Can QEMU and vhost-user devices be used on macOS and *BSD?

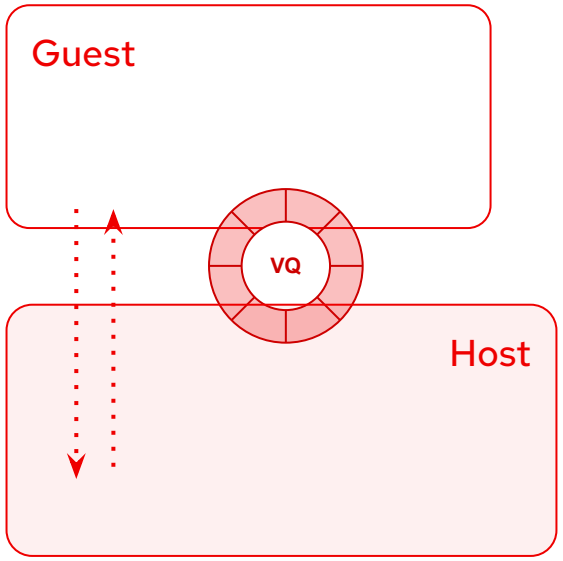
FOSDEM 2025

Stefano Garzarella

<sgarzare@redhat.com>

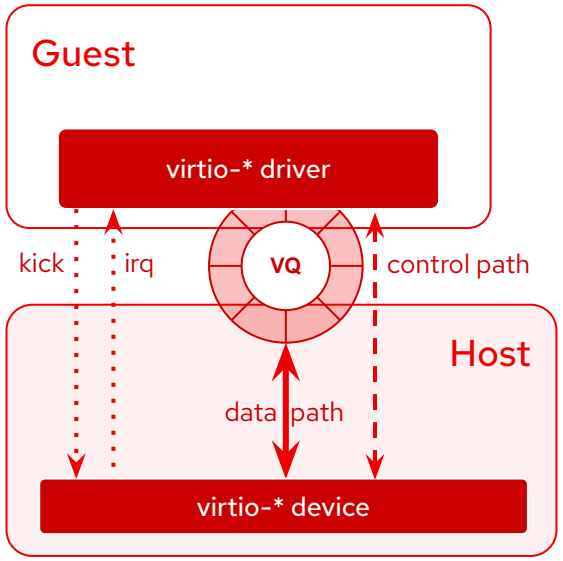
Red Hat

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.*
- 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.)

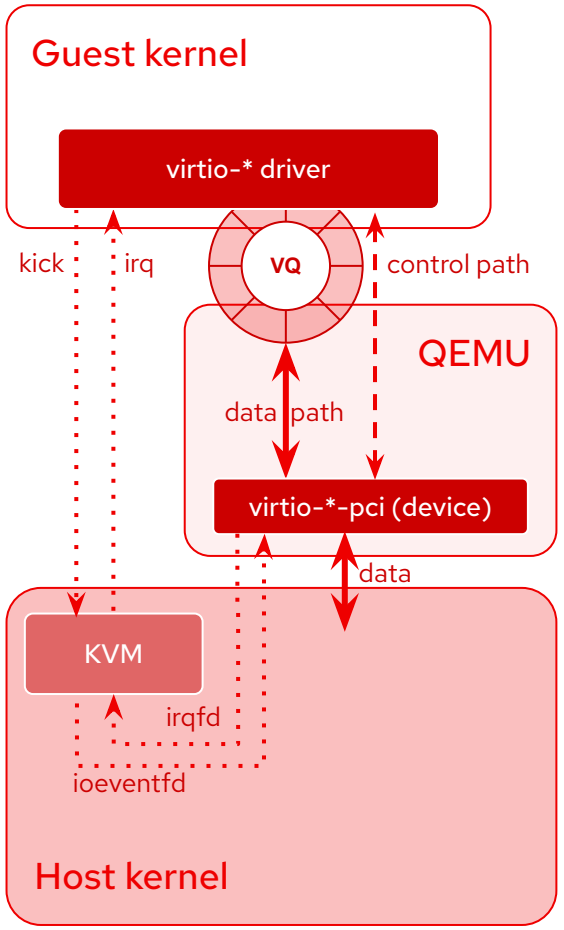
VIRTIO device & driver



- 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

Red Hat

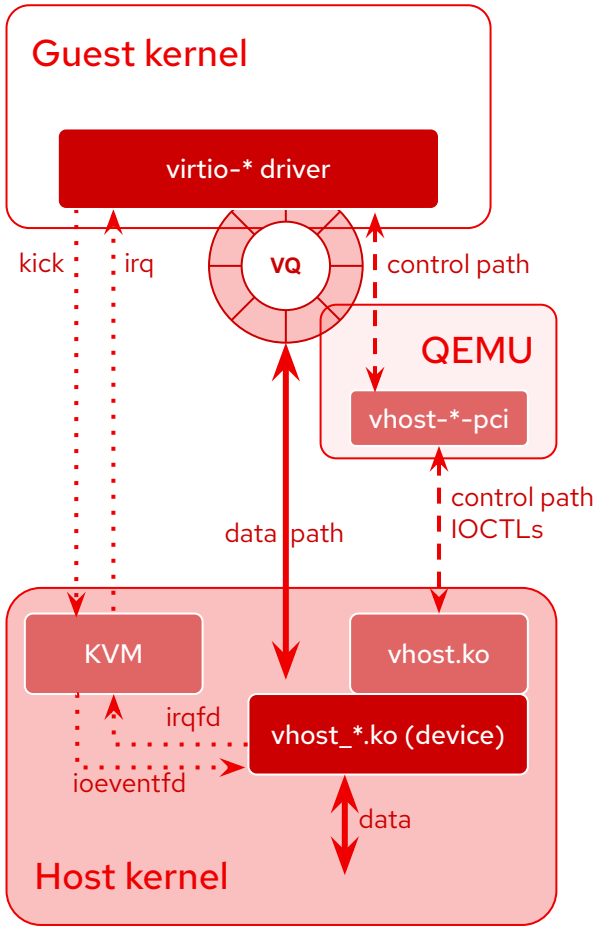
VIRTIO device emulated by the VMM



- Common scenario
- QEMU
  - de facto reference implementation for VIRTIO devices
- virtqueue
  - guest memory fully accessible by the VMM
- notifications
  - KVM
    - ioeventfd/irqfd

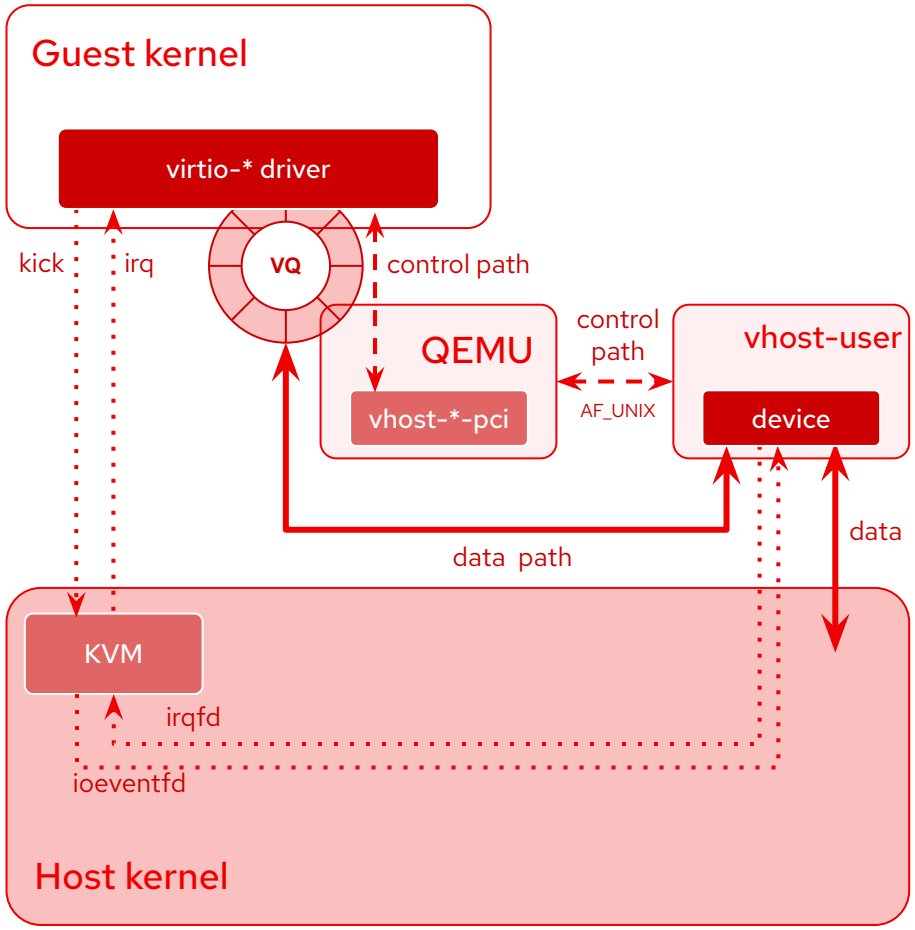
Red Hat

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
- Pros
  - Performance (less syscalls per request)
  - Easily to integrate with host kernel stacks (e.g. AF_VSOCK)
- Cons
  - Linux-specific
  - Safety
  - Device updates

vhost-user: VIRTIO device in an external process

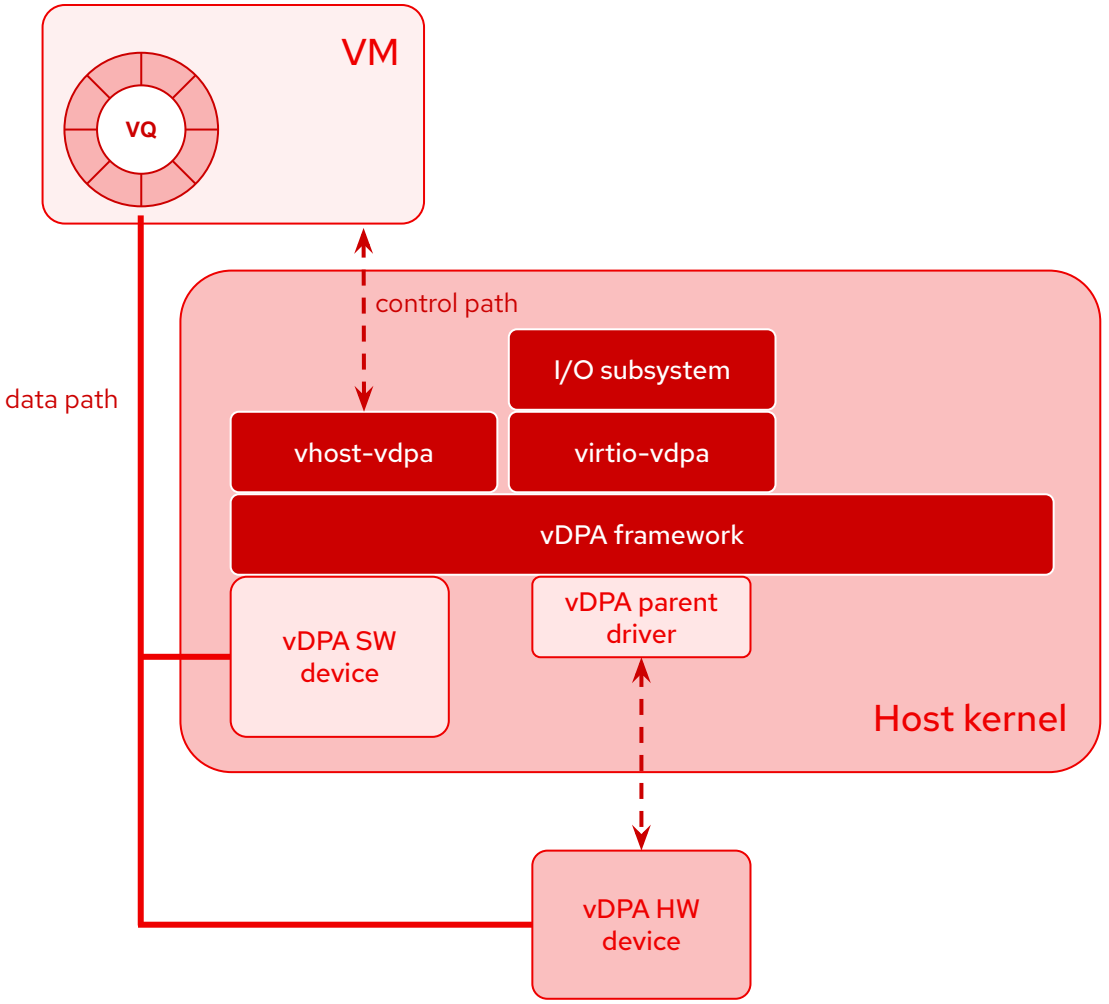


- Inspired by vhost
  - Control path
    - AF_UNIX
  - Data path
    - Shared memory through fd sharing (memfd, /dev/shm, etc.)
- Pros
  - Safety
  - Device updates
  - Different language from VMM (e.g. Rust)
  - More isolation
- Cons
  - Similar performance of in-VMM device
  - More coordination
    - can be hidden by management layer (e.g. libvirt)
  - ~~Linux-specific~~

Red Hat

vDPA: VIRTIO device in hardware

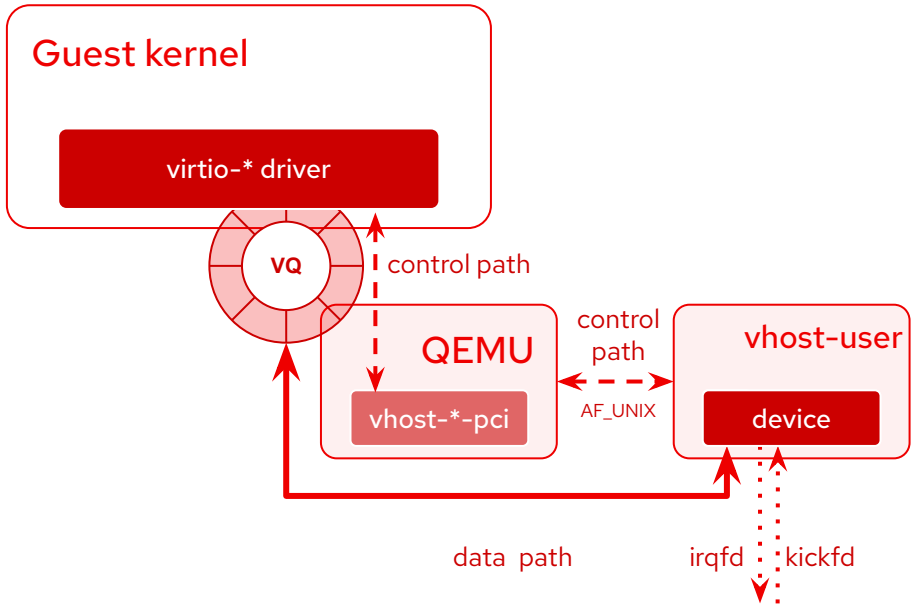
- virtio Data Path Acceleration
  - VIRTIO compliant data path
  - vendor specific control path
    - small vDPA driver for the control part
  - https://vdpa-dev.gitlab.io
- Designed for hardware accelerators
  - software accelerators also possible
- Unified software stack for vDPA devices
  - vhost-vdpa
    - interface for userspace/guest virtio driver
  - virtio-vdpa
    - interface for host virtio driver
    - bare metal or containerized applications



Picture from vdpa-blk: Unified Hardware and Software Offload for virtio-blk - Stefano Garzarella @ KVM Forum 2021

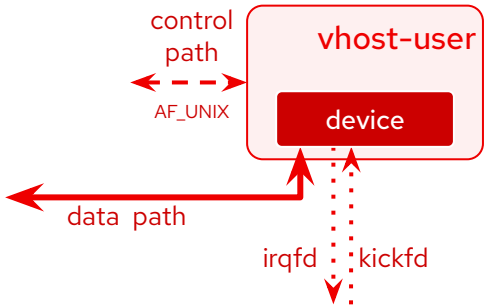
vhost-user protocol

- 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.



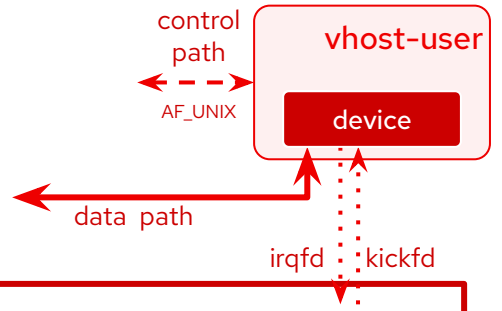
Red Hat

vhost-user on POSIX



- UNIX domain sockets
  - sys/un.h – definitions for UNIX domain sockets
    - https://pubs.opengroup.org/onlinepubs/9799919799/basedefs/sys_un.h.html
  - *The **sockaddr_un** structure is used to store addresses for UNIX domain sockets.*
- Ancillary data support
  - sys/socket.h – main sockets header
    - https://pubs.opengroup.org/onlinepubs/9799919799/basedefs/sys_socket.h.html
  - *The **cmsghdr** structure is used for storage of ancillary data object information.*
  - ***SCM_RIGHTS** Indicates that the data array contains the access rights to be sent or received.*
- Shared memory
  - shm_open – open a shared memory object
    - https://pubs.opengroup.org/onlinepubs/9799919799/functions/shm_open.html
  - *The **shm_open()** function shall establish a connection between a **shared memory** object and a **file descriptor**.*
- Notifications
  - pipe, pipe2 – create an interprocess channel
    - https://pubs.opengroup.org/onlinepubs/9799919799/functions/pipe.html
  - *The **pipe()** function shall create a pipe and place two file descriptors, … that refer to the open file descriptions for the read and write ends of the pipe, respectively.*

vhost-user on POSIX



- UNIX domain sockets
  - sys/un.h – definitions for UNIX domain sockets
    - https://pubs.opengroup.org/onlinepubs/9799919799/basedefs/sys_un.h.html
  - *The **sockaddr_un** structure is used to store addresses for UNIX domain sockets.*

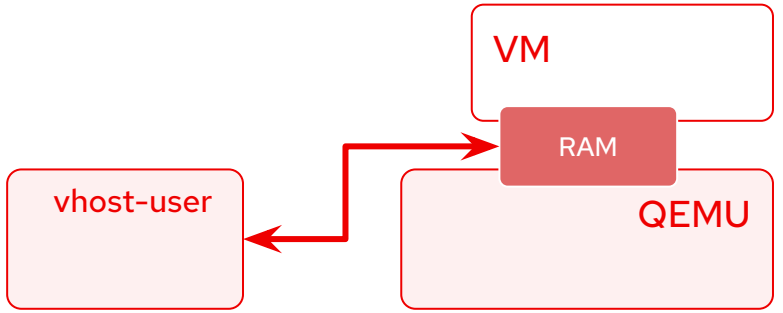
- Ancillary data support
  - sys/socket.h – main sockets header
    - https://pubs.opengroup.org/onlinepubs/9799919799/basedefs/sys_socket.h.html
  - *The **cmsghdr** structure is used for storage of ancillary data object information.*
  - ***SCM_RIGHTS** Indicates that the data array contains the access rights to be sent or received.*

QEMU did not support

- Shared memory
  - shm_open – open a shared memory object
    - https://pubs.opengroup.org/onlinepubs/9799919799/functions/shm_open.html
  - *The **shm_open()** function shall establish a connection between a **shared memory** object and a **file descriptor**.*

- Notifications
  - pipe, pipe2 – create an interprocess channel
    - https://pubs.opengroup.org/onlinepubs/9799919799/functions/pipe.html
  - *The **pipe()** function shall create a pipe and place two file descriptors, … that refer to the open file descriptions for the read and write ends of the pipe, respectively.*

QEMU memory backends



- QEMU main RAM
  - `-m 512M`
  - `-machine …,memory-backend='id'`
- Memory backends support **sharing access to guest RAM**
  - `share` boolean option determines whether the memory region is marked as private to QEMU, or shared.
- Memory backends supported by QEMU
  - `-object memory-backend-ram`
    - legacy RAM backend, same as `-m 512M` but with more control
  - `-object memory-backend-file`
    - memory file backend object, which can be used to back the guest RAM on
      - files
      - shared memory or huge page filesystem mounts (e.g. `/dev/shm`, `/dev/hugepages`, etc.)
        - Linux specific
  - `-object memory-backend-memfd`
    - anonymous memory file backend object
      - allows QEMU to share the memory with an external process
    - Linux only: `memfd_create(2)`
    - `share` boolean option is on by default
  - **`-object memory-backend-shm`**
    - **POSIX shared memory** backend object - `shm_open(3)`
      - more portable and less featureful version of memory-backend-memfd
    - `share` boolean option is on by default
    - since QEMU v9.1.0 (see next slides)
  - others
    - `-object memory-backend-epc`
      - EPC (Enclave Page Cache) for Intel SGX

QEMU changes to support vhost-user on any POSIX

- [PATCH v8 00/13] vhost-user: support any POSIX system (tested on macOS, FreeBSD, OpenBSD)
  https://patchew.org/QEMU/20240618100043.144657-1-sgarzare@redhat.com/

fixes

```
qapi: clarify that the default is backend dependent
libvhost-user: set msg.msg_control to NULL when it is empty
libvhost-user: fail vu_message_write() if sendmsg() is failing
libvhost-user: mask F_INFLIGHT_SHMFD if memfd is not supported
vhost-user-server: do not set memory fd non-blocking
contrib/vhost-user-blk: fix bind() using the right size of the address
```

support POSIX

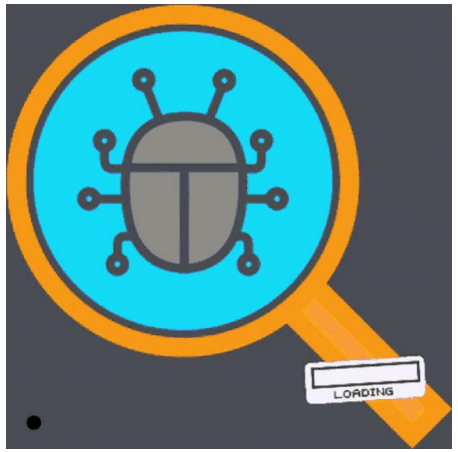
```
contrib/vhost-user-*: use QEMU bswap helper functions
vhost-user: enable frontends on any POSIX system
libvhost-user: enable it on any POSIX system
contrib/vhost-user-blk: enable it on any POSIX system
hostmem: add a new memory backend based on POSIX shm_open()
```

tests

```
tests/qtest/vhost-user-blk-test: use memory-backend-shm
tests/qtest/vhost-user-test: add a test case for memory-backend-shm
```

- Most of them are merged upstream
  - available since QEMU v9.1.0
  - red ones were not merged since CI was failing on FreeBSD

Issues with QEMU changes



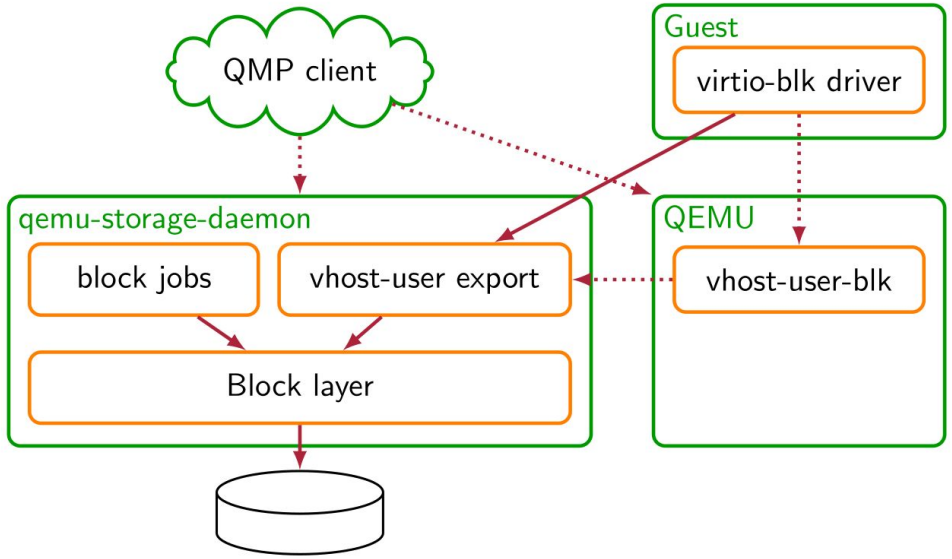
- FreeBSD/macOS qtests are failing
  - https://patchew.org/QEMU/20240618100043.144657-1-sgarzare@redhat.com/#suvpzkb3ppdodjdyo5zcpngz5mwrrlq3nfowemh4tqjghbc4si@hzrsr23fganv
  - `gmake --output-sync -j6 check-qtest-ppc64`
    - on FreeBSD x86_64 VM is failing every time, but check-qtest-aarch64 or check-qtest-x86_64 are working 🤷
  - `make --output-sync -j6 check-qtest-aarch64`
    - on macOS aarch64 host is failing frequently
- `vhost-user/reconnect` test
  - after the disconnection, the test doesn't receive `VHOST_USER_SET_MEM_TABLE` message
  - `wait_for_fds()` fails after the 5 sec timeout (increasing it doesn't help)

- https://gitlab.com/sgarzarella/qemu/-/tree/macos-vhost-user
  - Contains the missing patches rebased on recent master (Jan 2025)

```
vhost-user: enable frontends on any POSIX system
libvhost-user: enable it on any POSIX system
contrib/vhost-user-blk: enable it on any POSIX system
```

Try it: start vhost-user daemon first

- Let's use vhost-user with a virtio-blk device
  - QEMU repository includes 2 daemons that can expose a vhost-user-blk device
    - both of them can be executed on any POSIX system
      - https://gitlab.com/sgarzarella/qemu/-/tree/macos-vhost-user



- vhost-user-blk

```
$ make contrib/vhost-user-blk/vhost-user-blk
$ ./build/contrib/vhost-user-blk/vhost-user-blk -s /tmp/vhost.socket \
    -b Fedora-Cloud-Base-41-1.4.x86_64.raw
```

- qemu-storage-daemon

```
$ make storage-daemon/qemu-storage-daemon
$ ./build/storage-daemon/qemu-storage-daemon \
    --blockdev file,filename=Fedora-Cloud-Base-41-1.4.x86_64.qcow2,node-name=file \
    --blockdev qcow2,file=file,node-name=qcow2 \
    --export vhost-user-blk,addr.type=unix,addr.path=/tmp/vhost.socket,id=vub,num-queues=1,node-name=qcow2,writable=on
```


Picture from QEMU-Storage-Daemon and libblkio: Exploring New Shores for the QEMU Block Layer - Kevin Wolf & Stefano Garzarella @ KVM Forum 2022

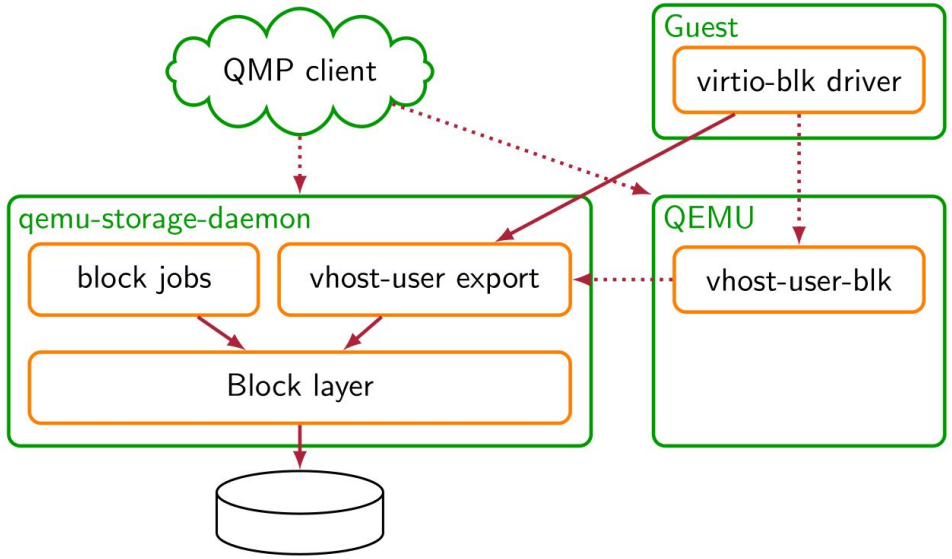
Try it!

- **Fedora Linux** 🐧 (x86_64): QEMU + KVM accelerator

```
$ qemu-system-x86_64 -smp 2 -M q35,accel=kvm,memory-backend=mem \
    -object memory-backend-shm,size="512M" \
    -device vhost-user-blk-pci,num-queues=1,chardev=char0 \
    -chardev socket,id=char0,path=/tmp/vhost.socket
```

- **FreeBSD** 😈**/OpenBSD** (x86_64): QEMU without accelerator

```
$ qemu-system-x86_64 -smp 2 -M q35,memory-backend=mem \
    -object memory-backend-shm,id=mem,size="512M" \
    -device vhost-user-blk-pci,num-queues=1,chardev=char0 \
    -chardev socket,id=char0,path=/tmp/vhost.socket
```



- **macOS** 🍏 (aarch64): QEMU + HVF accelerator

```
$ qemu-system-aarch64 -smp 2 -cpu host -M virt,accel=hvf,memory-backend=mem \
    -drive file=./build/pc-bios/edk2-aarch64-code.fd,if=pflash,format=raw,readonly=on \
    -device ramfb -device usb-ehci -device usb-kbd \
    -object memory-backend-shm,id=mem,size=512M \
    -device vhost-user-blk-pci,num-queues=1,chardev=char0 \
    -chardev socket,id=char0,path=/tmp/vhost.socket
```


Picture from QEMU-Storage-Daemon and libblkio: Exploring New Shores for the QEMU Block Layer - Kevin Wolf & Stefano Garzarella @ KVM Forum 2022

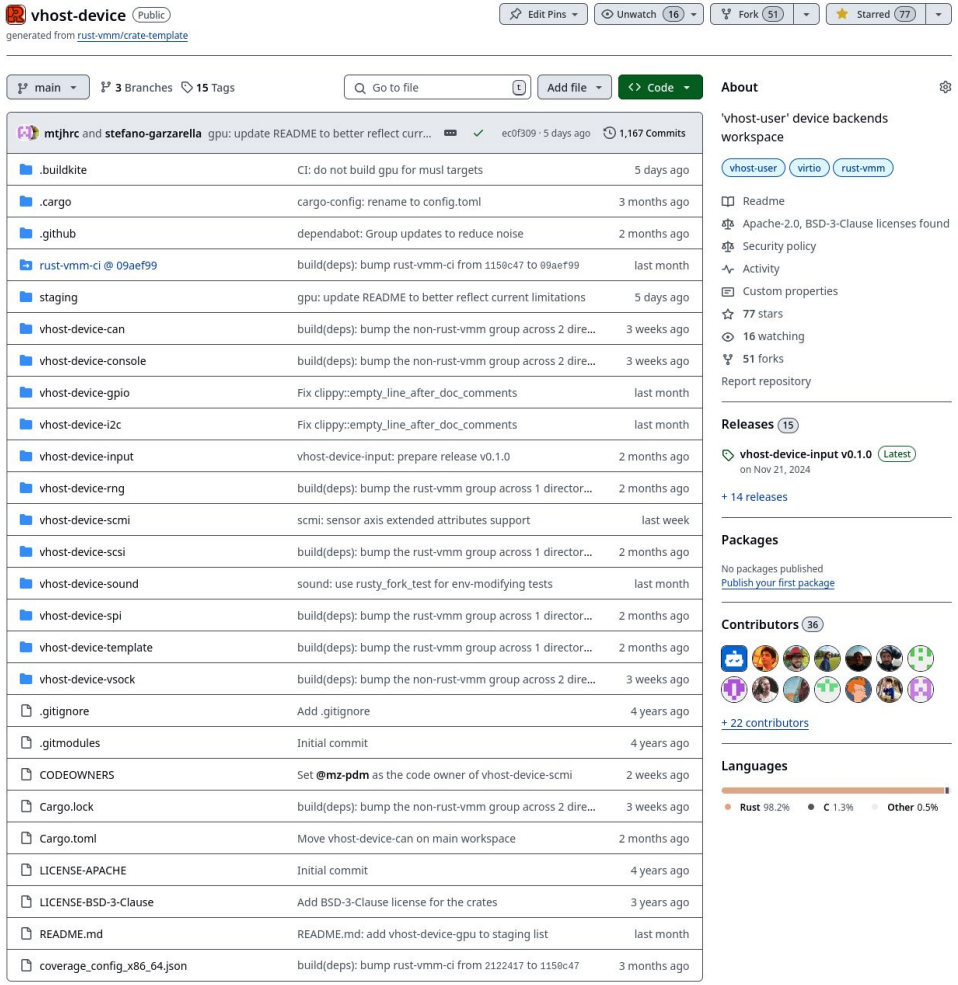
rust-vmm

- https://github.com/rust-vmm/community
  - ***open-source*** *project that empowers the community to* ***build custom*** *Virtual Machine Monitors (****VMMs****) and* ***hypervisors***
  - ***set of virtualization components*** *that any project can use to quickly develop virtualization solutions*
- Why rust-vmm?
  - Reduce code duplication
  - Faster development
  - Security & Testability
  - Clean interface
- Community Channels
  - Mailing list: http://lists.opendev.org/cgi-bin/mailman/listinfo/rust-vmm
  - Slack workspace: https://join.slack.com/t/rust-vmm/shared_invite/enQtODAxMzA2ODIyMTc2LWRhYjIwZmQ0YzUxODJlMTRhZWU2ZDBjYmJiNzBmOWVmYjg4MjY5YWRjYjM0YzQ5YzgyMTBmYzNlMjMzYmZlODU

Red Hat

rust-vmm: vhost-* crates

- vhost
  - A pure rust library for vDPA, vhost and vhost-user
- vhost-user-backend
  - A framework to implement vhost-user backend services
    - A daemon control object to start and stop the service daemon.
    - A vhost-user backend trait to handle vhost-user control messages and virtio messages.
    - A vring access trait to access virtio queues, and three implementations of the trait.
- vhost-user devices
  - https://github.com/rust-vmm/vhost-device
    - Supported devices:
      - can, console, gpio, i2c, input, rng, scmi, scsi, sound, spi, vsock
      - staging: gpu, video
  - https://gitlab.com/virtio-fs/virtiofsd
    - virtio-fs



Red Hat

rust-vmm crates on POSIX

Mac build support #110

Open

matejsp opened on Mar 18, 2022

Unable to build on mac Monterey 12.2.1. Any chance to get around this?

```
➜  virtiofsd git:(main) cargo build --release
   Compiling vhost v0.3.0
   Compiling futures-executor v0.3.19
   Compiling structopt v0.3.26
error[E0432]: unresolved import `vmm_sys_util::eventfd`
  --> /Users/myuser/.cargo/registry/src/github.com-1ecc6299db9ec823/vhost-0.3.0/src/backend.rs:16:19
   |
16 | use vmm_sys_util::eventfd::EventFd;
   |                   ^^^^^^^ could not find `eventfd` in `vmm_sys_util`

error[E0432]: unresolved import `vmm_sys_util::sock_ctrl_msg`
  --> /Users/myuser/.cargo/registry/src/github.com-1ecc6299db9ec823/vhost-0.3.0/src/vhost_user/connection.rs:18:1
   |
18 | use vmm_sys_util::sock_ctrl_msg::ScmSocket;
   |                   ^^^^^^^^^^^^ could not find `sock_ctrl_msg` in `vmm_sys_util`

   Compiling futures v0.3.19
error[E0599]: no method named `send_with_fds` found for struct `UnixStream` in the current scope
   --> /Users/myuser/.cargo/registry/src/github.com-1ecc6299db9ec823/vhost-0.3.0/src/vhost_user/connection.rs:142
    |
142 |         self.sock.send_with_fds(iovs, rfds).map_err(Into::into)
    |                   ^^^^^^^^^^^^ method not found in `UnixStream`

error[E0599]: no method named `recv_with_fds` found for struct `UnixStream` in the current scope
   --> /Users/myuser/.cargo/registry/src/github.com-1ecc6299db9ec823/vhost-0.3.0/src/vhost_user/connection.rs:317
    |
317 |         let (bytes, _) = unsafe { self.sock.recv_with_fds(&mut iovs, &mut [])? };
    |                                           ^^^^^^^^^^^^ method not found in `UnixStream`

error[E0599]: no method named `recv_with_fds` found for struct `UnixStream` in the current scope
   --> /Users/myuser/.cargo/registry/src/github.com-1ecc6299db9ec823/vhost-0.3.0/src/vhost_user/connection.rs:349
    |
349 |         let (bytes, fds) = self.sock.recv_with_fds(iovs, &mut fd_array)?;
    |                                       ^^^^^^^^^^^^ method not found in `UnixStream`

Some errors have detailed explanations: E0432, E0599.
For more information about an error, try `rustc --explain E0432`.
error: could not compile `vhost` due to 5 previous errors
warning: build failed, waiting for other jobs to finish...
error: build failed
```

Create sub-issue

- Not yet supported
  - https://github.com/rust-vmm/vmm-sys-util
    - *safe wrappers around common utilities for working with files, event file descriptors, ioctls and others*
    - Linux supported, Windows partially
    - POSIX support should be doable
      - `epoll(7)` can be replaced with https://github.com/smol-rs/polling
        - *Portable interface to epoll, kqueue, event ports, and IOCP*
      - `eventfd(2)`
        - https://github.com/rust-vmm/vmm-sys-util/blob/main/src/linux/eventfd.rs
        - `pipe()`/`pipe2()` fallback, like QEMU, can be implemented
- Open issues
  - vhost / vhost-user-backend
    - Mac build support #110 - rust-vmm/vhost
    - Consider changing the vhost-user-backend API #279 - rust-vmm/vhost
  - https://gitlab.com/virtio-fs/virtiofsd
    - Add macOS support #169 - virtio-fs/virtiofsd

Next steps

- QEMU
  - identify the issue with `vhost-user/reconnect` test
    - after that, we can post the missing patches upstream again
- rust-vmm crates
  - Improve POSIX support in vmm-sys-util
  - Replace Linux-specific syscalls (e.g. `epoll(7)`, `eventfd(2)`, etc.) in vhost-user-backend
  - Specific support for each vhost-user device
    - e.g. virtio-fs will need specific code to support other OSes besides Linux

Thank you!

Stefano Garzarella <sgarzare@redhat.com>

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

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat