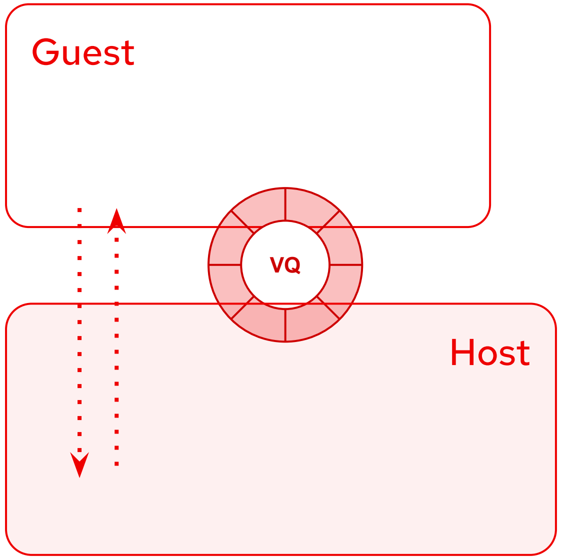
Where should my VIRTIO device live?

FOSDEM 2026

Stefano Garzarella

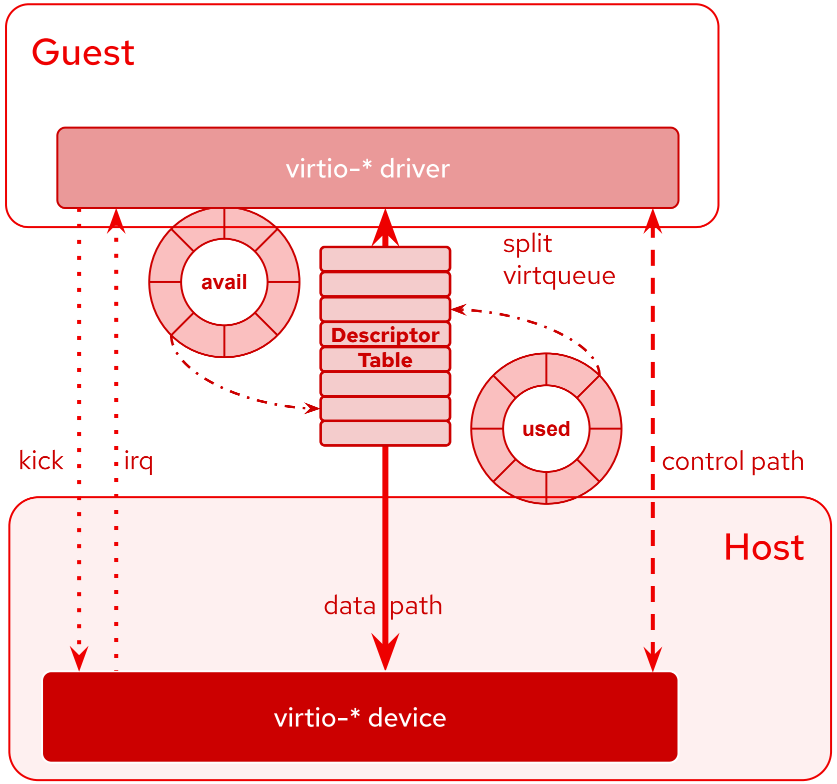
<sgarzare@redhat.com>

VIRTIO specification



- Virtual I/O Device (VIRTIO) Version 1.3
  - *The purpose of virtio and this specification is that virtual environments and guests should have a* ***straightforward, efficient, standard and extensible mechanism for virtual devices****, rather than boutique per-environment or per-OS mechanisms.*
  - VIRTIO 1.4 (Public Review Draft)
- https://github.com/oasis-tcs/virtio-spec
  - Authoritative source of the VIRTIO (Virtual I/O) Specification
- Virtual I/O devices
  - core components (features, notifications, configuration, virtqueues, etc.)
  - initialization steps
  - transports (PCI, MMIO, Channel I/O)
  - device types (e.g. net, block, vsock, sound, fs, etc.)

VIRTIO core components

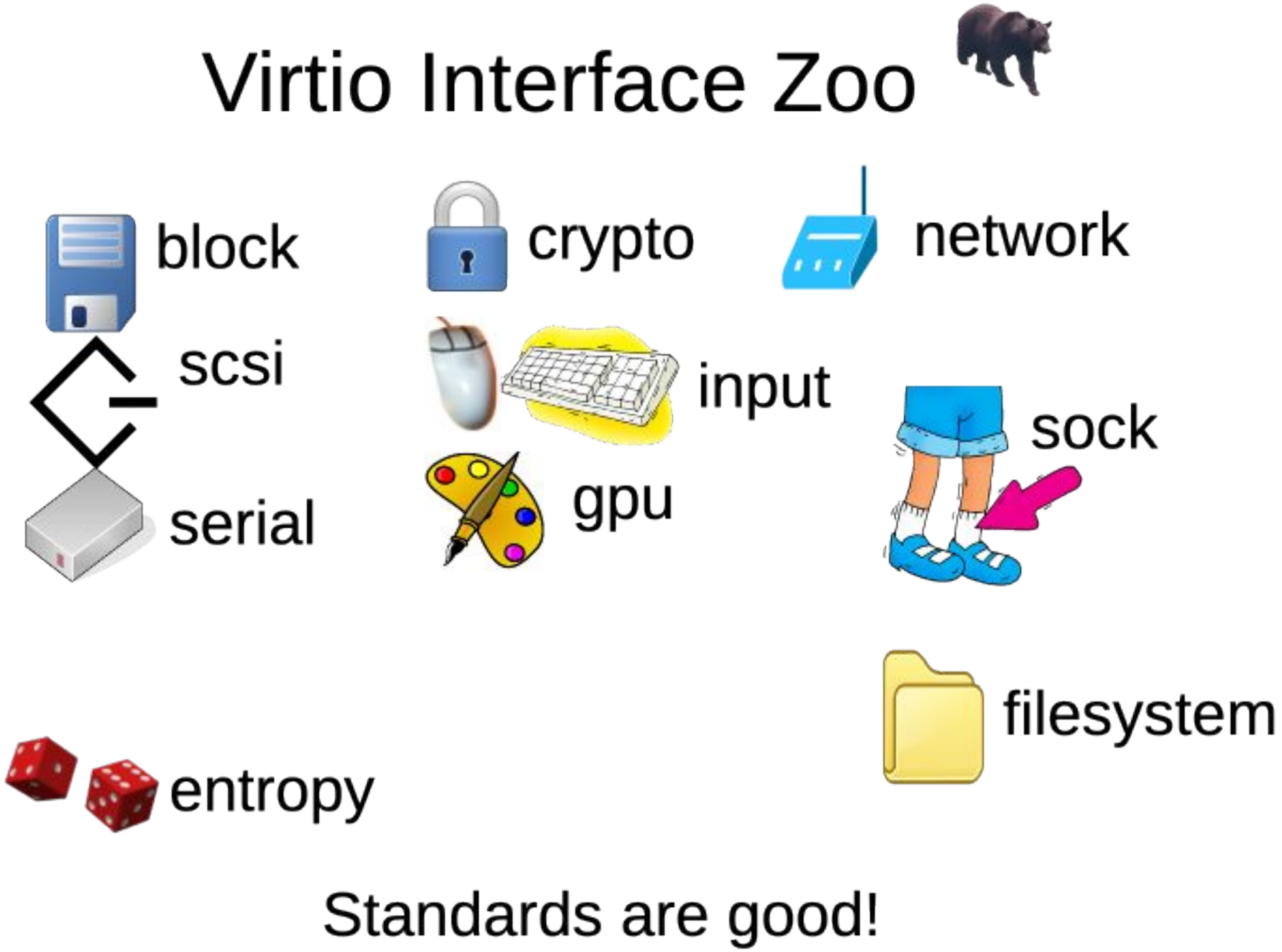


- Control path
  - features negotiation
  - configuration space
  - data path setup
- Data path
  - virtqueue
    - split / packed
    - always allocated by the guest

- Notifications
  - kick
    - guest -> host
  - irq
    - host -> guest

VIRTIO device types

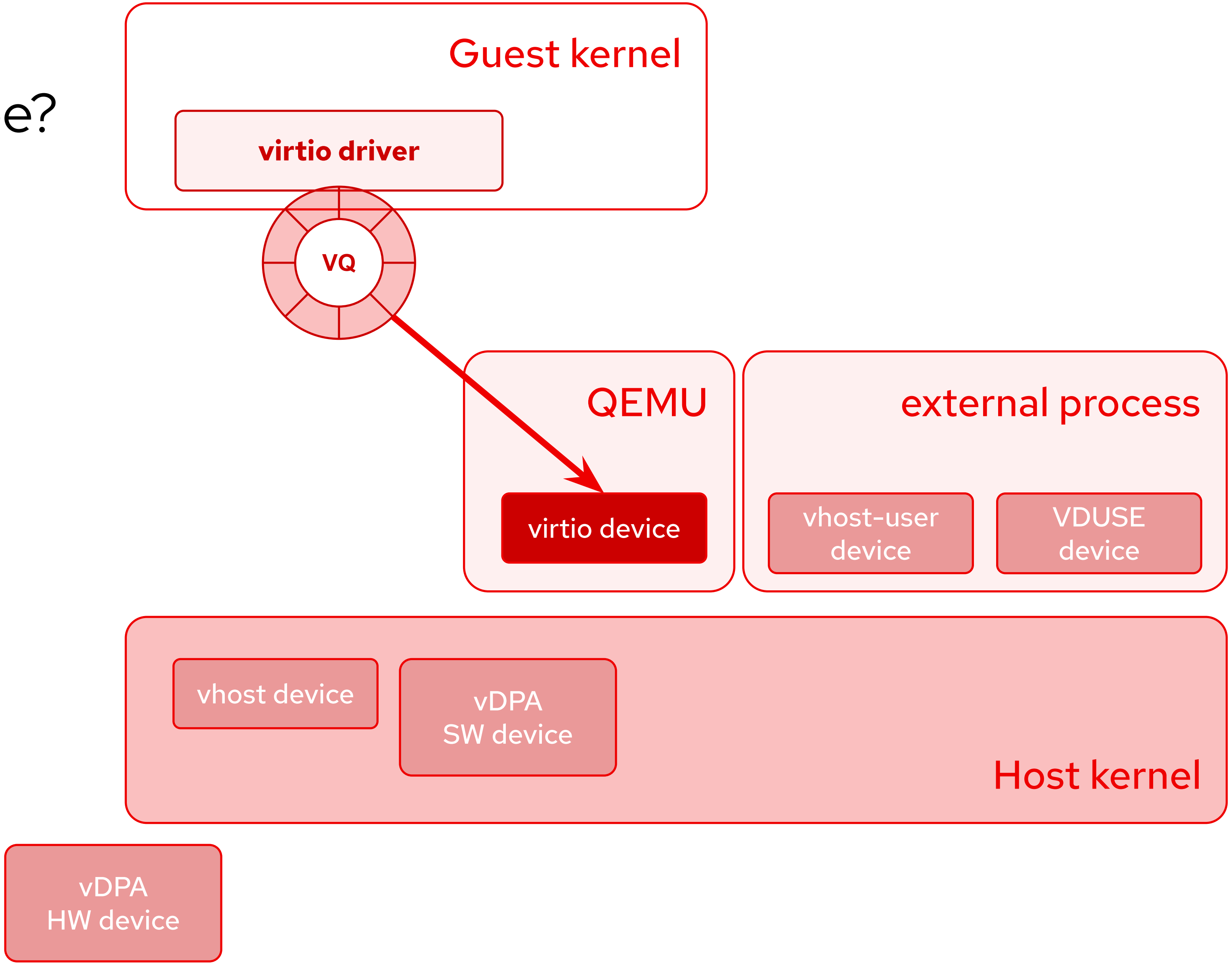
- Several device types in the specification
  - virtio-net
    - Network card
  - virtio-blk
    - Block device (HDD, SSD)
  - virtio-vsock
    - Virtual Socket
  - virtio-fs
    - File system (e.g. shared folder)
  - and others
- Modern / Legacy
  - Legacy (pre VIRTIO v1.0)
  - Modern



Picture from https://kvmforum2019.sched.com/event/TmxF/virtio-without-the-virt-towards-implementations-in-hardware-michael-tsirkin-red-hat

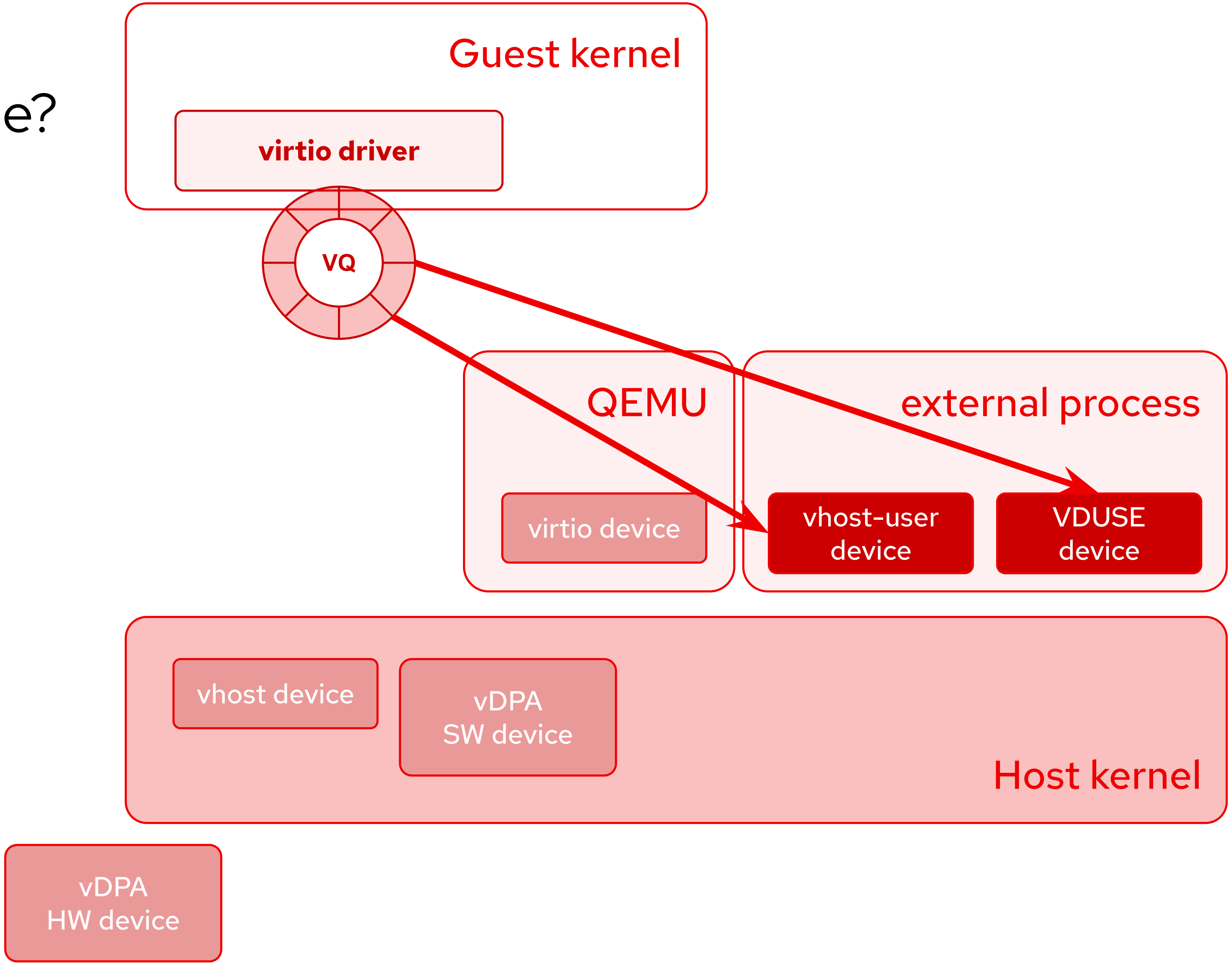
Where the device could be?

- **VMM built-in**
  - **e.g. in QEMU process**
- user space (host) device
  - vhost-user
  - VDUSE (vDPA in user space)
- kernel (host) device
  - vhost
  - vDPA SW device
- hardware
  - vDPA HW device



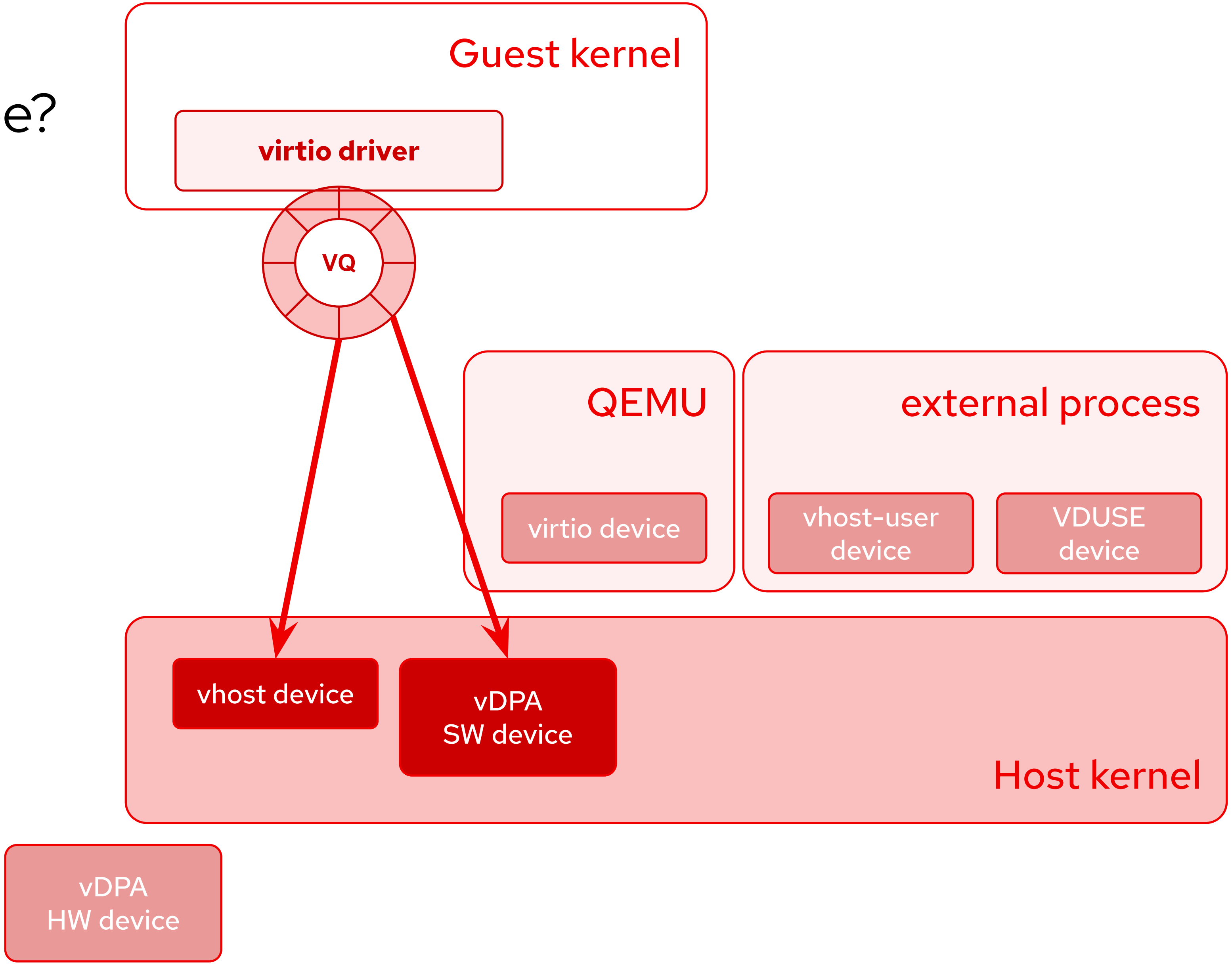
Where the device could be?

- VMM built-in
  - e.g. in QEMU process
- **user space (host) device**
  - **vhost-user**
  - **VDUSE (vDPA in user space)**
- kernel (host) device
  - vhost
  - vDPA SW device
- hardware
  - vDPA HW device



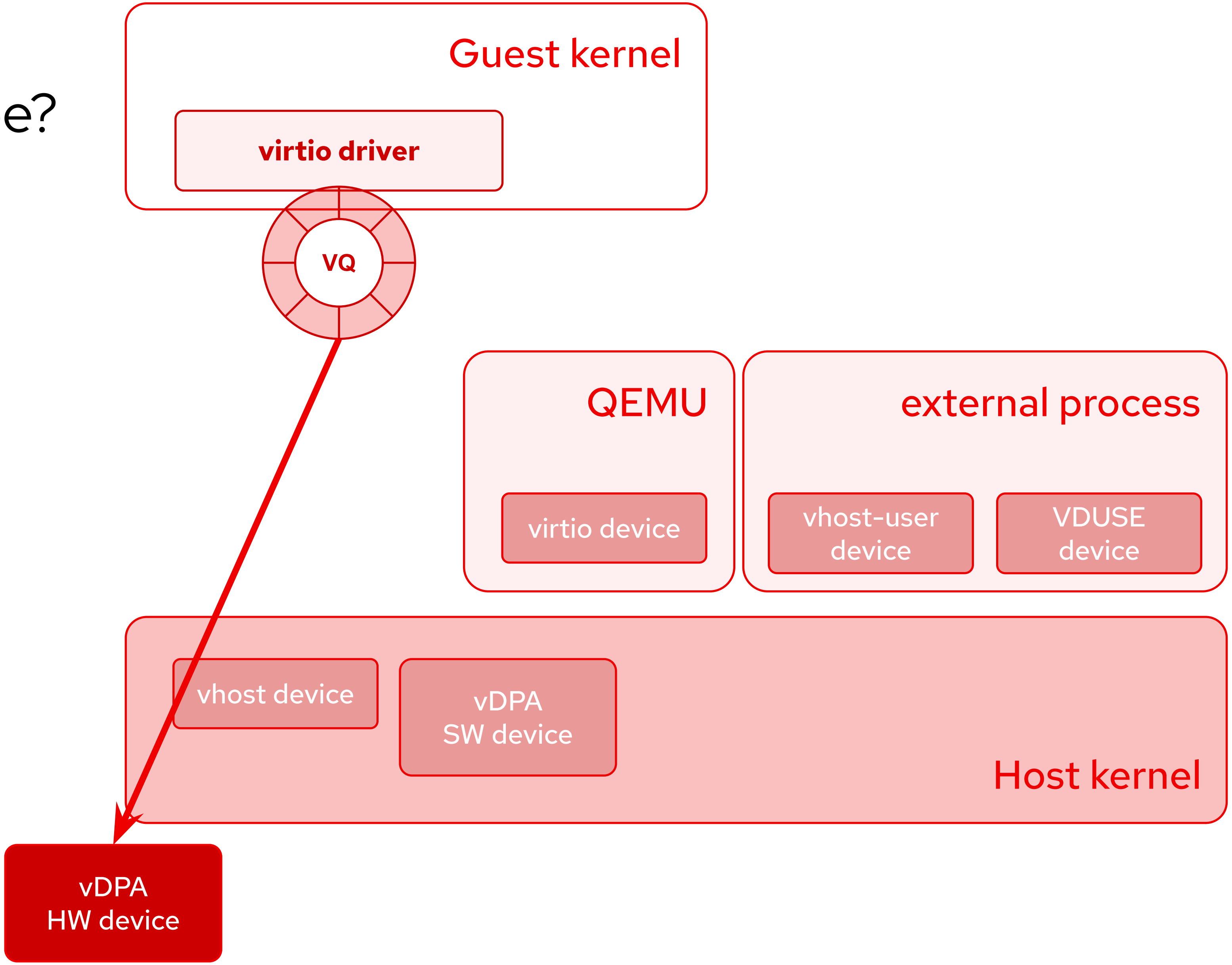
Where the device could be?

- VMM built-in
  - e.g. in QEMU process
- user space (host) device
  - vhost-user
  - VDUSE (vDPA in user space)
- **kernel (host) device**
  - **vhost**
  - **vDPA SW device**
- hardware
  - vDPA HW device

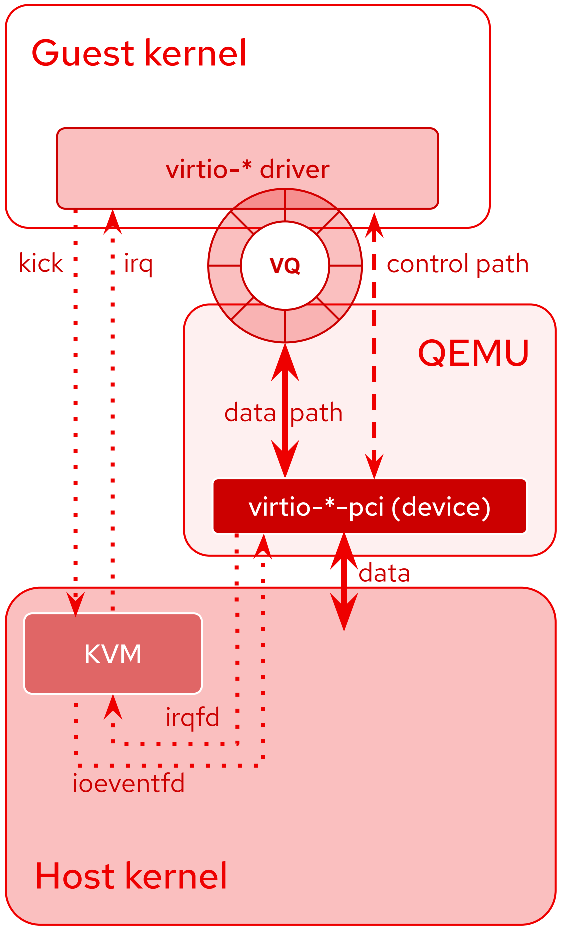


Where the device could be?

- VMM built-in
  - e.g. in QEMU process
- user space (host) device
  - vhost-user
  - VDUSE (vDPA in user space)
- kernel (host) device
  - vhost
  - vDPA SW device
- **hardware**
  - **vDPA HW device**

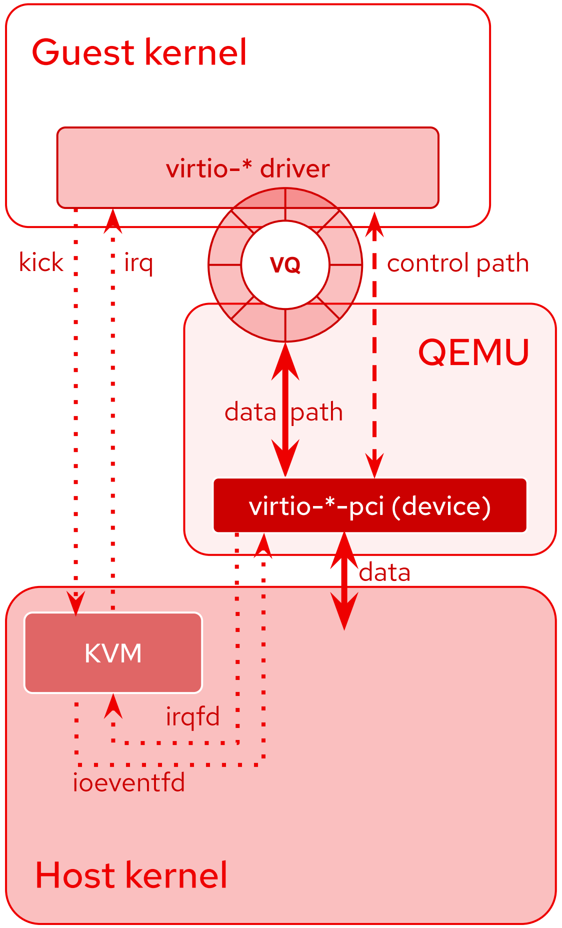


VIRTIO device emulated by the VMM



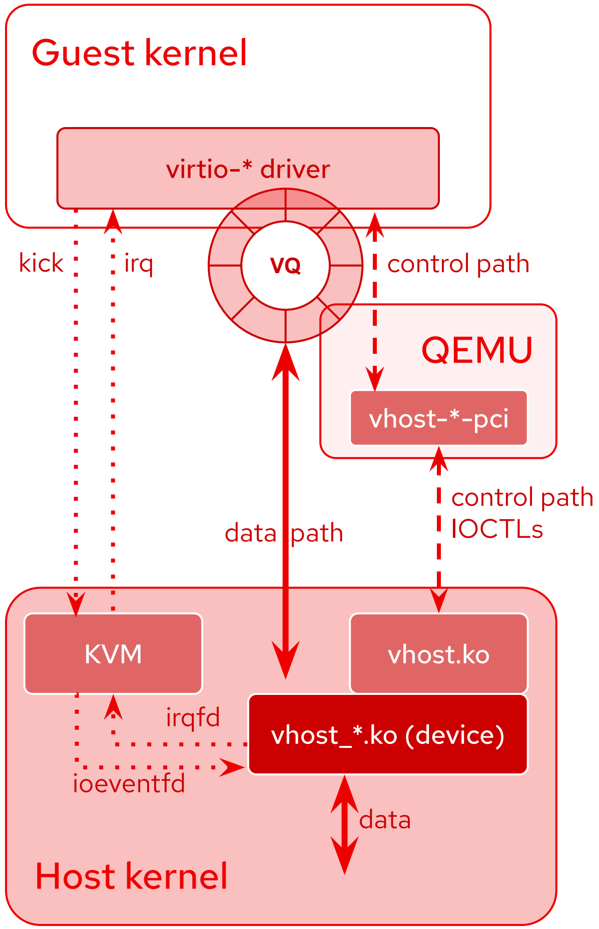
- Common scenario
- QEMU
  - de facto reference implementation for VIRTIO devices
- Why?
  - Simplicity
  - Portability
- Drawbacks
  - Reliability risk
  - Performance overhead

VIRTIO device emulated by the VMM



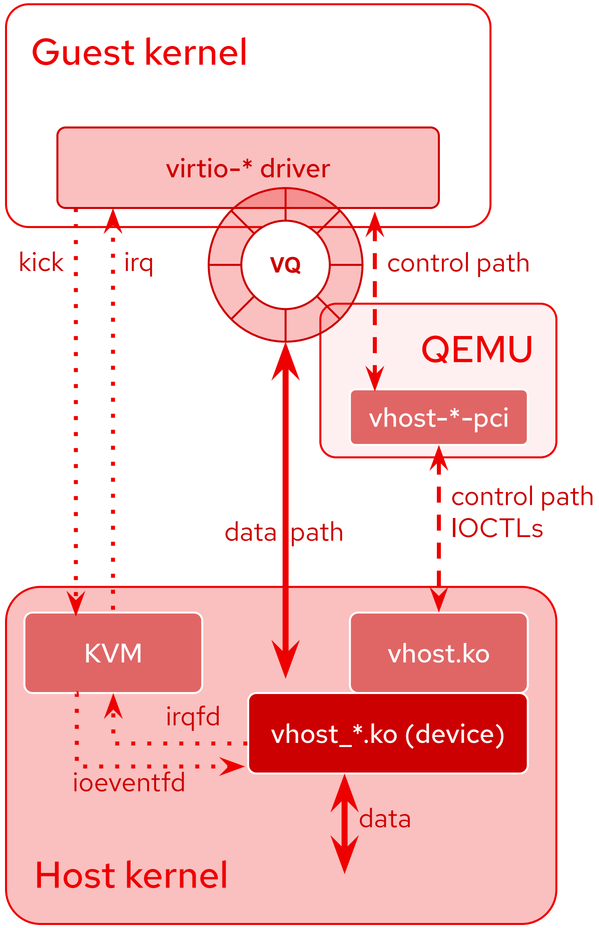
```
$ qemu-system-x86_64 -smp 2 -m 2G \
    -M q35,accel=kvm \
    -blockdev file,filename=fedora.qcow2,node-name=file \
    -blockdev qcow2,file=file,node-name=qcow2 \
    -device virtio-blk-pci,drive=qcow2
```

vhost: VIRTIO device in the host kernel



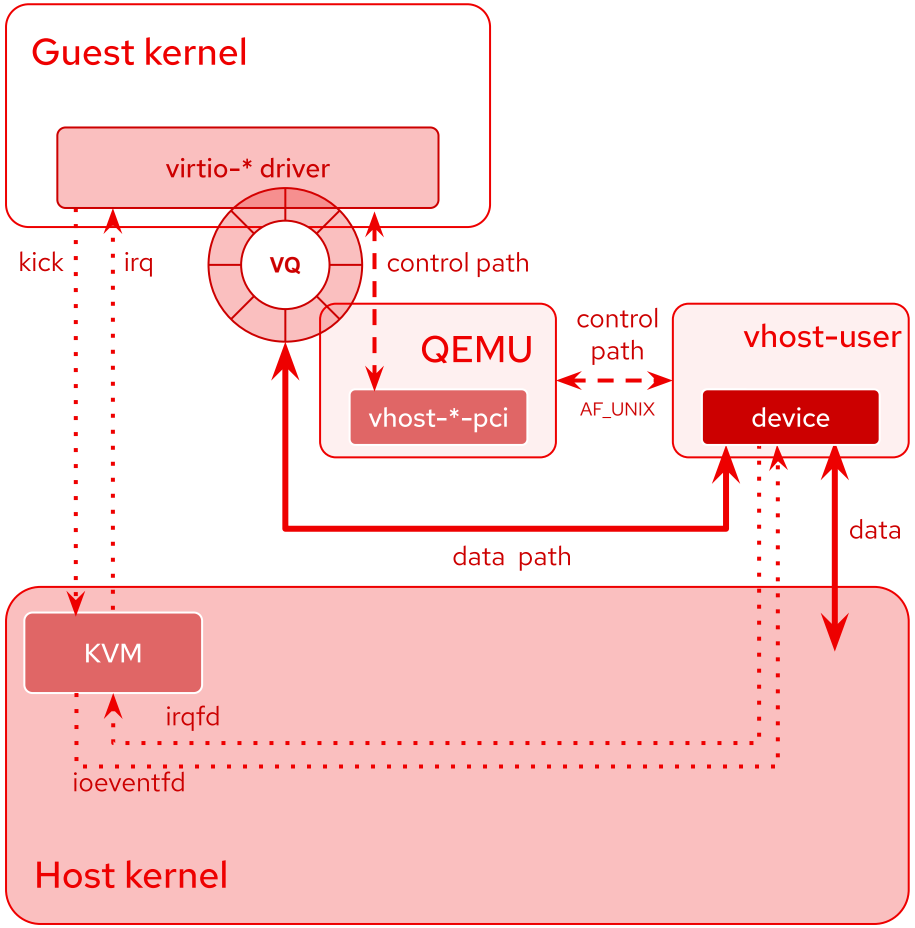
- Initially introduced to increase performance of virtio-net device
  - Control path
    - IOCTLs
  - Data path
    - kthread/vhost_task attaches VMMs address space
- Linux kernel supports
  - vhost-net, vhost-scsi, vhost-vsock
- Why?
  - Performance
  - Easily to integrate with host kernel stacks (e.g. AF_VSOCK)
- Drawbacks
  - Security & reliability risk
  - Upgradability
  - Portability
    - Linux-specific

vhost: VIRTIO device in the host kernel



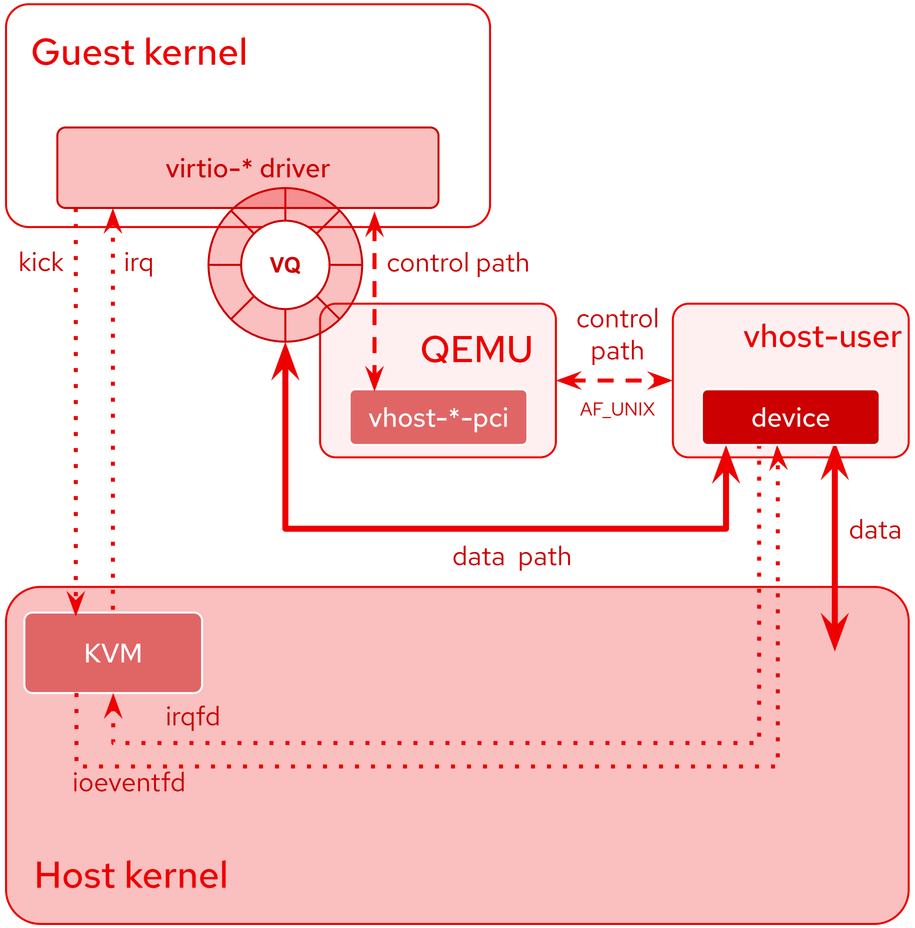
```
$ qemu-system-x86_64 -smp 2 -m 2G \
    -M q35,accel=kvm \
    -drive file=fedora.qcow2,format=qcow2,if=virtio \
    -device vhost-vsock-pci,guest-cid=42
```

vhost-user: VIRTIO device in an external process



- Inspired by vhost
    - Control path
        - AF_UNIX
    - Data path
        - Shared memory through fd sharing (memfd, /dev/shm, etc.)
- Why?
    - Security & reliability risks mitigated
    - Upgradability
    - Flexibility
        - Different language from VMM (e.g. Rust)
- Drawbacks
    - More resources used
    - More coordination
        - can be hidden by management layer (e.g. libvirt)
    - ~~Portability~~
        - see FOSDEM 2025 - Can QEMU and vhost-user devices be used on macOS and *BSD?

vhost-user: VIRTIO device in an external process



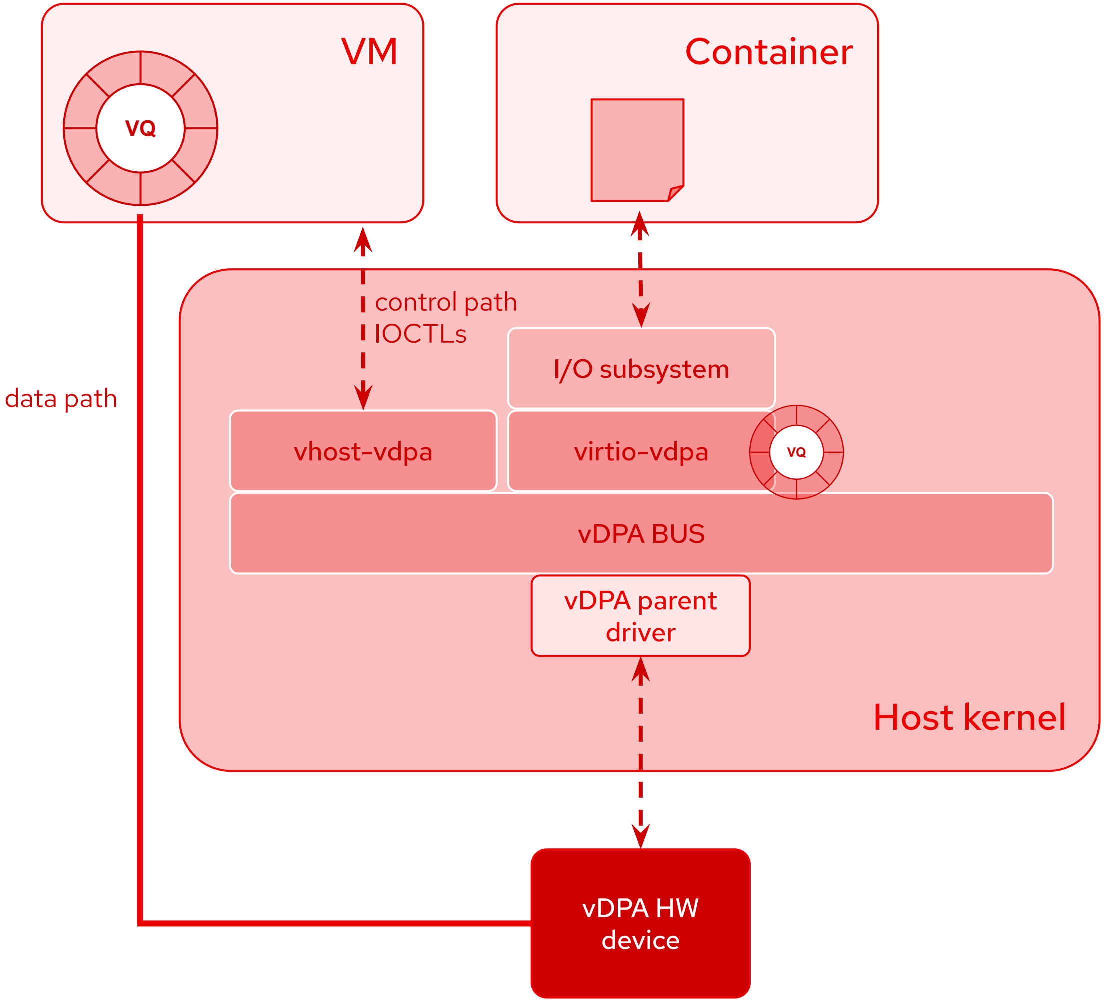
```
$ qemu-storage-daemon \
  --blockdev file,filename=fedora.qcow2,node-name=file \
  --blockdev qcow2,file=file,node-name=qcow2 \
  --export vhost-user-blk,id=vbu,node-name=qcow2,\
num-queues=1,writable=on,\
addr.type=unix,addr.path=/tmp/vhost.socket
```

```
$ qemu-system-x86_64 -smp 2 \
    -M q35,accel=kvm,memory-backend=mem \
    -chardev socket,id=char0,path=/tmp/vhost.socket \
    -device vhost-user-blk-pci,num-queues=1,chardev=char0 \
    -object memory-backend-memfd,id=mem,size="2G"
```

```
# -object memory-backend-shm,id=mem,size="2G"
# can eventually be used on any POSIX host OS (available from QEMU 9.1)
```

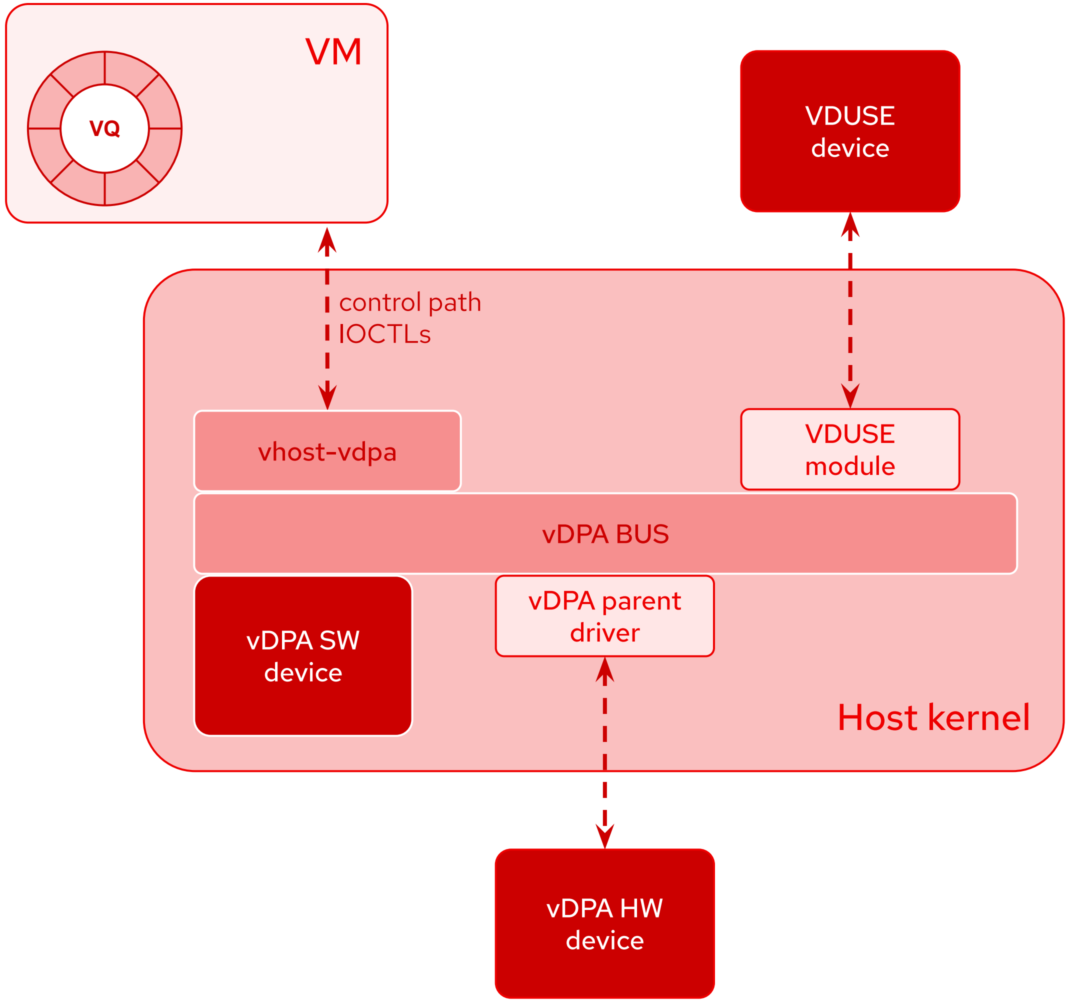
vDPA: VIRTIO device (also) in hardware

- virtio Data Path Acceleration
  - VIRTIO compliant data path
  - vendor specific control path
    - small vDPA driver for the control path
- Why?
  - Performance
    - Designed for hardware accelerators
      - software accelerators also possible
  - Support both VMs and containers workloads
    - vhost-vdpa
      - interface for userspace/guest virtio driver
    - virtio-vdpa
      - interface for host virtio driver
      - bare metal or containerized applications



vDPA: virtio Data Path Acceleration

- Unified software stack for vDPA devices
  - Hardware device
    - small parent driver needed
  - Software device
    - in-kernel
    - in-userspace
      - **VDUSE**: vDPA device in Userspace
- Drawbacks
  - Portability
    - Linux-specific
  - Maturity
    - supported by few hardware devices
    - support few virtio types
      - net, block (VDUSE)
  - Cost (HW devices)



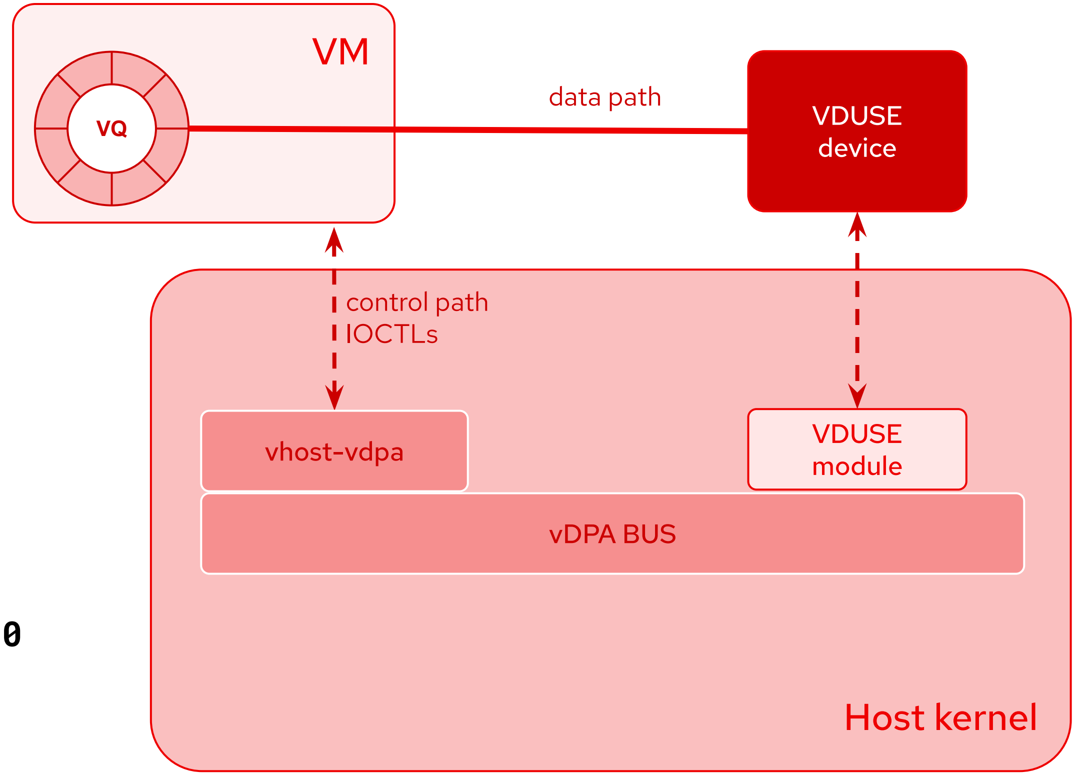
vDPA: virtio Data Path Acceleration

```
$ qemu-storage-daemon \
  --blockdev file,filename=fedora.qcow2,node-name=file \
  --blockdev qcow2,file=file,node-name=qcow2 \
  --export vduse-blk,id=vduse1,node-name=qcow2,\
num-queues=1,writable=on,name=vduse1

# instantiate the `vduse1` device (same name used in QSD)
$ vdpa dev add name vduse1 mgmtdev vduse

# the device is identified as `vhost-vdpa-0` in the host
$ ls /sys/bus/vdpa/devices/vduse1/
driver  driver_override  power  subsystem  uevent  vhost-vdpa-0

$ qemu-system-x86_64 -smp 2 \
    -M q35,accel=kvm,memory-backend=mem \
    -object memory-backend-memfd,id=mem,size="2G" \
    -device vhost-vdpa-device-pci,vhostdev=/dev/vhost-vdpa-0
```



More examples: https://stefano-garzarella.github.io/posts/2024-02-12-vdpa-blk/

Thank you!

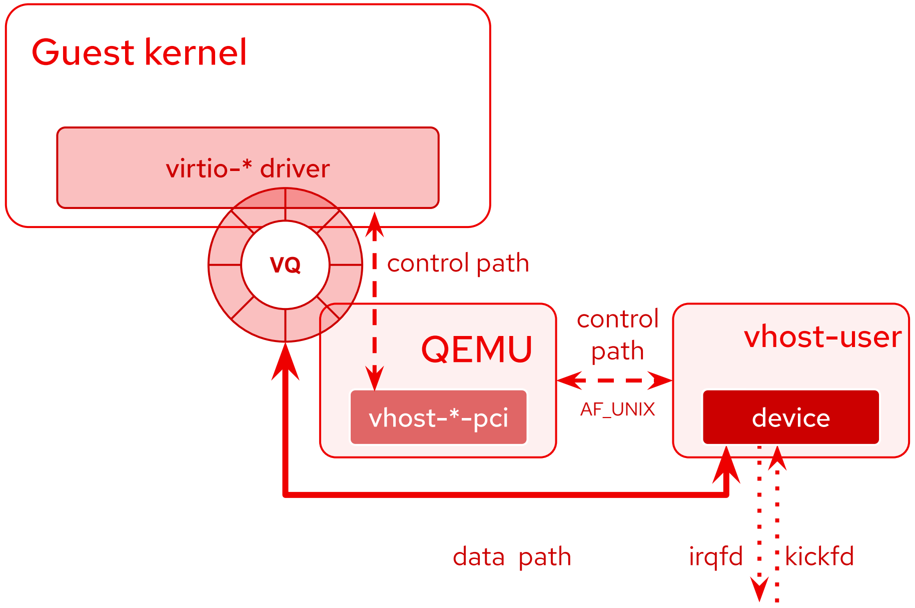
Stefano Garzarella <sgarzare@redhat.com>

https://stefano-garzarella.github.io/

in linkedin.com/company/red-hat	f facebook.com/redhatinc
youtube.com/user/RedHatVideos	twitter.com/RedHat

vhost-user protocol

- https://qemu-project.gitlab.io/qemu/interop/vhost-user.html
  - *control plane needed to establish virtqueue sharing with an user space process on the same host*
  - **frontend**
    - application that shares its virtqueues (i.e. VMM like QEMU)
  - **backend**
    - consumer of the virtqueues (i.e. virtio device emulation)
- Key components
  - UNIX domain socket (**AF_UNIX**)
    - + ancillary data support to exchange file descriptors
      - shared memory, notifications (irqfd, kickfd), etc.
  - **shared memory** represented by a file descriptor
    - so it can be passed over a UNIX domain socket and then mapped by the other process
  - notifications
    - **eventfd** or **pipe**/**pipe2**
      - on platforms where eventfd is not available, QEMU will automatically fall back to pipe2 or, as a last resort, pipe.


Guest kernel
virtio-* driver
VQ
control path
QEMU
vhost-*-pci
control path
AF_UNIX
vhost-user
device
data path
irqfd
kickfd