vdpa-blk: Unified Hardware and Software Offload for virtio-blk

KVM Forum 2021

Stefano Garzarella <sgarzare@redhat.com>

Senior Software Engineer @ Red Hat

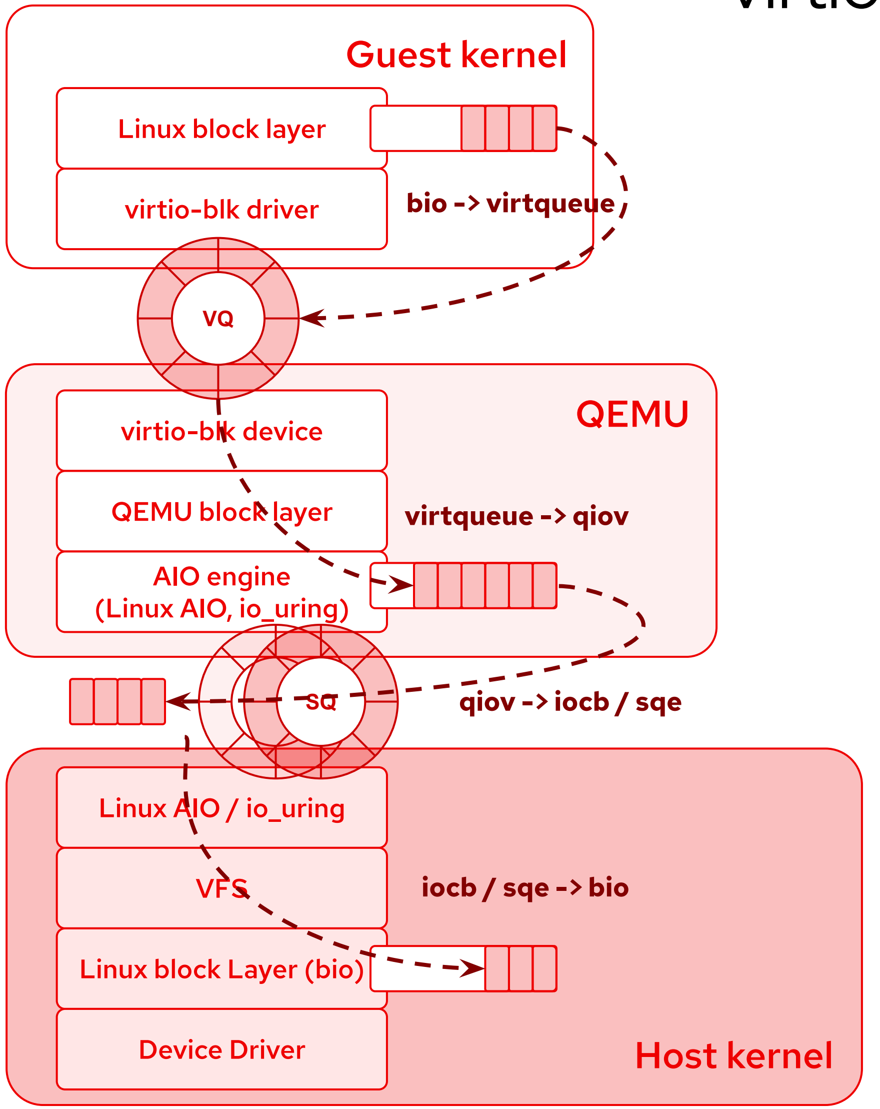
Agenda

- Goals and benefits of vdpa-blk
- virtio-blk request path
    - vhost acceleration
    - io_uring passthrough
- vDPA
    - overview
    - virtio-blk devices
- QEMU
    - block layer features
    - auto-switching: fast and slow path
- Current status and next steps

Goals and benefits of vdpa-blk

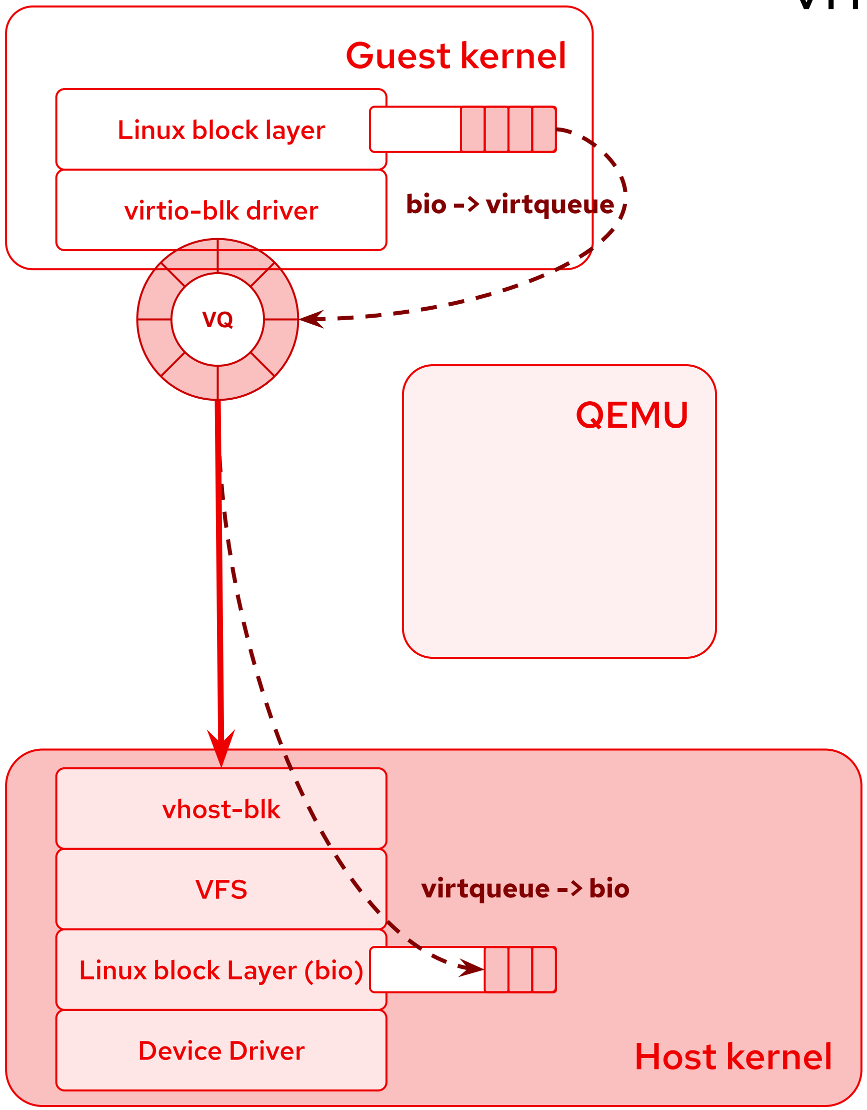
- **Unified software stack**
  - supports **virtual machines**, **containers**, and **applications**
  - supports both **hardware** and **software** virtio-blk devices
  - **QEMU**'s storage virtualization features (image file formats, block jobs, etc) available for virtual machines
  - **high-performance** implementation suitable for high IOPS NVMe drives
- Developing new accelerators PCI devices?
  - **participate and take advantage of the vdpa-blk stack**!
- vDPA website: **https://vdpa-dev.gitlab.io**

