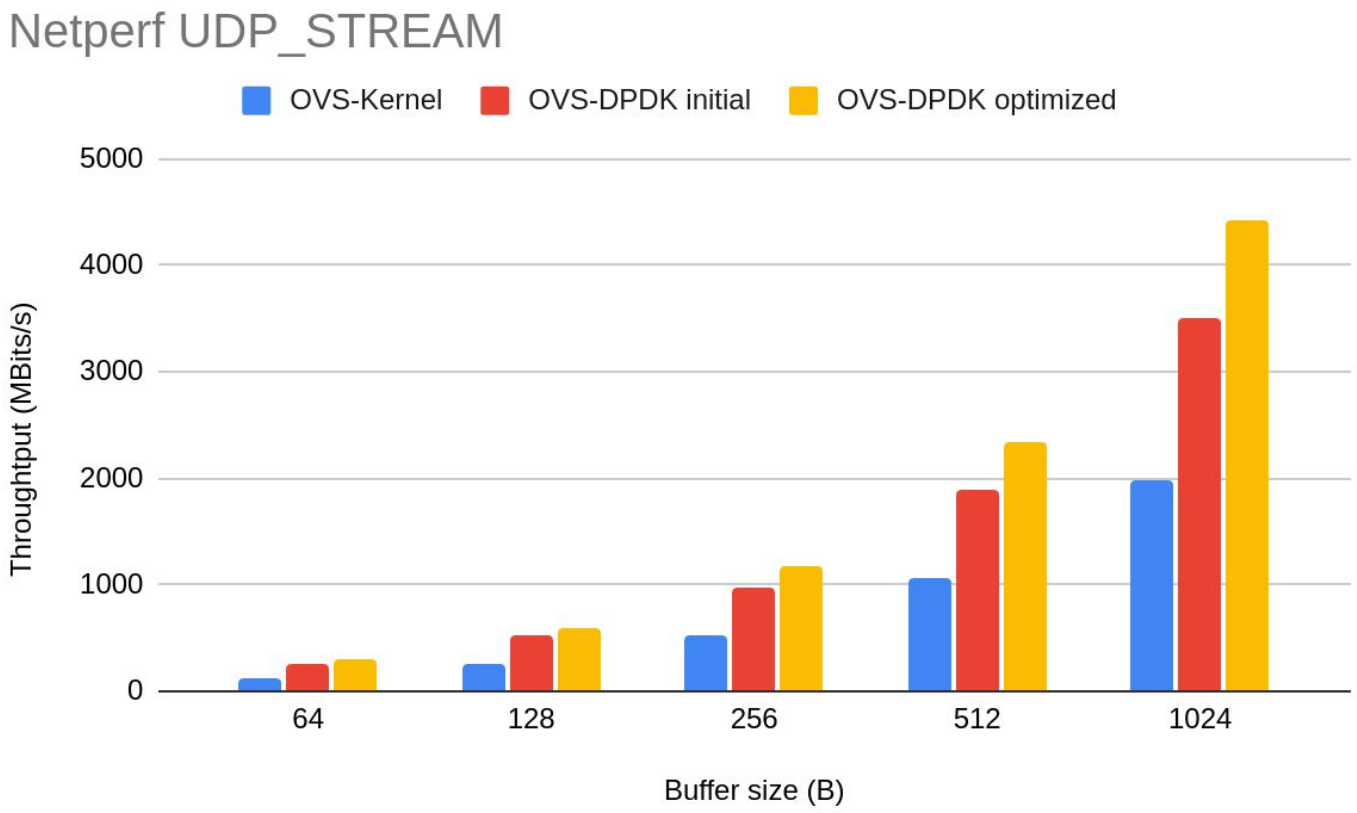
Inter-node TCP benchmark - pmd-sleep-max

- **Solution: Virtio-net IRQ coalescing feature**
  - The driver request the device to inject IRQ only every X packets or before Y usecs via the control queue
  - Supported in upstream Kernel Virtio-net driver for both Rx (auto and manual) and Tx (manual only)
  - Prototyped in DPDK Vhost library for VDUSE backend
- Advantages
  - Reduce the number of IRQs, and so the number of syscalls in the PMD thread and vCPU load
  - Improves batching
- Drawback
  - Impact on latency, but only for Rx queues (from guest PoV)

