redhat.

Can VMs networking benefit from DPDK?

Virtio/Vhost-user status & updates

Maxime Coquelin – Victor Kaplansky
2017-01-27

AGENDA

Can VMs networking benefit from DPDK?

- Overview
- Challenges
- New & upcoming features

Overview

DPDK – project overview

Overview

DPDK is a set of userspace libraries aimed at fast packet processing.

- Data Plane Development Kit
- **Goal:**
  - Benefit from software flexibility
  - Achieving performance close to dedicated HW solutions

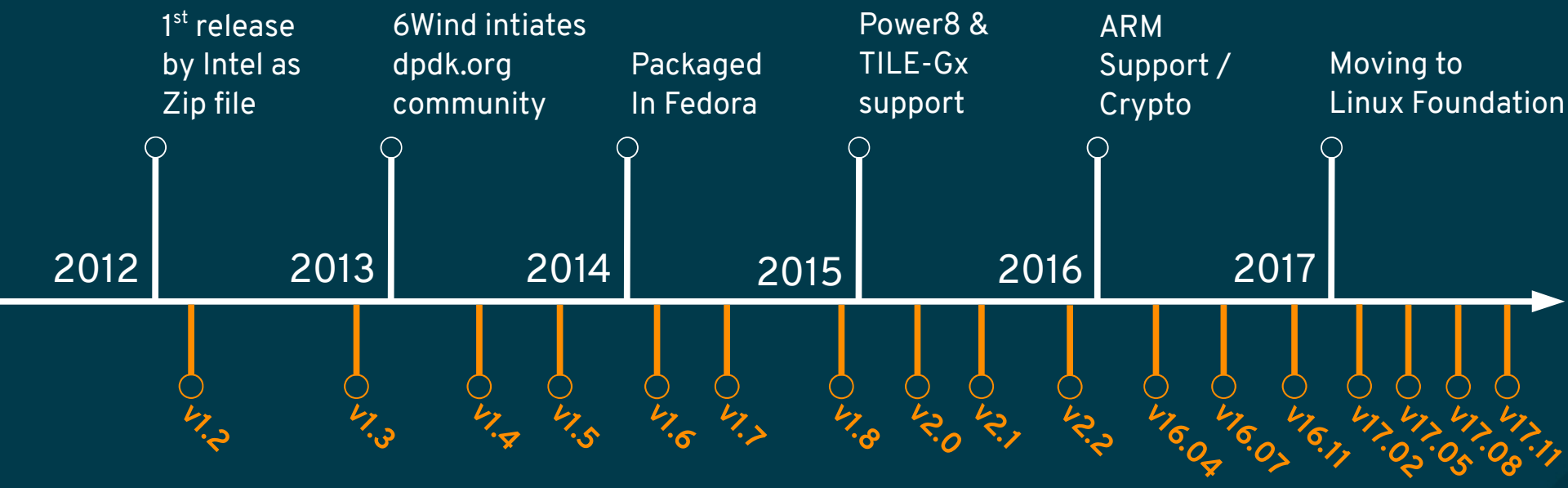
DPDK – project overview

Overview

- **License:** BSD
- **CPU architectures:** x86, Power8, TILE-Gx & ARM
- **NICs:** Intel, Mellanox, Broadcom, Cisco,...
- **Other HW:** Crypto, SCSI for SPDK project
- **Operating systems:** Linux, BSD

DPDK - project history

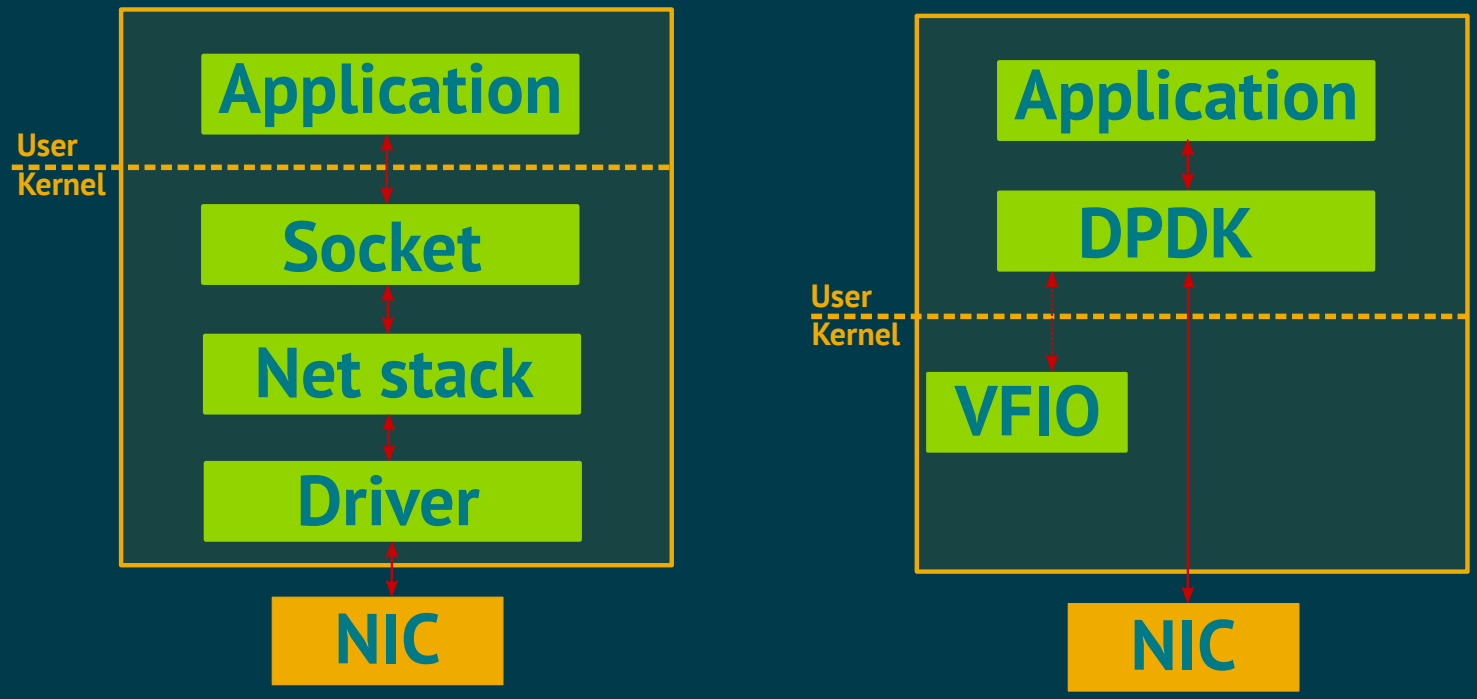
Overview



v16.11: ~750K LoC / ~6000 commits / ~350 contributors

DPDK – comparison

Overview



DPDK – performance

Overview

DPDK uses:

- **CPU isolation/partitioning & polling**
  - → Dedicated CPU cores to poll the device
- **VFIO/UIO**
  - → Direct devices registers accesses from user-space
- **NUMA awareness**
  - → Resources local to the Poll-Mode Driver's (PMD) CPU
- **Hugepages**
  - → Less TLB misses, no swap

DPDK – performance

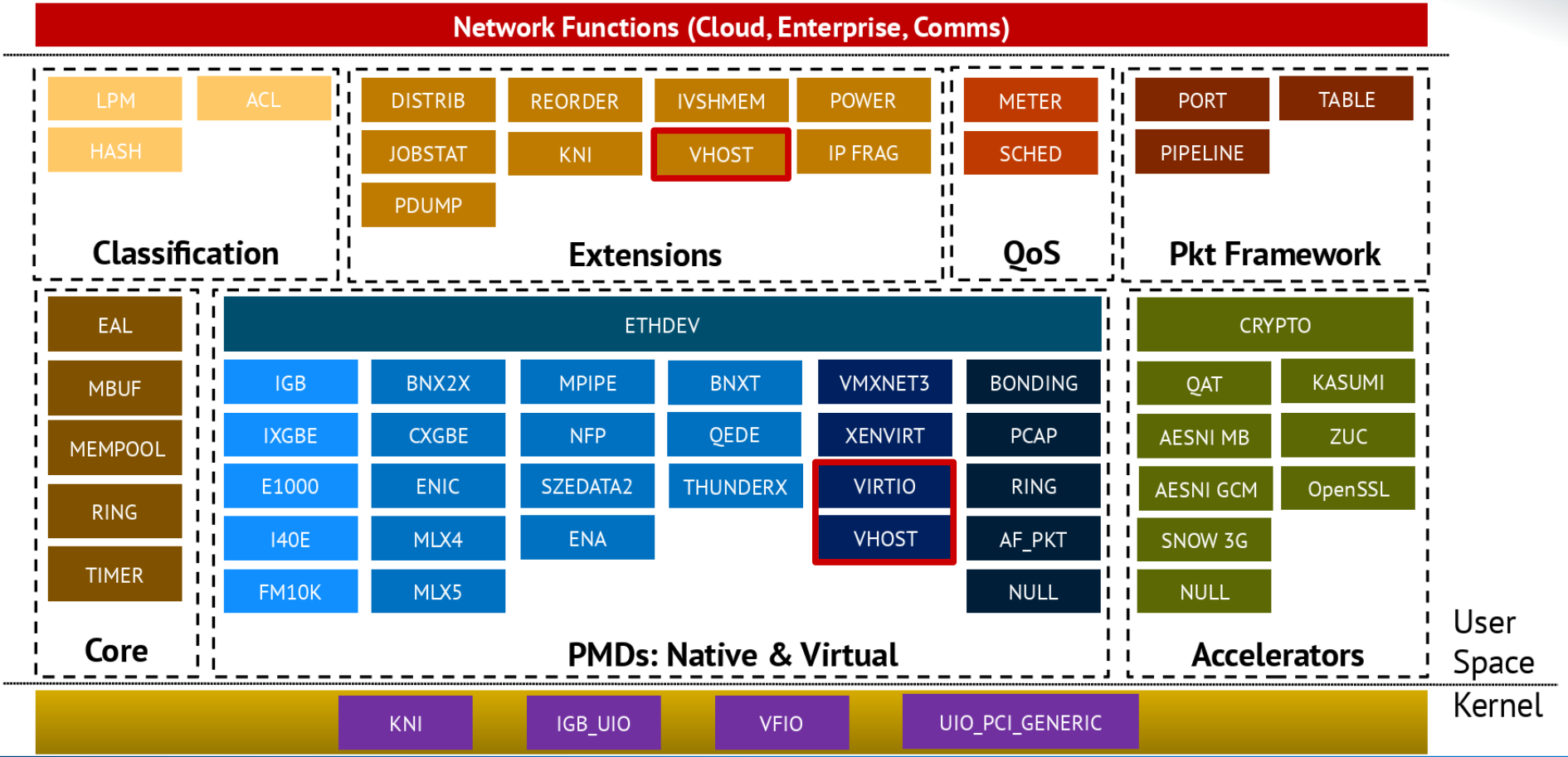
Overview

To avoid:

- **Interrupt handling**
  → Kernel's NAPI polling mode is not enough
- **Context switching**
- **Kernel/user data copies**
- **Syscalls overhead**
  → More than the time budget for a 64B packet at 14.88Mpps

DPDK - components

Overview



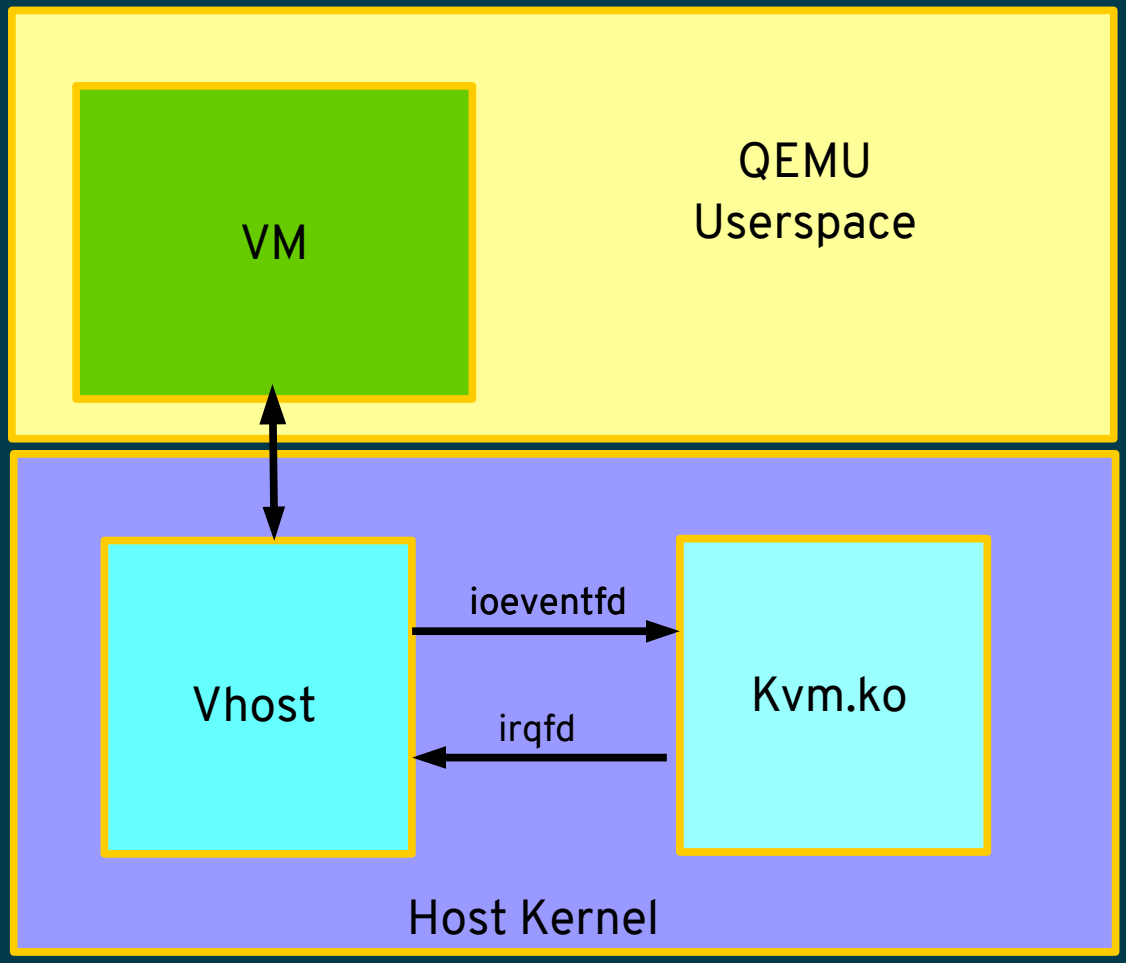
Credits: Tim O'Driscoll - Intel

Virtio/Vhost

Overview

- Device emulation, direct assignment, VirtIO
- Vhost: In-kernel virtio device emulation
- device emulation code calls to directly call into kernel subsystems
- Vhost worker thread in host kernel
- Bypasses system calls from user to kernel space on host

Vhost driver model



In-kernel device emulation

Overview

- In-kernel restricted to virtqueue emulation
- QEMU handles control plane, feature negotiation, migration, etc
- File descriptor polling done by vhost in kernel
- Buffers moved between tap device and virtqueues by kernel worker thread

