virtio-gpu
Where are we now?

Implementing virtio-gpu in rust-vmm project

Dorinda Bassey
dbassey@redhat.com

Matej Hrica
mhrica@redhat.com

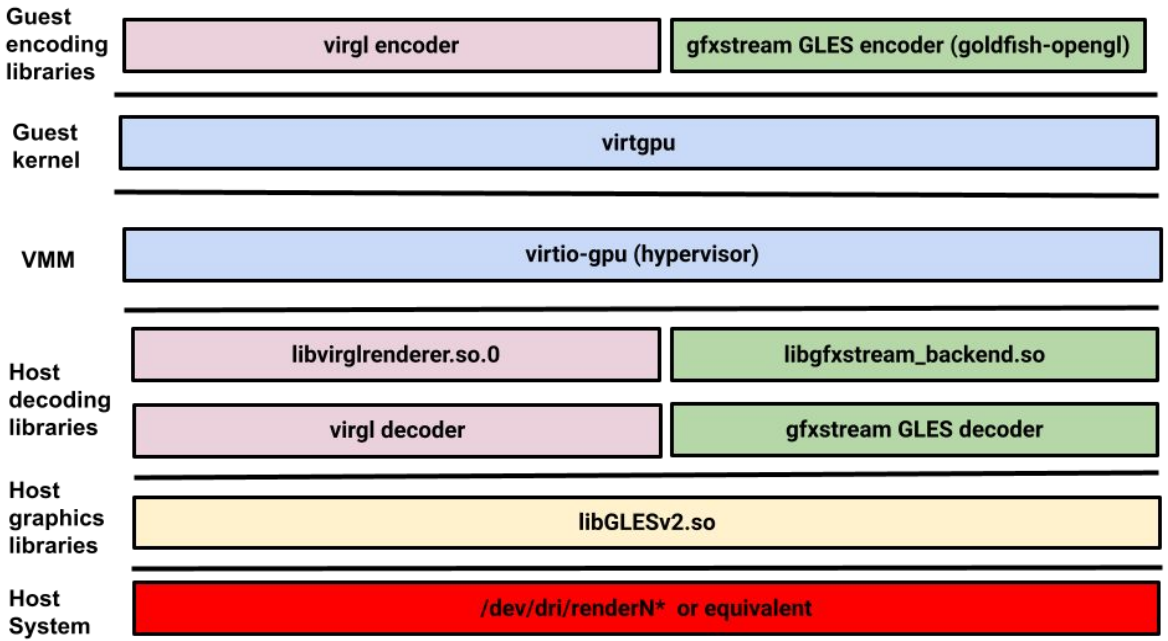
Red Hat

Automotive Goals

- Virtio-gpu enablement for Android virtualized workloads
  - Android Reference platform requirements
- Rely on virtio interfaces to allow automotive operating systems running on VMs to share the available GPU through virtio devices with minimal performance overhead.

Red Hat

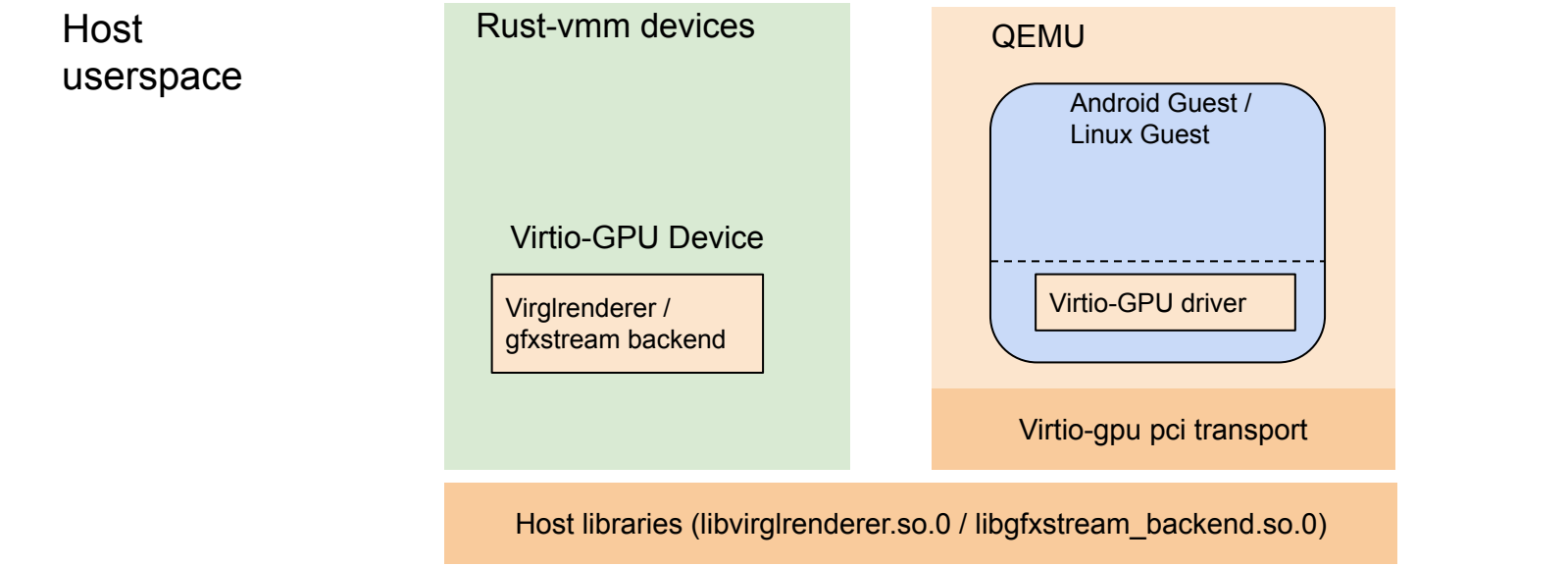
AAOS as a guest VM



https://source.android.com/docs/automotive/virtualization/architecture#graphics

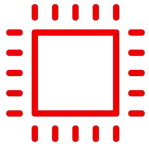
Red Hat

Architecture



Red Hat

Virtio-gpu in a nutshell



Paravirtualized GPU device

A virtualized GPU that allows guest VMs to use the host's GPU resources with minimal overhead.

Virtio Spec 1.2

The Specification defines the virtio-gpu device, providing a framework for paravirtualized graphics device in VMs
5.7 GPU Device

Components

Consists of the virtio-gpu driver, PCI bus transport, and the virtio-gpu device.
supports 2d and 3d graphical acceleration

Red Hat

Historical development of virtio-gpu

Device	QEMU 2.4 release virtio-gpu 2d mode virtio-gpu-pci support
Driver	Linux kernel 4.2 June - 2d mode Linux kernel 4.4 Oct - 3d / virgl support

2015

2018

Device	QEMU 2.5 release virtio-gpu 2d mode 3d acceleration + virglrenderer
Driver	Linux kernel 5.0 Nov - EDID feature
Spec	Virtio 1.1 2D mode VIRTIO_GPU_F_VIRGL VIRTIO_GPU_F_EDID

2019

Device

QEMU 4.0 release
EDID support

2020

Driver

Linux kernel 5.10
Sep - blob resource
Aug - UUID-based resource sharing

2021

Driver

Linux kernel 5.16
context initialization

2022

Spec

Virtio 1.2
3D commands support
VIRTIO_GPU_F_RESOURCE_UUID
VIRTIO_GPU_F_RESOURCE_BLOB
VIRTIO_GPU_F_CONTEXT_INIT

Red Hat

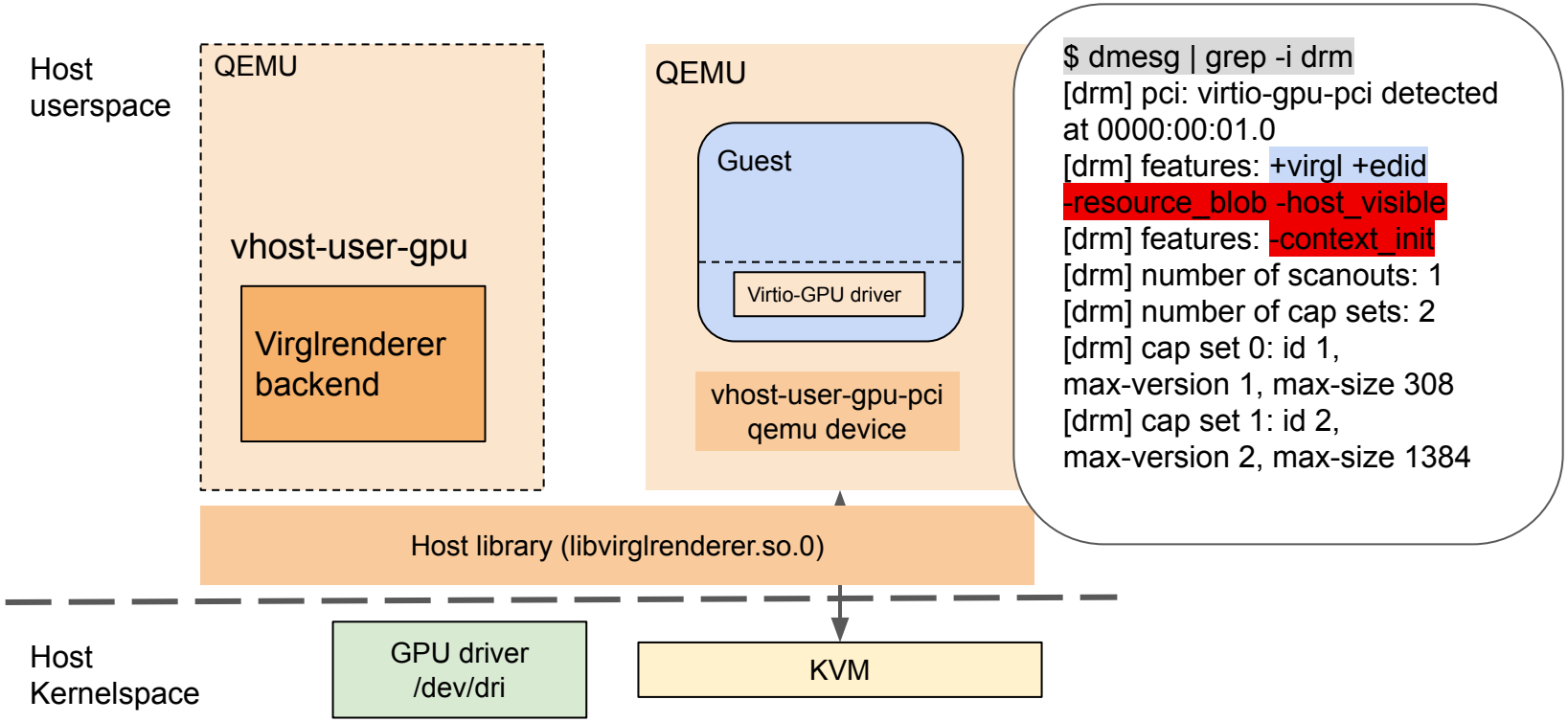
Current Status

- Many virtio-gpu devices are now being upstreamed in various projects:
  - CROSVM
  - libkrun
  - QEMU
  - RUST-VMM
- Virtio-gpu device under the rust-vmm umbrella (vhost-device-gpu) supporting:
  - virglrenderer component from rutabaga
  - gfxstream component from rutabaga
    - Why? Android automotive OS
  - The existing D-Bus display and GTK display in QEMU, with focus on D-Bus
- Developments is based on the virtio 1.2 specification

Current Status: Why rust?

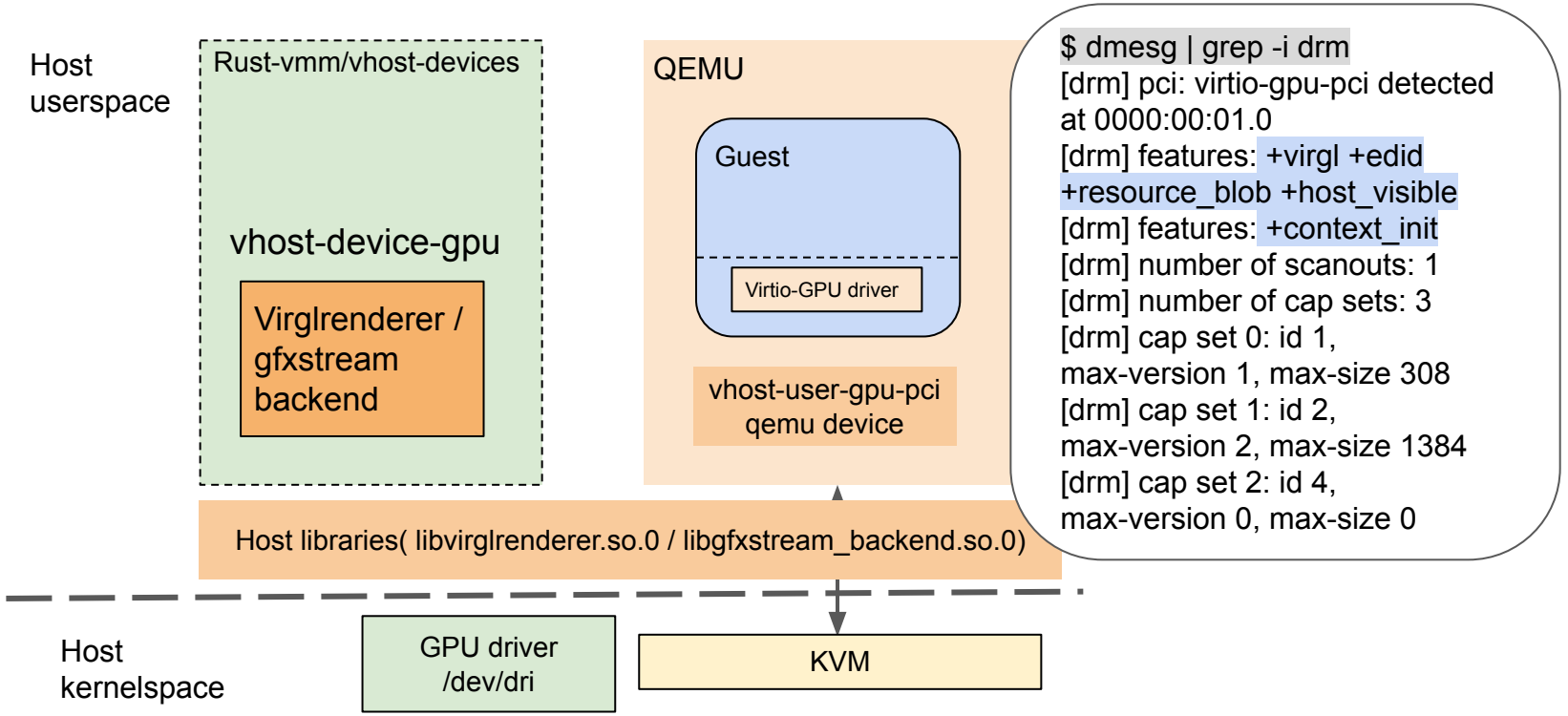
- Device implementation can be used by multiple VMMs
  - QEMU, Crosvm, etc.
  - Hypervisor agnostic
- Operate as an independent process separate from QEMU
  - reduces the attack surface
- Leverage on the vhost-user framework
  - Recently added VHOST_USER_GPU_SET_SOCKET
- Benefit from features of the Rust language:
  - Memory, thread safety, etc.

QEMU vhost-user-gpu - C



Red Hat

Rust-vmm vhost-device-gpu - Rust



https://github.com/rust-vmm/vhost-device/pull/668

Red Hat

Virtio-gpu feature bit negotiation

- At device initialization
- set feature flags based on the feature bits negotiated

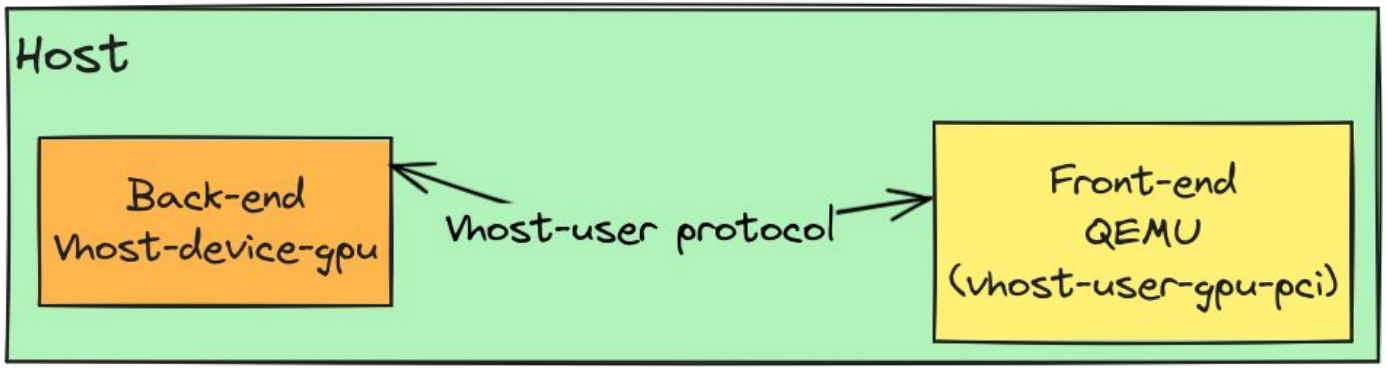
```
fn features(&self) -> u64 {
    1 << VIRTIO_F_VERSION_1
        | 1 << VIRTIO_F_RING_RESET
        | 1 << VIRTIO_F_NOTIFY_ON_EMPTY
        | 1 << VIRTIO_RING_F_INDIRECT_DESC
        | 1 << VIRTIO_RING_F_EVENT_IDX
        | 1 << VIRTIO_GPU_F_VIRGL
        | 1 << VIRTIO_GPU_F_EDID
        | 1 << VIRTIO_GPU_F_RESOURCE_BLOB
        | 1 << VIRTIO_GPU_F_CONTEXT_INIT
        | VhostUserVirtioFeatures::PROTOCOL_FEATURES.bits()
}
```

Red Hat

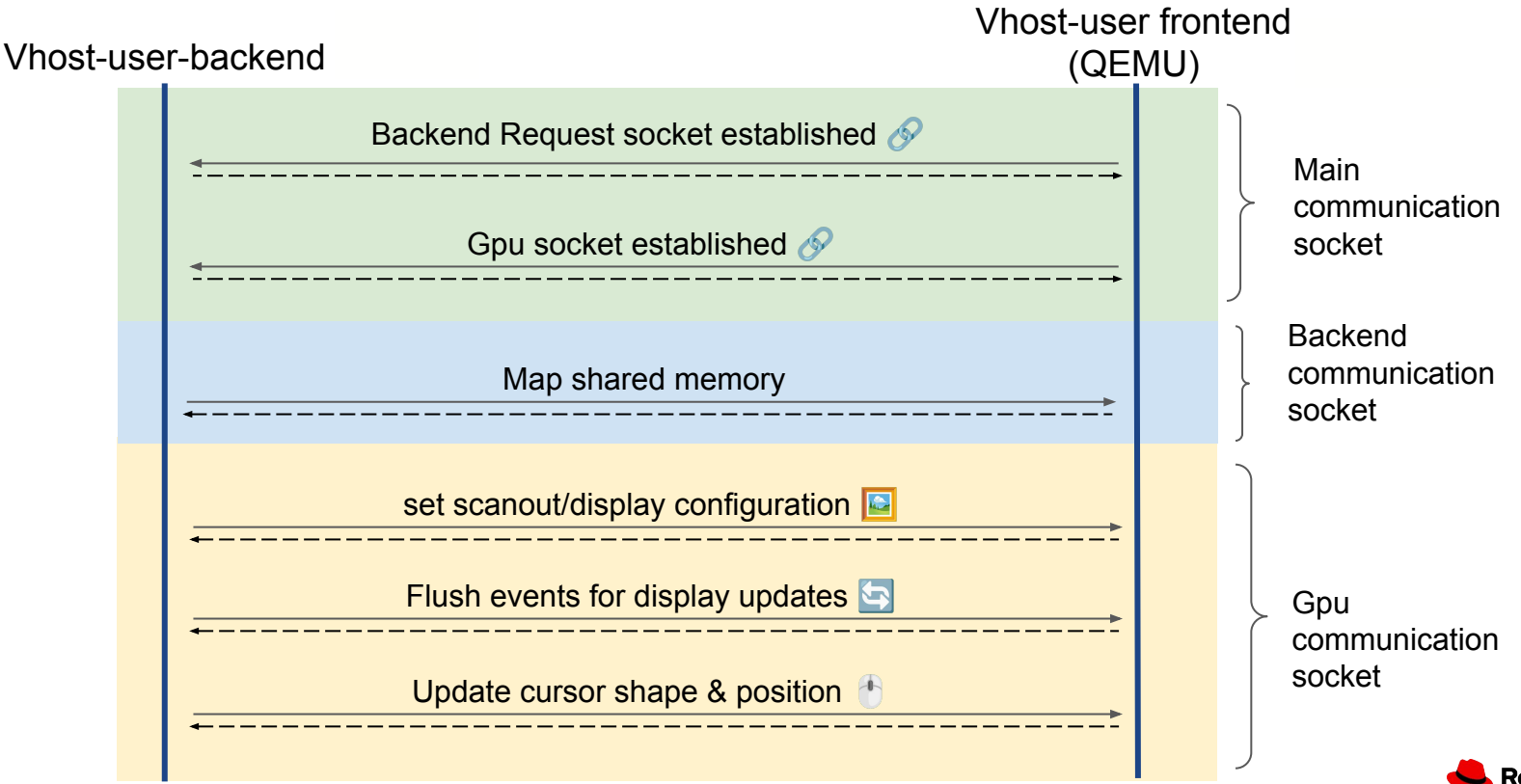
Protocol Overview

Vhost-user protocol

- 2 sides of communication
  - Frontend
  - Backend
- Control plane: establish Virtqueues sharing
- Communication over Unix domain sockets



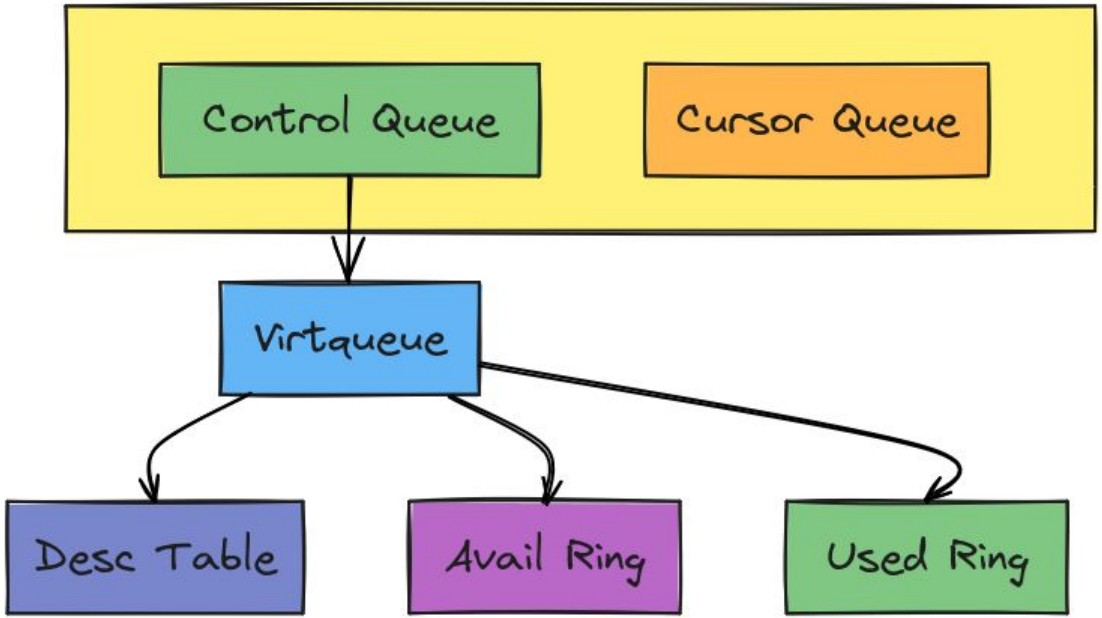
Vhost-user-gpu Protocol Overview



Red Hat

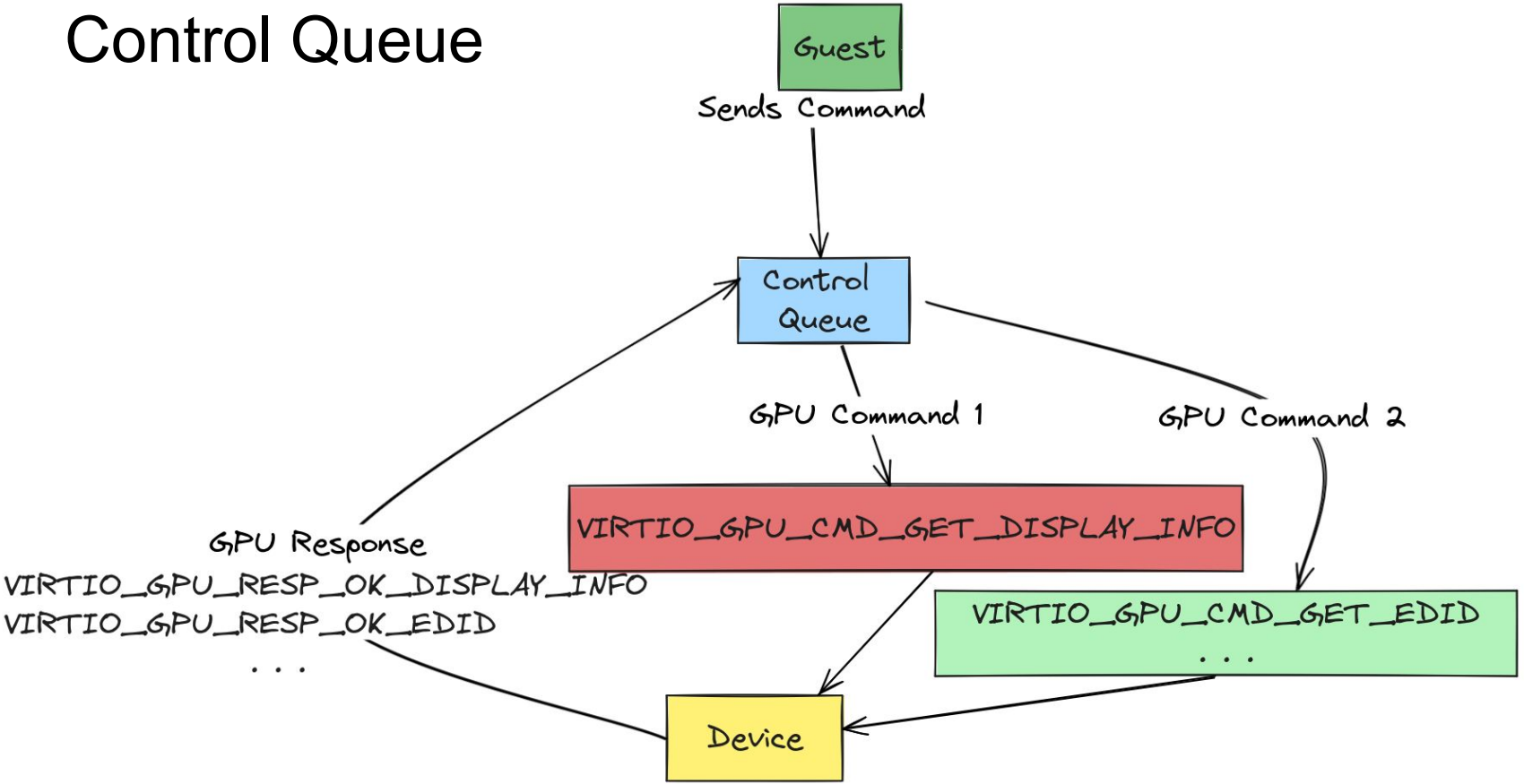
Inner workings

- 2 Virtqueues



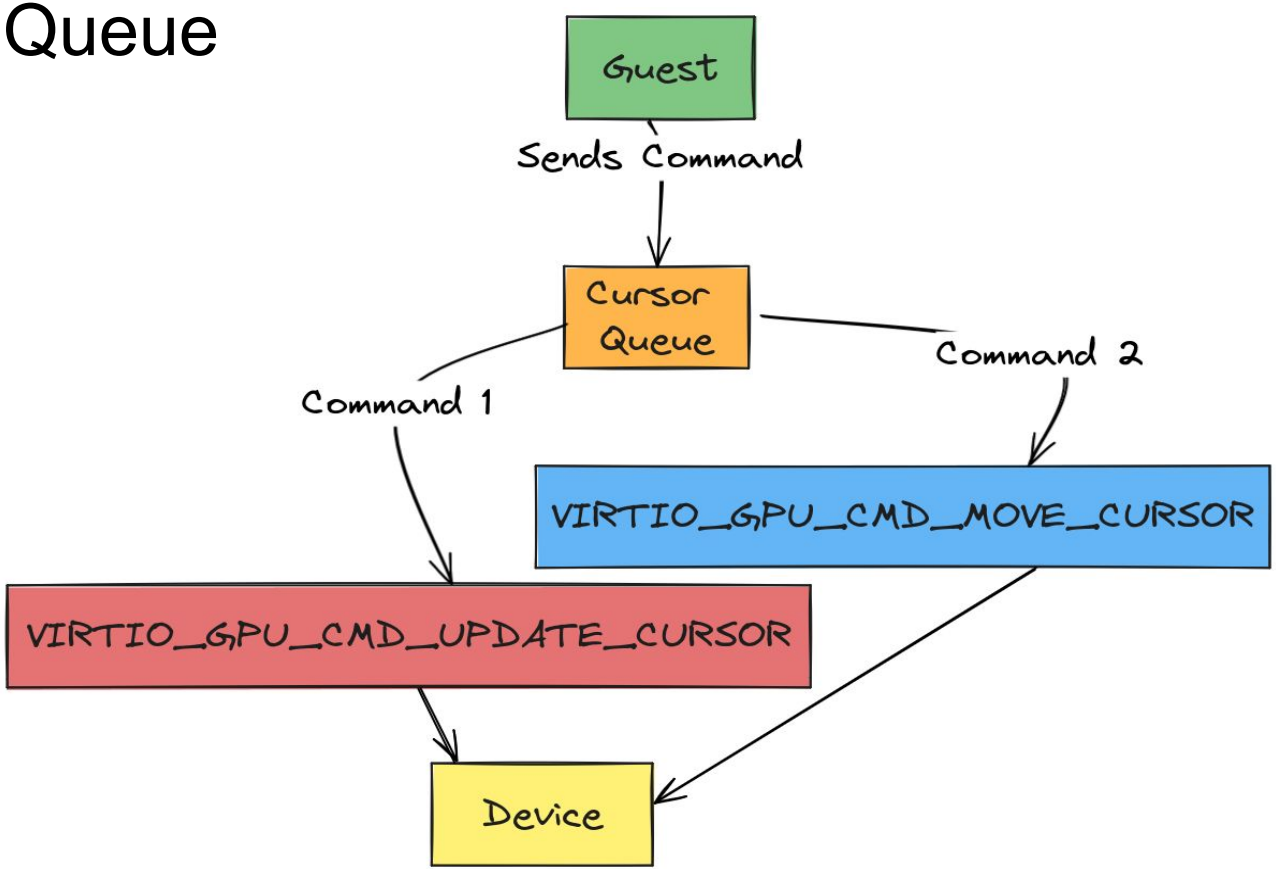
Red Hat

Control Queue



Red Hat

Cursor Queue



Red Hat

Shared Memory Regions

- Support shared memory regions:

```
$ qemu .... -device  vhost-user-gpu-pci,chardev=char0,hostmem=2G
```

```
drm] features: +host_visible
```

- host visible memory region:
  - **VIRTIO_GPU_SHM_ID_HOST_VISIBLE**
- Required for Vulkan (Venus) and gfxstream support

QEMU Patches

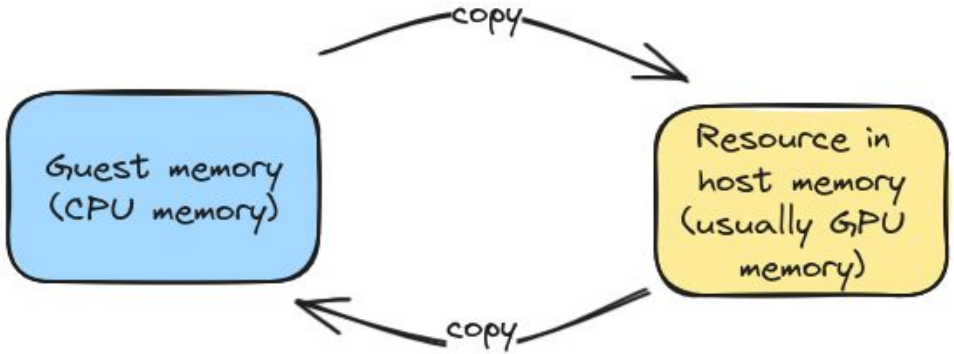
- vhost-user: Add SHMEM_MAP/UNMAP requests
- vhost-user-gpu: Add support for blob resources

Rust-vmm/vhost PR

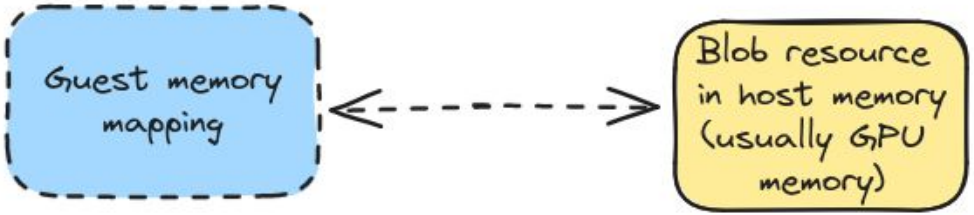
- Vhost-user: Add support for SHMEM_MAP/UNMAP

Red Hat

Normal Resources

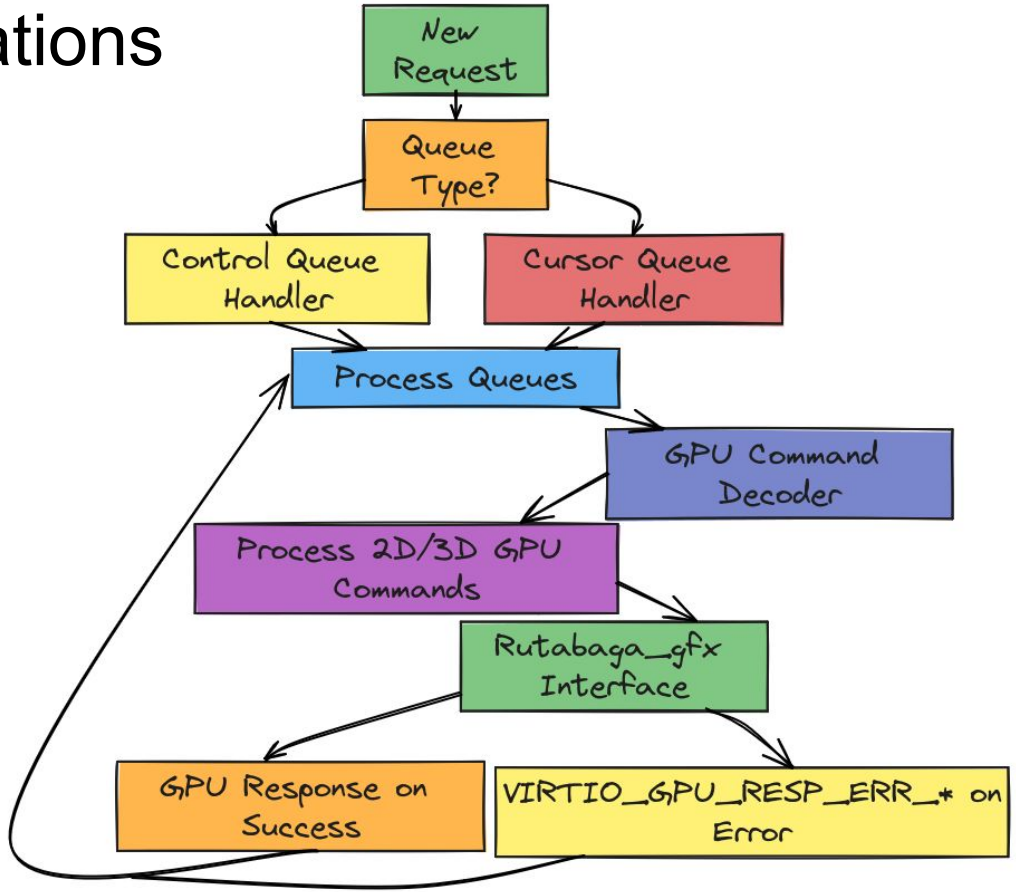


Blob Resources using shared memory regions



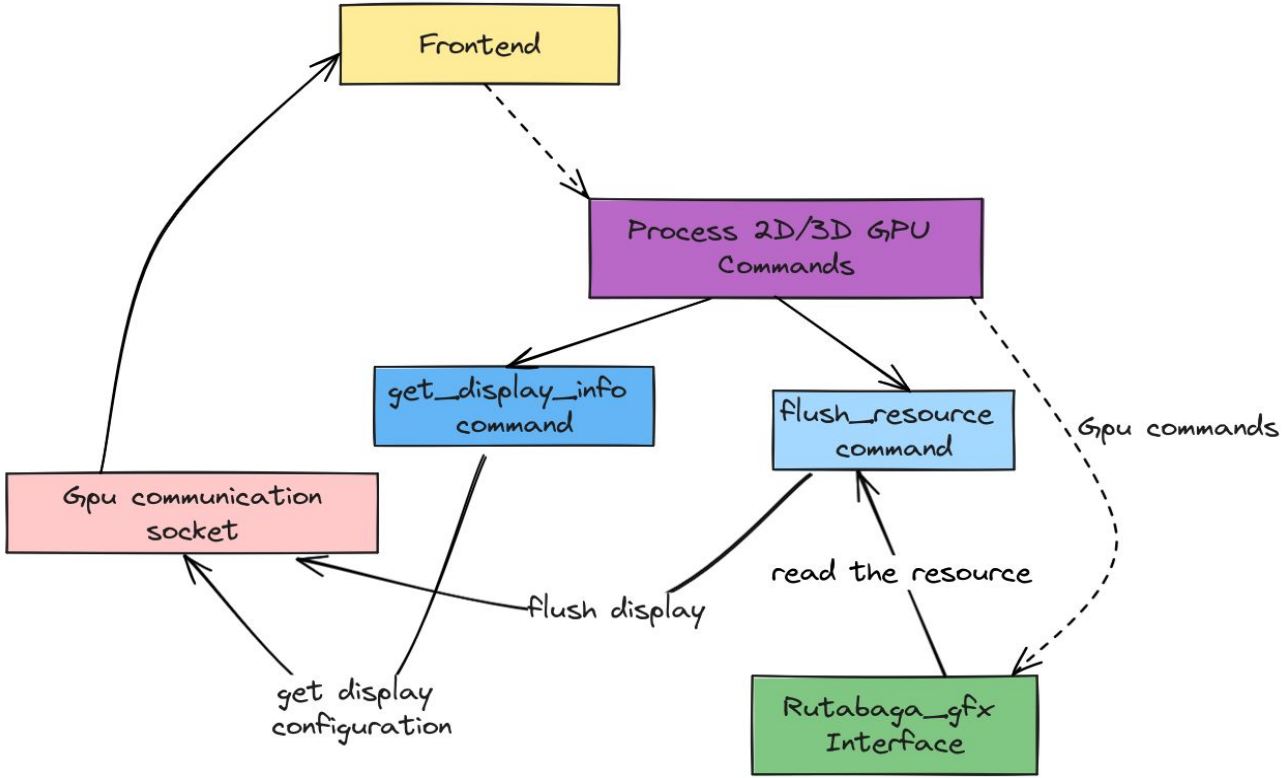
Red Hat

Device operations



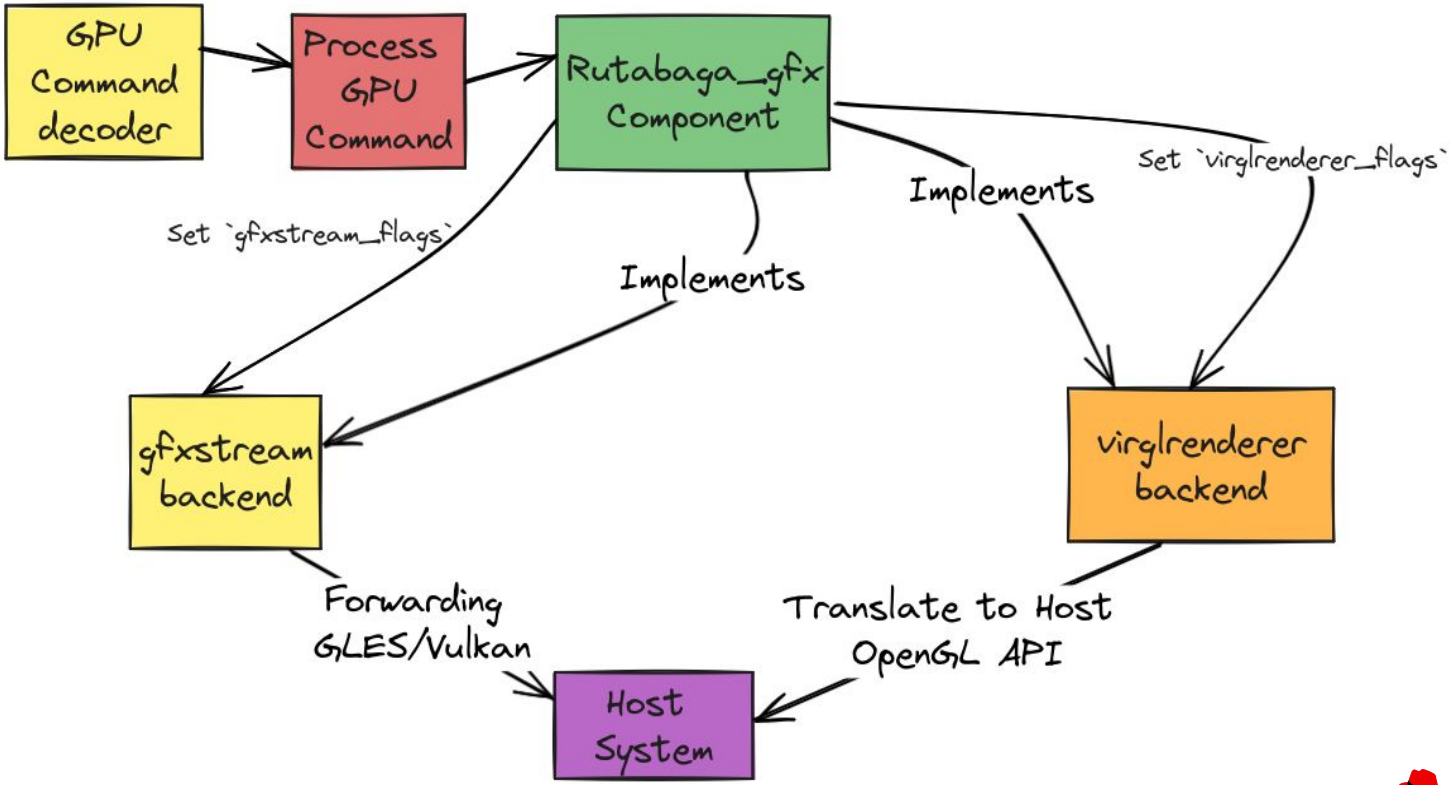
Red Hat

Device operations: Processing gpu commands



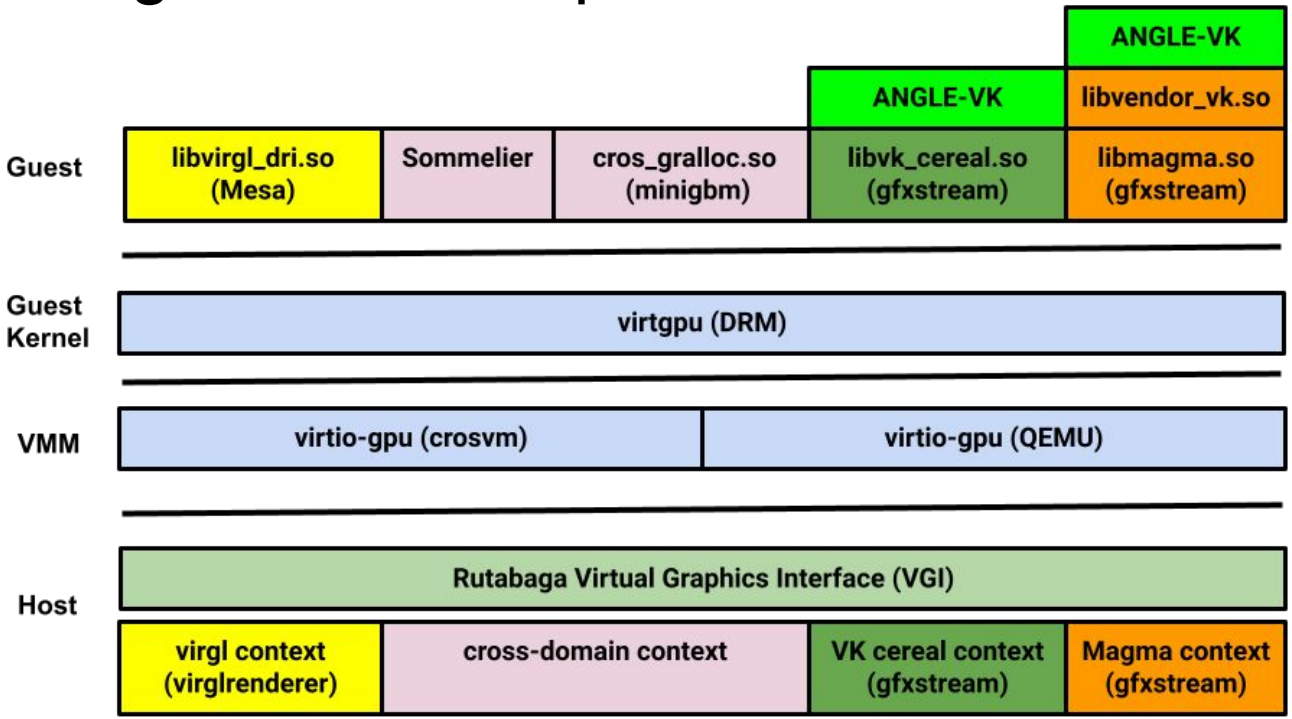
Red Hat

Device operations: rutabaga_gfx components



Red Hat

Rutabaga Virtual Graphics Interface



https://crosvm.dev/book/appendix/rutabaga_gfx

Red Hat

Problems faced:

- Rutabaga Context Initialization
  - Tied to the process, can only be initialized **once**
- Challenges with Rutabaga and Thread Safety
  - `rutabaga` struct is !Send
- Complications with vhost_backend
- Dependence on Crosvm git_submodules
  - Reduce rutabaga dependencies on third_party modules?
  - Enable other projects to use the crate

How does it benefit Automotive?

Gfxstream perspective

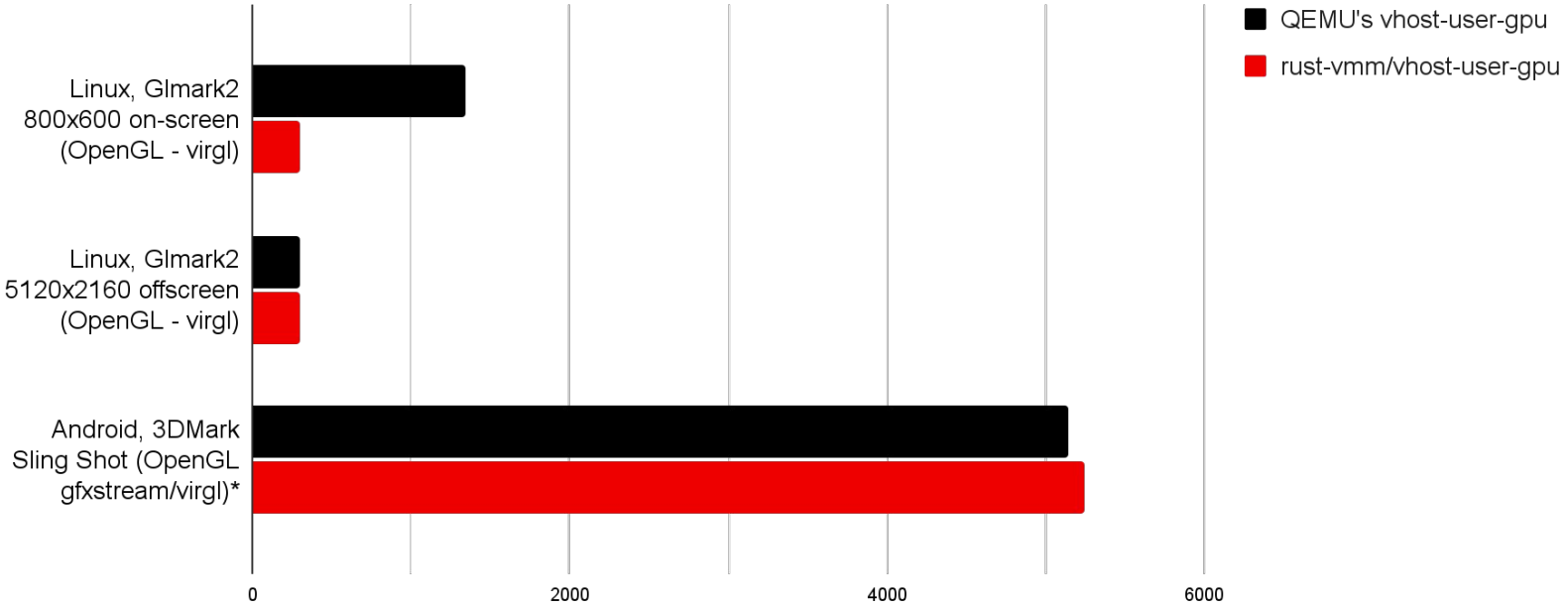
- Optimised for Android Automotive OS
  - Supports both OpenGL ES and Vulkan APIs
- Accelerate Graphics for IVI
  - Automotive infotainment systems require high-performance graphics rendering for smooth user interfaces.
- Useful for Linux guests too?
  - Recently gfxstream got merged into Mesa:
  - https://gitlab.freedesktop.org/mesa/mesa/-/merge_requests/27246

How does it benefit Automotive?

VirglRenderer perspective

- More suitable for automotive use cases that involve Linux-based guests needing graphics acceleration
- virglrenderer Primarily supports OpenGL, Vulkan supports (Venus) exists but is not enabled everywhere:
  - Guest driver: mesa's Venus may not be enabled in all distributions such as Fedora

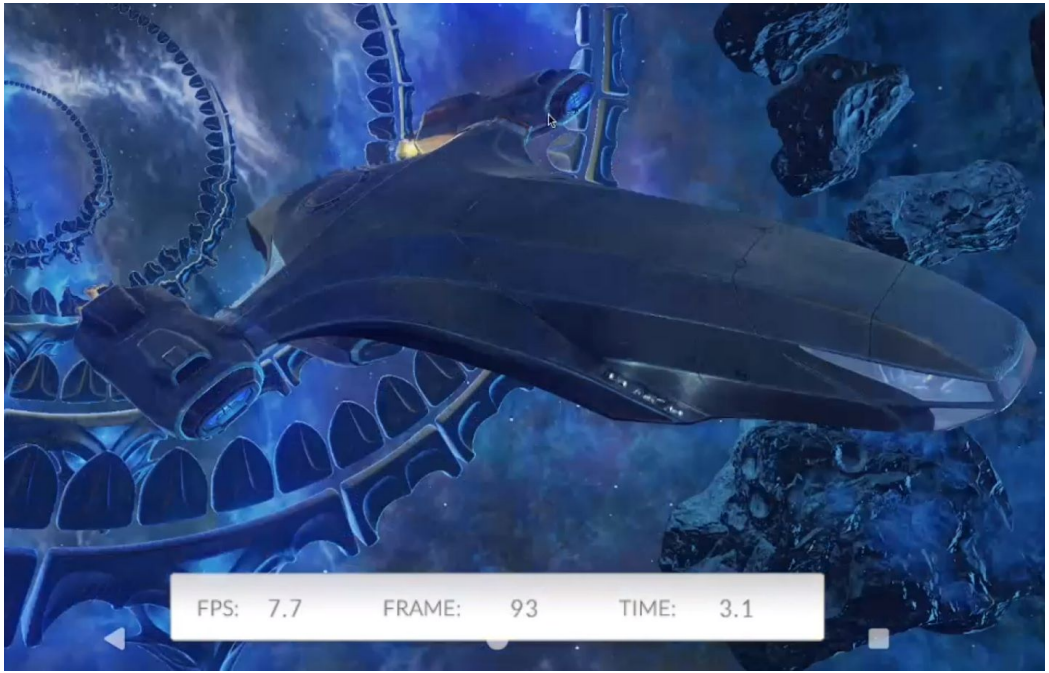
Performance measurements

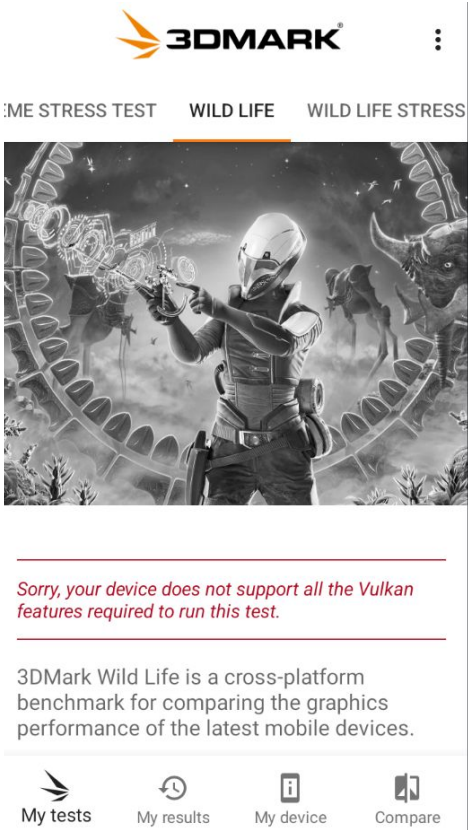


*Using Android 13 for the QEMU's vhost-user gpu and Android 14 for the rust-vmm's vhost-user-gpu.

Red Hat

Performance measurements





Red Hat

Ongoing work

- Upstream
  - QEMU patches
  - Gpu device into rust-vmm
- Improve scanout performance
  - Virglrenderer (expose dmabuf export functionality in Rutabaga?)
  - Gfxstream (we could use blob resources?)

Future work

- Improve scanout performance using dmabuf
- Add support for snapshotting?
- Currently using virglrenderer and gfxstream via rutabaga_gfx. Food for thought:
  - Virglrenderer rust crate?
  - Gfxstream rust crate?

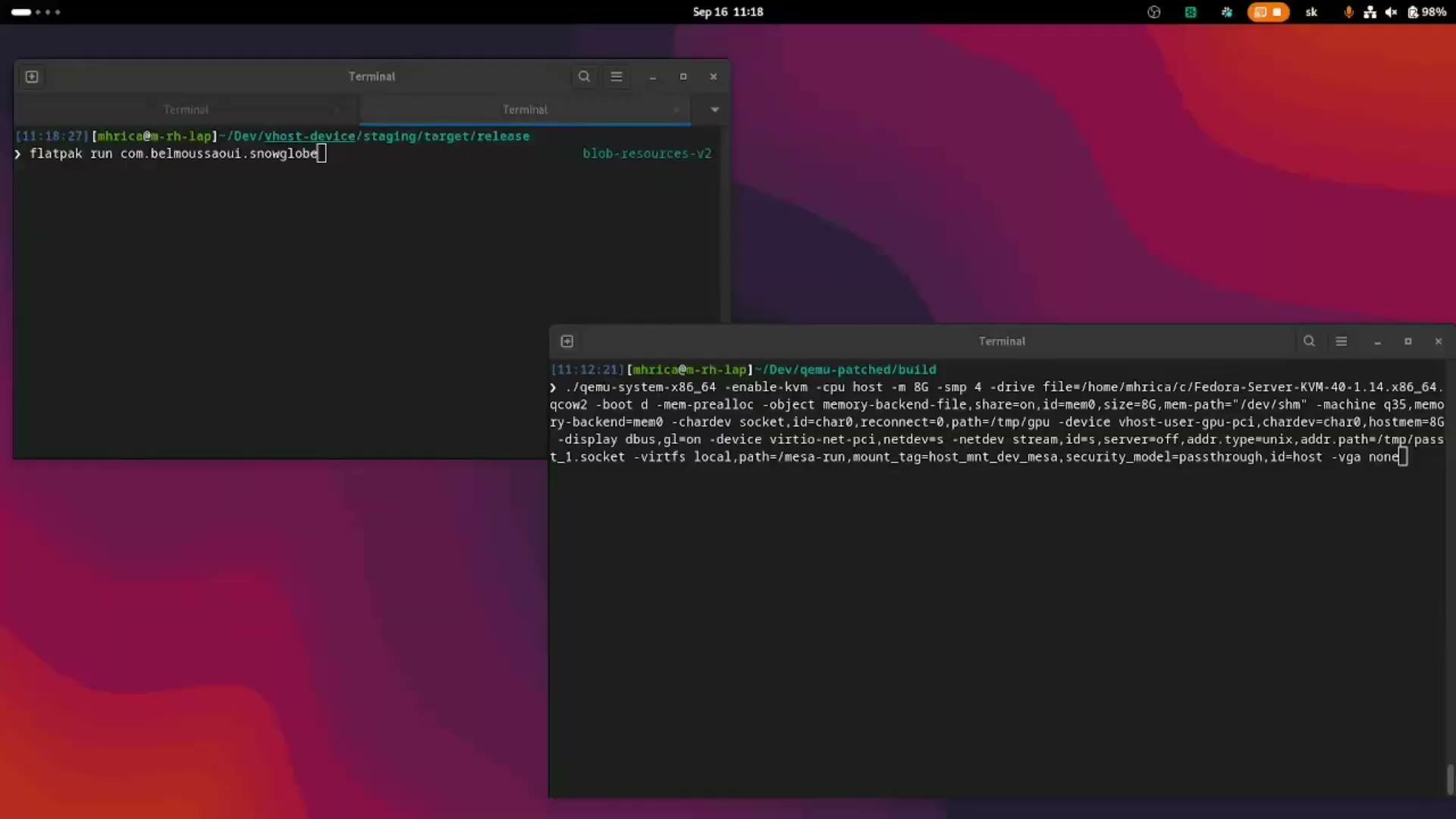
Vhost-device-gpu Example

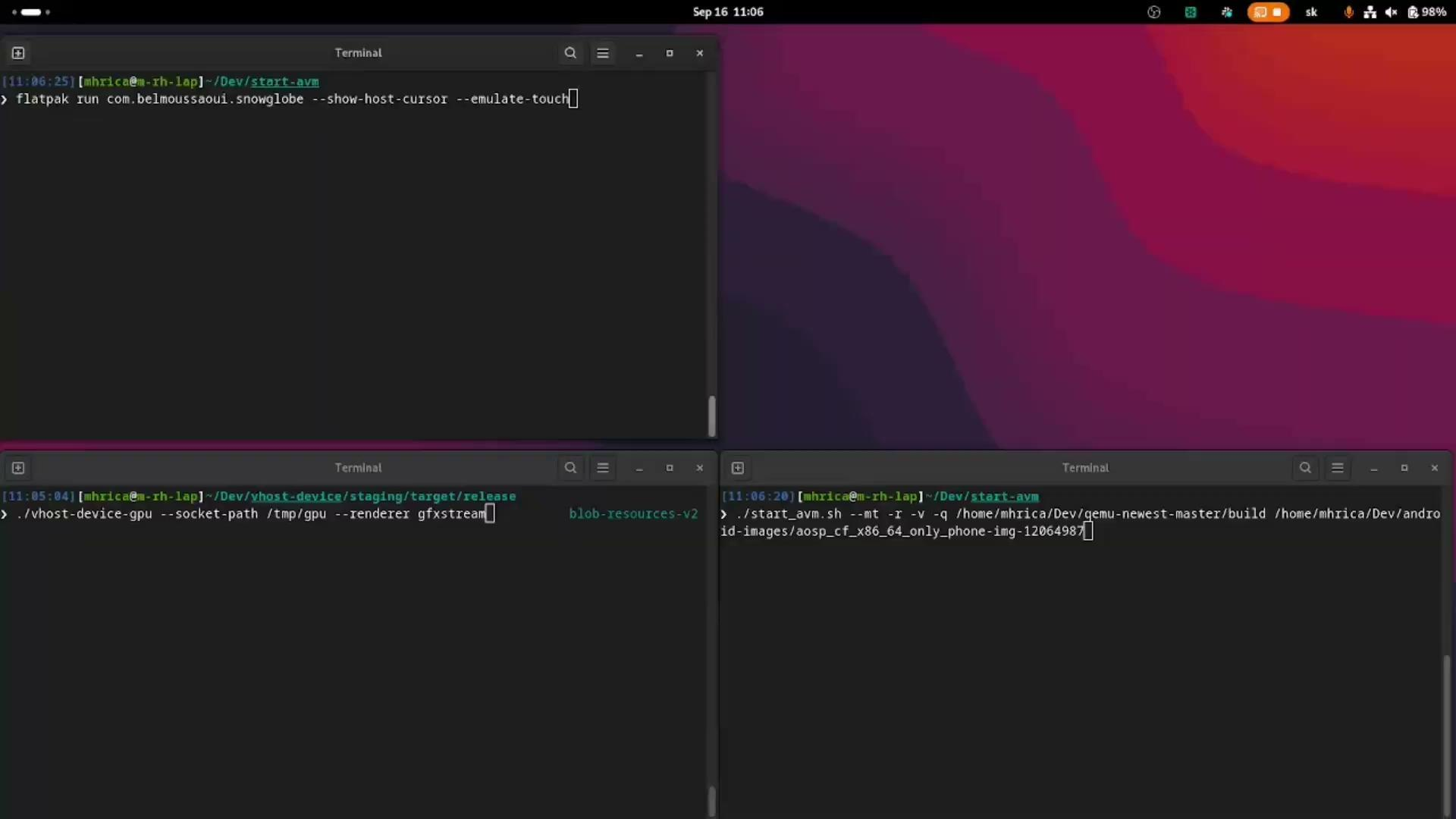
First launch the vhost-user-gpu, using the command:
`$ vhost-user-gpu --socket-path /tmp/gpu --renderer RENDERER`

Where `RENDERER` is either: `gfxstream` or `virglrenderer`

Then when starting QEMU pass the device using the following arguments:

```
-chardev socket,id=char0,reconnect=0,path=/tmp/gpu -device
vhost-user-gpu-pci,chardev=char0,hostmem=2G
```


Sep 16 11:18
sk
98%
Terminal
Terminal
Terminal
[11:18:27][mhrica@m-rh-lap]~/Dev/vhost-device/staging/target/release
❯ flatpak run com.belmoussaoui.snowglobe
blob-resources-v2
Terminal
[11:12:21][mhrica@m-rh-lap]~/Dev/qemu-patched/build
❯ ./qemu-system-x86_64 -enable-kvm -cpu host -m 8G -smp 4 -drive file=/home/mhrica/c/Fedora-Server-KVM-40-1.14.x86_64.qcow2 -boot d -mem-prealloc -object memory-backend-file,share=on,id=mem0,size=8G,mem-path="/dev/shm" -machine q35,memory-backend=mem0 -chardev socket,id=char0,reconnect=0,path=/tmp/gpu -device vhost-user-gpu-pci,chardev=char0,hostmem=8G -display dbus,gl=on -device virtio-net-pci,netdev=s -netdev stream,id=s,server=off,addr.type=unix,addr.path=/tmp/pass_1.socket -virtfs local,path=/mesa-run,mount_tag=host_mnt_dev_mesa,security_model=passthrough,id=host -vga none


Sep 16 11:06
sk
98%
Terminal
[11:06:25][mhrica@m-rh-lap]~/Dev/start-avm
❯ flatpak run com.belmoussaoui.snowglobe --show-host-cursor --emulate-touch
Terminal
[11:05:04][mhrica@m-rh-lap]~/Dev/vhost-device/staging/target/release
❯ ./vhost-device-gpu --socket-path /tmp/gpu --renderer gfxstream
blob-resources-v2
Terminal
[11:06:20][mhrica@m-rh-lap]~/Dev/start-avm
❯ ./start_avm.sh --mt -r -v -q /home/mhrica/Dev/qemu-newest-master/build /home/mhrica/Dev/android-images/aosp_cf_x86_64_only_phone-img-12064987

Thank you

Demo links:


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat