qemu-storage-daemon and libblkio

Exploring new shores for the QEMU block layer

Kevin Wolf <kwolf@redhat.com>
Stefano Garzarella <sgarzare@redhat.com>

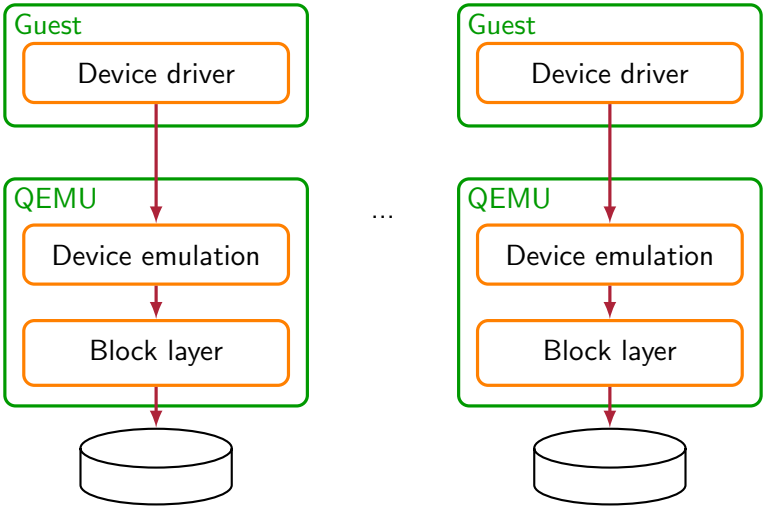
KVM Forum 2022

Section 1

What we have today

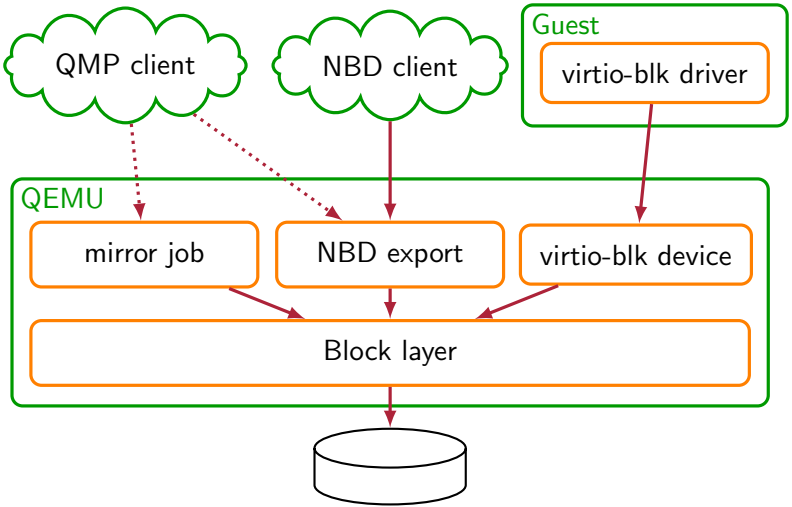
Red Hat

Traditional use of the QEMU block layer



Red Hat

Slightly less traditional use



Red Hat

Summary

- Everything runs in the QEMU process
- Serves a single VM
- Only available while the VM is running
- Sharing images: For read-only backing files
  No snapshot deletion etc. in the shared chain

