# The Linux graphics stack, Optimus and the Nouveau driver

## Cooperative rendering across GPUs on Linux

Martin Peres

Nouveau developer
PhD student at LaBRI
X.Org Foundation board member

September 24, 2014

# Summary

1. Introduction to the Linux graphics stack
   - General overview
   - Kernel space
   - User space
2. Optimus
3. Kernel
4. Userspace
5. Tools
6. Community

# General overview of the Linux Graphics stack

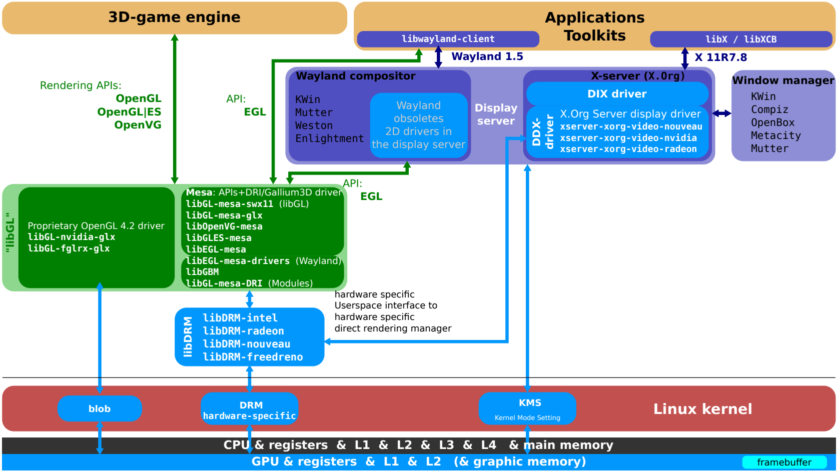
## The graphics stack before 2005

- The X-Server provided everything:
  - Modesetting (CRTC & plane management);
  - 2D/3D acceleration;
  - Video rendering acceleration;
  - Input management.
- The X-Server talked to the GPU directly, as root.

## The current graphics stack

- The X-Server got split into more than 200 components:
  - Privileged operations moved to the kernel;
  - 2D drivers got put into different shared objects;
  - 3D acceleration got put in mesa;
  - The list is too long ;)

by Shmuel Csaba Otto Traian; CC-BY-SA 4.0 intl; created 2013-08-24; last updated 2014-03-25



3D-game engine

Applications
Toolkits

libwayland-client

libX / libXCB

Wayland 1.5

X 11R7.8

Rendering APIs:
OpenGL
OpenGL|ES
OpenVG

API:
EGL

Wayland compositor
KWin
Mutter
Weston
Enlightment

Wayland obsoletes 2D drivers in the display server

Display server

X-server (X.Org)

DIX driver

DDX-driver

X.Org Server display driver
xserver-xorg-video-nouveau
xserver-xorg-video-nvidia
xserver-xorg-video-radeon

Window manager
KWin
Compiz
OpenBox
Metacity
Mutter

"libGL"

Proprietary OpenGL 4.2 driver
libGL-nvidia-glx
libGL-fglrx-glx

Mesa: APIs+DRI/Gallium3D driver
libGL-mesa-swx11 (libGL)
libGL-mesa-glx
libOpenVG-mesa
libGLES-mesa
libEGL-mesa
libEGL-mesa-drivers (Wayland)
libGBM
libGL-mesa-DRI (Modules)

API:
EGL

hardware specific
Userspace interface to
hardware specific
direct rendering manager

libDRM
libDRM-intel
libDRM-radeon
libDRM-nouveau
libDRM-freedreno

blob

DRM
hardware-specific

KMS
Kernel Mode Setting

Linux kernel

CPU & registers & L1 & L2 & L3 & L4 & main memory

GPU & registers & L1 & L2 (& graphic memory)

framebuffer

Figure: General overview of the Linux graphics stack

# The kernel space

## Direct Rendering Manager (DRM) : The common code

- This common code provides:
  - Kernel ModeSetting (KMS): CRTC & plane management;
  - Video memory management via GEM (with a TTM backend?);
  - Nodes with different capabilities (master or render nodes).

## DRM open source drivers

- i810/i915: Intel;
- nouveau: NVIDIA;
- radeon: AMD/ATI;
- vmwgfx: VMware;
- many SoC GPUs (armada, exynos, msm, omap, tegra, ...).

# Architecture of the X-Server

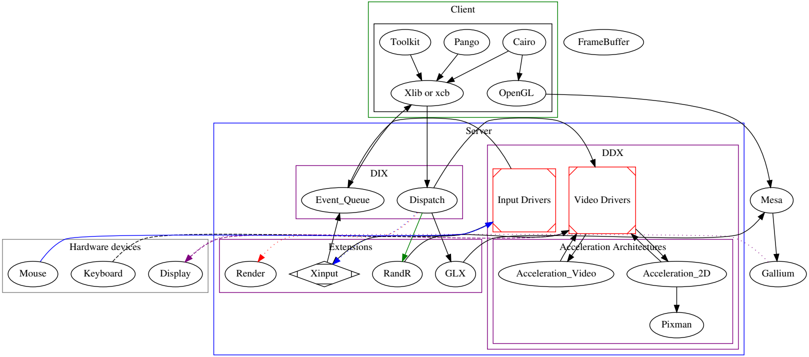


Figure: General overview of the X-Server's internal architecture

# Architecture of Mesa

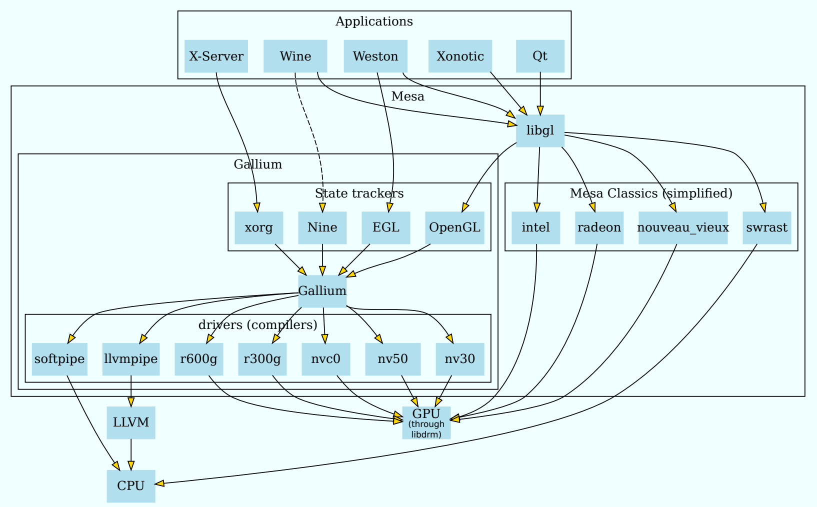


Figure: General overview of Mesa's internal architecture

# Summary

1 Introduction to the Linux graphics stack

2 Optimus

- Introduction
- Turning the dGPU on/off
- Driving the right outputs

3 Kernel

4 Userspace

5 Tools

6 Community

# Great performance, great battery-life

## Optimus

- Laptops can be equipped with two GPUs;
- The Intel IGP is great for battery-life;
- NVIDIA's discrete GPU (dGPU) is great for performance;
- Dynamic switch between the 2: get the best of both worlds!

## Challenges

- When/How the dGPU should be turned on/off?
- Who drives the outputs?
- How to copy buffers from a driver to another?
- How should we handle the HDMI "sound card"?

# Turning the dGPU on/off

## How

- Optimus laptops have ACPI functions to do that;
- Two ways of calling them:
  - bbswitch: Old kernel module for manual management;
  - vgaswitcheroo: Manual or automatic state management.

## When: The case of vgaswitcheroo

- Turn off the dGPU when it has been idle for 5 seconds;
- Idle?:
  - no graphics context allocated;
  - no output is being used;
  - no sound interface used (not done);
  - no call to the drm driver has been made;

# Handling the outputs : Hardware multiplexer

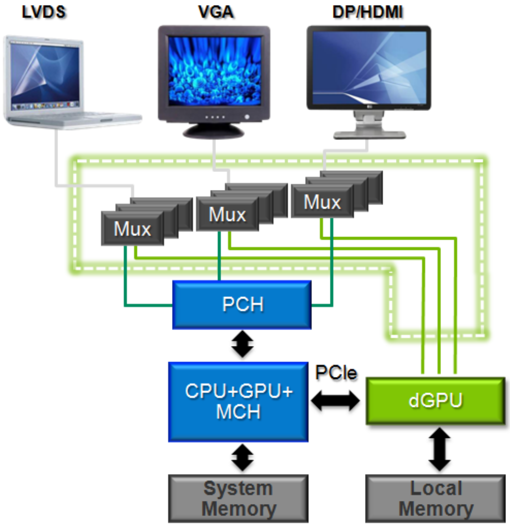


Figure: Switchable graphics

# Handling the outputs : Software multiplexer

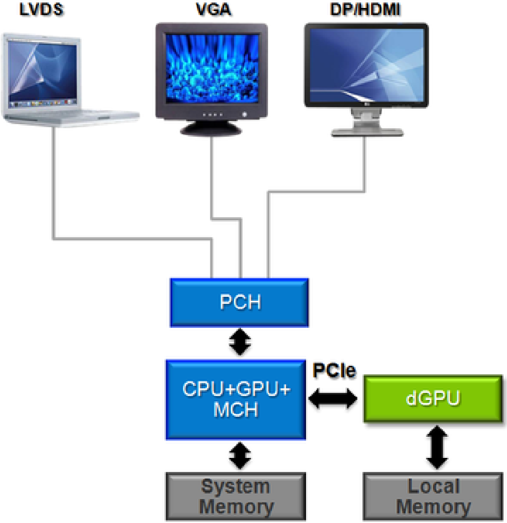


Figure: The "real" optimus architecture

# Switching from one GPU to another : How windows does it

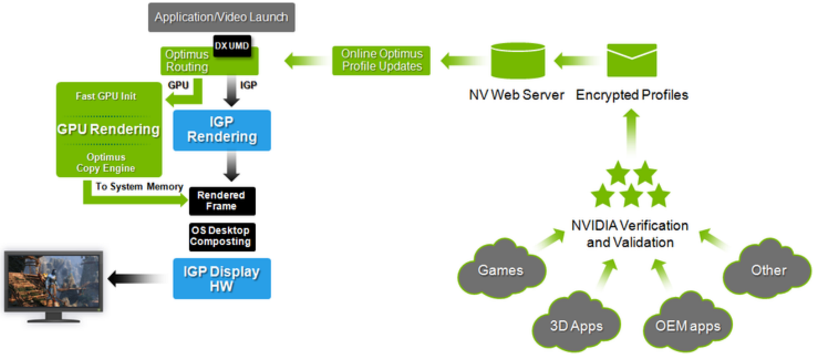


Figure: The global hardware/software infrastructure

# Sharing buffers across drivers

## Challenges

- The memory representation for buffers is different from hardware to hardware:
    - pitch: number of pixels per row;
    - tiling: technique that increases the spatial locality.
- Synchronising rendering across drivers.

## DMA-Buf

The memory representation for buffers is different from hardware to hardware:

- pitch: number of pixels per row;
- tiling: technique that increases the spatial locality.

# Summary

1. Introduction to the Linux graphics stack
2. Optimus
3. Kernel
    - Optimus/prime
    - Power Management
4. Userspace
5. Tools
6. Community

# Prime

## Prime

Prime is the name for all the open source technologies that make hybrid graphics possible:

- vgaswitcheroo: switching graphics ;
- running the nouveau ddx;

## List of requirements

- running nouveau/radeon drm;
- running the nouveau ddx;