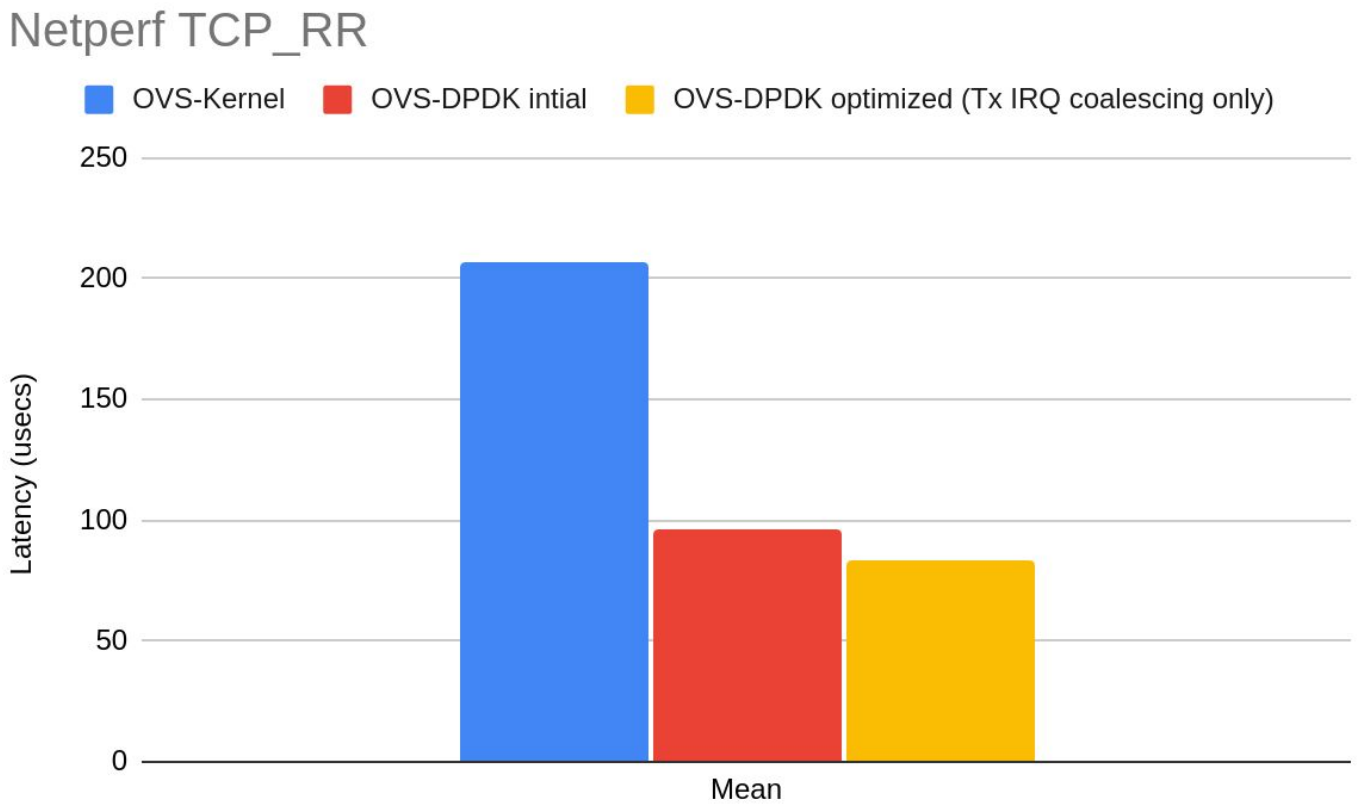
Inter-node TCP benchmark - Optimized



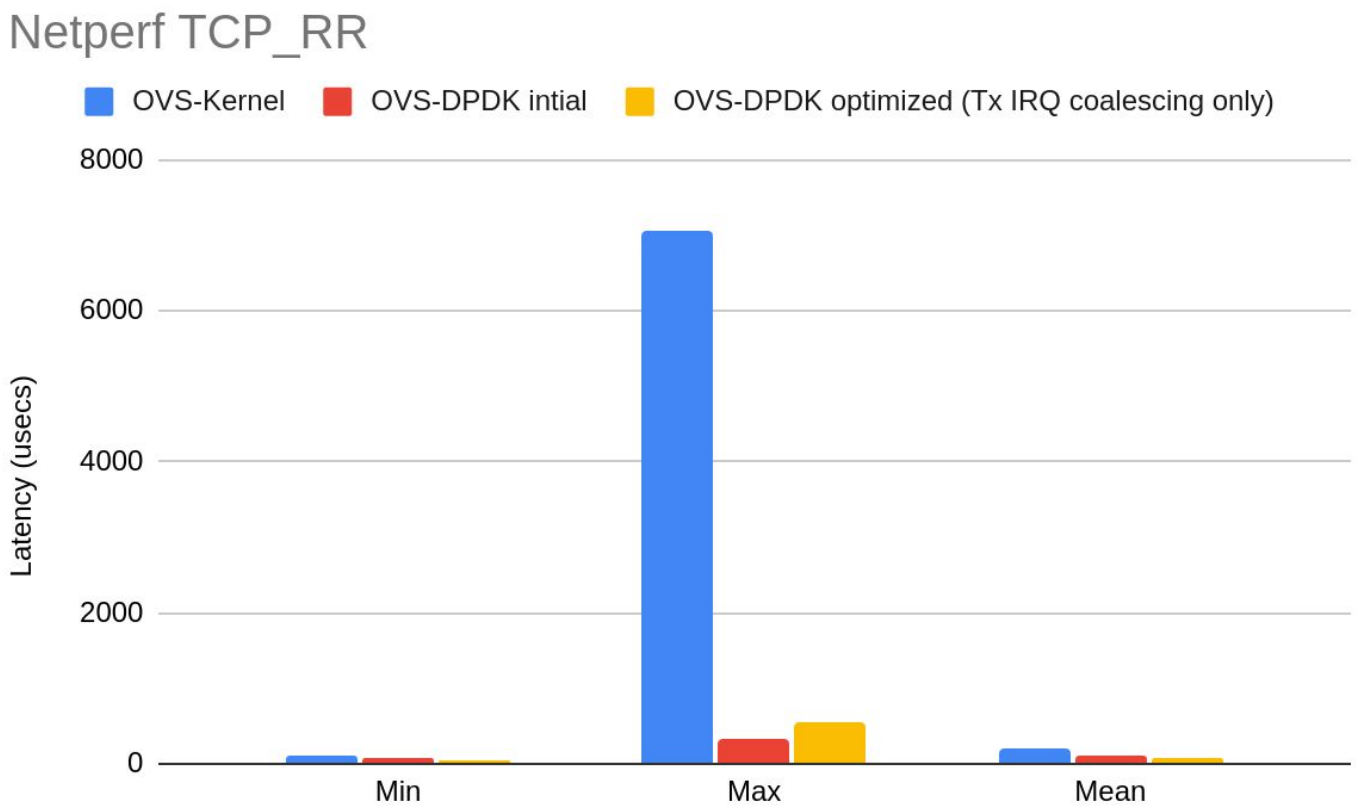
Inter-node UDP benchmark



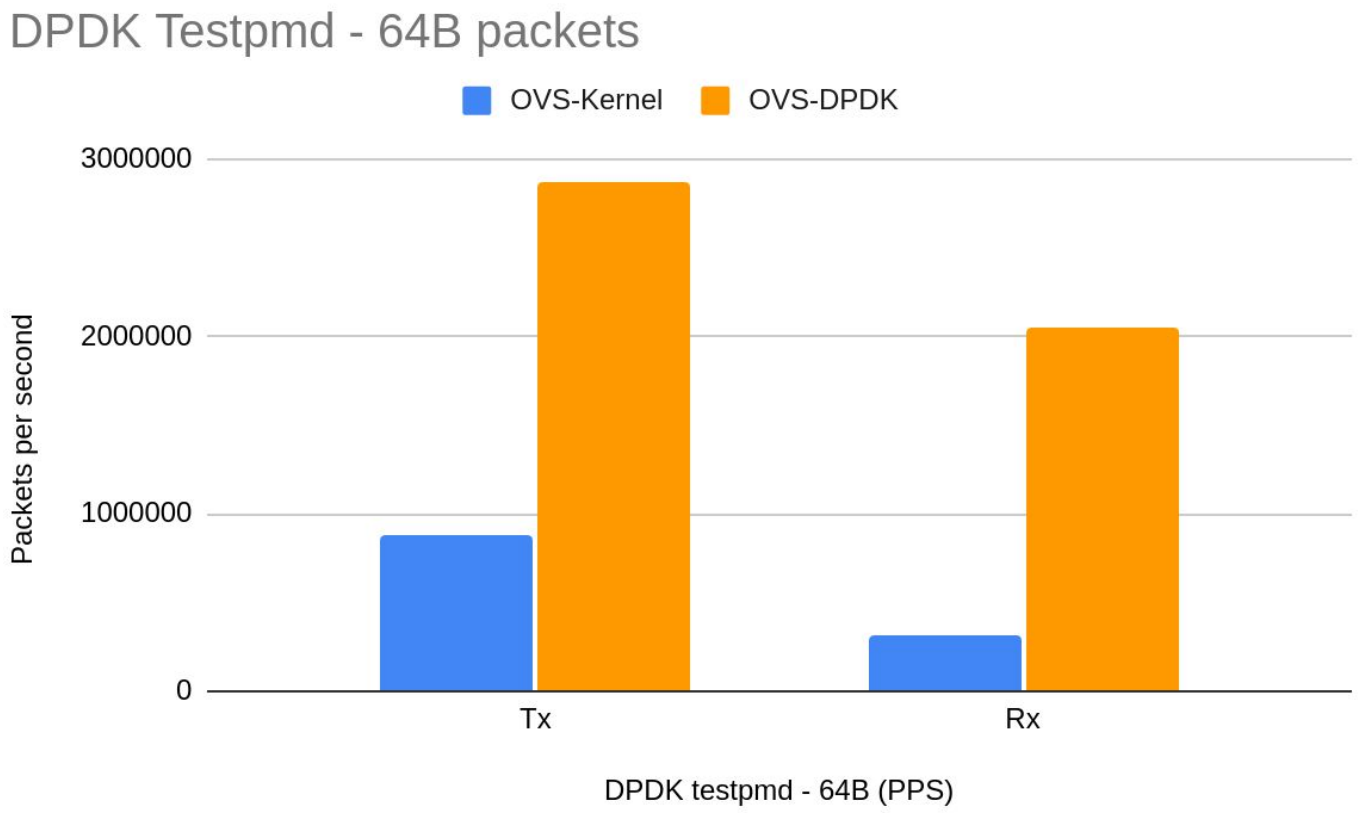
Inter-node latency



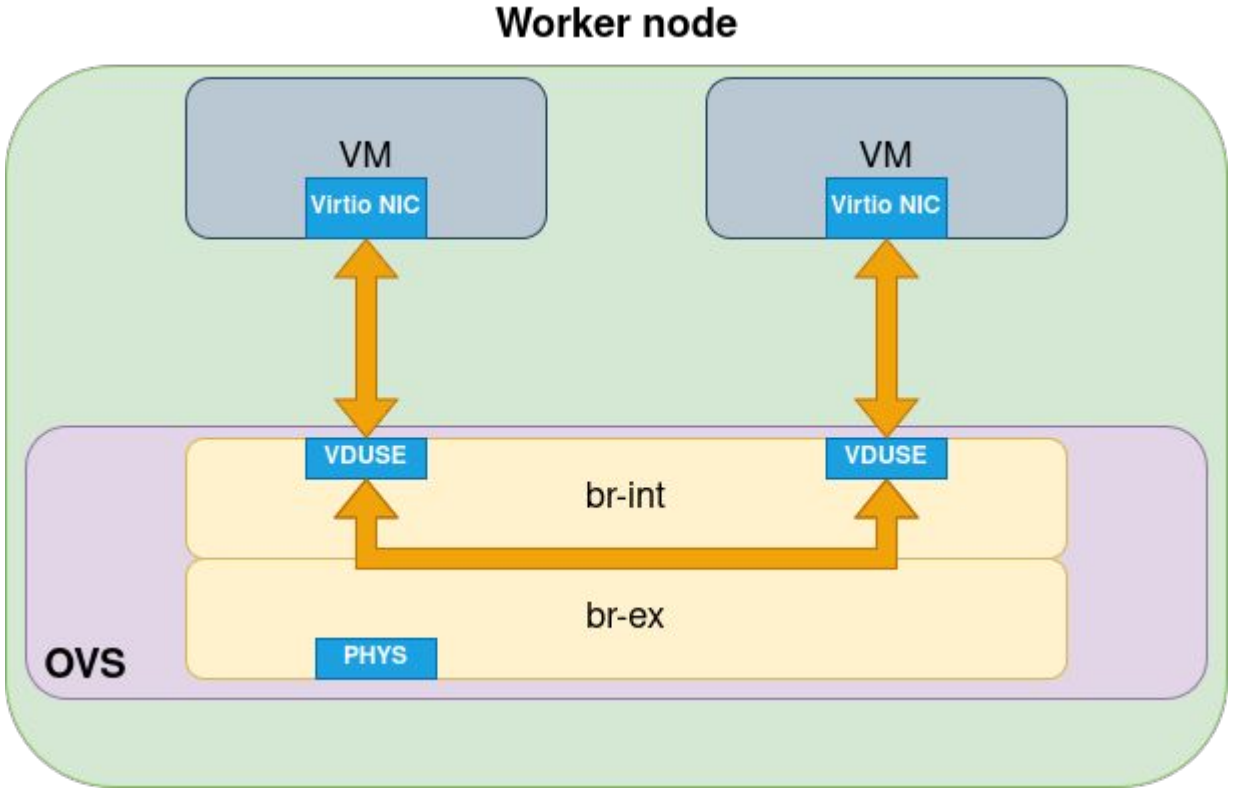
Inter-node latency



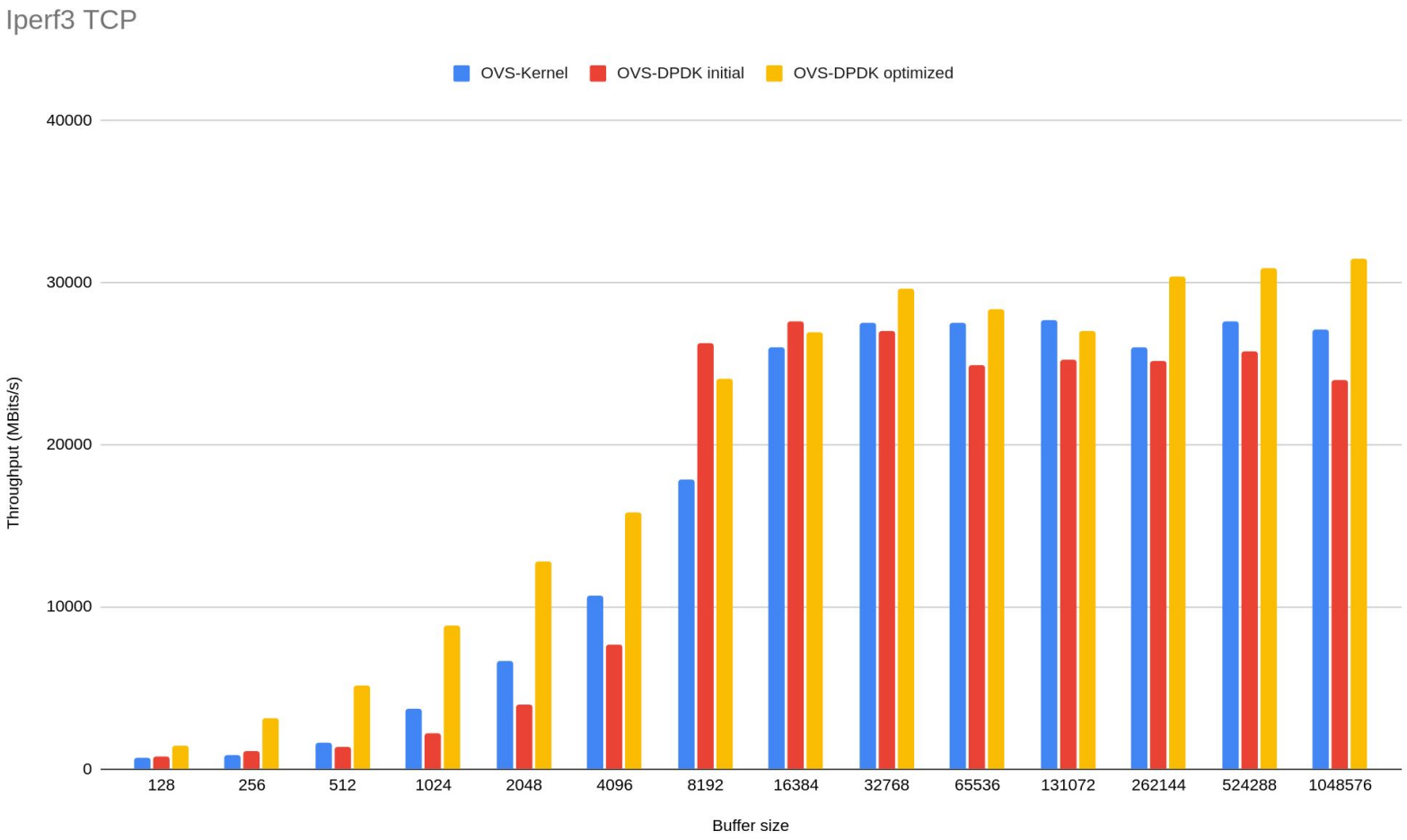
Inter-node DPDK



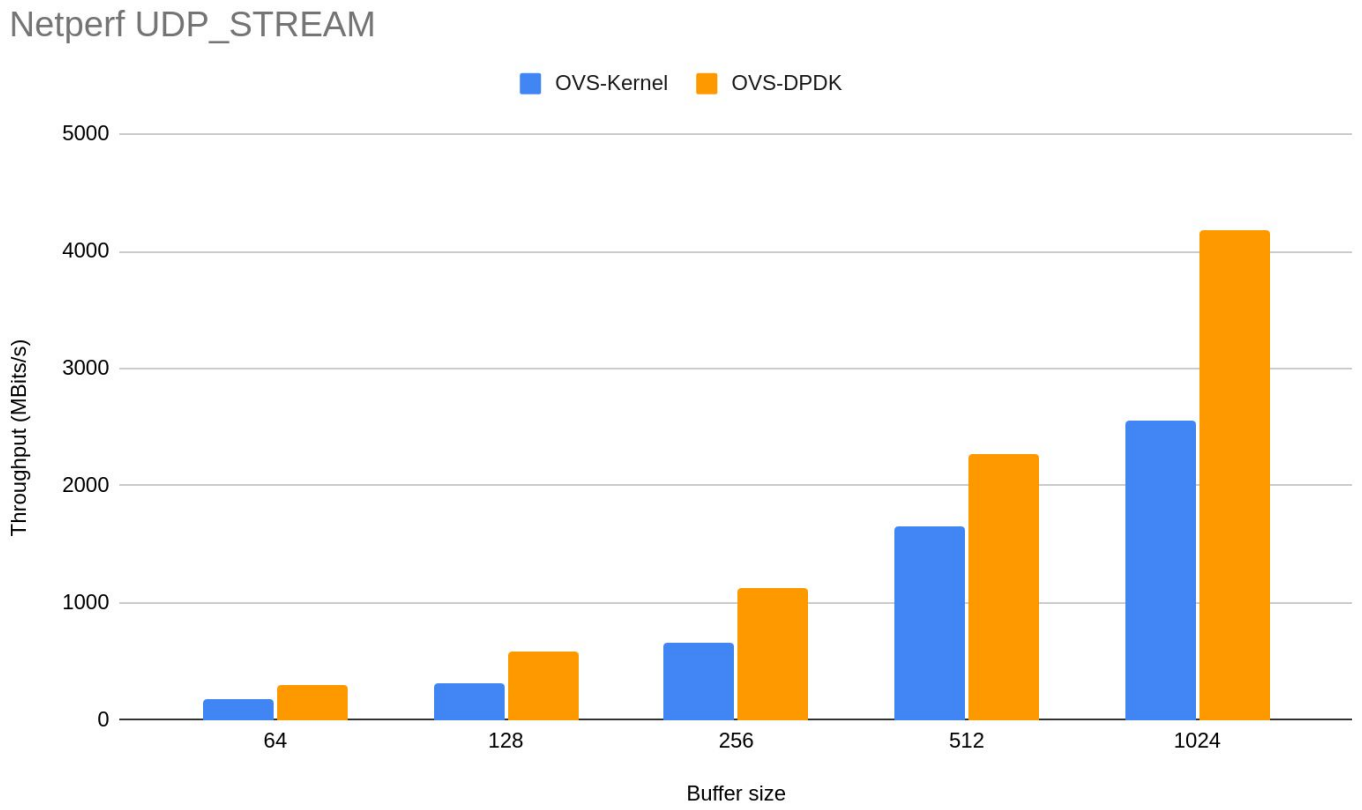
Intra-node VM to VM benchmarking



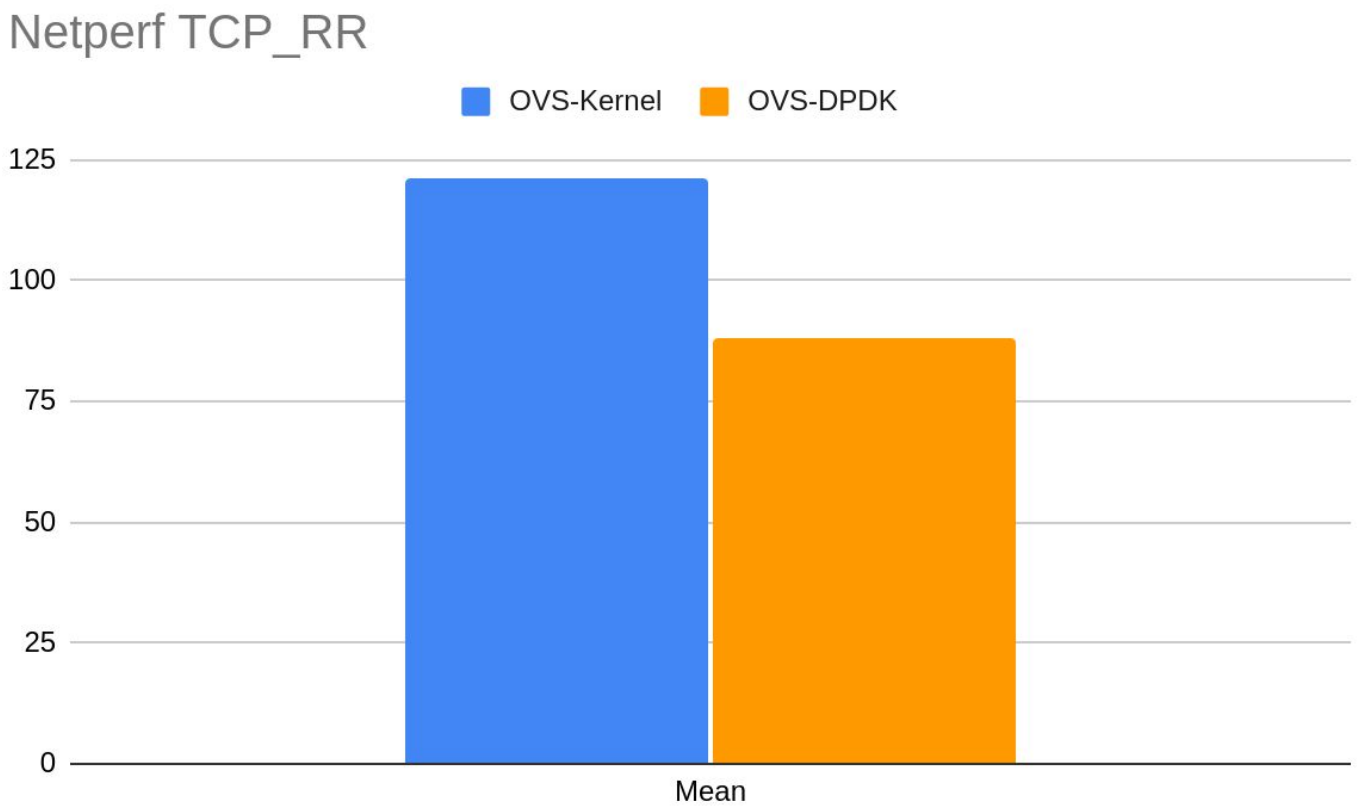
Intra-node TCP benchmark



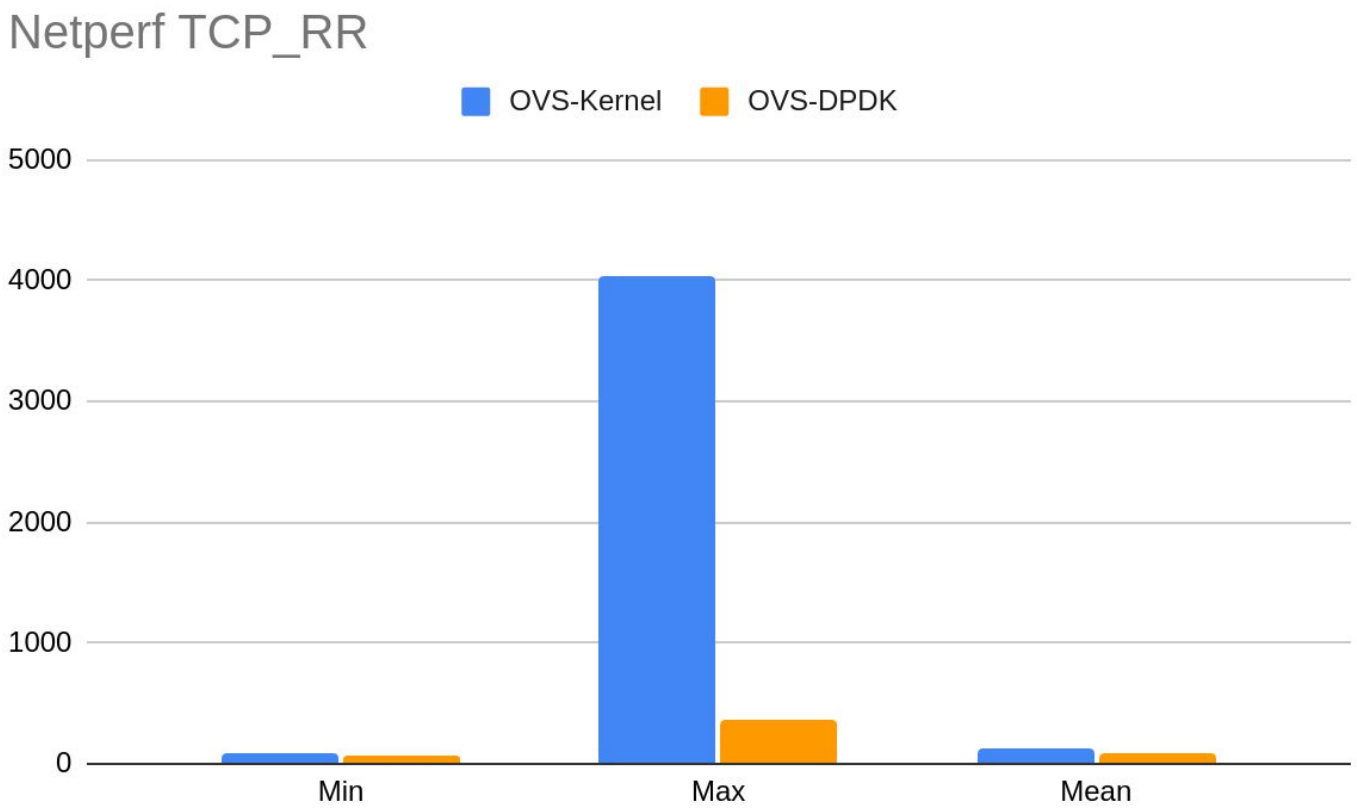
Intra-node UDP benchmark



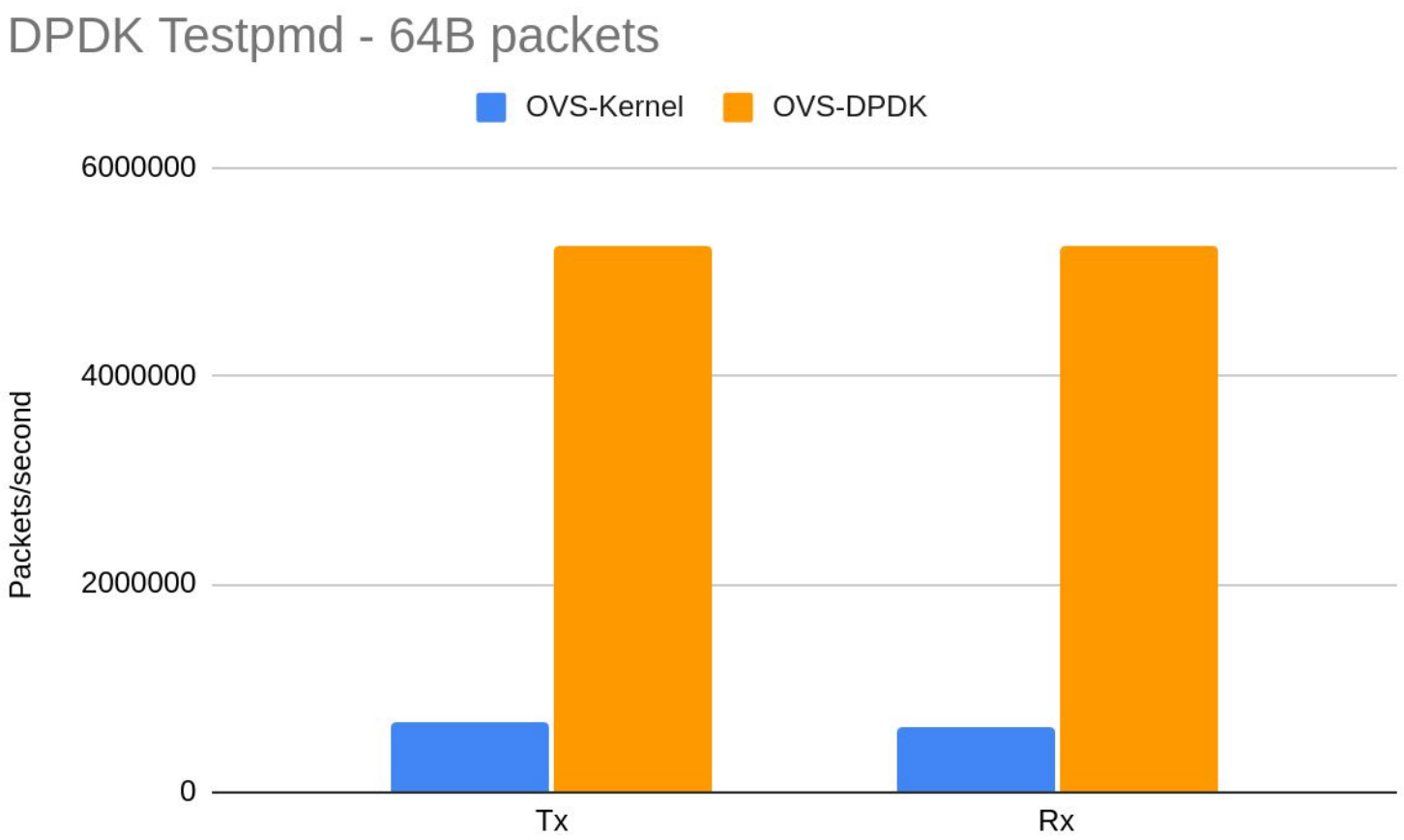
Intra-node latency benchmark



Intra-node latency benchmark



Intra-node DPDK benchmark



Conclusion

Evaluation

Pros

- VDUSE/vhost-vDPA
