THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA

Embedded Linux Conference

Demystifying VirtIO-GPU: Building a Graphics Virtualization Bridge From Scratch

Yung-Tse Cheng (National Taiwan Normal University)
Sheng-Wen (Colin) Cheng (The University of Texas at Austin)

May 18, 2026

Outline

- Motivation and Background
- Introduction to virtio-gpu 2D
- Role of the Graphics Stack
- Pitfalls Found During Implementation
- virtio-gpu 3D, Supplementary Topics and Demos

Motivation and Background

Motivation

- Most virtio-gpu talks focus on acceleration mechanisms.
- This talk starts one layer below:
  the building blocks that make virtio-gpu work.
- Inspired by Dave Airlie's Virgil talk,
  we focus on the implementation big picture: what has to exist before 3D acceleration becomes meaningful?

GPU Virtualization Paradigms

Class	Guest model	Host work	Examples	VGPU?
Device emulation software GPU model	Guest talks to an emulated GPU	Software reproduces device behavior	Old-hardware emulators	
API / protocol remoting renderer protocol boundary	Guest sends a graphics protocol	Host translates it to host graphics work	virtio-gpu 3D, VirGL, Venus	✓
Fixed pass-through one whole GPU per VM	Guest loads the native GPU driver	Host assigns a whole physical GPU to one VM	QEMU/KVM VFIO GPU passthrough	
Mediated pass-through shared physical GPU	Multiple guests use part of one GPU	Hardware, firmware, driver, and platform coordinate sharing	NVIDIA vGPU, SR-IOV	

History: Prior Attempts Before Virgil

Attempt	What it proved	Why it was not enough	Lesson for Virgil
VMware SVGA II existing VM graphics stack	Virtual GPU demand already existed	Closed source Early 3D centered around DirectX 9	QEMU needed a design it could reuse and evolve
VirtualBox adapter open-source adapter	Open VM graphics was possible	Too much abstraction built on OpenGL as a huge API boundary	Avoid exposing OpenGL as the virtual device API
vGallium closer to Dave's direction	Gallium-like GPU virtualization was feasible	Host still needed a special Gallium driver API	Gallium-like guest model, normal host OpenGL backend
Missing piece what Virgil had to provide	Previous designs showed the direction and demand	No general interface that fit QEMU cleanly	QEMU-friendly virtual GPU Start with Linux guest stack: kernel / X.Org / Mesa / QEMU

Introduction to virtio-gpu 2D

(Resource, Backing, Scanout, Transfer, and Flush)

OPEN SOURCE SUMMIT NORTH AMERICA
Embedded Linux Conference

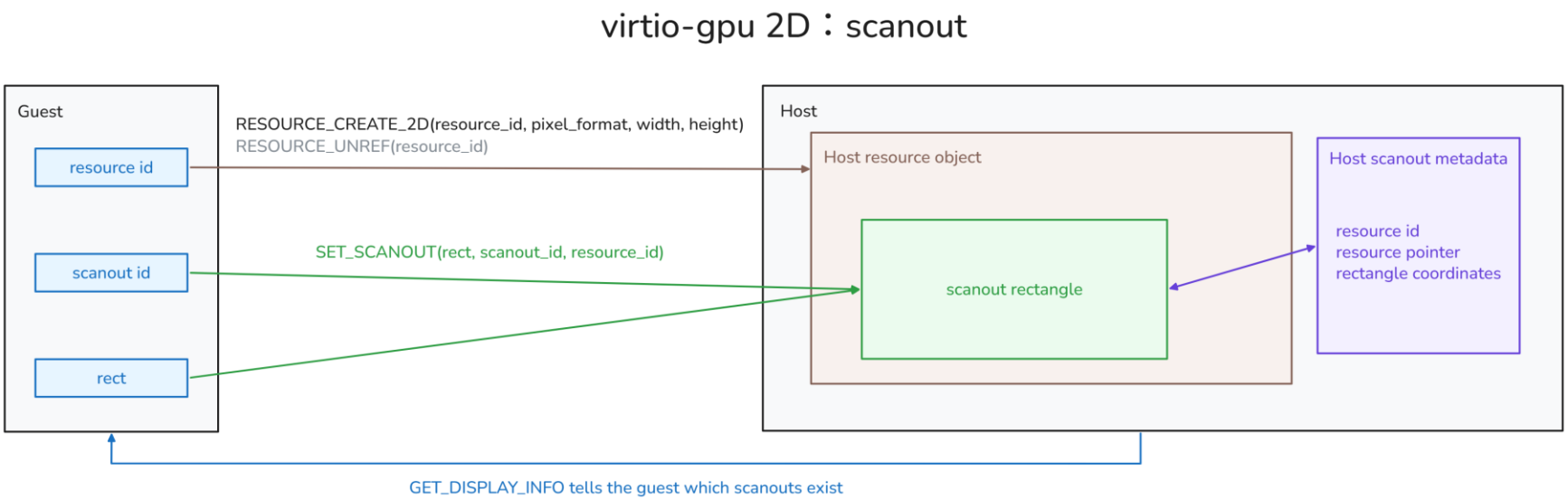
Benefits of starting with virtio-gpu 2D

- The 2D path avoids the complexity of 3D acceleration.
- This lets us focus on the display infrastructure.
- We use Xorg modesetting + Mesa swrast / drisw as the example environment.

Requirement for a Minimal 2D Device

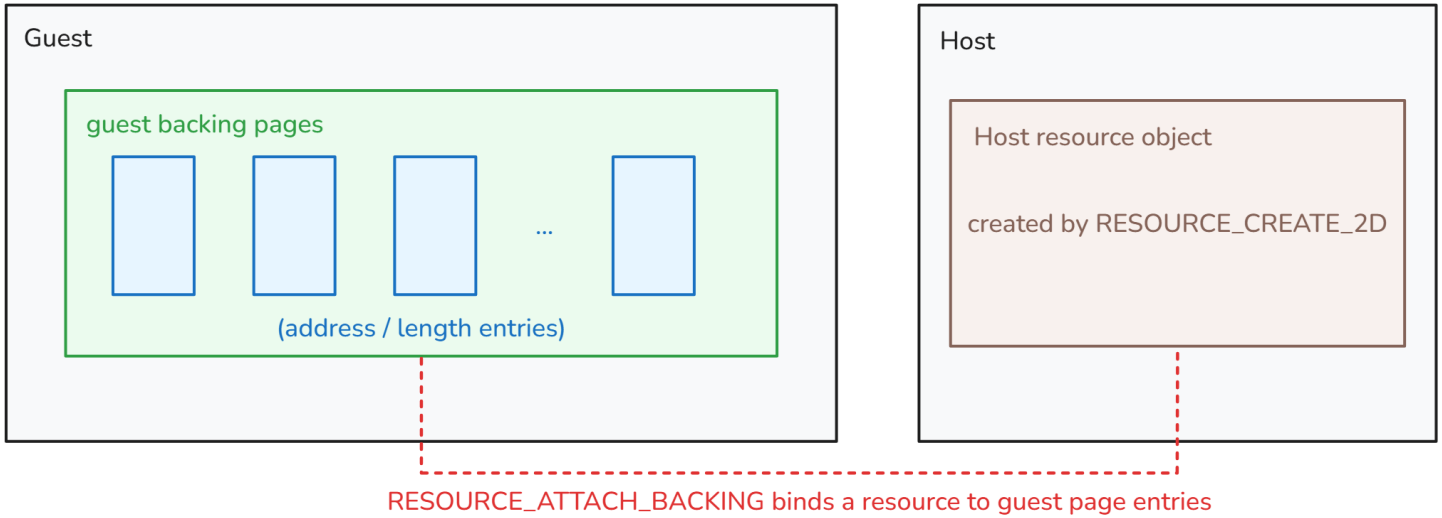
- Virtqueue handling virtio-input
  - controlq
  - cursorq
- GET_DISPLAY_INFO (first command invoked by virtio-gpu)
  - display outputs
  - scanout / resolution
- KMS modesetting state (basic validation)

Resource and Scanout



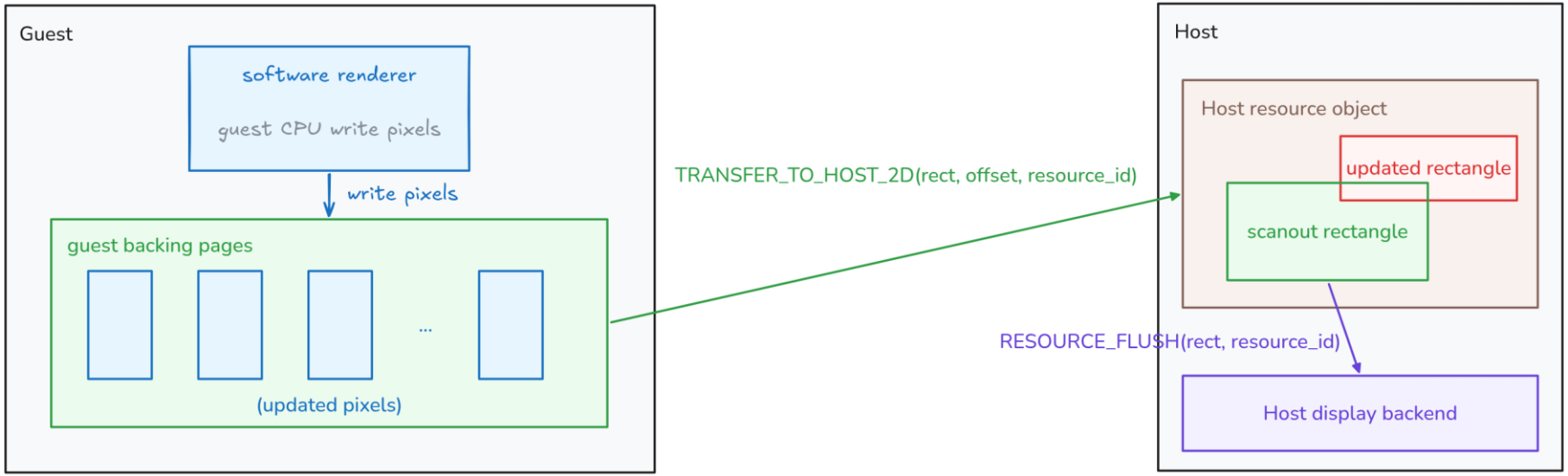
Resource Backing

virtio-gpu 2D：resource + backing



Transfer and Flush

virtio-gpu 2D：transfer + flush



Role of the Graphics Stack

(Xorg, Mesa swrast / dri-sw, and Front Buffer Updates)

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
NORTH AMERICA
Embedded Linux Conference

Buffers Setup: Screen Pixmap & Front BO

Object	Backing / kernel object	What it describes	virtio-gpu relation
Xorg screen front BO	DRM GEM BO DRM framebuffer / fb_id	screen storage selected by KMS plane / CRTC	creates virtio-gpu 2D resource and attaches guest backing

Buffers Setup: X11 Objects

Object	Backing / kernel object	What it describes	virtio-gpu relation
Xorg screen front BO	DRM GEM BO DRM framebuffer / fb_id	screen storage selected by KMS plane / CRTC	creates virtio-gpu 2D resource and attaches guest backing
X11 Window Drawable / Pixmap	X server software pixmap storage	where pixels are presented and how regions are clipped	stays inside X server; no virtio-gpu resource creation here

Buffers Setup: Mesa Client Buffer

Object	Backing / kernel object	What it describes	virtio-gpu relation
Xorg screen front BO	DRM GEM BO DRM framebuffer / fb_id	screen storage selected by KMS plane / CRTC	creates virtio-gpu 2D resource and attaches guest backing
X11 Window Drawable / Pixmap	X server software pixmap storage	where pixels are presented and how regions are clipped	stays inside X server; no virtio-gpu resource creation here
Mesa drisw / swrast client-side color buffer	software display target userspace memory / XShm	guest CPU writes the rendered pixels here	client-side software buffer; handed to X server at swap

Step 1: Rendering

Action	Runs in	Target / result	virtio-gpu relation
Rendering	Application + Mesa swrast	guest CPU writes pixels into client-side color buffer	no virtio-gpu work; all rendering is software

Step 2: Present

Action	Runs in	Target / result	virtio-gpu relation
Rendering	Application + Mesa swrast	guest CPU writes pixels into client-side color buffer	no virtio-gpu work; all rendering is software
Present	Mesa drisw + X11 request	pixels are sent to X server by XPutImage() / XShmPutImage()	still outside virtio-gpu; this is window-system handoff

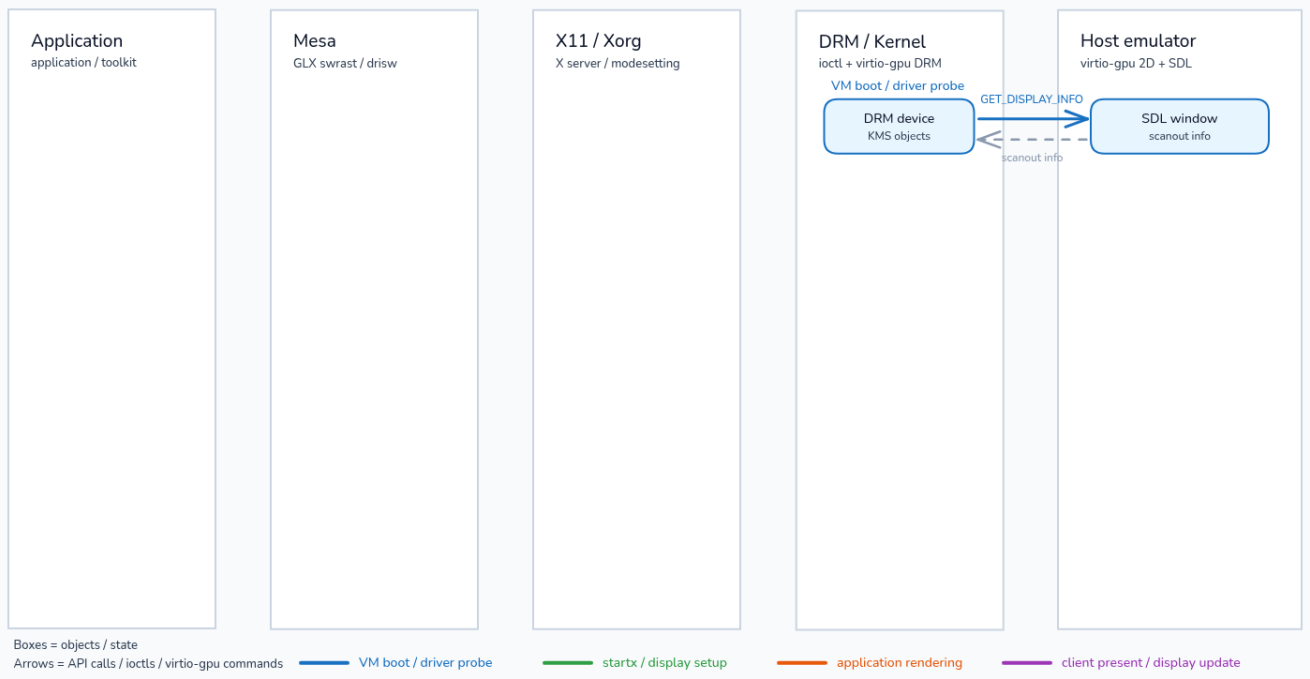
Step 3: Composition

Action	Runs in	Target / result	virtio-gpu relation
Rendering	Application + Mesa swrast	guest CPU writes pixels into client-side color buffer	no virtio-gpu work; all rendering is software
Present	Mesa drisw + X11 request	pixels are sent to X server by XPutImage() / XShmPutImage()	still outside virtio-gpu; this is window-system handoff
Composition	X server	visible regions are copied into screen pixmap / front BO	still outside virtio-gpu; front BO content changes; scanout source is now dirty

Step 4: Scanout Update

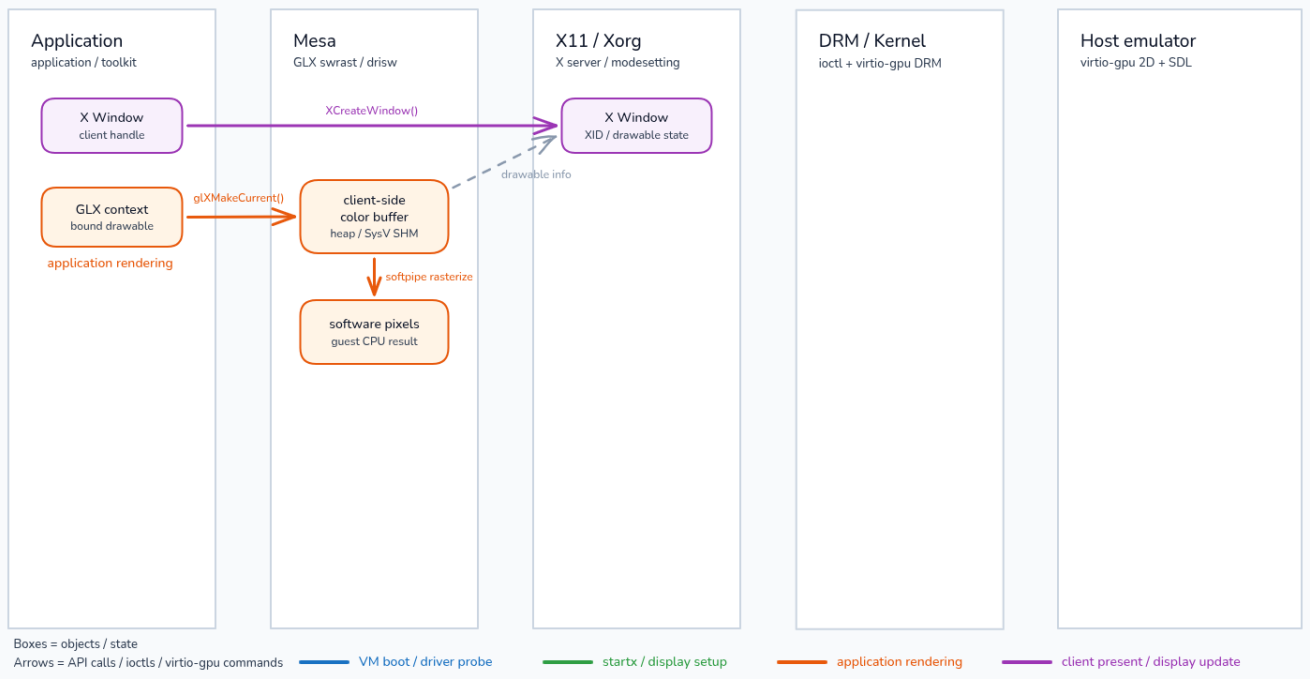
Action	Runs in	Target / result	virtio-gpu relation
Rendering	Application + Mesa swrast	guest CPU writes pixels into client-side color buffer	no virtio-gpu work; all rendering is software
Present	Mesa drisw + X11 request	pixels are sent to X server by XPutImage() / XShmPutImage()	still outside virtio-gpu; this is window-system handoff
Composition	X server	visible regions are copied into screen pixmap / front BO	still outside virtio-gpu; front BO content changes; scanout source is now dirty
Scanout update	Xorg modesetting + DRM/KMS	dirty front BO region becomes display update	TRANSFER_TO_HOST_2D then RESOURCE_FLUSH

Example: GLX Application Flow (Device Probe)

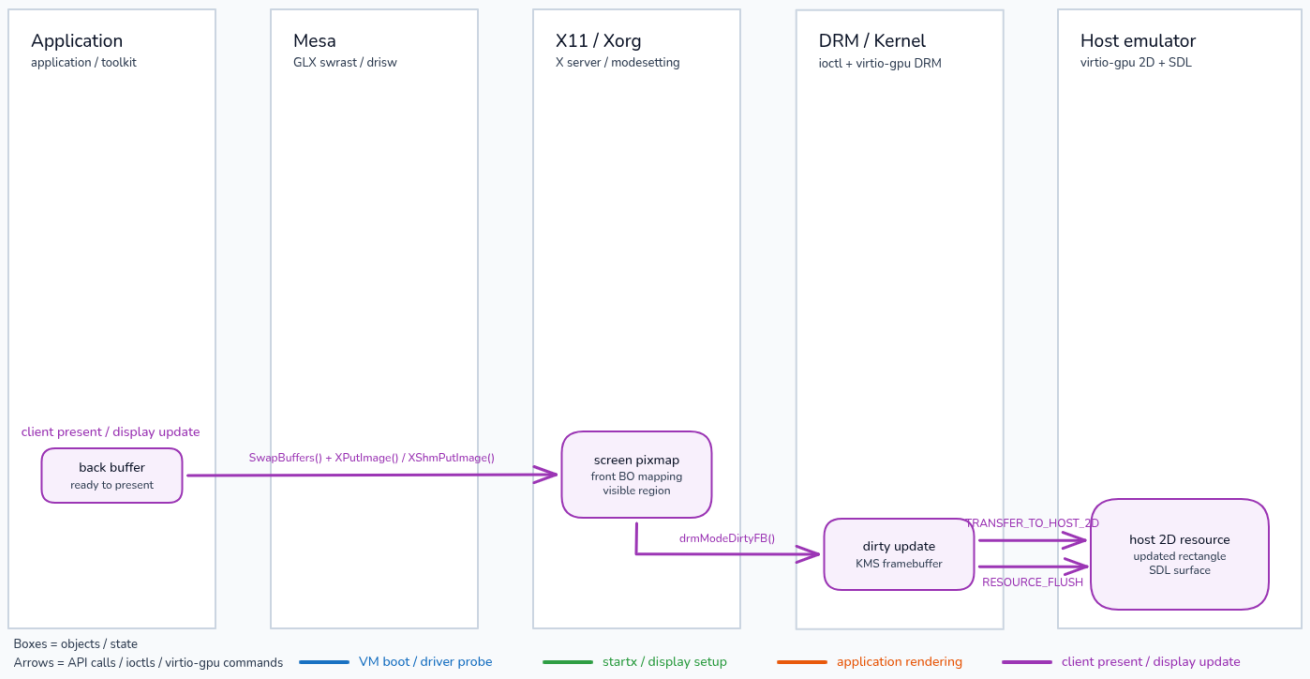


Example: GLX Application Flow (Display Setup)

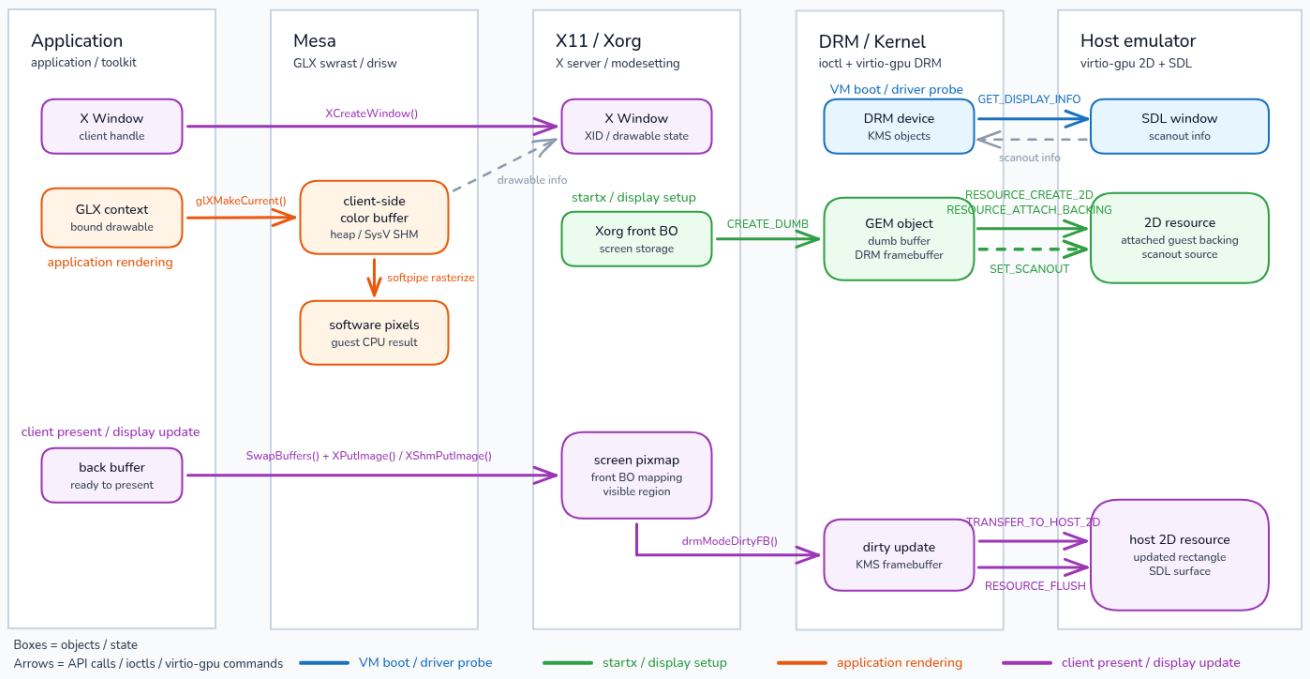
Example: GLX Application Flow (Rendering)



Example: GLX Application Flow (Display Update)



Example: GLX Application Flow (Summary)

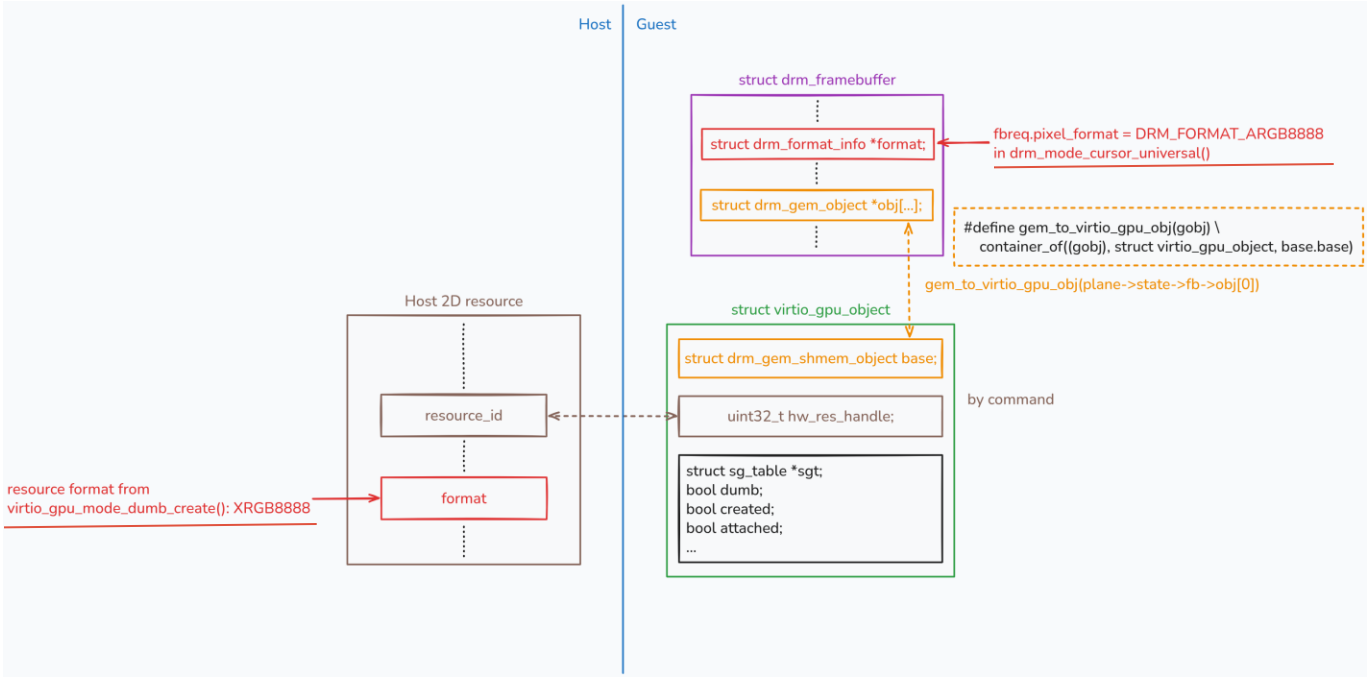


Pitfalls Found During Implementation

(Protocol Alone Is Not Enough)

OPEN SOURCE SUMMIT NORTH AMERICA
Embedded Linux Conference

Issue 1: Cursor RGB format Handling

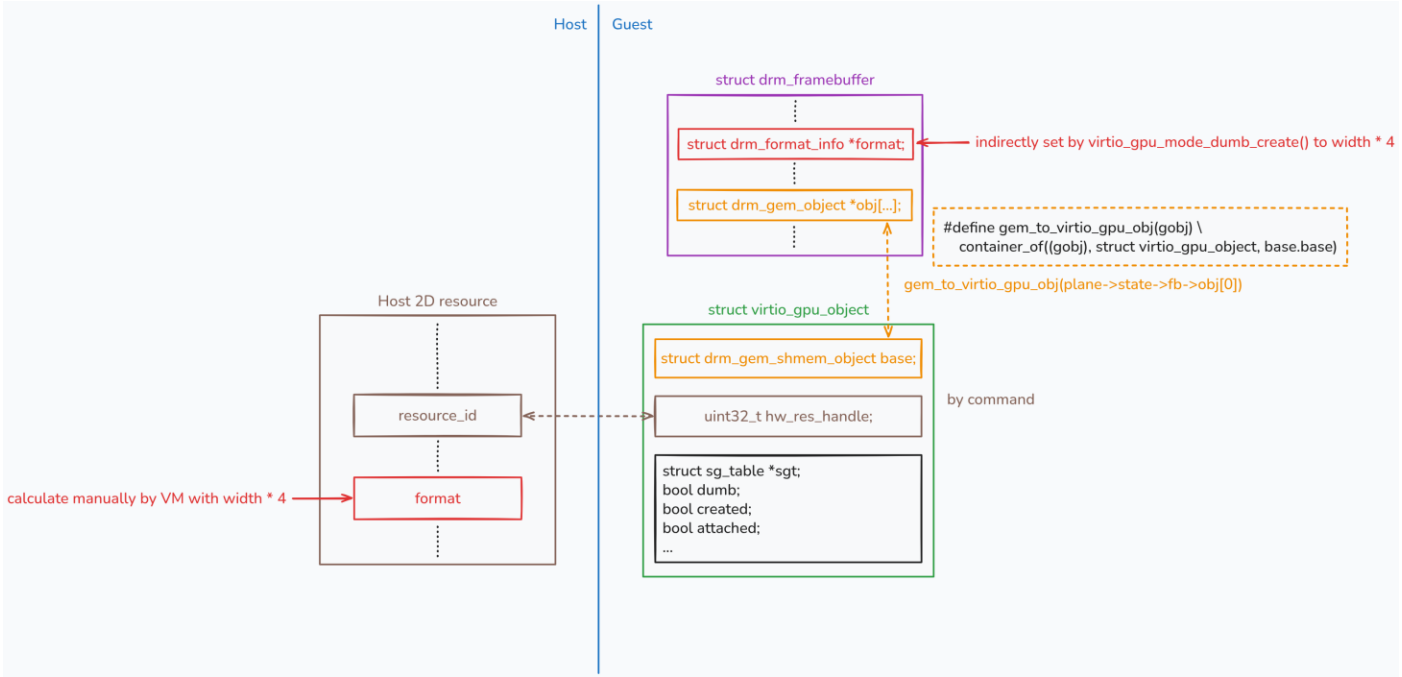


QEMU's solution to Cursor RGB format Handling issue

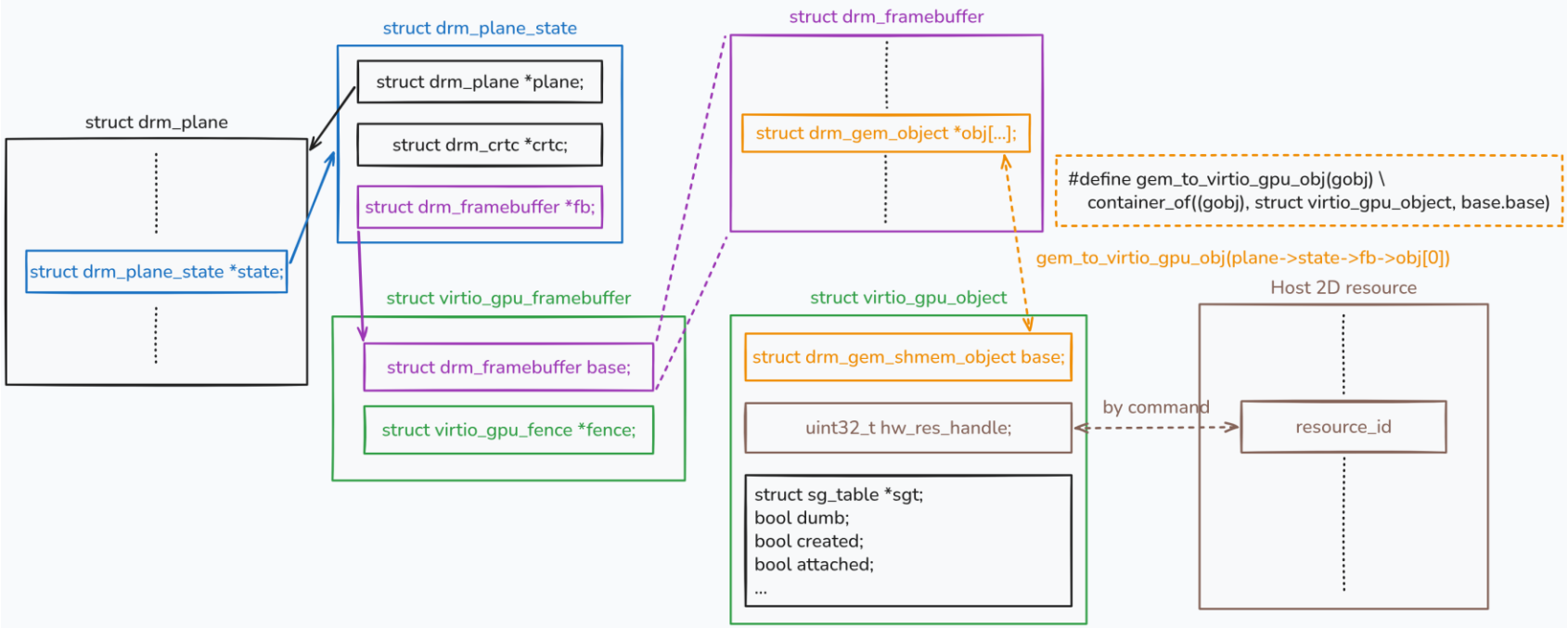
```
/* cursor data format is 32bit RGBA */
typedef struct QEMUCursor {
    uint16_t            width, height;
    int                 hot_x, hot_y;
    int                 refcount;
    uint32_t            data[];
} QEMUCursor;
```

```
guest_sprite_surface =
    SDL_CreateRGBSurfaceFrom(c->data, c->width, c->height, 32, c->width * 4,
                             0xff0000, 0x00ff00, 0xff, 0xff000000);
```

Issue 2: 2D Commands Do Not Carry Layout



From KMS Plane to virtio-gpu 2D Resource

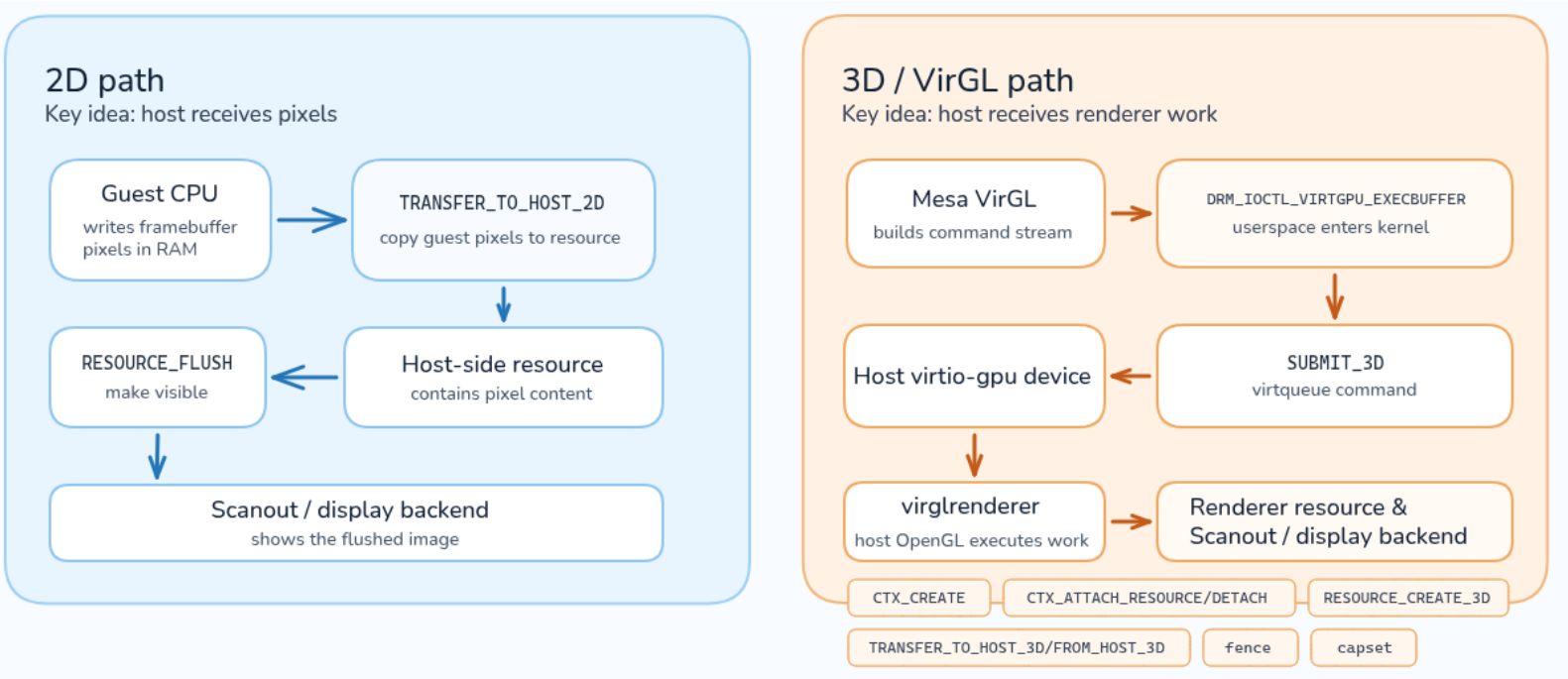


Virtio-gpu 3D, Supplementary Topics and Demos

(virtio-gpu 3D Acceleration)

OPEN SOURCE SUMMIT NORTH AMERICA
Embedded Linux Conference

virtio-gpu 2D / 3D Comparison



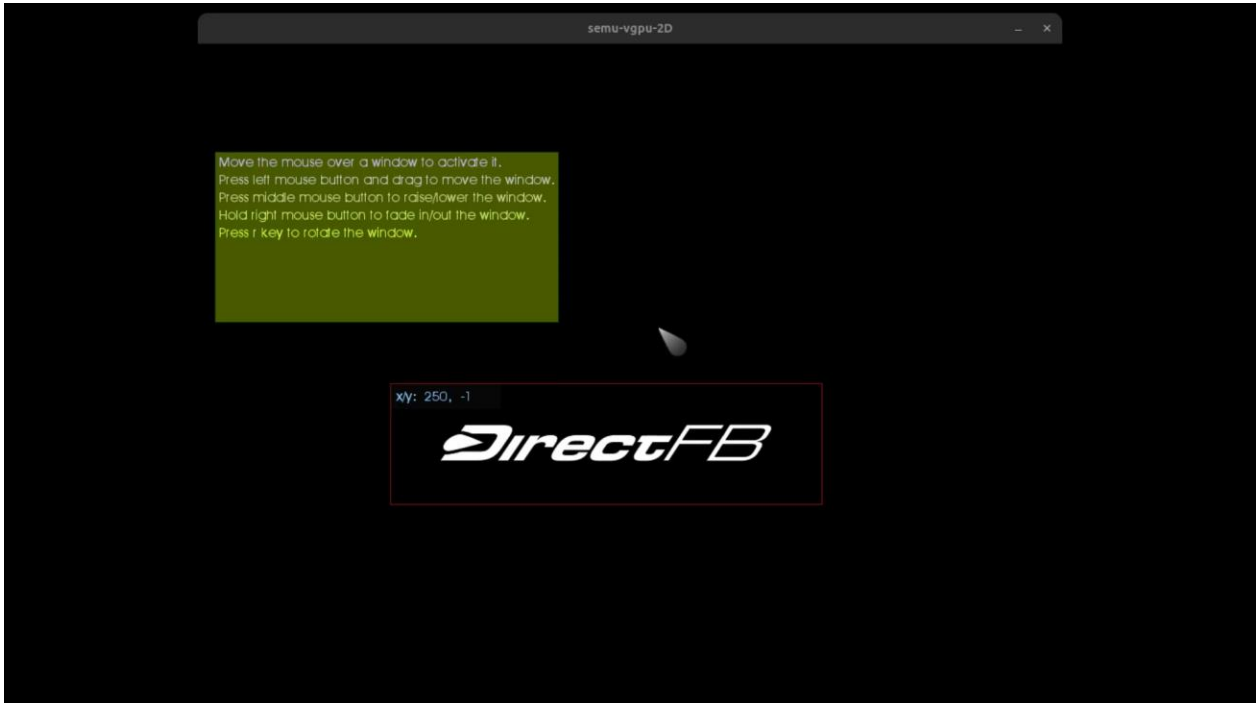
Beyond VirGL

- VirGL proved the model: OpenGL over virtio can work.
- Venus: Vulkan over virtio
- vDRM: native driver protocol over virtio (Rob Clark's talk)
- Recent Mesa VirGL deprecation and unmaintained announcement
  - https://lists.freedesktop.org/archives/mesa-announce/2026-May/000849.html

Recommended Implementation Order

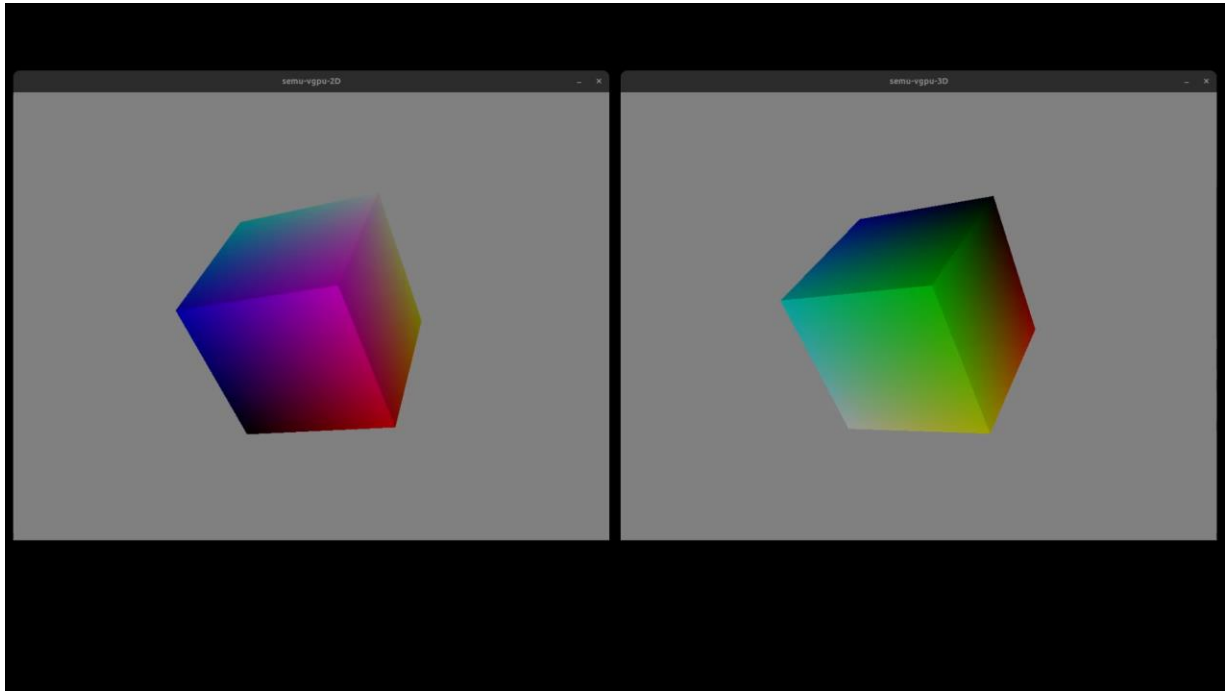
- Ensure virtio correctness
- Implement minimal 2D display support
- Add virtio-input for mouse and keyboard
- Integrate VirGL / 3D acceleration path
- Verify the actual rendering backend

Demo: DirectFB2



https://www.youtube.com/watch?v=HXCayIWxO0M

Demo: kmscube



https://www.youtube.com/watch?v=E_do6xjRoz4

Demo: X11

https://www.youtube.com/watch?v=SseiS5zDdxM

Questions?