virtio-blk request path



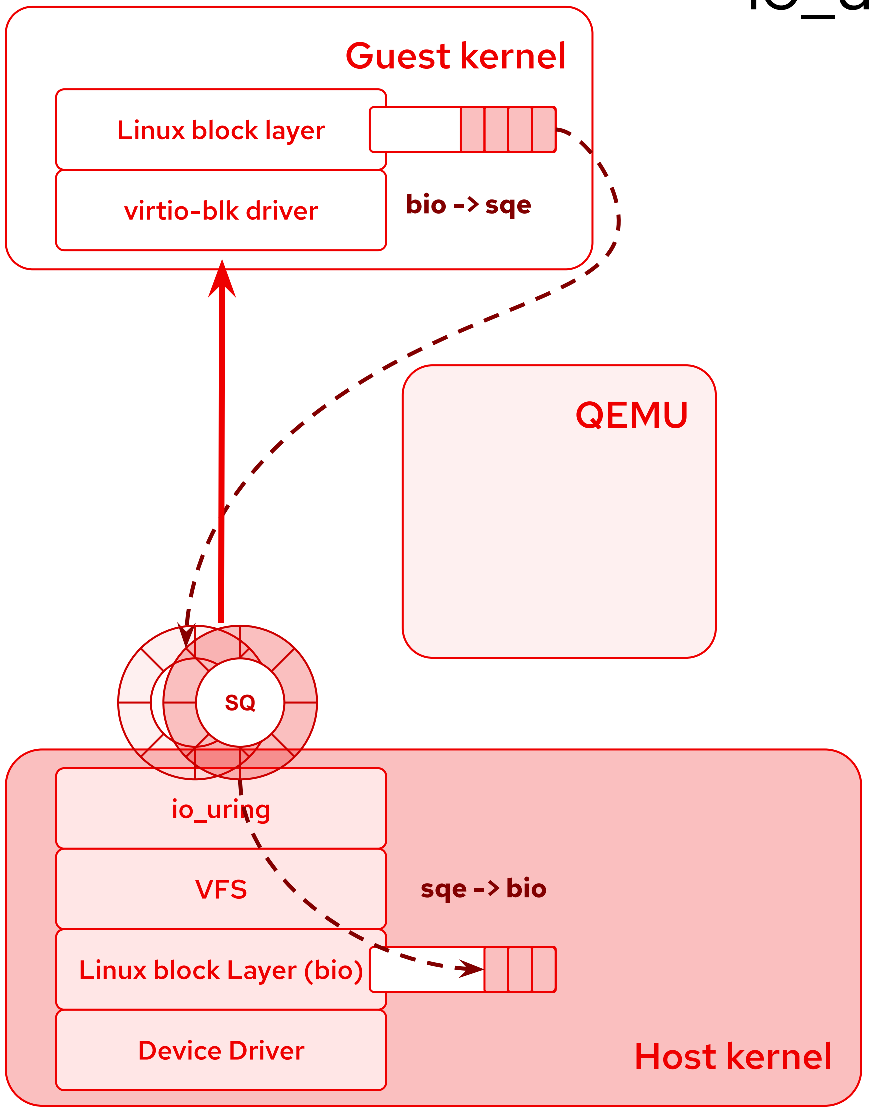
- Multiple layers to cross
  - Linux block layer -> virtio-blk
  - virtio-blk -> QEMU block layer
  - QEMU block layer -> Linux AIO / io_uring
  - Linux AIO / io_uring -> VFS
  - VFS -> Linux block layer
- Multiple request translations
- Multiple queues
- System calls to interact with host kernel

vhost acceleration



- in-kernel virtio device emulation
- QEMU bypassed
  - QEMU's storage features not available (image file formats, block jobs, etc)
- vhost-blk
  - proposed multiple times, but never merged upstream
    - Asias He's vhost-blk [2012] https://lore.kernel.org/patchwork/patch/344823/
      - bio API
    - Vitaly Mayatskih's vhost-blk [2018] https://patchwork.kernel.org/cover/10665995/
      - VFS API

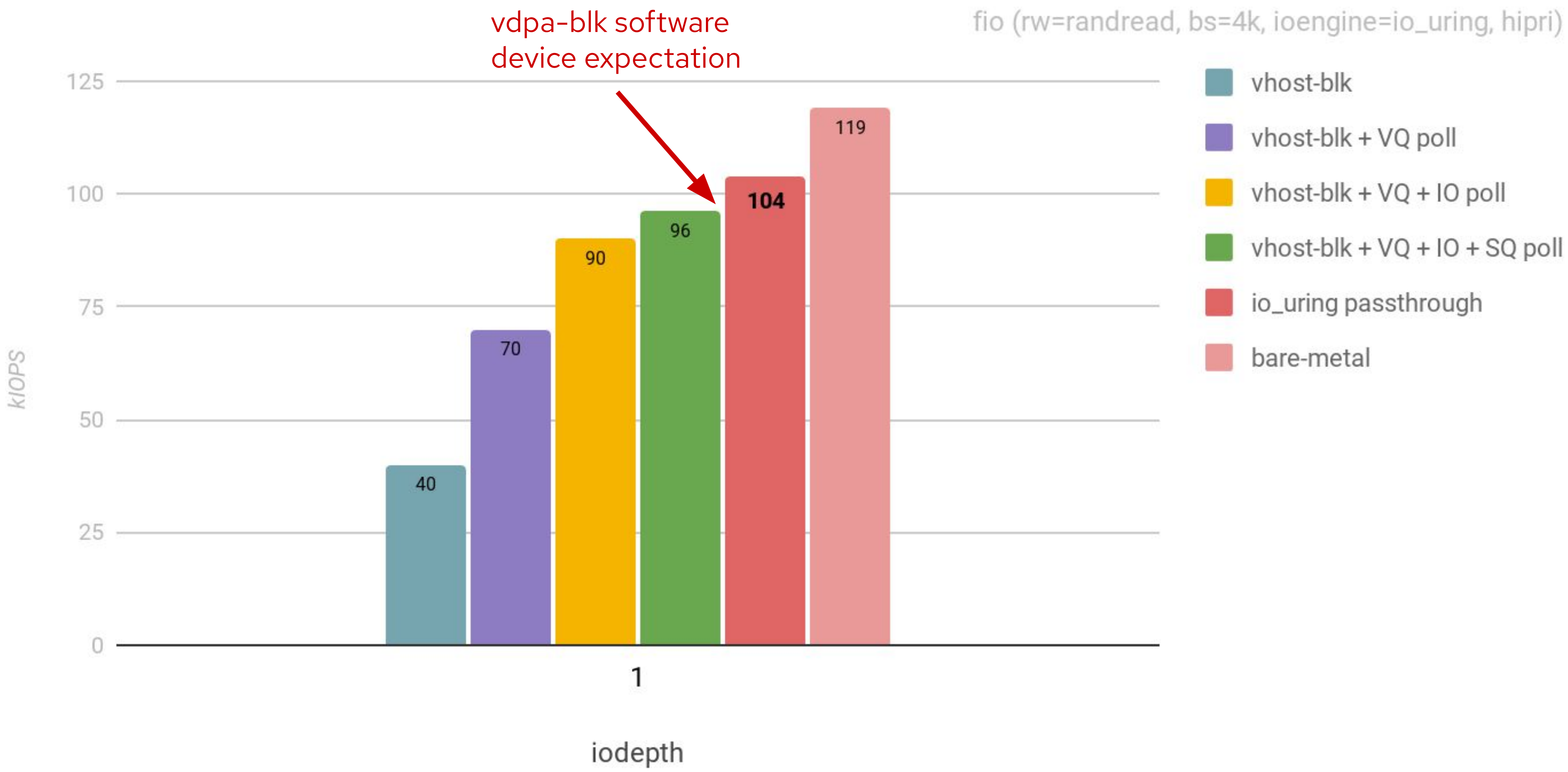
io_uring passthrough



- io_uring's SQ/CQ are memory mapped in the guest
  - require changes in the guest kernel driver
- virtio-blk driver modified
  - handle io_uring's SQ/CQ memory mapped
  - eventfd registered to inject interrupts (irqfd)
- Polling
  - SQPOLL enabled in the host to avoid notification from the guest (vmexit)
  - IOPOLL enabled in the host to avoid IRQs in the host

from Speeding Up VM's I/O Sharing Host's io_uring Queues With Guests - Stefano Garzarella @ KVM Forum 2020