  - improves packet processing performance by **800%** for VM DPDK workloads
  - reduces max latency by **10x** and mean latency by **1.3x** on VM kernel interfaces
  - accelerates UDP workloads by **50%** on VM kernel interfaces
- VDUSE/virtio-vDPA improves performance for small and medium iperf3 buffer sizes
- Compute resources divided into system, network, and workload partitions
- Unified stack for primary and secondary networks

Cons

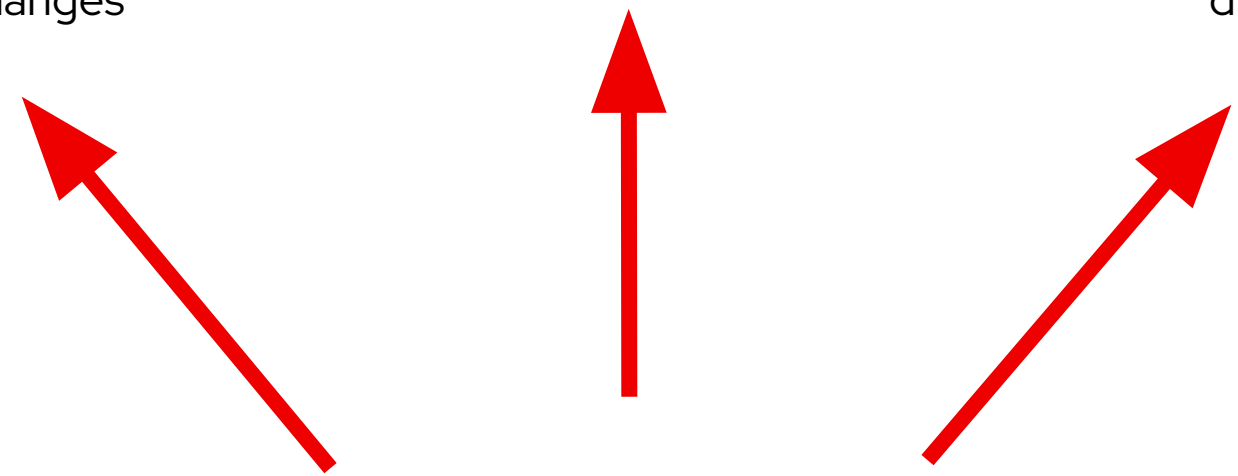
- Higher memory bandwidth usage for VDUSE/virtio-vDPA due to bounce buffer copies.
- Increased complexity, i.e. workload partitioning, hugepages, NUMA tuning. Mitigation with product integration and documentation.
- Increased CPU utilization due to PMDs. Mitigation with PMD thread load-based sleeping (increases wakeup latency).