Vhost as user space interface

Overview

- Vhost architecture is not tied to KVM
- Backend: Vhost instance in user space
- Eventfd is set up to signal backend when new buffers are placed by guest (kickfd)
- Irqfd is set up to signal the guest about new buffers placed by backend (callfd)
- The beauty: backend only knows about guest memory mapping, kick eventfd and call eventfd
- Vhost-user implemented in DPDK in v16.7

Challenges

Performance

Challenges

DPDK is about **performance**, which is a trade-off between:

- Bandwidth → achieving line rate even for small packets
- Latency → as low as possible (of course)
- CPU utilization → $$$

→ Prefer bandwidth & latency at the expense of CPU utilization

→ Take into account HW architectures as much as possible

Reliability

Challenges

0% packet-loss

- Some use-cases of Virtio cannot afford packet loss, like NFV
- Hard to achieve max perf without loss, as Virtio is CPU intensive

  → Scheduling “glitches” may cause packets drop

Migration

- Requires restoration of internal state including the backend
- Interface exposed by QEMU must stay unchanged for cross-version migration
- Interface exposed to guest depends on capabilities of third-party application
- Support from the management tool is required

Security

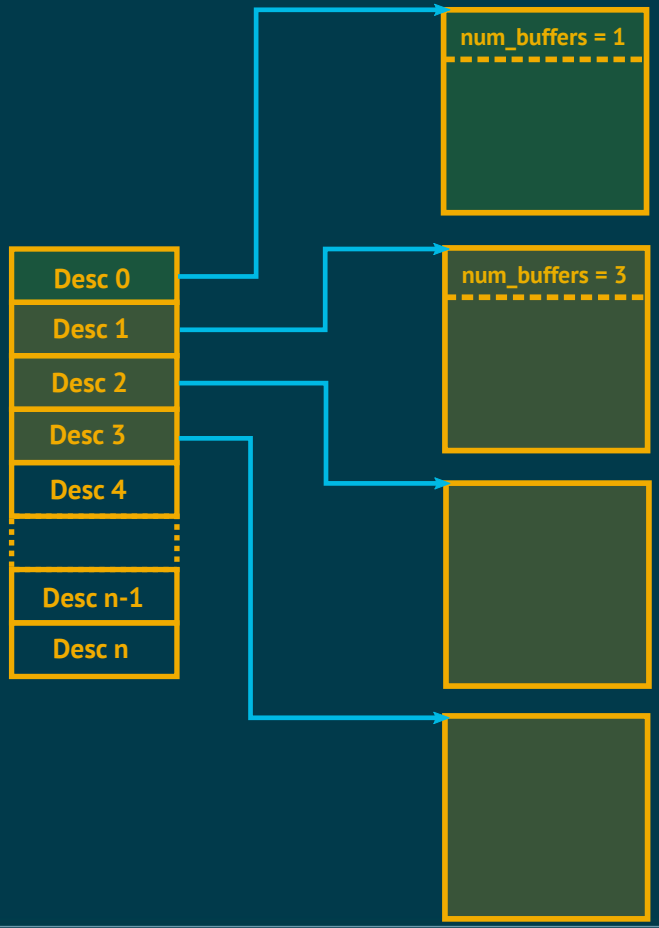
Challenges

- Isolation of untrusted guests
- Direct access to device from untrusted guests
- Current implementations require mediator for guest -to-guest communication.
- Zero-copy is problematic from security point of view

New & upcoming features

Rx mergeable buffers

New & upcoming features



Pro:

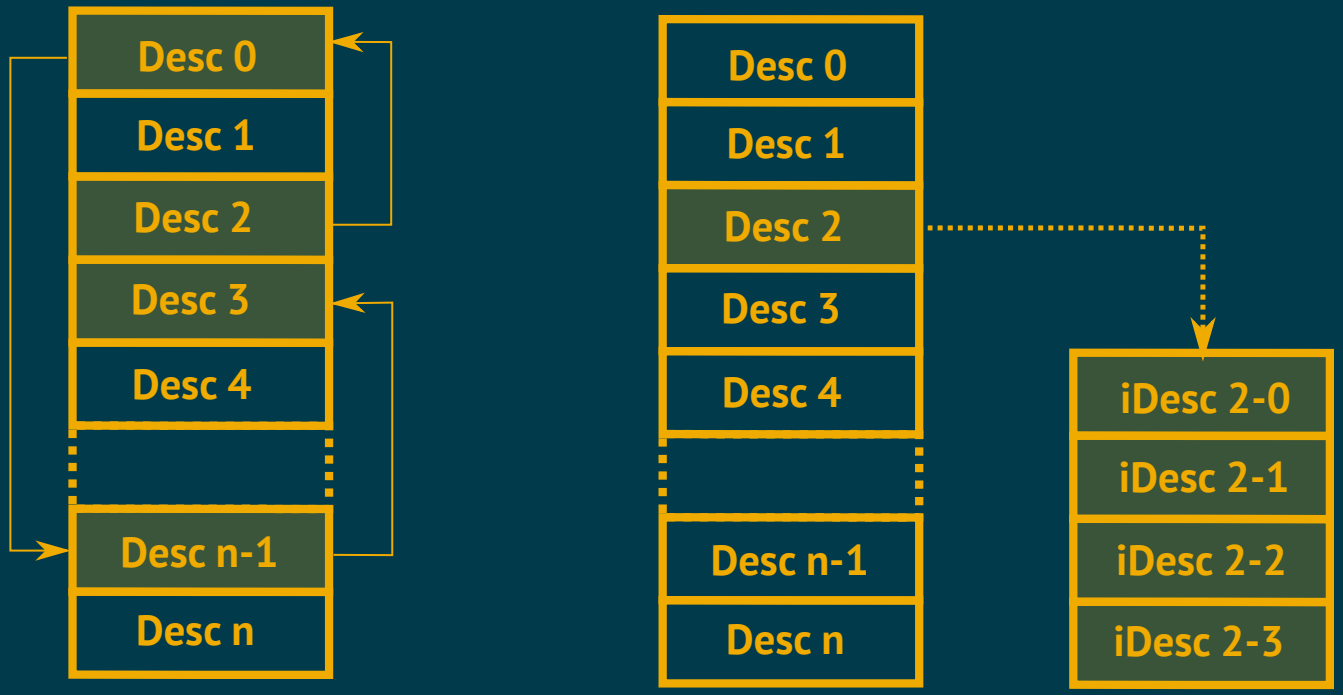
- Allows receiving packets larger than descriptors' buffer size

Con:

- Introduce extra-cache miss in the dequeue path

Indirect descriptors (DPDK v16.11)

New & upcoming features



Direct descriptors chaining

Indirect descriptors table

Indirect descriptors (DPDK v16.11)

New & upcoming features

Pros:

- Increase ring capacity
- Improve performance for large number of large requests
- Improve 0% packet loss perf even for small requests

  → If system is not fine-tuned

  → If Virtio headers are in dedicated descriptor

Cons:

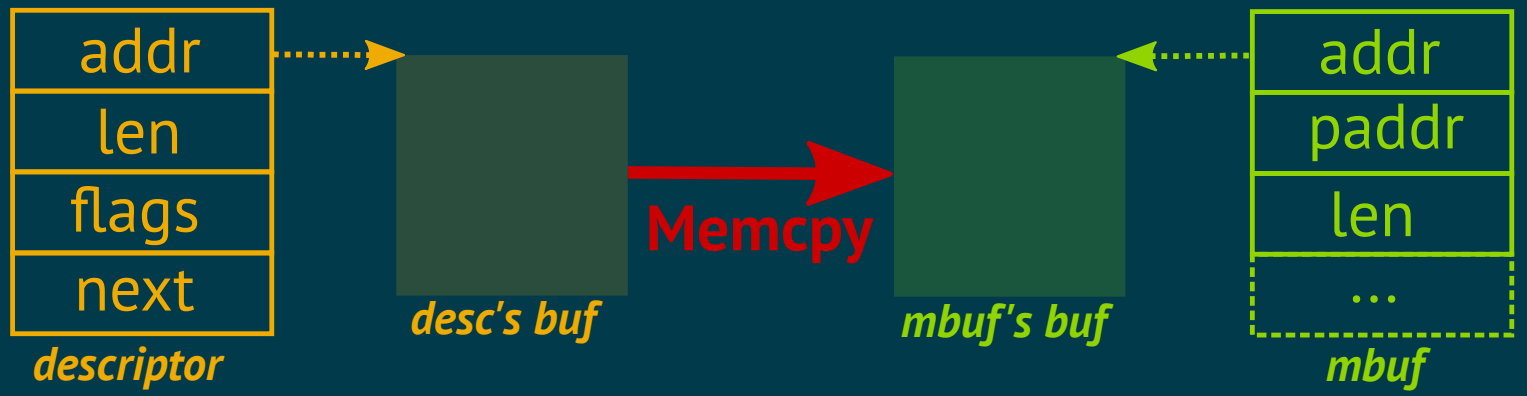
- One more level of indirection

  → Impacts raw performance (~-3%)

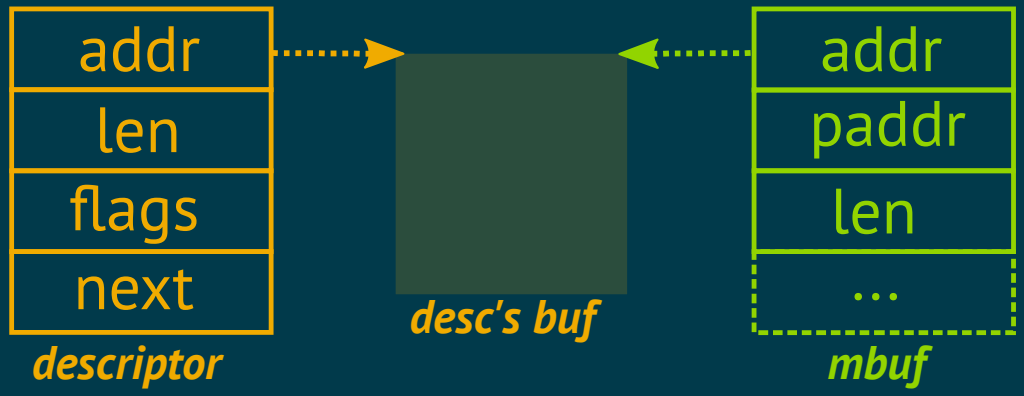
Vhost dequeue 0-copy (DPDK v16.11)

New & upcoming features

Default dequeuing



Zero-copy dequeuing



Vhost dequeue 0-copy (DPDK v16.11)

New & upcoming features

Pros:

- Big perf improvement for standard & large packet sizes

  → More than +50% for VM-to-VM with iperf benchs

- Reduces memory footprint

Cons:

- Performance degradation for small packets

  → But disabled by default

- Only for VM-to-VM using Vhost lib API (No PMD support)
- Does not work for VM-to-NIC

  → Mbuf lacks release notif mechanism / No headroom

MTU feature (DPDK v17.05?)

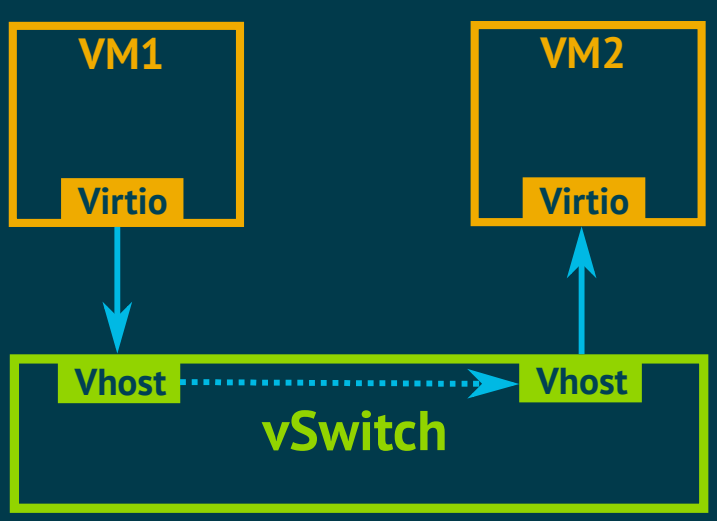
New & upcoming features

Way for the host to share its max supported MTU

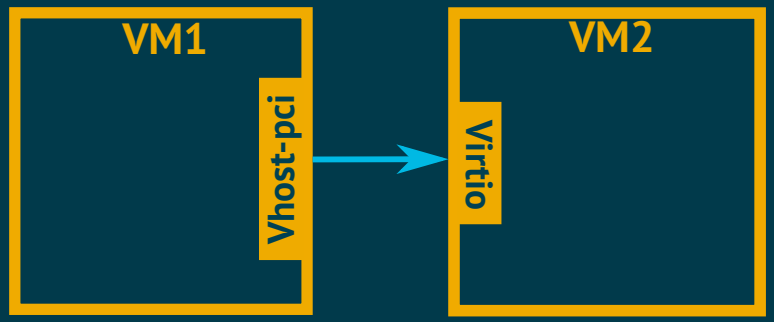
- Can be used to set MTU values across the infra
- Can improve performance for small packet
  - → If MTU fits in rx buffer size, disable Rx mergeable buffers
  - → Save one cache-miss when parsing the virtio-net header

Vhost-pci (DPDK v17.05?)

New & upcoming features



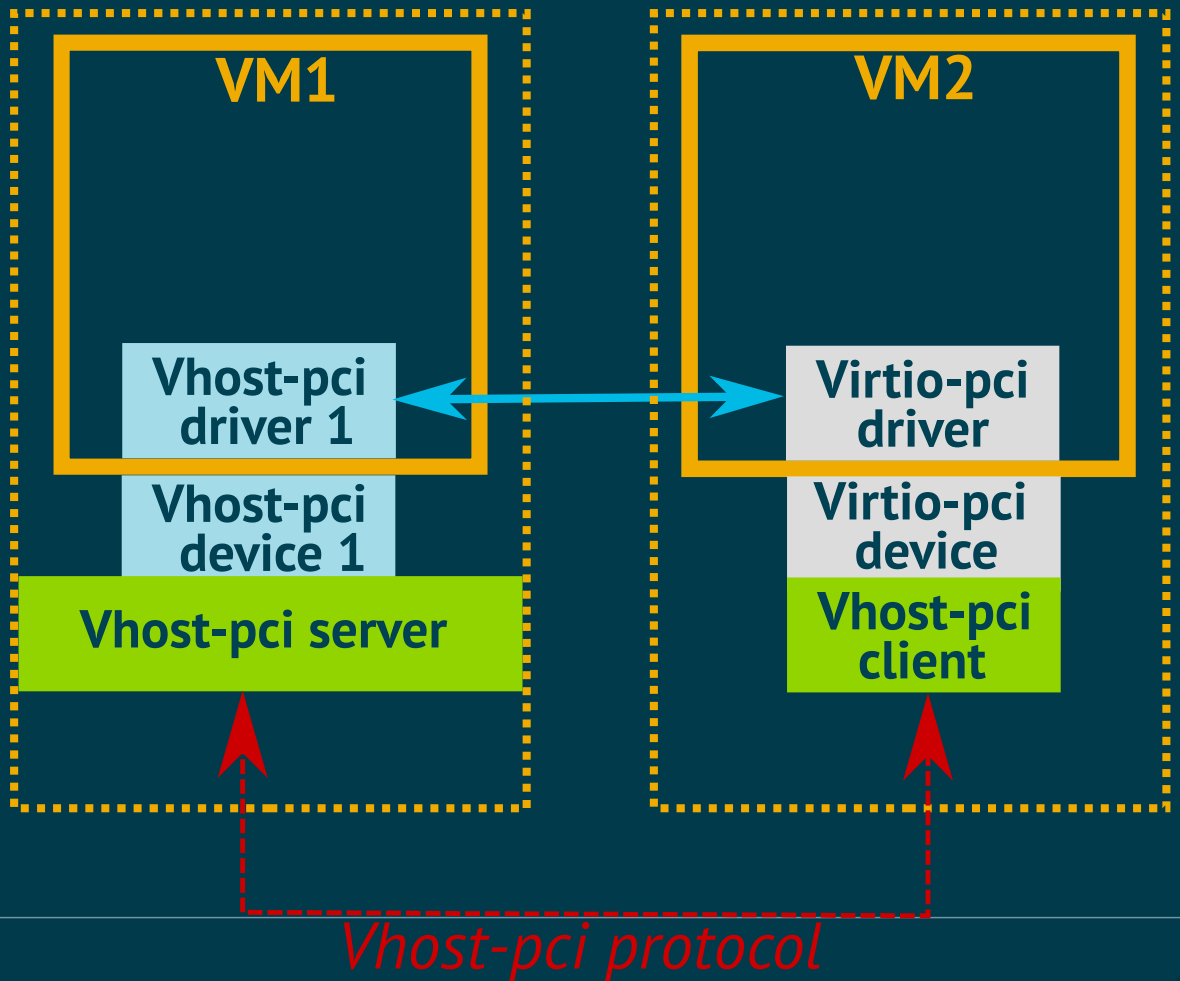
Traditional VM to VM communication



Direct VM to VM communication

Vhost-pci (DPDK v17.05?)

New & upcoming features



Vhost-pci (DPDK v17.05?)

New & upcoming features

Pros:

- Performance improvement
  → The 2 VMs share the same virtqueues
  → Packets doesn't go through host's vSwitch
- No change needed in Virtio's guest drivers

Vhost-pci (DPDK v17.05?)

New & upcoming features

Cons:

- Security

  → Vhost-pci's VM maps all Virtio-pci's VM memory space

  → Could be solved with IOTLB support

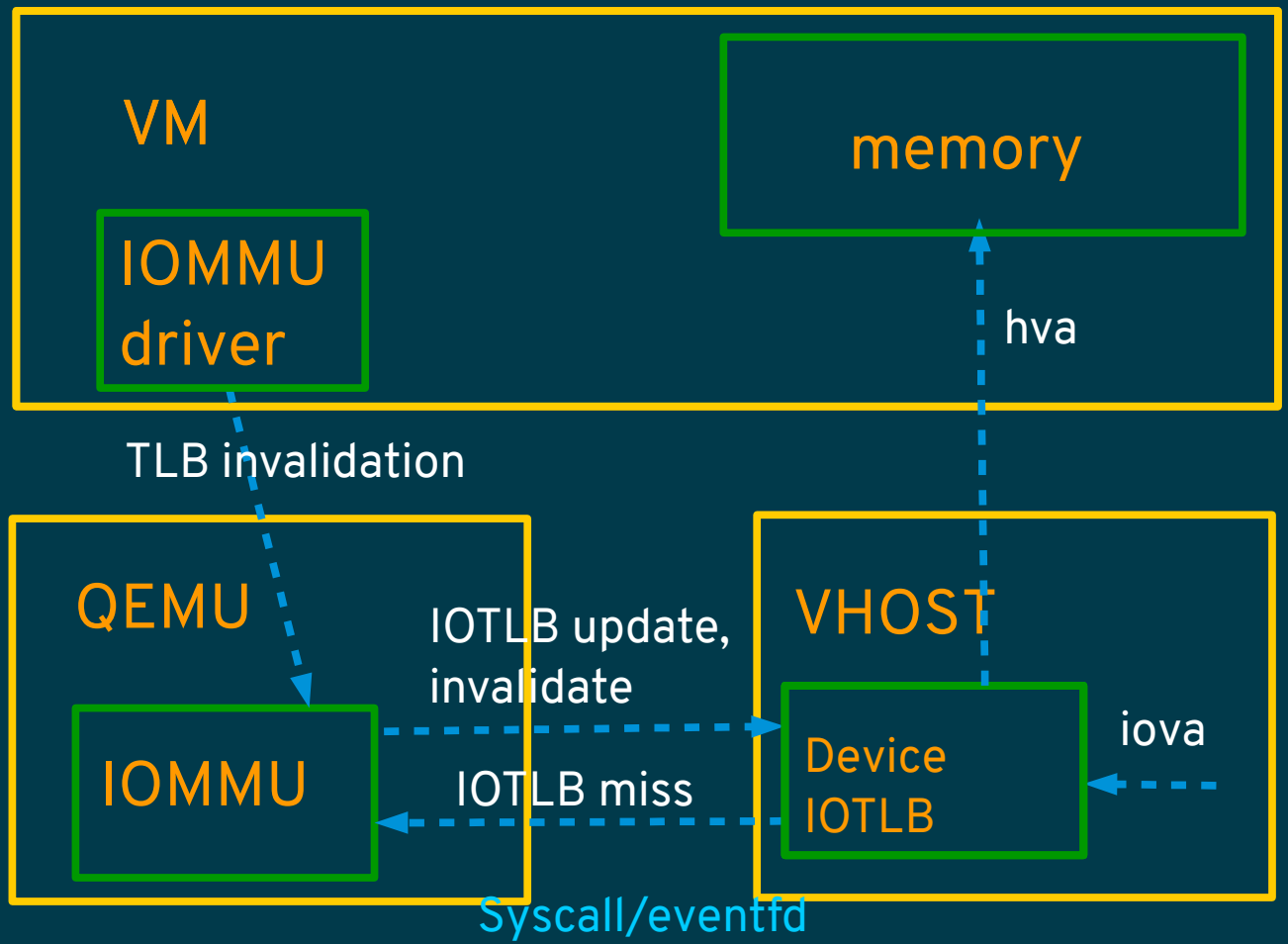
- Live migration

  → Not supported in current version

  → Hard to implement as VMs are connected to each other through a socket

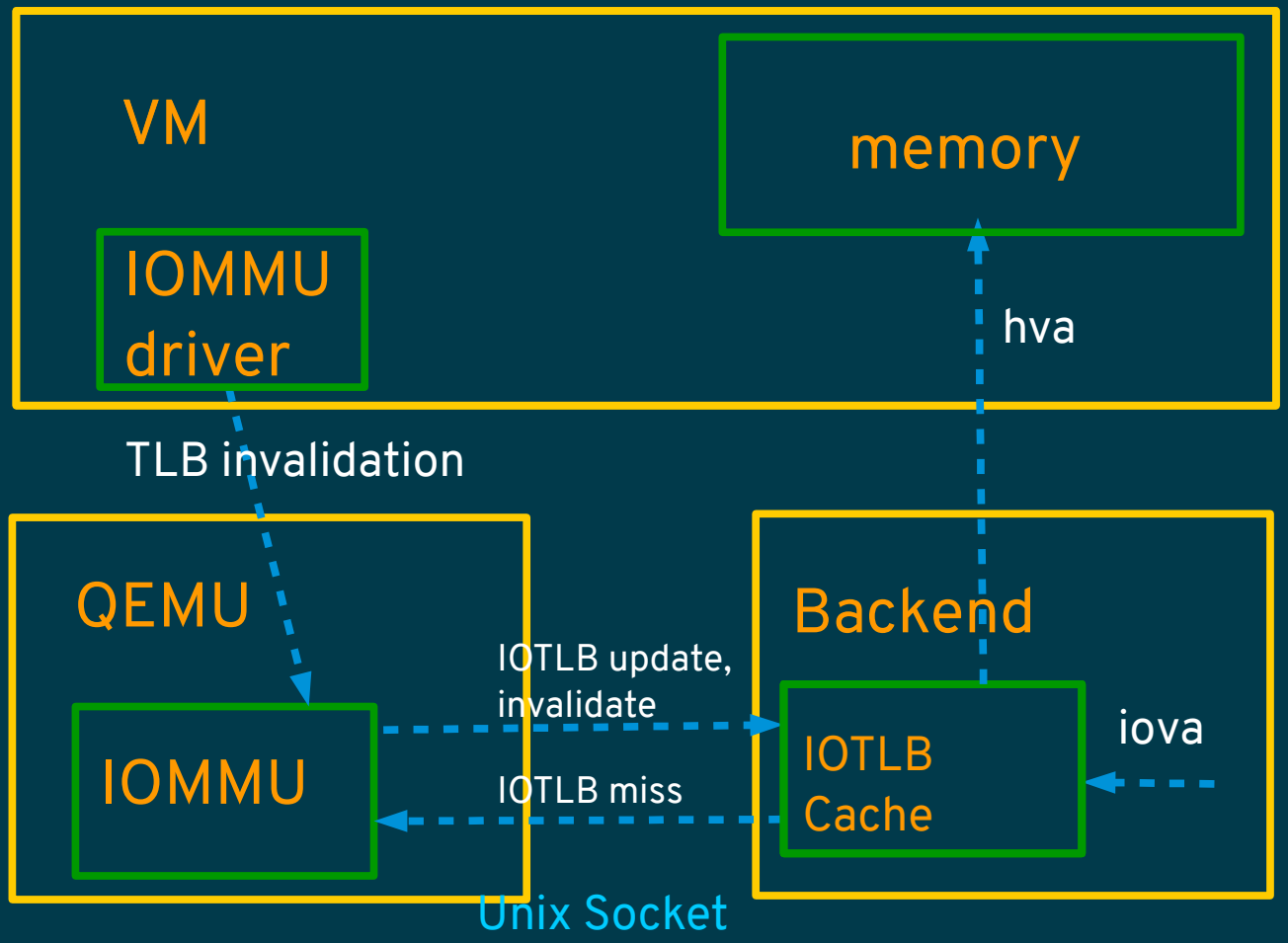
IOTLB in kernel

New & upcoming features



IOTLB for vhost-user

New & upcoming features



Conclusions

- DPDK support for VM is in active development
- 10M pps is real in VM with DPDK
- New features to boost performance of VM networking
- Accelerate transition to NFV / SDN

Q / A

redhat.

THANK YOU