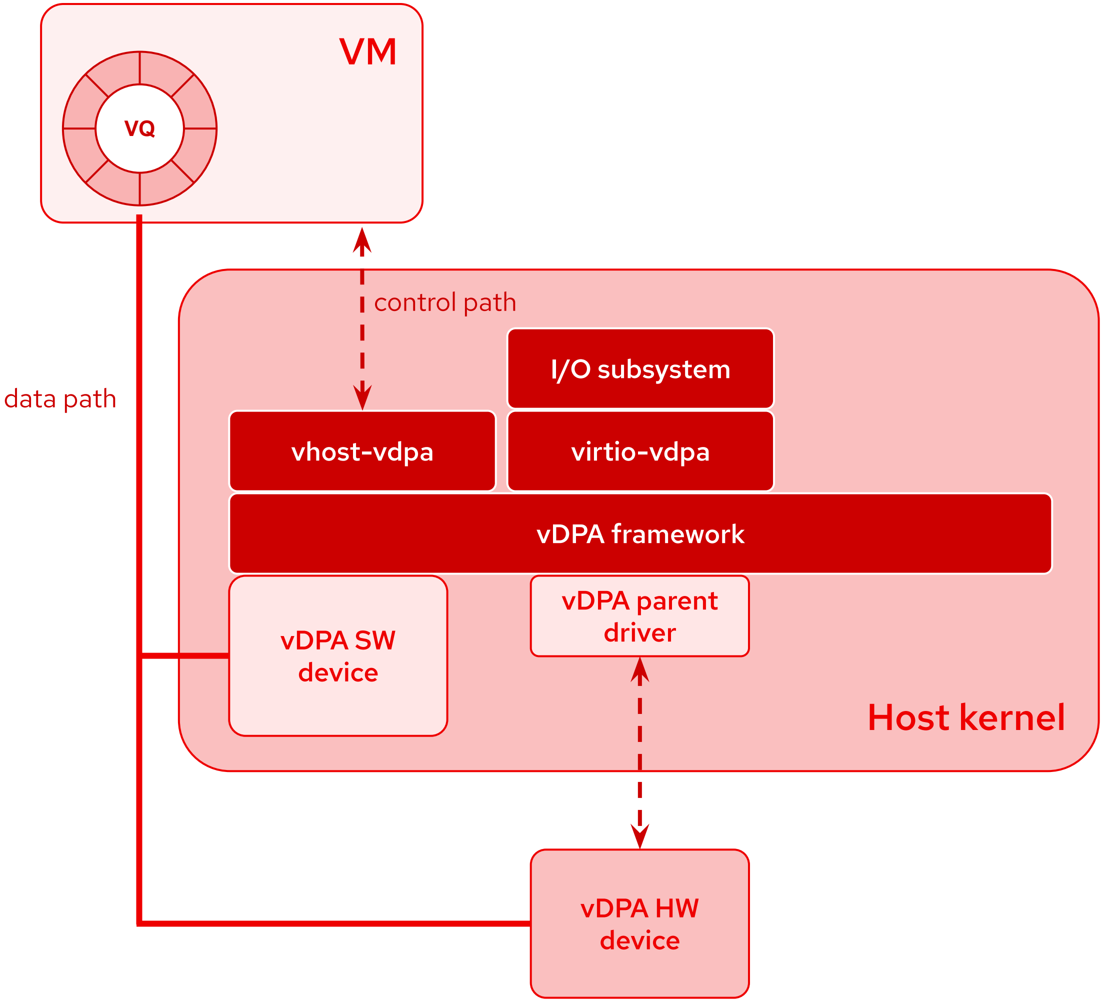
vhost-blk vs io_uring passthrough



from Speeding Up VM's I/O Sharing Host's io_uring Queues With Guests - Stefano Garzarella @ KVM Forum 2020

virtio Data Path Acceleration (vDPA)

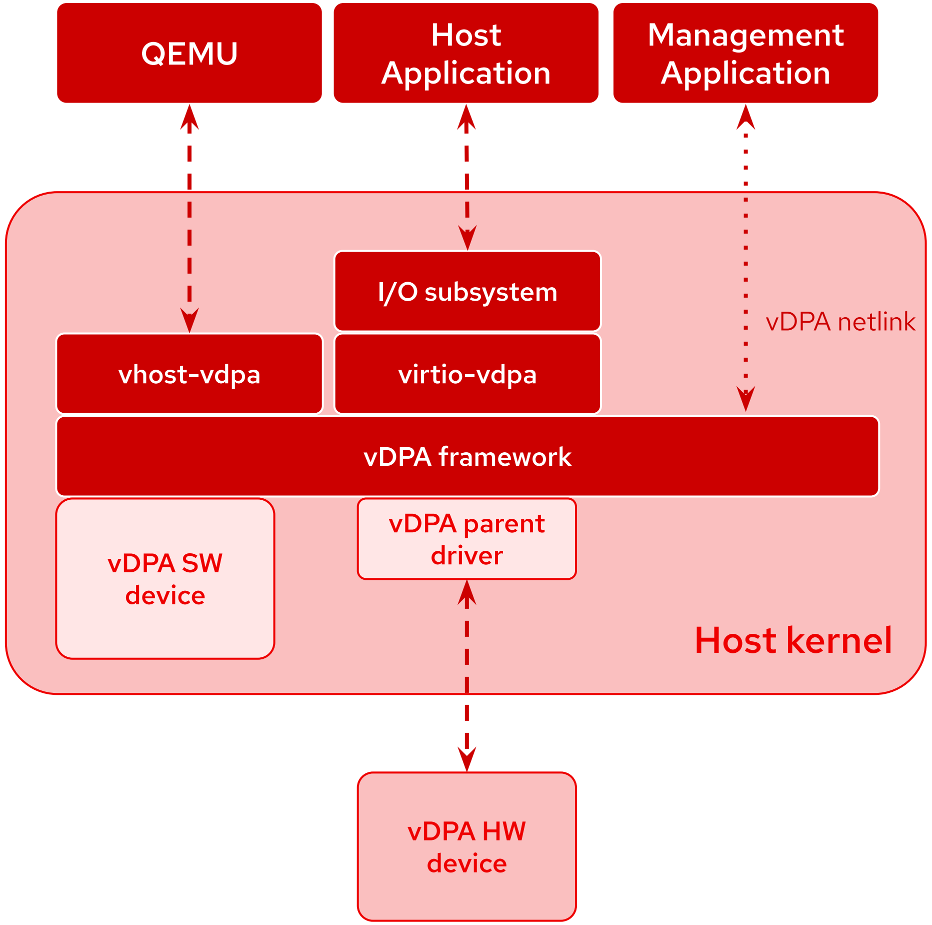
- vDPA device
  - **VIRTIO compliant data path**
  - **vendor specific control path**
    - small vDPA driver for the control part
- Designed for hardware accelerators
  - software accelerators also possible
  - guest memory locked
    - memory overcommit not supported yet
    - fast access to virtqueues
      - vhost needs `copy_in/copy_out`