Future work

Develop VDUSE/
vhost-vDPA CNI

Reduce tech debt,
upstream changes

Tailor solutions to
distinct markets



Thank you!

- Cindy Lu (QE)
- Yanhui Ma (QE)
- Jason Wang (Kernel)
- Benat Gartzia (KubeVirt)
- Jakob Meng (OpenShift)
- David Marchand (OVS, DPDK)
- Maxime Coquelin (OVS, DPDK)
- Eugenio Perez Martin (Kernel, QEMU)
- Leonardo Milleri (Kubernetes, KubeVirt)
- Flavio Leitner (Management, Consulting)
- Adrian Moreno Zapata (Kubernetes, KubeVirt)

> Christmas tree order <

Materials

Project

- Epic FDP-1284, PoC (v3) FDP-1286
- PoC code and docs
- OpenShift Networking Transformed: Fully Embracing DPDK Datapaths in OVN-K8s!? (Recording↗ / Slides↗ from OVS+OVN 2024)

vDPA/VDUSE

- vDPA and VDUSE Overview, Blog Posts, Presentations, ...
- Introducing VDUSE: a software-defined datapath for virtio

CPU affinity in OpenShift

- 5G Core enablement materials from Franck Baudin
- CPU affinity 201: PerformanceProfile, reserved & isolated CPUs from Franck Baudin
- Better networking pinning from Martin Sivák

Thank you


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


x.com/RedHat

Red Hat