

# 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 26, 2014

# Summary

- 1 Introduction to the Linux graphics stack
  - General overview
  - Kernel space
  - User space
- 2 Optimus
- 3 Prime
- 4 Nouveau
- 5 Q&A

# 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 (and boring) ;)

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

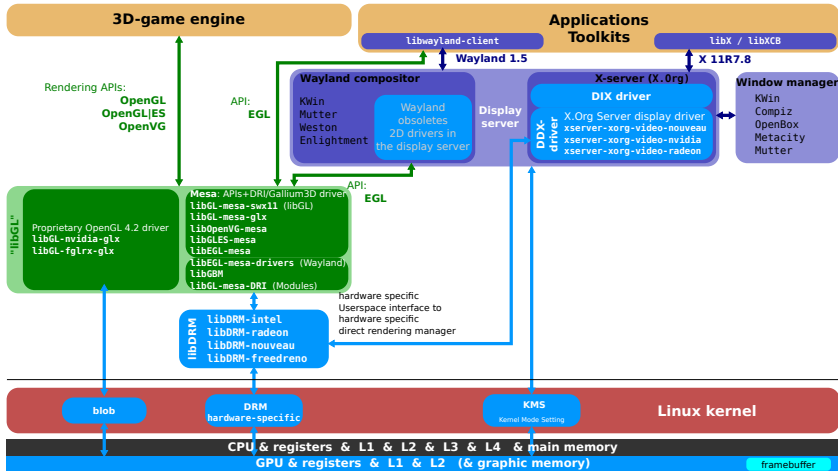


Figure: General overview of the Linux graphics stack

# 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

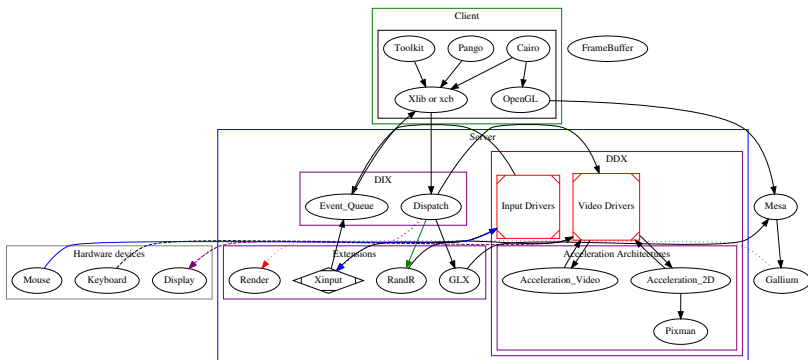


Figure: General overview of the X-Server's internal architecture

# Architecture of Mesa

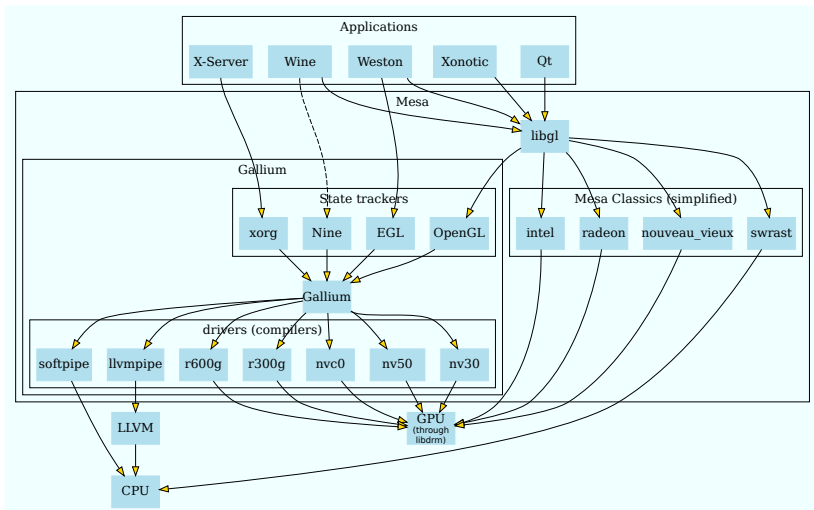


Figure: General overview of Mesa's internal architecture

# Summary

- 1 Introduction to the Linux graphics stack
- 2 **Optimus**
  - Introduction
  - Turning the dGPU on/off
  - Driving the right outputs
  - How to share buffers across drivers?
- 3 Prime
- 4 Nouveau
- 5 Q&A



# 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 one driver to another?
- How should we do application migration?
- 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 power 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

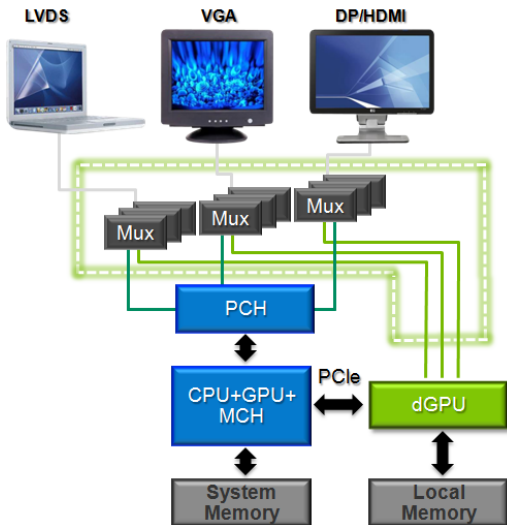


Figure: Switchable graphics

# Handling the outputs : Software multiplexer

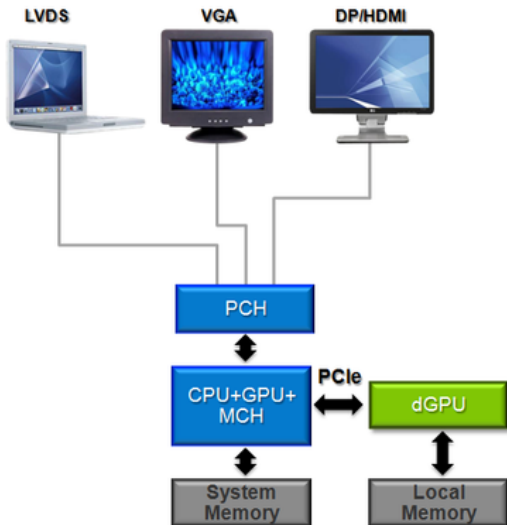


Figure: The “real” Optimus architecture

# Sharing buffers across drivers

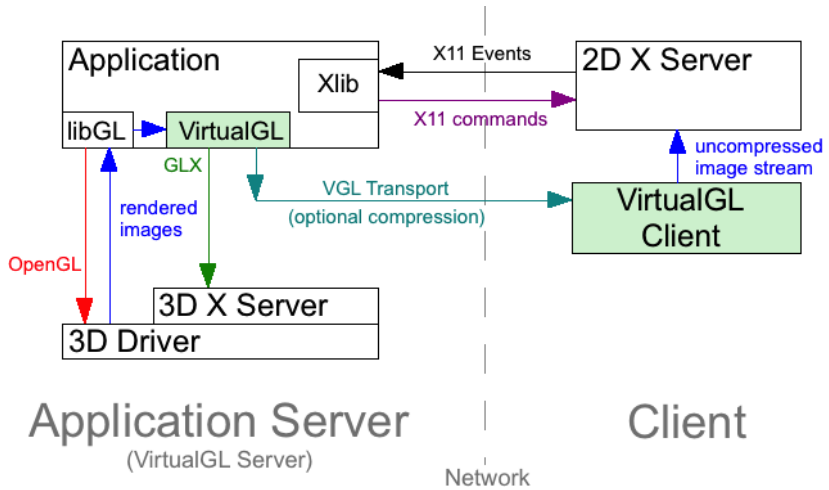
## Cross-driver BO sharing : Challenges

- 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.

## Solutions

- VirtualGL: Remote rendering solution. Redirects drawing commands to a distant GPU, read back the rendered frame;
- Primus: Same solution as VirtualGL except in a more lightweight fashion!
- DMA-Buf: A Linux-only solution that allows sharing buffers between different GPUs without copies.

# Sharing buffers across drivers : VirtualGL



# Sharing buffers across drivers : DMA-Buf

## Driver roles

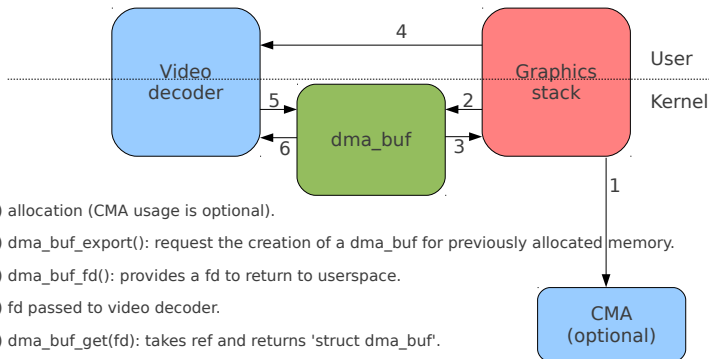
- Exporter: Being able to export a buffer;
- User: Being able to import a buffer.

## General overview

- No standardised memory allocation: It's up to the exporter;
- An arbitrary buffer can be wrapped into a DMA buffer;
- An fd can be returned to the userspace to reference this buffer;
- The fd can be passed to another process;
- The fd can be mmaped and accessed by the CPU;
- The fd can be imported by any driver supporting the user role.



# dma\_buf usage flow



- 1) allocation (CMA usage is optional).
- 2) `dma_buf_export()`: request the creation of a `dma_buf` for previously allocated memory.
- 3) `dma_buf_fd()`: provides a fd to return to userspace.
- 4) fd passed to video decoder.
- 5) `dma_buf_get(fd)`: takes ref and returns 'struct dma\_buf'.
- 6) `dma_buf_attach()` + `dma_buf_map_attachment()`: to get info for dma
  - a) `dev->dma_parms` should be expanded to tell if receiving device needs contiguous memory or any other special requirements
  - b) allocation of backing pages could be deferred by exporting driver until it is known if importing driver requires contiguous memory.. to make things a bit easier on systems without IOMMU



# How should we do application migration?

## Short answer

**We don't!**

## Why?

- The context is hardware-dependent;
- The context can be HUGE (hundreds of MB);
- The best way is to ask the application to re-upload its context: `GL_ARB_robustness`.

# Optimus : How windows does it

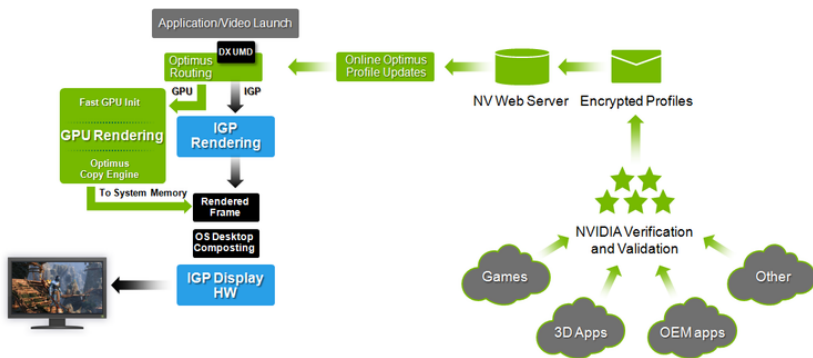


Figure: The global hardware/software infrastructure

# Summary

- 1 Introduction to the Linux graphics stack
- 2 Optimus
- 3 Prime**
  - Introduction
  - How to
  - Demos
- 4 Nouveau
- 5 Q&A

# Prime

## Prime

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

- DMA-Buf: sharing buffers across drivers (Linux 3.3);
- vgaswitcheroo: switching graphics (Linux 3.12);
- Cross-device fence mechanism: make a driver wait on another driver to complete a task (Linux 3.17);
- DMA-Buf synchronisation: Wait for rendering completion of a DMA-Buf before compositing to avoid tearing (Linux 3.19?).

## List of requirements

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

# Prime: Simplified how-to for Nouveau

## vgaswitcheroo

- # cd /sys/kernel/debug/vgaswitcheroo/
- # cat switch
- # echo (DIGD|DDIS) > switch
- (Re)start your desktop environment.

## XRandr

- \$ xrandr --listproviders
- \$ xrandr --setprovideroffloadsink nouveau Intel
- This sets the order: Nouveau == offload, Intel == default

# Prime: Simplified how-to for Nouveau

## Usage

- `DRI_PRIME=1 glxgears # Use the NVIDIA GPU`
- `DRI_PRIME=0 glxgears # Use the Intel GPU`
- `glxgears # Use the Intel GPU`

## Longer How-to for Nouveau

<http://nouveau.freedesktop.org/wiki/Optimus>

# Prime: Demos

## Current setup

- This is an Optimus laptop (Sandy Bridge + NVIDIA NVD9);
- All the outputs are connected to the Intel IGP.

## List of demos

- Selecting the GPU and checking with `glxinfo`;
- Performance difference in `glxgears`;
- Video decoding with VDPAU on the NVIDIA GPU.

# Summary

- 1 Introduction to the Linux graphics stack
- 2 Optimus
- 3 Prime
- 4 Nouveau**
  - Introduction
  - Current work
  - Involvement from NVIDIA
- 5 Q&A



# Nouveau: Introduction



**NOUVEAU**

## Nouveau: An Open Source Linux driver for NVIDIA GPUs

- Merged in Linux 2.6.33 & left staging on Linux 3.4;
- Mostly developed by Red Hat and students.

## Current features

- Modesetting support for almost all NVIDIA GPUs;
- 2D, 3D and video-rendering accel on NV04-;
- Video decoding accel on NV40-NV117 (non-free).

# Nouveau: Current developments

## Current work

- Maxwell support:
  - Released in two times (March then September);
  - Modesetting: DONE;
  - 2D/3D support: MOSTLY DONE;
  - Video decoding: TODO
  - Open source firmware: WIP
- Manual reclocking support:
  - nv40-a3: crude support, disabled by default;
  - nva3-ac: good chances of working;
  - Fermi: crude support, disabled by default;
  - Kepler: WIP, good chances of partial support;
  - Maxwell: TODO
- Adding new OpenGL extensions:
  - Everything is done up to Fermi;
  - OpenGL 4 for the other GPUs.

# Involvement from NVIDIA

## NV

- 1998(?): NVIDIA releases “nv”, a Linux OSS 2D driver;
- 1998: Obfuscation commit, release only pre-processed source.

## Little hope of NVIDIA ever working again on an OSS driver

“It’s so hard to write a graphics driver that open-sourcing it would not help [...] In addition, customers aren’t asking for opensource drivers.”

Andrew Fear, NVIDIA software product manager, April 2006

# Short history of Nouveau

## Nouveau

- 2005: Stephane Marchesin improves nv and works on 3D
- 2008: Open Arena runs on nv40
- 2009: KMS driver based on TTM for memory management
- 2010: Merged in Linux 2.6.33
- 2010: Nv is deprecated by NVIDIA, “use VESA”.

## A new hope from NVIDIA

- September 2013: NVIDIA releases some vbios documentation;
- January 2014: NVIDIA starts adding support for their Tegra K1 in Nouveau, as requested by its clients;
- Full support for the Tegra K1 expected by the end of 2014.

# Summary

- 1 Introduction to the Linux graphics stack
- 2 Optimus
- 3 Prime
- 4 Nouveau
- 5 Q&A**

## Questions?

Thank you for listening! Questions?

Martin Peres: [martin.peres@free.fr](mailto:martin.peres@free.fr)