vDPA Support in Linux Kernel - Jason Wang @ KVM Forum 2020

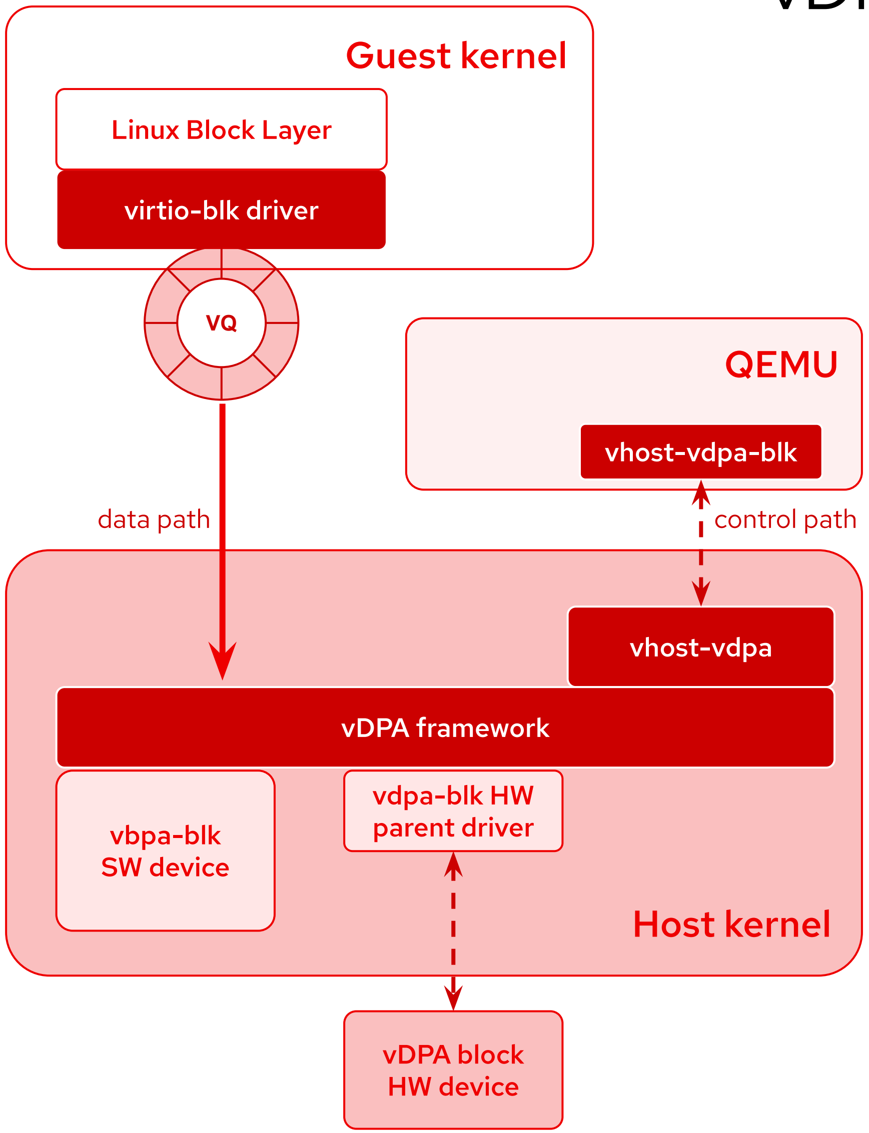
virtio Data Path Acceleration (vDPA)

- **Unified software stack for vDPA devices**
  - vhost-vdpa
    - interface for userspace/guest virtio driver
    - vhost generic uAPI + vhost-vdpa uAPI for full device abstraction
  - virtio-vdpa
    - interface for host virtio driver
    - bare metal or containerized applications
  - Management API



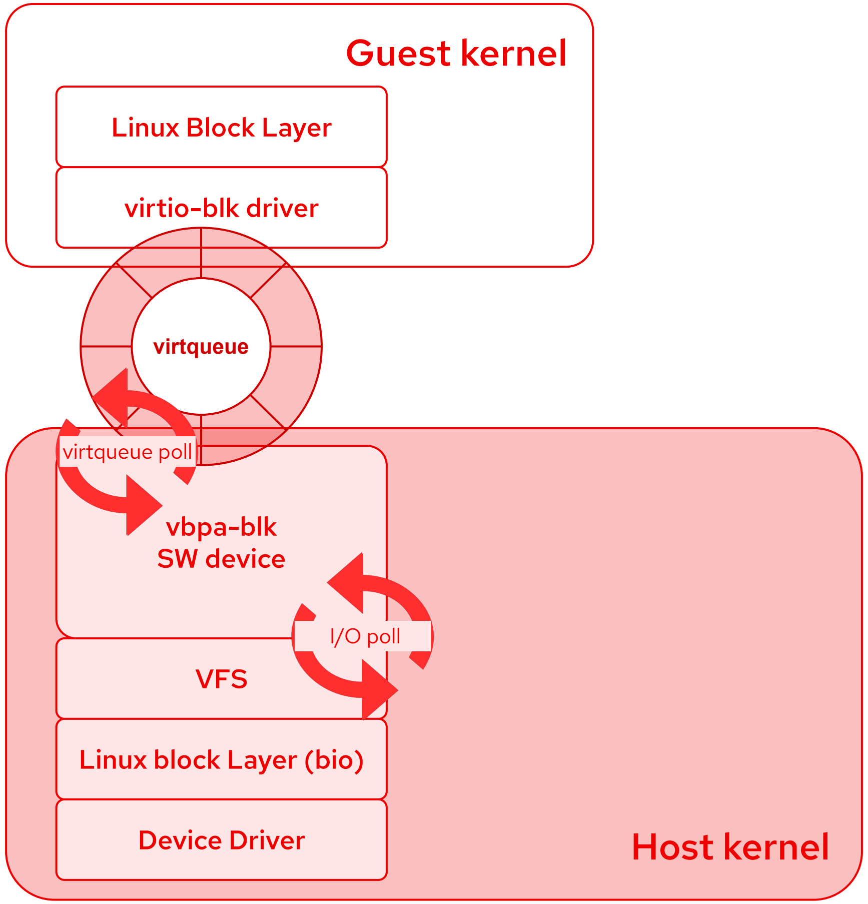
vDPA Support in Linux Kernel - Jason Wang @ KVM Forum 2020

vDPA block devices



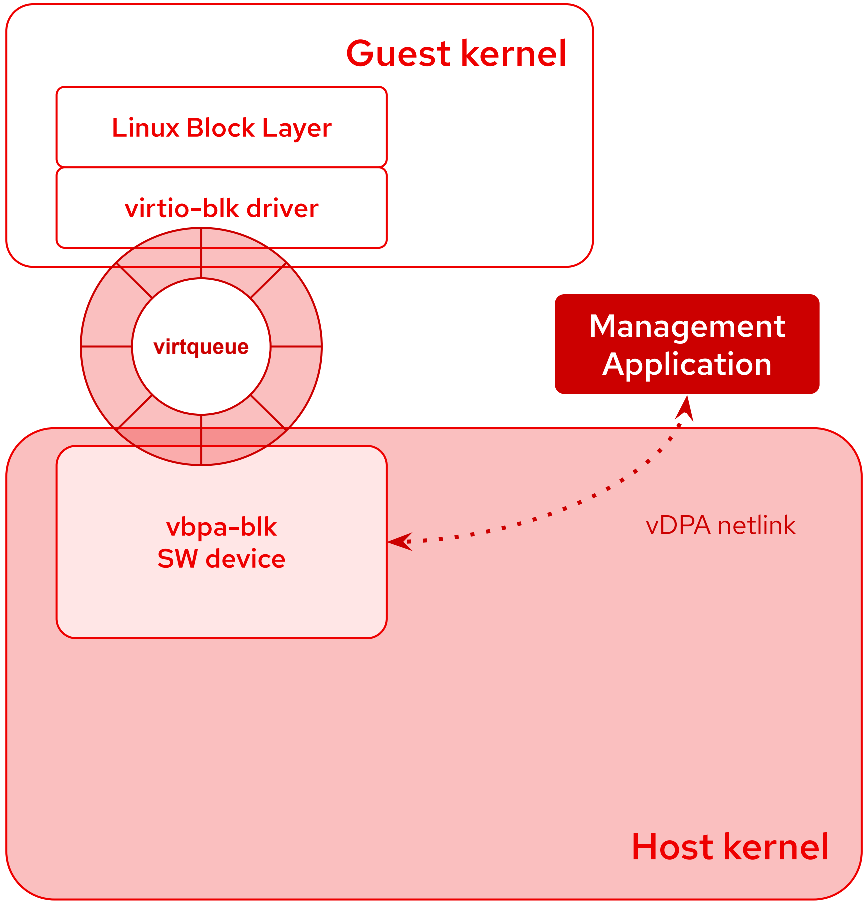
- Unified software stack for **software and hardware virtio-blk devices**
  - Guest kernel (virtio-blk device driver)
  - QEMU
  - Host kernel (vDPA framework, vhost-vdpa)
- Custom code
  - vDPA parent driver
    - custom hardware, Smart NIC, FPGA
  - vDPA software device
    - in-kernel virtio-blk device emulation

vDPA block: software device



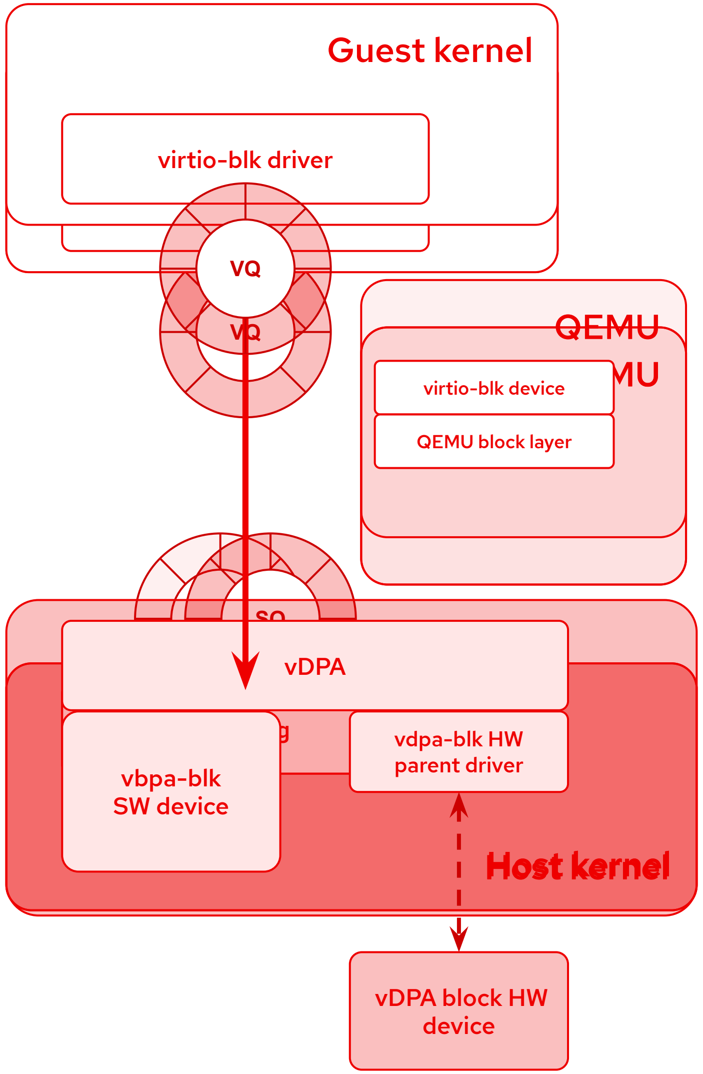
- VFS integration
  - virtqueue <-> VFS (struct kiocb)
- Fallback when HW accelerators are not available
- Polling
  - virtqueue polling (similar to io_uring's SQPOLL)
    - potentially guests can submit I/Os without vmexits
  - I/O polling (similar to io_uring's IOPOLL)
    - busy-waiting for an I/O completion
    - opposed to get notifications via an asynchronous IRQ
    - file system or block device must support polling

vDPA block: software device



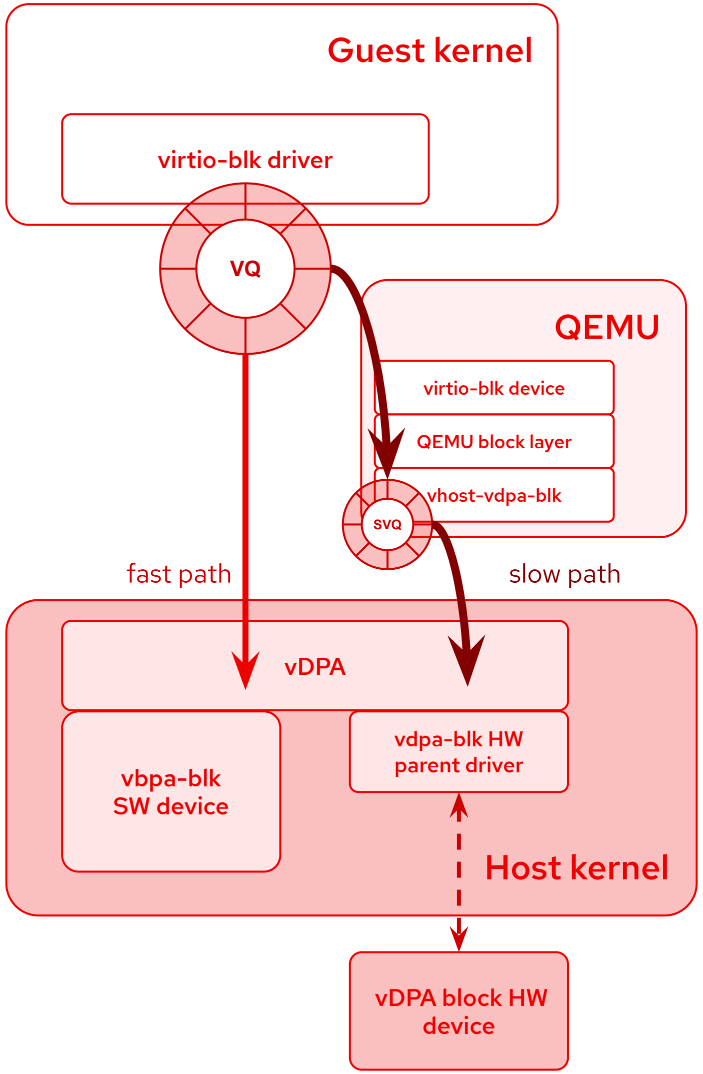
- Management API
  - based on vDPA netlink API
  - create/destroy vdpa-blk software devices
  - setup virtio parameters
    - virtqueues parameters (e.g. queue size)
    - virtio-blk configuration (e.g. block size)
- Custom API
  - attach to block devices / raw files
  - custom parameters (e.g. polling)

QEMU block layer



- Bypassed when using accelerators
  - Hardware
  - Software
- **QEMU storage virtualization features**
  - Image file formats (e.g. qcow2)
  - I/O throttling
  - Snapshot
  - Encryption
  - Incremental backup

QEMU auto-switching: fast and slow path

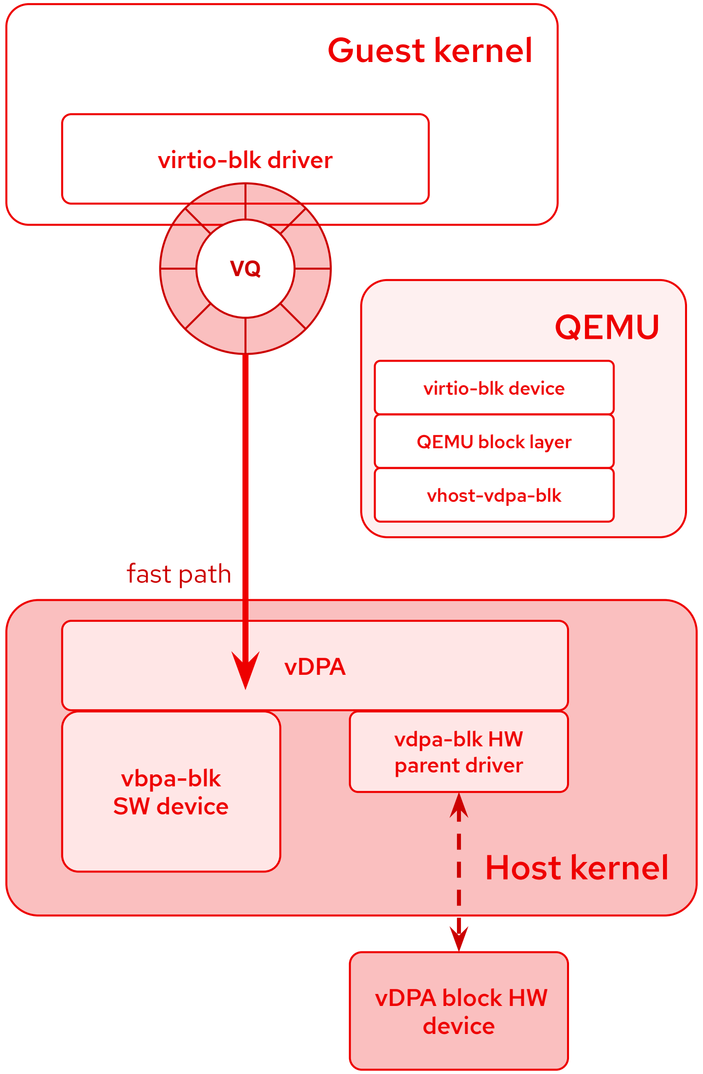


- **Fast path**
  - HW / SW vdpa-blk accelerators
    - raw files / block devices through software device
  - guest virtqueue (VQ) exposed directly to the vDPA device
- **Slow path**
  - QEMU storage features needed
  - Guest RAM overcommit
  - Live migration
  - QEMU processes guest virtqueue (VQ)
    - virtio-blk device already available
    - shadow virtqueue (SVQ) exposed to the vDPA device
      - Eugenio Pérez is working on this topic

        `[RFC v3 00/29] vDPA software assisted live migration`

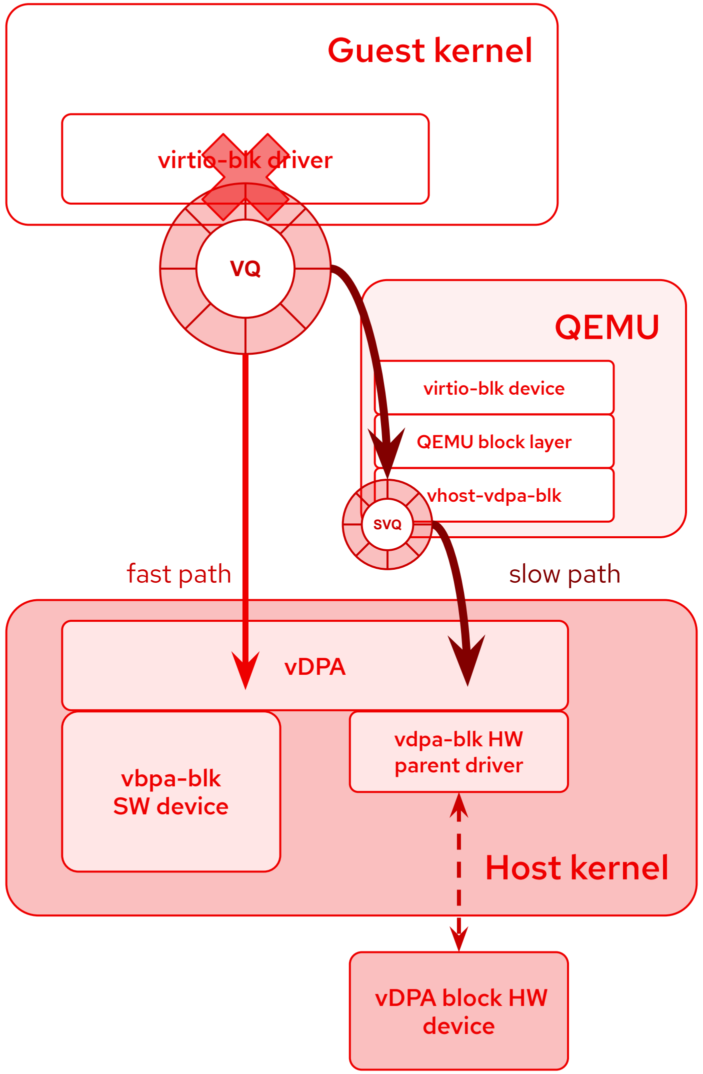
        https://lore.kernel.org/qemu-devel/20210519162903.1172366-1-eperezma@redhat.com/

Runtime switching between fast and slow path



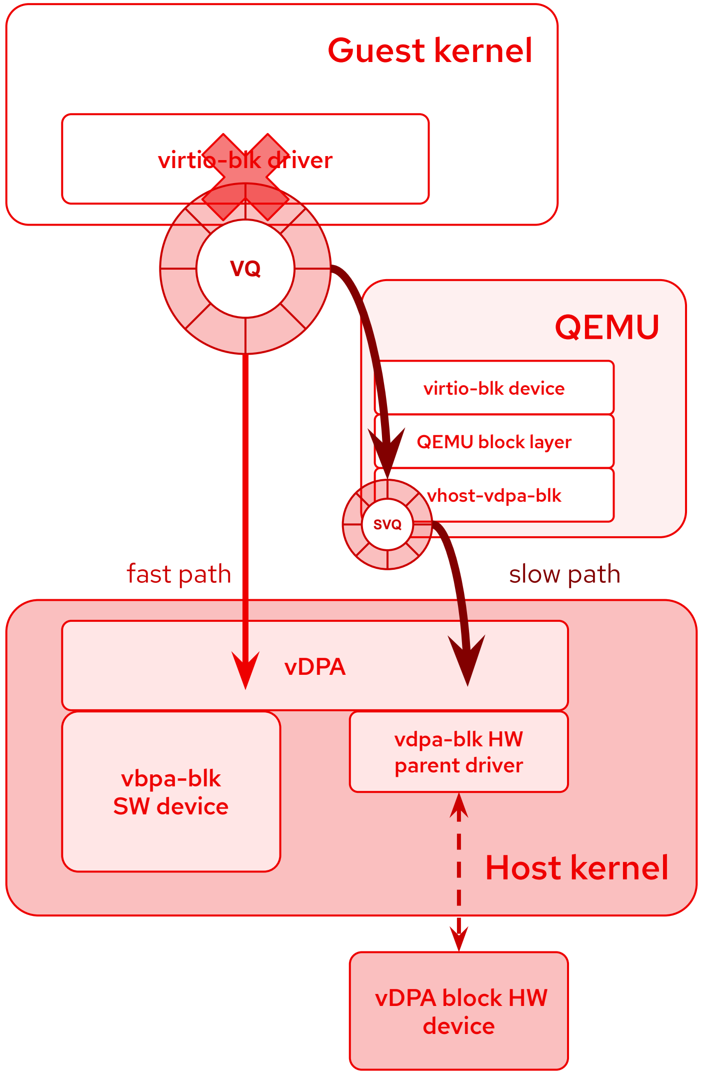
- Several operations may require switching at runtime
  - Live migration
  - I/O throttling
  - Snapshot
- Example
  - guest is using the fast path
    - QEMU storage features not needed
    - guest virtqueue (VQ) processed by vDPA device
  - an operation is requested where we need QEMU to process the virtqueue ...

Runtime switching between fast and slow path



- ... example continue
  - switch to slow path
    - stop guest driver from queuing new requests
    - wait for the vDPA device to complete all in-flight requests
    - QEMU takes over the Guest virtqueue (VQ)
      - QEMU allocates Shadow virtqueue (SVQ) and exposes it to the vDPA device
    - re-start guest driver to queue new request
  - operation is terminated (e.g. live migration) or is no longer required (e.g. I/O throttling) ...

Runtime switching between fast and slow path



- … example continue
  - switch back to fast path
    - stop guest driver from queuing new requests
    - wait for in-flight requests to complete
    - QEMU passes control of guest virtqueue (VQ) to the vDPA device
    - re-start guest driver to queue new request

vdpa-blk: current status and next steps

- Merged upstream
  - vDPA block simulator in the Linux kernel 5.13+
- **Work to do** (collaborations are welcome :-)
  - Linux
    - vdpa-blk software device
  - QEMU
    - vdpa-blk support
    - fast / slow path auto-switching
  - Hardware (custom accelerators, Smart NICs, FPGA)
- **Join us and take advantage of the vdpa-blk stack!**
  - **https://vdpa-dev.gitlab.io**

Thank you!

Stefano Garzarella <**sgarzare@redhat.com**>

Blog: [https://**stefano-garzarella.github.io/**](https://stefano-garzarella.github.io/)

IRC: **sgarzare** on #qemu irc.oftc.net

in linkedin.com/company/red-hat

f facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat