



Virtio-GPU for Automotive

Implementing Libkrun + Vhost-User

Dorinda Bassey
dbassey@redhat.com

The Automotive Challenge

Traditional: Separate ECUs

ASIL-D
Braking

ASIL-B
Cluster

QM
Apps

IVI
Android

Modern: Central Compute Platform CCU

ASIL-D
Partition
RTOS

Hypervisor

ASIL-B
Cluster

QM
Apps
(krun)

QM
AAOS
(krun/QEMU)

libkrun scope: QM partition only

- Infotainment, media, third-party apps
- Isolated from ASIL-D workloads
- Needs: fault isolation, security, performance

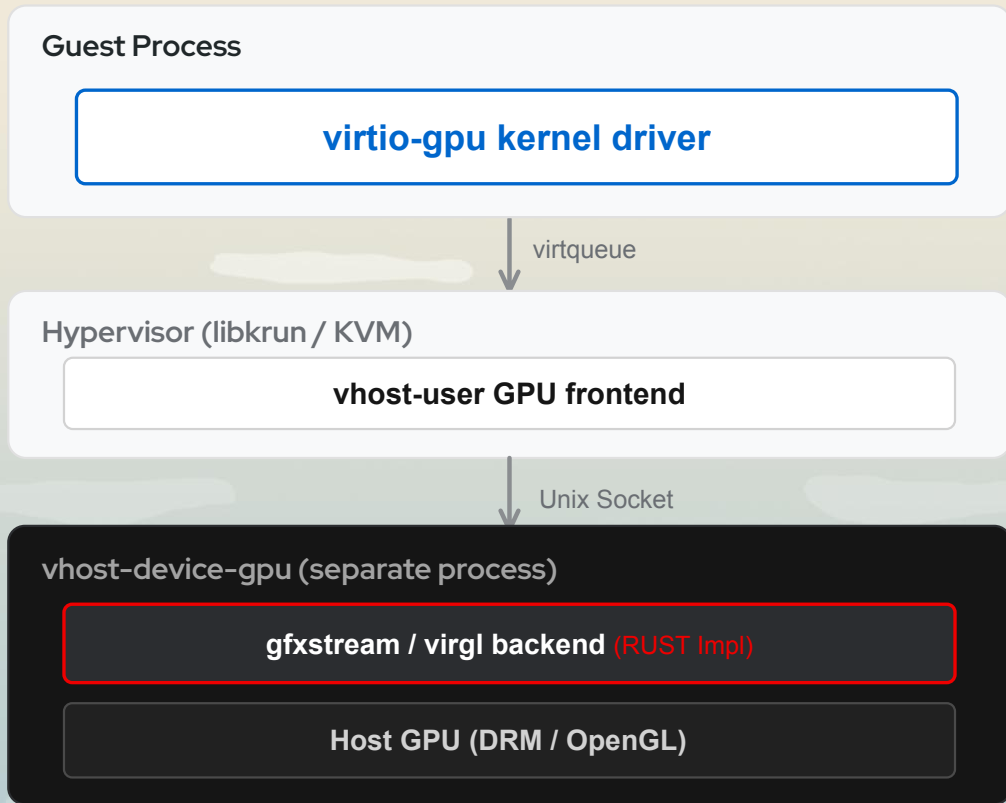


LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Architecture Overview



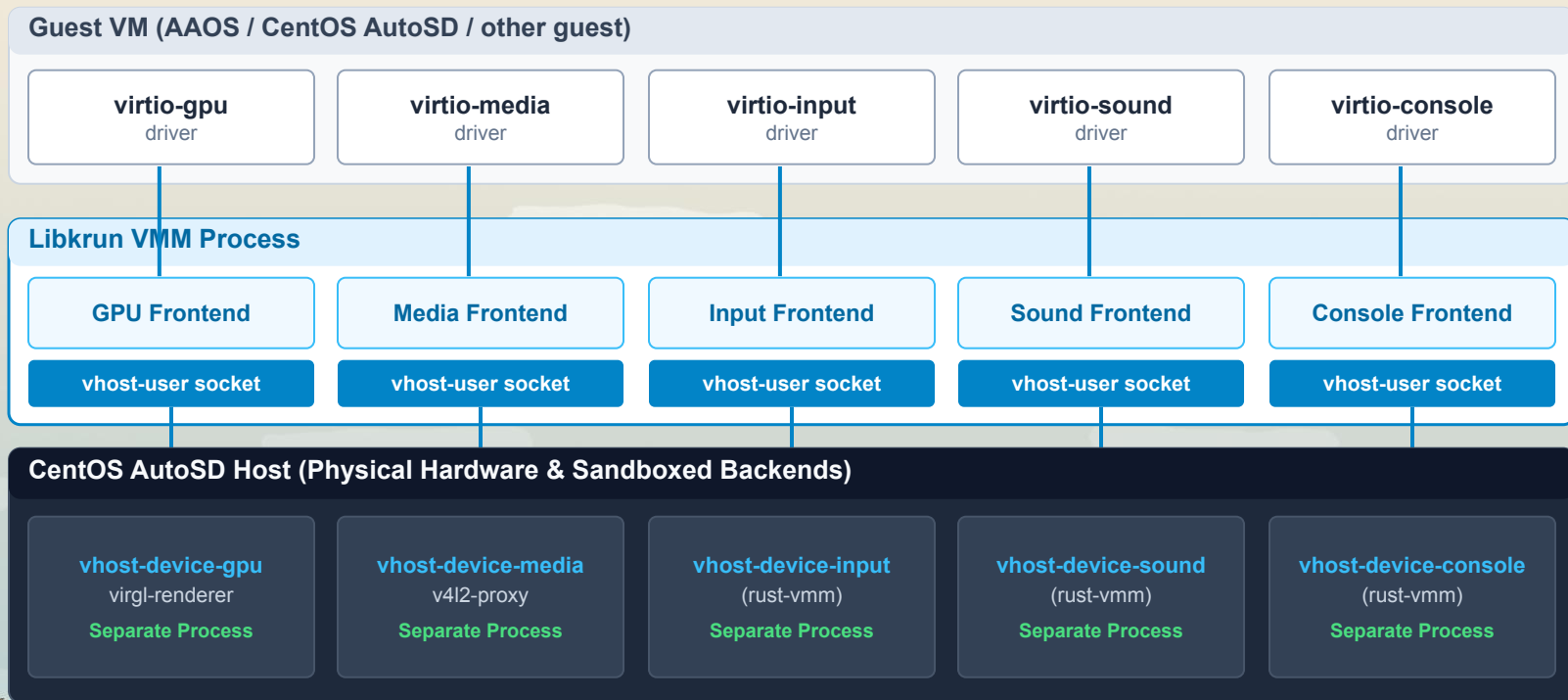
LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Benefits: Fault isolation, zero-copy, reduced vmm attack surface



Broader Vhost-User in Libkrun



Implementation Status

1. Headless GPU Compute

Without Scanouts

Guest VM

Application
(GPU compute)

virtio-gpu
(num_scanouts=0)

Resource create / attach
commands

GPU Backend (virglrenderer/gfxstream)

No display output (Headless execution)
Direct offscreen rendering
num_scanouts = 0 works (kernel accepts,
spec unclear)



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Implementation Status

1. Headless GPU Compute

Without Scanouts

Guest VM

Application
(GPU compute)

virtio-gpu
(num_scanouts=0)

Resource create / attach
commands

GPU Backend (virglrenderer/gfxstream)

No display output (Headless execution)
Direct offscreen rendering
num_scanouts = 0 works (kernel accepts,
spec unclear)

2. Software Scanout

Full Frame Pixel Copy

Guest VM Application → virtio-gpu

GPU Backend (gfxstream)

CPU rendering (llvmpipe) → Read pixels (CPU
copy)

Pixel buffer via gfx

VMM Display (GTK)

alloc_frame() → Malloc pixel data →
present_frame()

Tested: modetest, SMPTE bars, kmscube,
glmark.

⚠ Performance: High latency, high CPU
(llvmpipe software rendering)



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Implementation Status

1. Headless GPU Compute

Without Scanouts

Guest VM

Application
(GPU compute)

virtio-gpu
(num_scanouts=0)

Resource create / attach
commands

GPU Backend (virglrenderer/gfxstream)

No display output (Headless execution)
Direct offscreen rendering
num_scanouts = 0 works (kernel accepts,
spec unclear)

2. Software Scanout

Full Frame Pixel Copy

Guest VM Application → virtio-gpu

GPU Backend (gfxstream)

CPU rendering (llvmpipe) → Read pixels (CPU
copy)

Pixel buffer via gfx

VMM Display (GTK)

alloc_frame() → Malloc pixel data →
present_frame()

Tested: modetest, SMPTE bars, kmscube,
glmark.

⚠ Performance: High latency, high CPU
(llvmpipe software rendering)

3. Zero-Copy DMABUF Scanout

Shared GPU Memory

Guest VM Application → virtio-gpu

GPU Backend (virglrenderer)

GPU rendering (hw accel) → Export DMABUF FD

DMABUF FD via virgl

VMM Display (GTK)

GPU memory ↔ GTK texture (shared)
(zero-copy)

Eliminates: CPU rendering and pixel copies

Benefits: Lower latency, lower CPU usage,
GPU acceleration

Target: Production automotive displays



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Back pressure & Frame Synchronization

The Problem: Missing VSync

- Guest GPU rendering at 2037 FPS (both paths!)
- Display refresh: 60 Hz
- 97% of frames wasted, never visible
- No throttling = resource exhaustion

The Fix: Guest-Side Vblank Timer

- Upstream kernel patches (Linux 6.13+)
- drm/vblank: Add vblank timer
- drm/virtgpu: Use vblank timer
- Guest paces frames to display refresh
- Result: 60 FPS (proper VSync)



Spec vs. Kernel: num_scanouts Divergence

Virtio-GPU Spec v1.4 (section 5.7.4.1)

"num_scanouts: specifies the maximum number of scanouts supported by the device. Minimum value is 1, maximum value is 16."

Linux Kernel (virtgpu_kms.c:279-284)

```
if (!IS_ENABLED(CONFIG_DRM_VIRTIO_GPU_KMS)
    || !vgdev->num_scanouts) {
    DRM_INFO("KMS disabled\n");
    vgdev->num_scanouts = 0;
    vgdev->has_edid = false;
}
```

// Accepts 0, disables KMS, works for headless

The Question: How should we resolve this divergence?

- Update spec to permit num_scanouts = 0?
- Update kernel to enforce minimum = 1?
- Document divergence & let VMMs choose?

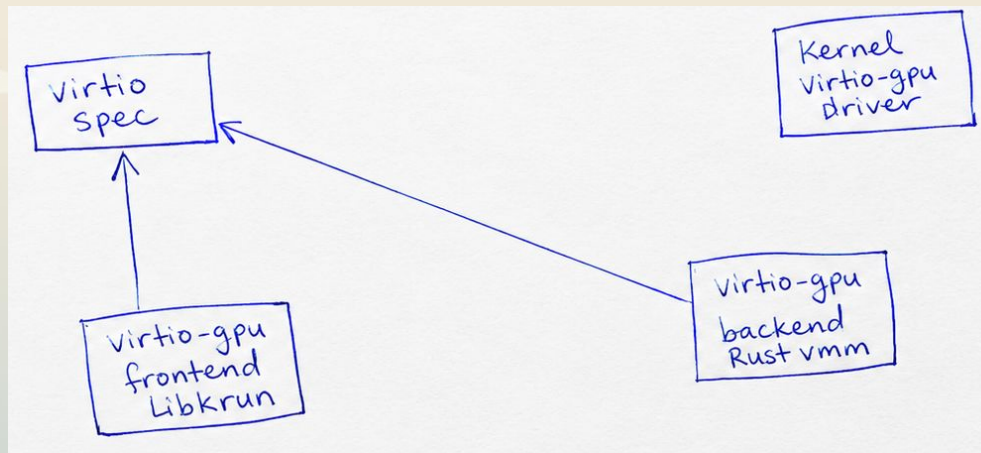


LINU

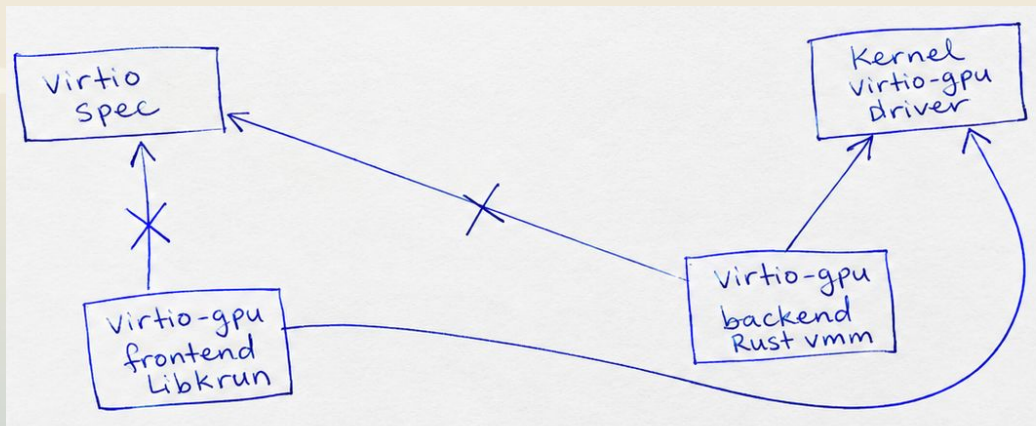
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

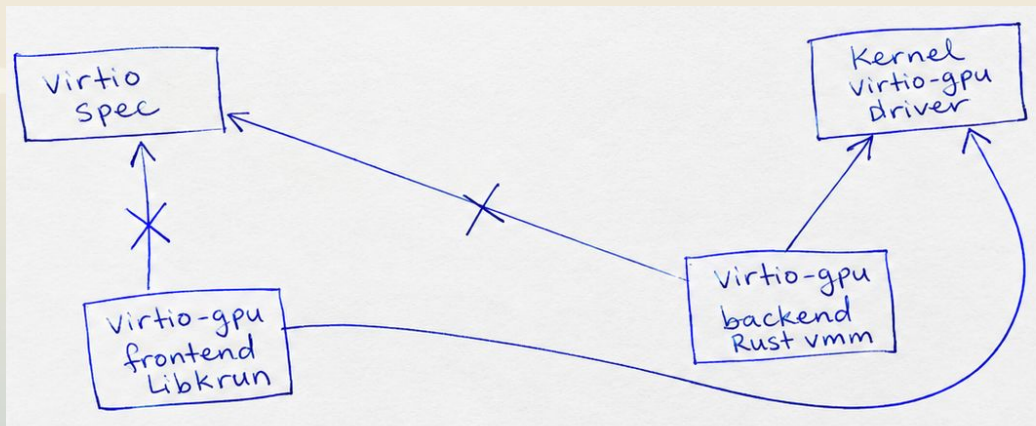
Spec vs. Kernel: num_scanouts Divergence



Spec vs. Kernel: num_scanouts Divergence



Spec vs. Kernel: num_scanouts Divergence



Linux Kernel (virtgpu_kms.c:279-284)

```
if (!IS_ENABLED(CONFIG_DRM_VIRTIO_GPU_KMS)
    || !vgdev->num_scanouts) {
    DRM_INFO("KMS disabled\n");
    vgdev->num_scanouts = 0;
    vgdev->has_edid = false;
}
```

// Accepts 0, disables KMS, works for headless



Headless Mode Complexity

Two Types of "Headless"

True Headless (GPU Compute Only)

```
num_scanouts = 0
```

- **KMS disabled** in kernel
- **Use cases:** OpenCL, Vulkan compute, AI/ML workloads
- No DRM graphics APIs available

Headless Graphics (Offscreen rendering mode)

```
num_scanouts ≥ 1 (disconnected)
```

- **KMS enabled**, no display output
- **Use cases:** glmark2-drm
--off-screen, DRM-based headless rendering
- Requires **"zombie displays"** to function



The Governance Question

When Spec and Kernel Diverge: Who Decides?

1. Conflict Precedence

When specification and implementation conflict, which of the two takes precedence in the ecosystem?



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



The Governance Question

When Spec and Kernel Diverge: Who Decides?

1. Conflict Precedence

When specification and implementation conflict, which of the two takes precedence in the ecosystem?

2. Alignment Responsibility

- Virtio TC updates the specification?
- Kernel enforces spec compliance?
- VMM developers choose their own path?



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



The Governance Question

When Spec and Kernel Diverge: Who Decides?

1. Conflict Precedence

When specification and implementation conflict, which of the two takes precedence in the ecosystem?

2. Alignment Responsibility

- Virtio TC updates the specification?
- Kernel enforces spec compliance?
- VMM developers choose their own path?

3. Developer Framework Need

- What do other VMMs do? (crosvm, QEMU, etc.)
- Should we align and document best practices?
- Establish clear paths for VMM developers

This isn't just about **num_scanouts**, it's about defining process and governance across the entire ecosystem.



LIN

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Infrastructure Lessons: Building Generic SHMEM Support

The path to unifying device memory sharing and protocols

Software Scanout

PR #816

✓ No SHMEM needed

Guest buffer

↓ virtqueue

GPU Backend

↓ Copy pixels

VMM display

Copy path, simple

Minimal requirements; higher CPU and memory bandwidth overhead.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

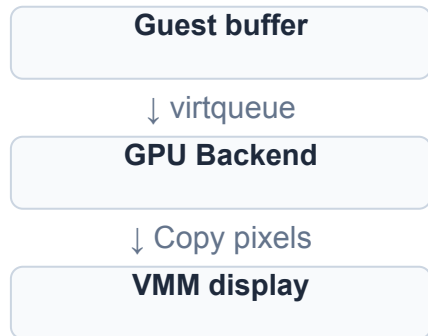
Infrastructure Lessons: Building Generic SHMEM Support

The path to unifying device memory sharing and protocols

Software Scanout

PR #816

✓ No SHMEM needed



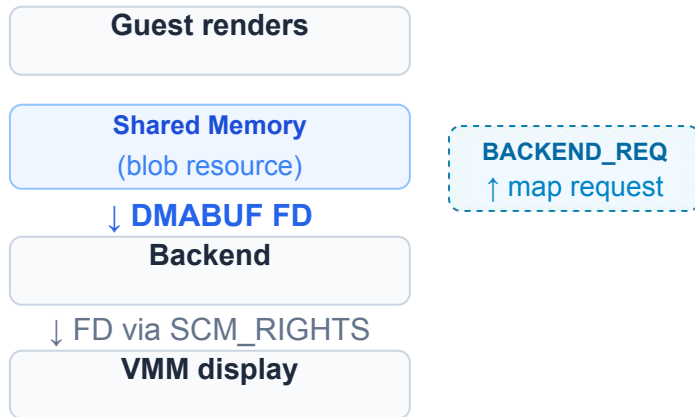
Copy path, simple

Minimal requirements; higher CPU and memory bandwidth overhead.

DMABUF Zero-Copy

PR #862

⚠ Requires Shmem Infrastructure



Zero-copy, needs:

- SHMEM regions
- BACKEND_REQ protocol



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

Infrastructure Lessons: Building Generic SHMEM Support

The path to unifying device memory sharing and protocols

First Use Case: Media Device

`vhost-user-media`

V4L2 proxy protocol

- Translates V4L2 protocol over virtio
- **4GB SHMEM** for MMAP buffers
- **BACKEND_REQ** for notifications



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Infrastructure Lessons: Building Generic SHMEM Support

The path to unifying device memory sharing and protocols

First Use Case: Media Device

`vhost-user-media`

V4L2 proxy protocol

- Translates V4L2 protocol over virtio
- **4GB SHMEM** for MMAP buffers
- **BACKEND_REQ** for notifications

Device Quirks

GPU: 8GB SHMEM

Media: 4GB SHMEM

RNG: 0 (No SHMEM)

Sound: 0 (No SHMEM)

Each device requires distinct queue counts & message formats.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Infrastructure Lessons: Building Generic SHMEM Support

The path to unifying device memory sharing and protocols

First Use Case: Media Device

`vhost-user-media`

V4L2 proxy protocol

- Translates V4L2 protocol over virtio
- **4GB SHMEM** for MMAP buffers
- **BACKEND_REQ** for notifications

Device Quirks

GPU: **8GB SHMEM**

Media: **4GB SHMEM**

RNG: **0 (No SHMEM)**

Sound: **0 (No SHMEM)**

Each device requires distinct queue counts & message formats.

API Design

Explicit Caller Control

```
krun_add_vhost_user_device(...,  
shm_size)
```

- **0** = no SHMEM
- **>0** = allocate amount

```
#define  
KRUN_VHOST_USER_MEDIA_SHM_  
SIZE
```

⚠ Warning

Wrong size = runtime errors (not prevented)



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Where Should SHMEM Size Requirements Live?

The Issue

Negotiation **EXISTS** (`GET_SHMEM_CONFIG`), but **Validation is MISSING**. Frontend can allocate less memory than requested, and the backend has no mechanism to reject it.

Who decides SHMEM size?

Backend

Device Implementation



Knows actual requirements

Buffer format, resolution limits, and memory layout.



Cannot enforce

Caller (VMM user)

Host Configuration



Must know device-specific needs

Caller is burdened with guessing backend requirements.



Wrong size = runtime failure

Memory mapping errors and guest display crashes.



LINUX
PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Where Should SHMEM Size Requirements Live?

The Issue

Negotiation **EXISTS** (`GET_SHMEM_CONFIG`), but **Validation is MISSING**. Frontend can allocate less memory than requested, and the backend has no mechanism to reject it.

Option A: Device Specs

Standardize in Virtio Specs

(e.g., *virtio-media* requires $\geq 2\text{GB}$)

- + Clear interoperability & compliance
- + Essential for proving FFI
- Inflexible across diverse hardware



Where Should SHMEM Size Requirements Live?

The Issue

Negotiation **EXISTS** (`GET_SHMEM_CONFIG`), but **Validation is MISSING**. Frontend can allocate less memory than requested, and the backend has no mechanism to reject it.

Option A: Device Specs Standardize in Virtio Specs

(e.g., `virtio-media` requires $\geq 2\text{GB}$)

- + Clear interoperability & compliance
- + Essential for proving FFI
- Inflexible across diverse hardware

Option B: Protocol Extension

Add vhost-user Validation

(Backend can reject connection if SHMEM is too small)

- + Fail-fast at startup
- + Flexible per-device requirements
- Requires protocol change



Where Should SHMEM Size Requirements Live?

The Issue

Negotiation **EXISTS** (`GET_SHMEM_CONFIG`), but **Validation is MISSING**. Frontend can allocate less memory than requested, and the backend has no mechanism to reject it.

Option A: Device Specs Standardize in Virtio Specs

(e.g., `virtio-media` requires $\geq 2\text{GB}$)

- + Clear interoperability & compliance
- + Essential for proving FFI
- Inflexible across diverse hardware

Option B: Protocol Extension

Add vhost-user Validation

(Backend can reject connection if SHMEM is too small)

- + Fail-fast at startup
- + Flexible per-device requirements
- Requires protocol change

Option C: Status Quo Implementation-Defined

(VMMs set constants; hope configurations match)

- + Maximum implementation freedom
- No portability guarantees
- Unclear safety compliance



Open Questions & Discussion?

Q1: Divergence

How should we resolve this divergence?

- Update spec to permit
`num_scanouts = 0`?
- Update kernel to enforce minimum = 1?
- Document divergence & let VMMs choose?

Q2: Precedence

When spec and implementation conflict, which takes precedence?

- Virtio TC updates the specification?
- Kernel enforces spec compliance?
- VMM developers choose their own path?

Q3: SHMEM Specs

Where should SHMEM size requirements live?

- Virtio specification?
- Vhost-user validation in the protocol?
- Follow status quo (VMMs set constants, backends pick numbers)?



Thank you!



**LINUX
PLUMBERS CONFERENCE**

PRAGUE, CZECHIA / OCT. 5-7, 2026