Section 2

qemu-storage-daemon

Red Hat

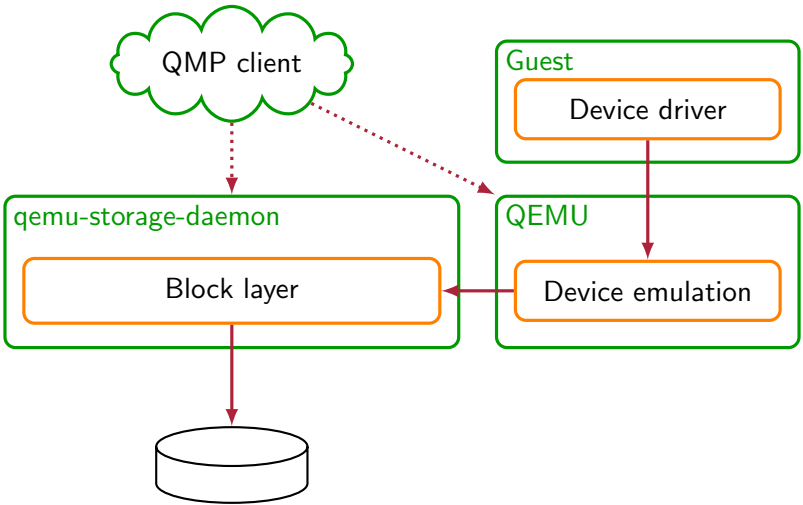
What is qemu-storage-daemon?

- QEMU block layer functionality without a VM
  - More features than qemu-nbd
  - Smaller footprint than QEMU
- Supports the relevant subset of QMP
- Export block devices to QEMU or other clients

```
qemu-storage-daemon
--blockdev file,filename=test.raw,node-name=disk
--export vhost-user-blk,id=exp0,node-name=disk,
         addr.type=unix,addr.path=/tmp/vhost.sock
```

Red Hat

Now we can do this instead



Red Hat

Why should we? – Isolation

- Storage code runs in a separate process
- Less privileges needed on both sides
  e.g. SELinux policies can be more specific
- Potential vulnerabilities are more contained

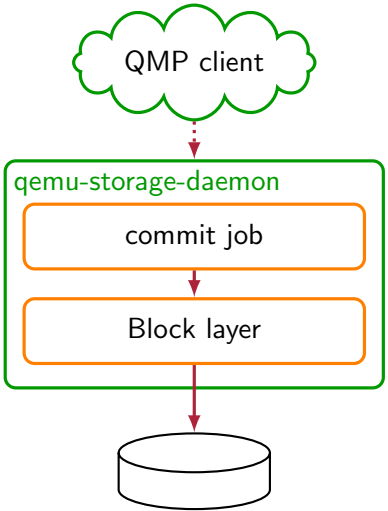
Red Hat

Why should we? – Separation of concerns

- Managing storage is a complex task
- Managing VMs is a separate complex task
- No need to couple both tightly
- e.g. Kubevirt leaves storage to external entity
- Separation hardly possible with both in one QEMU process

Red Hat

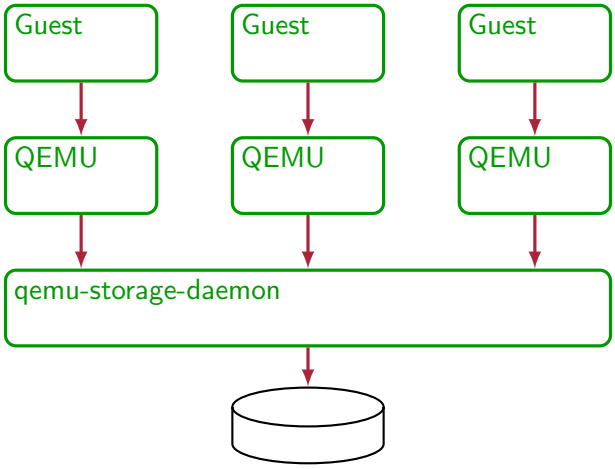
Why should we? – Offline block jobs



- Job can run while VM is shut down
- Export to QEMU can be added when VM is started

Red Hat

We can also serve multiple VMs



Red Hat

Why should we? – Sharing a backing chain

- Multiple VMs based on the same backing image
- Backing chain opened by multiple QEMU processes is read-only
- Opening them once in QSD allows e.g. deleting snapshots

Red Hat

Why should we? – Sharing a CPU for polling

- Polling can help in high performance use cases
- Requires setting a CPU aside per I/O thread
- Multiple VMs can share one I/O thread now

Red Hat

Why should we? – Sharing a hardware device

- e.g. the NVMe userspace driver in QEMU
- Split a single disk for multiple VMs
- Multiple disks in a single IOMMU group

Red Hat

Why should we? – Attaching to non-VMs

- Export storage to be mounted on the host
- Attach directly to an application
- Bring QEMU storage functionality to containers

Section 3

Block exports

Red Hat

NBD

- Export storage over the network
- Has existed for many years in QEMU
- Usually involved in live storage migration
- Not suitable for high performance use cases

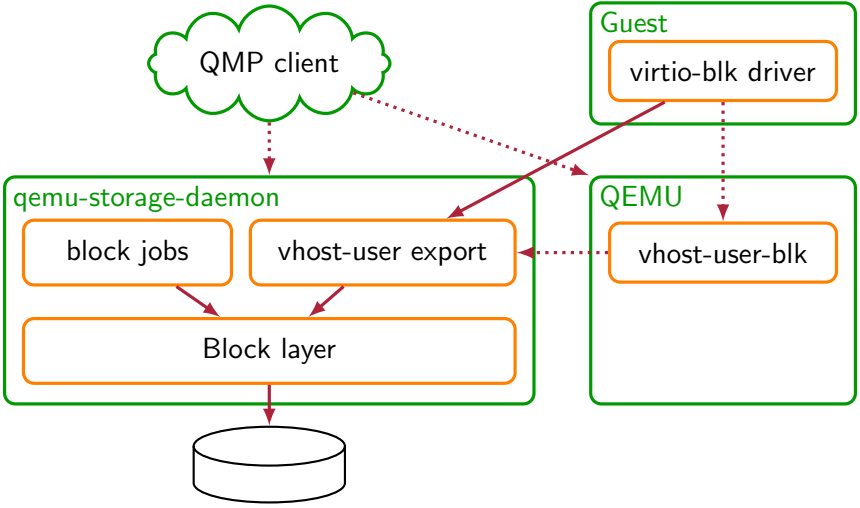
Red Hat

FUSE

- Mount a QEMU block backend as a host file
- Introduced in QEMU 6.0
- Works, but still fully synchronous

Red Hat

vhost-user-blk (with virtio-blk guest device)



Red Hat

VDUSE

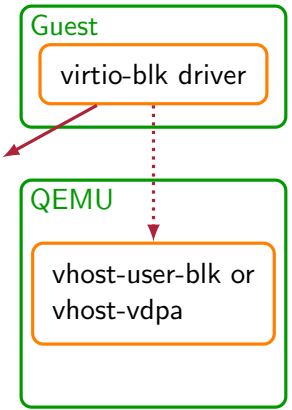
- Expose a QEMU block backend as a vDPA block device
- **v**DPA **d**evice in **use**rspace
  - Based on vDPA kernel framework
  - Software-emulated vDPA device in userspace
  - **Support both VM and container workloads**
- Introduced in QEMU 7.1
  - Linux $\geq$ 5.15 required (`vduse.ko`)

Section 4

libblkio

Red Hat

virtio-blk exports: vhost-user and VDUSE



- QEMU's block layer bypassed
  - virtio-blk device directly exposed to the guest
  - e.g. block device emulation (e.g. ide-hd) not supported
- How to support QEMU's block layer?
  - **QEMU needs a virtio-blk driver** to access the device

Red Hat

libblkio

- Introducing the libblkio High-performance Block I/O API
  - Stefan Hajnoczi & Alberto Faria, Red Hat
  - Wednesday, September 14 / 17:30 - 18:10 @ Liffey A
- Single API for efficiently accessing block devices
  - https://libblkio.gitlab.io/libblkio/
  - Supported drivers
    - Linux io_uring
    - NVMe (io_uring cmd)
    - **virtio-blk** (vhost-user, vhost-vdpa, and VFIO PCI)

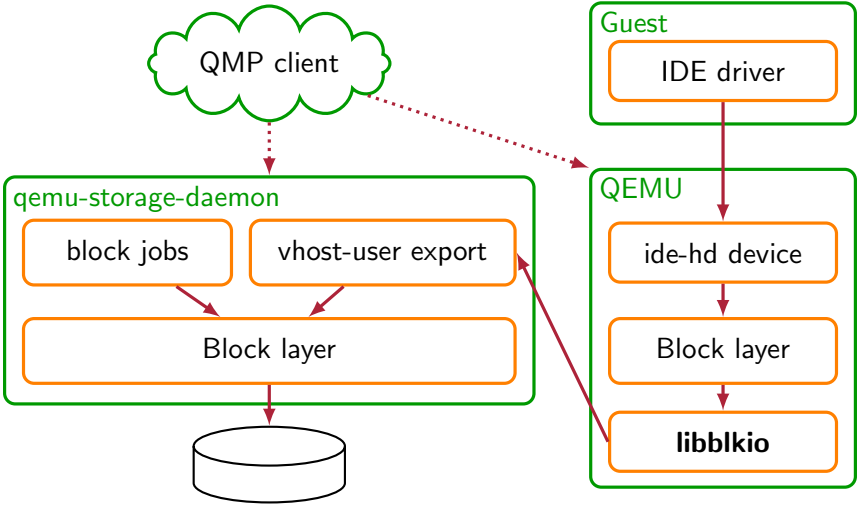
Red Hat

libblkio: virtio-blk drivers

- Applications can use **libblkio API** to access virtio-blk devices
  - Configuration and data path abstracted
  - Queue requests directly to virtio-blk devices
  - No need to implement the virtio spec
- virtio-blk drivers
  - **virtio-blk-vhost-user**: vhost-user front-end implementation to communicate with vhost-user back-ends (e.g. QSD)
  - **virtio-blk-vhost-vdpa**: vhost-vdpa is the interface to access vDPA device from userspace

Red Hat

vhost-user-blk QSD export (with libblkio)



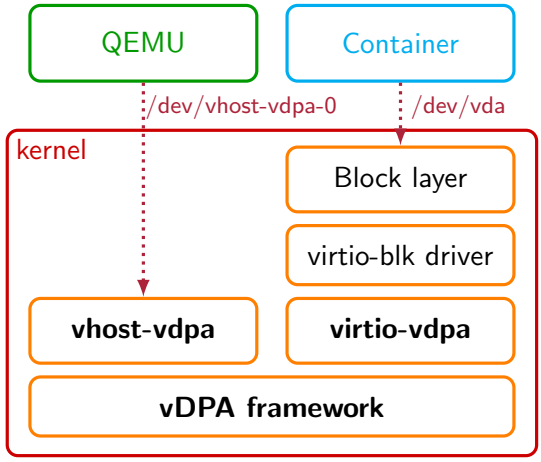
Section 5

vDPA

Red Hat

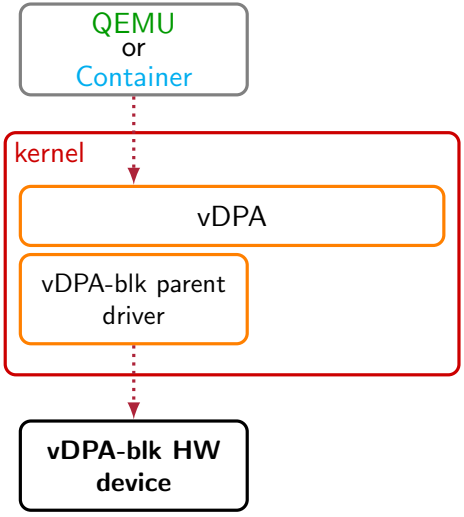
vDPA: virtio Data Path Acceleration

- Hardware and software virtio accelerators
  - Virtio spec compliant data path
  - Vendor specific control path
- vDPA BUS drivers
  - vhost-vdpa: VM workloads
  - virtio-vdpa: container workloads
- https://vdpa-dev.gitlab.io/



Red Hat

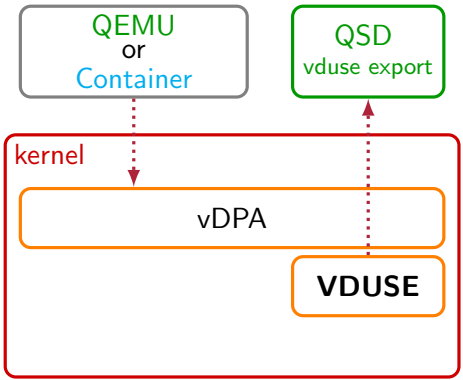
vDPA: hardware device



- Best performance
- **SmartNICs**
  - Network block protocols (Ceph RBD, iSCSI, etc.)
- Require only a small driver for the control path

Red Hat

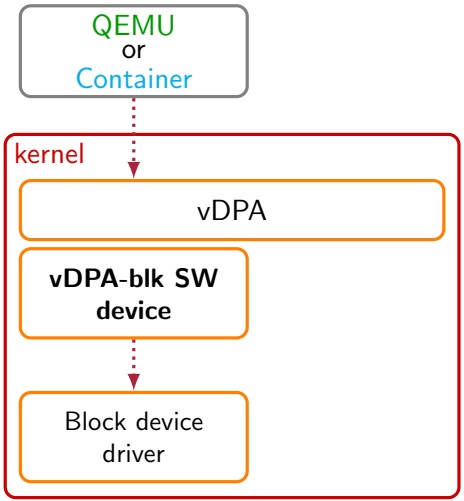
vDPA: software device in userspace



- **VDUSE**
- Security, flexibility
- Similar to vhost-user, but it can serve both **VM** and **container** workloads
- Introduced in Linux 5.15

Red Hat

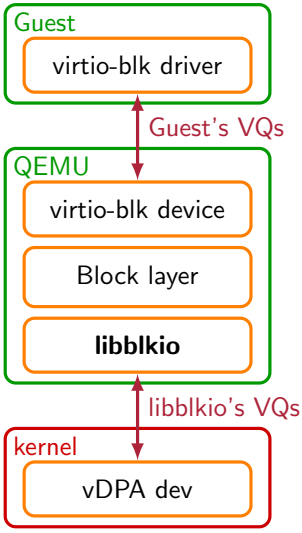
vDPA: software device in kernel



- Similar to vhost devices, but we can **reuse** the **vDPA software stack** for both HW and SW accelerators
- Good performance when vDPA is not supported by the HW
- Work in progress

Red Hat

vDPA & libblkio (VM workloads)



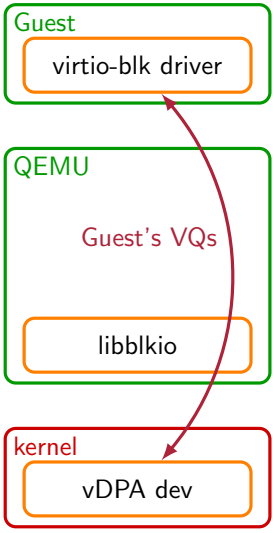
- QEMU storage virtualization features available
- QEMU can emulate any block device
  - virtio-blk: 2 virtqueues
    1. guest driver ↔ QEMU device emulation
    2. libblkio ↔ vDPA device
- **Slow path**

Section 6

Future plans

Red Hat

libblkio: virtio-blk queue passthrough



- libblkio will provide API for virtio-blk queue passthrough
- If QEMU block layer is not needed (**fast-path**)
  - Guest's VQs can be exposed to the device (**vDPA**, **vhost-user**)
  - Fast and slow path auto-switching
- Work in progress

Red Hat

Future plans

- virtio-blk queue passthrough in libblkio & QEMU
- Storage daemon support in libvirt
- Potentially alternative storage daemon
  - Chance to move to Rust
- vDPA in-kernel software device
  - PoC showed good results, comparable with the last attempt to implement vhost-blk