GPU on KVM

Gabriel Laskar <gabriel@lse.epita.fr>

Introduction

- How can we have 3D acceleration on VMs?
- Goals:
  - Portability
  - Security
  - and of course Quake !

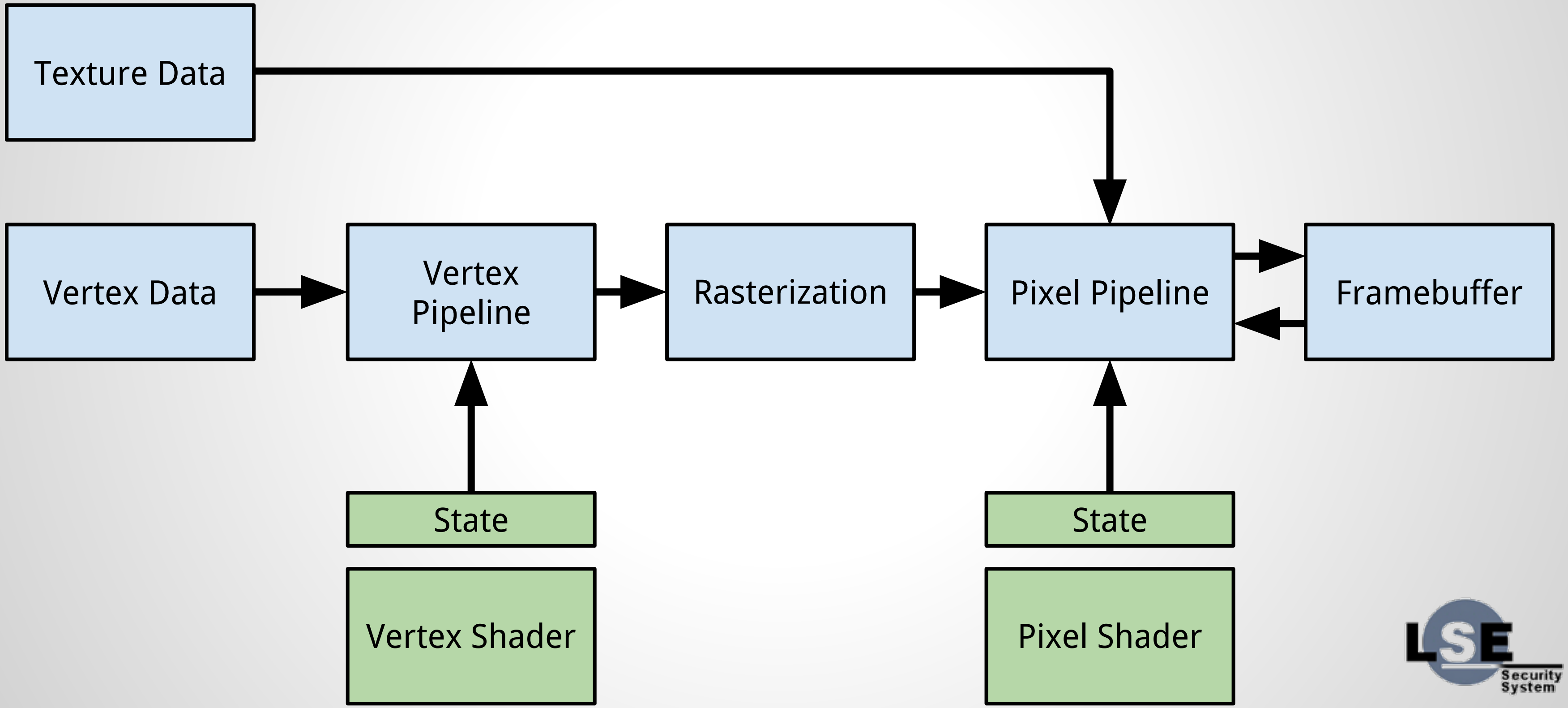
Outline

- What is a GPU?
- How can we bring something to the screen?
- What is a VM?
- A virtualized device?
- How can we bring all this together?

What is a Graphic Card?

- Display
- Video Playback
- 2D & 3D Graphics
- Computation

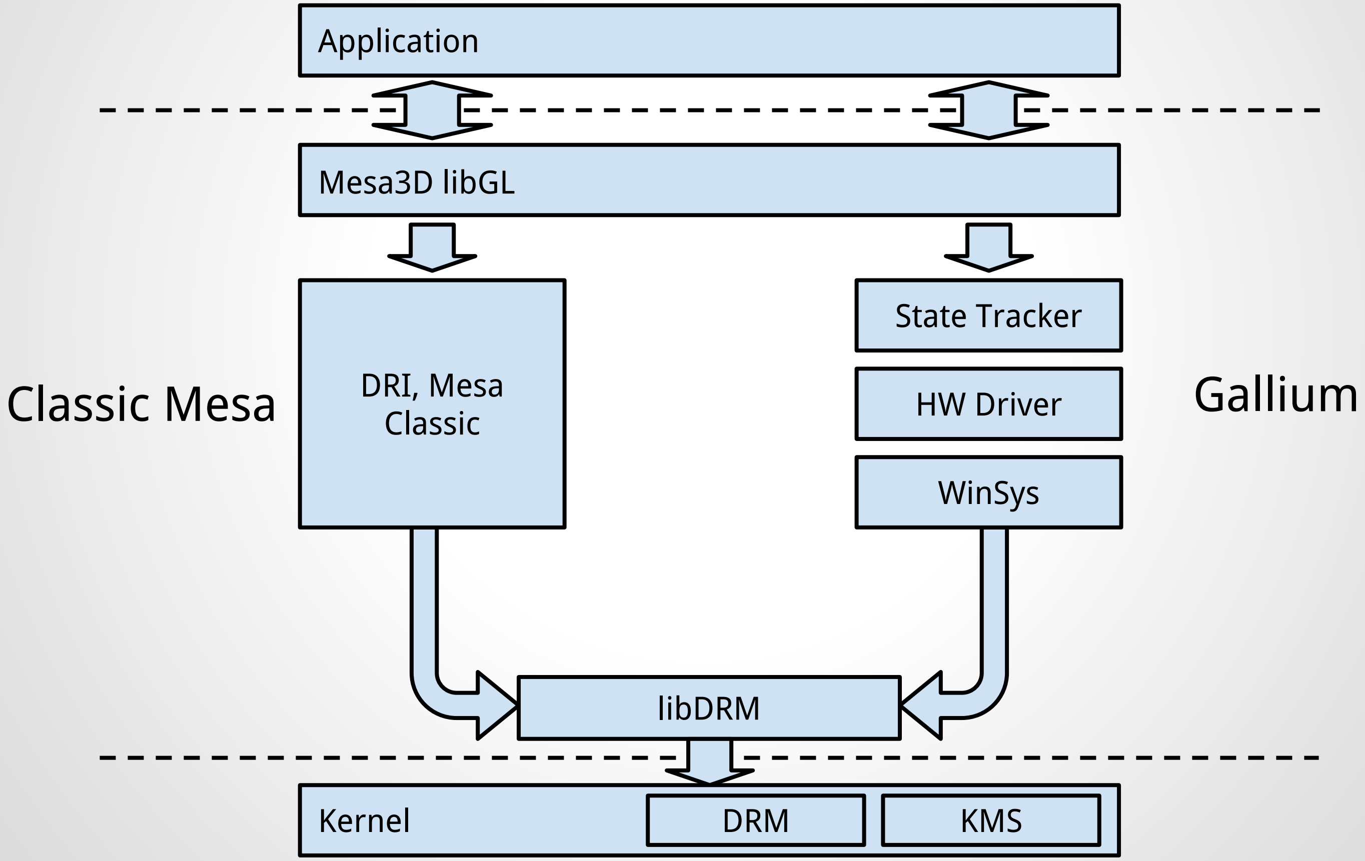
GPUs are complex



GPU API

- Specs unknown
- Every GPU is different
- State is enormous (>1GB size)
- DMA & Computation

Graphic Stack



Mesa3D

- Provides High Level APIs
  - 3D Acceleration: OpenGL/OpenGLES
  - Video Acceleration: XVMC, VAAPI, VDPAU
- Device dependant
- Divided in 2 parts: Mesa classic & Gallium3D

Gallium3D

- New architecture for graphic devices
- Allows code-sharing between drivers
- Used by radeon, nouveau & others
- Provides software fallback

Gallium API

- **Screen:** screen access, context & resource creation
- **Resource:** texture or buffer
- **Surface:** resource binded as a framebuffer
- **Sampler view:** resource for shader use
- **Context:** constant state, resources

Gallium Shaders: TGSI

- Intermediate language for shaders
- Text based
- API not stable yet

TGSI Example

```
VERT

DCL IN[0]
DCL IN[1]
DCL OUT[0], POSITION
DCL OUT[1], COLOR

IMM FLT32 { 0.2, -0.1, 0.0, 0.0 }

ADD OUT[0], IN[0], IMM[0]
MOV OUT[1], IN[1]

END
```

EGL/GLX

- Graphics context management
- Surface/buffer binding
- EGL is an interface between OpenGL and the windowing system

DRM

- Low level access to the GPU
- Perform Kernel Mode Setting
- Export GPU Primitives
  - Context allocations
  - Command queues
  - VRAM management with GEM & TTM
  - Buffer sharing with GEM & DMA-buf
- libDRM in userland wraps the interface

KMS

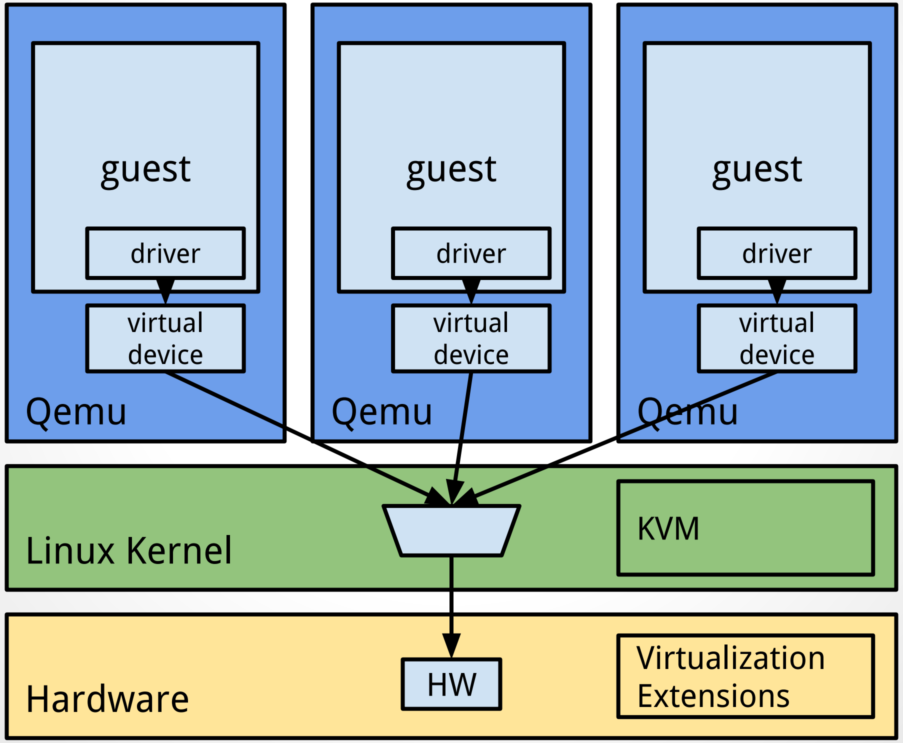
- In kernel API for modsetting
- Allow modesetting without root access
- Glitch-free boot
- Fast VT-Switch
- Kernel crash log
- Better power management

How can we have all of this inside a VM ?

Qemu/KVM

- Linux Hypervisor
- Leverage existing Linux APIs
- Use Qemu for VM Creation & Device Emulation

Qemu/KVM


guest
driver
virtual device
Qemu
guest
driver
virtual device
Qemu
guest
driver
virtual device
Qemu
Linux Kernel
KVM
Hardware
HW
Virtualization Extensions

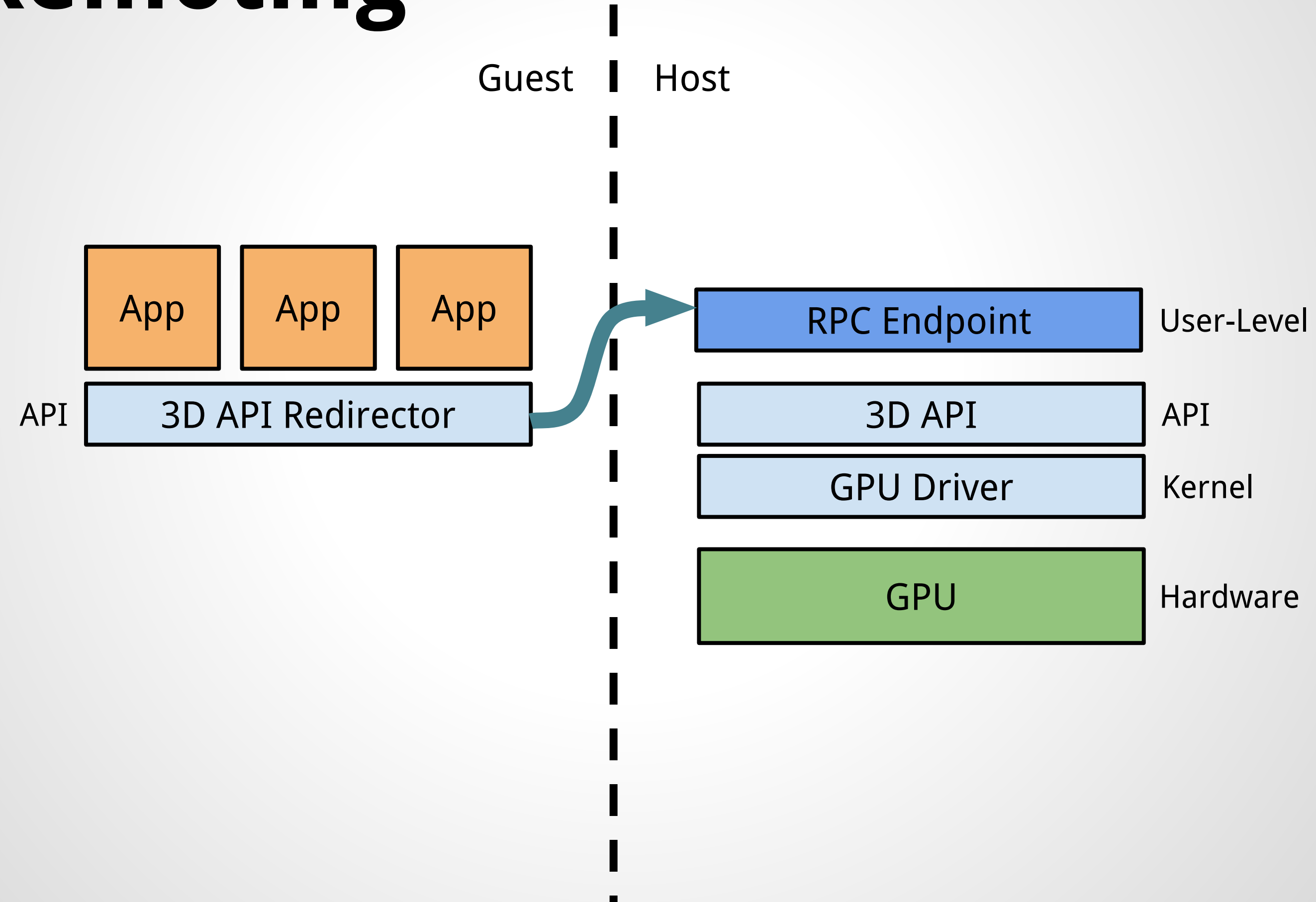
Device Virtualization

- Emulated Devices
  - Too Slow
  - GPU are too complex
- Virtualized Devices
  - Hard to do it right
  - What we want to do
- Hardware Passthrough
  - Good performances
  - No sharing

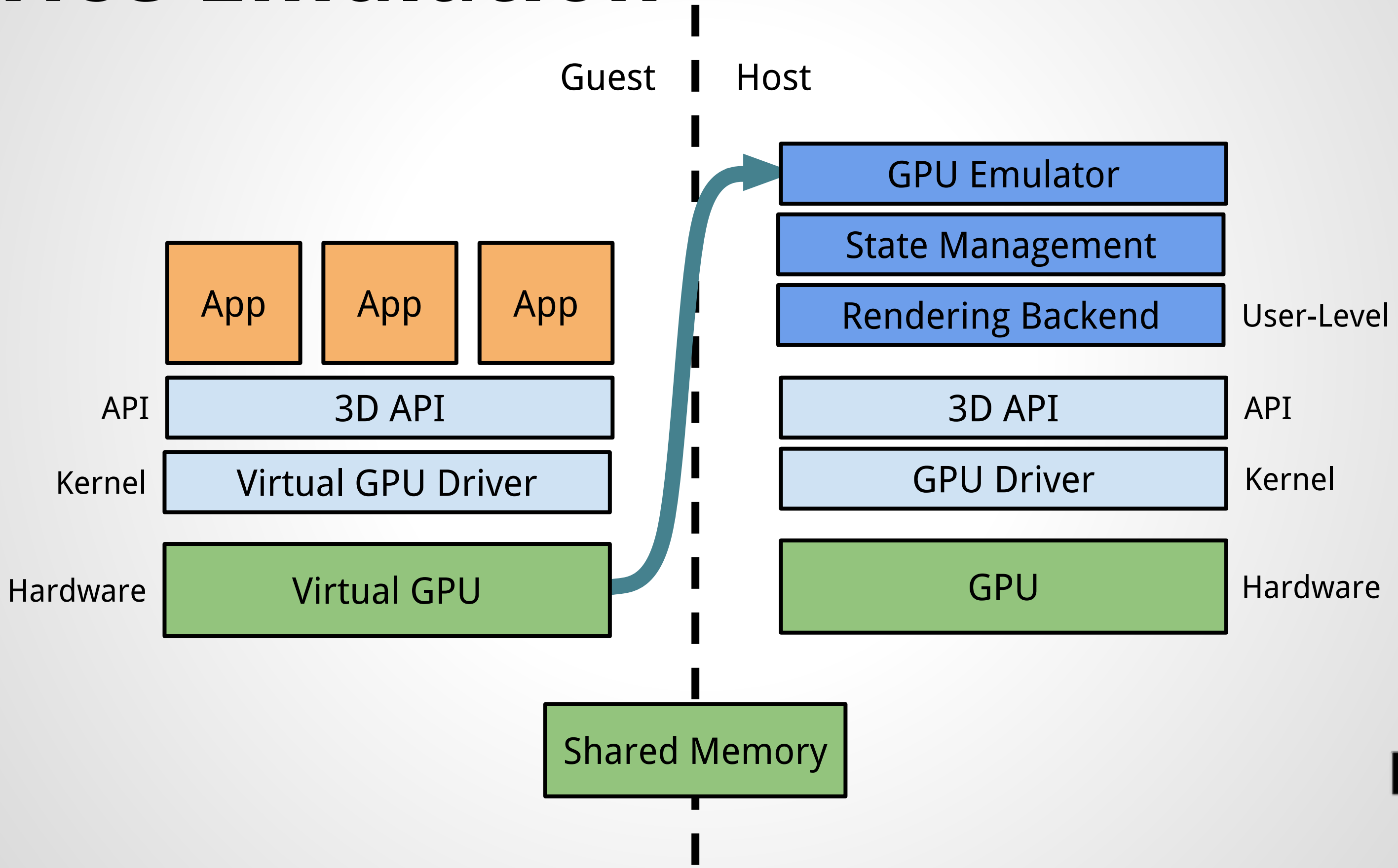
Virtualized OpenGL stack

- API remoting
- Virtual GPU
- HW Sharing: XenGT, Grid

API Remoting



Device Emulation



Virtio Devices

- Standard for virtualized devices
- Already used for net, block & console
- Easy code reuse

Virgl

- Virtio based virtual gpu
- Developed by David Airlie
- 2d and 3d version
- Portability: no need for a specific gpu

Virgl: What’s in it?

- vga device in Qemu
- Renderer Backend in Qemu (for 2D & 3D)
- KMS kernel driver for the guest
- Xorg DDX driver for guest
- Mesa Gallium3D based driver for guest

Virgl: virtio

- Single virtio queue used to send commands to the host
- Protocol is based on Gallium3D
- IRQ for Cursor & Fence IRQ

Current Status

- Error handling
- Capabilities (OpenGL 3.0 for the moment)
- GL versioning
- GLES in guest or host
- Not usable for production yet

Want to try it?

http://virgil3d.github.io/

Questions?