Status of GPU offloading on Wayland

Axel Davy

FOSDEM 2014

Using a device

Traditional way:

- A DRM Master
- Clients need to be authenticated by the DRM Master to render

New way: Render-nodes. Allow to render without authentication (but without some functionalities)

Sharing the buffers

Access:

- VRAM: per-device
- RAM with GTT: cross-device

Sharing:

- Handles → per context
  Use example: Mesa internally, KMS

- Gem names → per device *insecure*
  Use example: DRI2 DDX to allocate a buffer for Mesa

- Prime/Dma-buf fd → to share *secure*
  Use example: Wayland, DRI2 GPU offloading, DRI3

Memory Speed

Speed:

- VRAM/RAM: fast.
  DDR3 900Mhz/128bits $\rightarrow$ read 14,4 GB/s + write 14,4 GB/s
- PCI express 2.0 x8: 8 x 500Mhz = 4 GB/s
- Thunderbolt $\approx$ 1 GB/s

A 1080p screen buffer: $\approx$ 8 MB

60 screen buffer transfer per second: $\approx$ 480 MB/s

Memory Speed

My system:

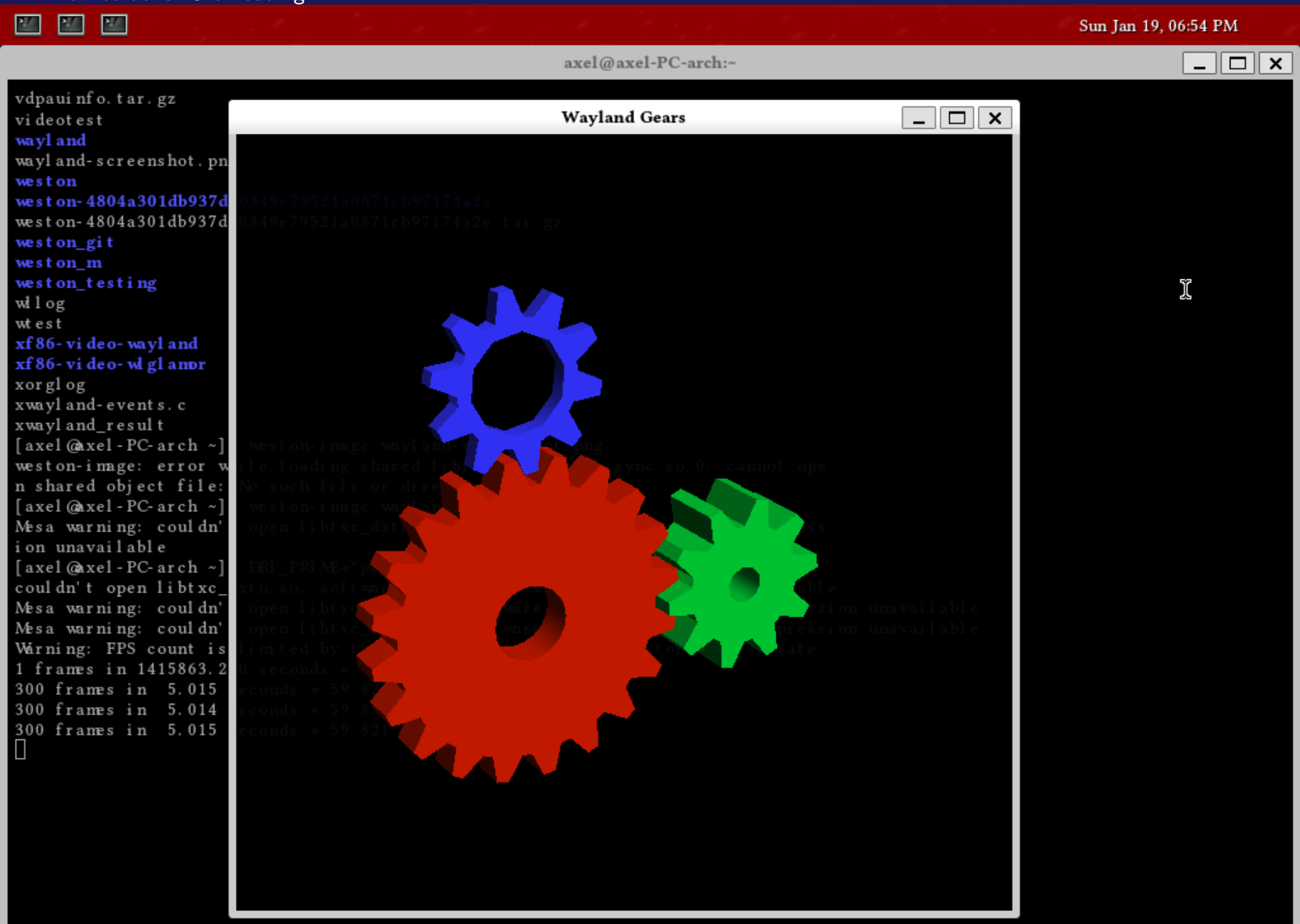
- intel HD4000. Ram DDR3 800Mhz.
- Amd HD7730m. VRAM DDR3 900Mhz. PCI express 2.0 x8.

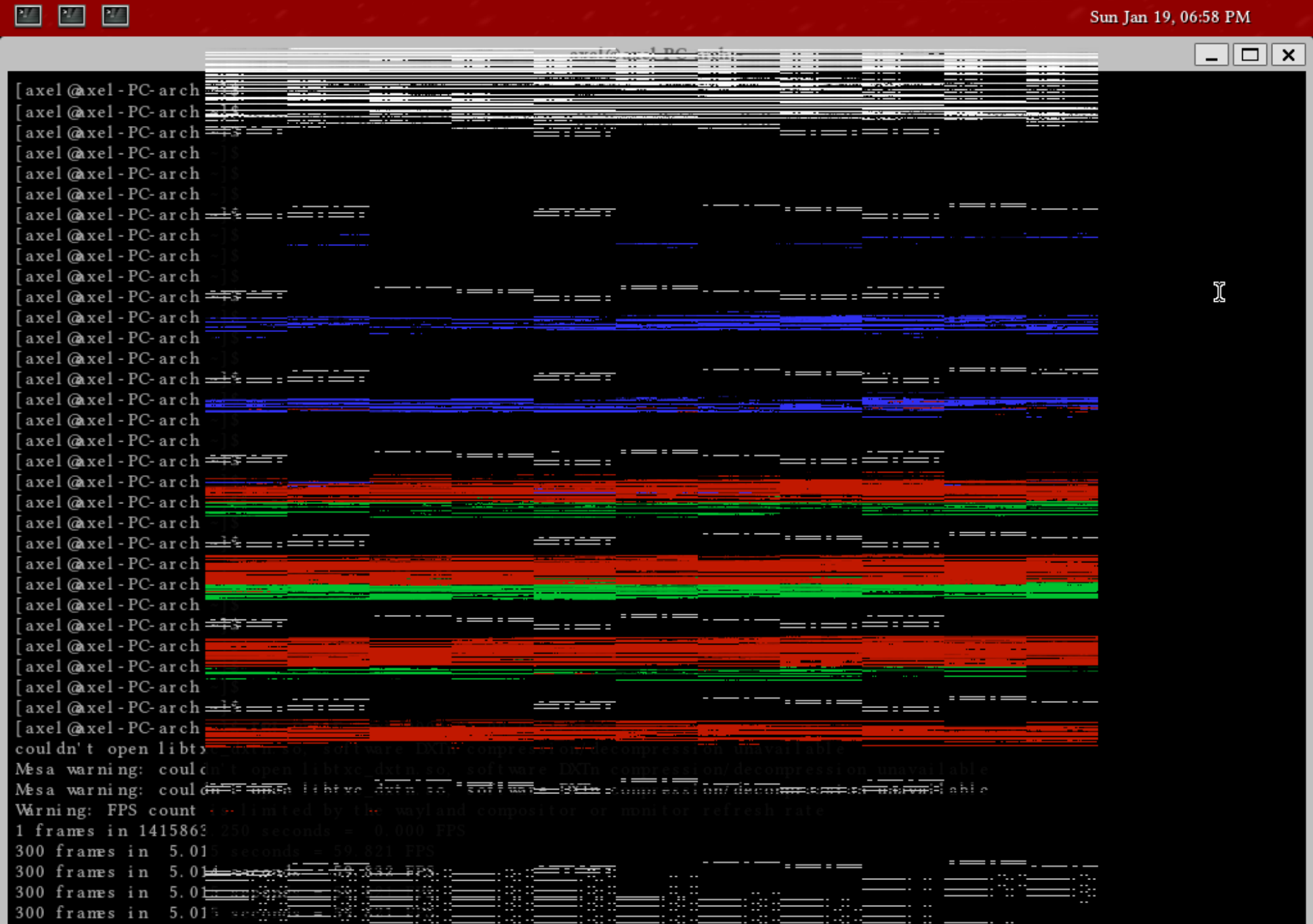
Rendering glmark2 on wayland ('build' test) in RAM:
Intel HD4000: 1320 fps $\approx$ 10.5 GB/s
Amd HD7730m: 250 fps $\approx$ 2 GB/s

Tiling

- Tiling: Special pixel ordering optimized to exploit local spatial coherence
  $\rightarrow$ good for performance !
- Not understandable between different card models/generations !

Example: Intel HD4000. OpenArena
tiling $\rightarrow$ 32 fps
no tiling $\rightarrow$ 10 fps


Sun Jan 19, 06:54 PM
axel@axel-PC-arch:~
vdpauinfo.tar.gz
videotest
wayland
wayland-screenshot.pn
weston
weston-4804a301db937d
weston-4804a301db937d
weston_git
weston_m
weston_testing
wlog
wtest
xf86-video-wayland
xf86-video-wlglamor
xorglog
xwayland-events.c
xwayland_result
[axel@axel-PC-arch ~]
weston-image: error w
n shared object file:
[axel@axel-PC-arch ~]
Mesa warning: couldn'
ion unavailable
[axel@axel-PC-arch ~]
couldn't open libtxc_
Mesa warning: couldn'
Mesa warning: couldn'
Warning: FPS count is
1 frames in 1415863.2
300 frames in 5.015
300 frames in 5.014
300 frames in 5.015
Wayland Gears


Sun Jan 19, 06:58 PM
[axel@axel-PC-arch
couldn't open libtx
Mesa warning: could
Mesa warning: could
Warning: FPS count
1 frames in 1415863
300 frames in 5.01
300 frames in 5.01
300 frames in 5.01
300 frames in 5.01

Dmabuf fences

Work in progress by Maarten Lankhorst
http://cgit.freedesktop.org/~mlankhorst/linux

→ will remove remaining glitches!

Associate to each Dma-buf:

- One write fence
- Several read fences

Extra feature: userspace can poll a dma-buf

X DRI2

Main mechanism:

- Client gets the device path, opens it and authenticates to the server.
- Client gets a buffer from the X server. It renders to it.
- Client tells X it has finished. X copies the buffer content to a correct location.

- A DDX per device/provider
- Manual configuration in xorg.conf or automatic
- GPU offloading configured with XRandr. Two modes:
  - One gpu for display/One gpu for rendering
  - One gpu for display + rendering/One gpu for offloading DRI_PRIME to specify the GPU to use (by indicated the provider number)

- With Prime, a buffer is created, shared between the two devices, and with no tiling.
  $\rightarrow$ this requires special DDX code
- DRI2 copy is done to this buffer.
- When the client is fullscreen, this buffer is used for the screen pixmap, else there will need compositing to make the content be copied to the screen pixmap.
- Everytime a part of the shared buffer is damaged, the whole buffer is damaged.

Current issues

No synchronization $\rightarrow$ tearings.

BUT Content ok

Wayland

Main mechanism:

- Client gets the path of the device used by the compositor, opens it and authenticates to the server (or opens the render-node of another device).
- Client creates a set of buffers and lets the compositor know their existence. Renders to one, tell the compositor it has rendered to it, then render to another one. Will wait the compositor has released a buffer to use it again.

What we would want to improve over DRI2

- Tearings
- Synchronization
- Need of server side support for Prime
- No configuration needed
- Support for every graphic device
- Hot plug support
- Buffer compatibility with the main graphic device handled client side, not server side

First scheme.

- Server advertises the cards it can authenticate to.
- Client can ask to authenticate to these cards.
- Client sends a buffer the server's card can read (linear tiling).

What we want to improve:

- Less server side code
- Simplificate the code

New Scheme

Rely on render-nodes:

- The server doesn't need to know the existence of the other cards
- No need of extra code!

No provider number here.
→ ID_PATH_TAG, tag given by udev.

Example: launching glmark2-wayland on my dedicated card:

DRI_PRIME="pci-0000_01_00_0" glmark2-wayland
or (not for compositors)
DRI_PRIME=1 glmark2-wayland

→ hotplug, external devices, etc can be supported!

Rendering to linear buffer isn't optimal.
$\rightarrow$ Render to a tiled buffer, and copy to a linear buffer shared with the compositor

Two ways:

- Embed clients in an Wayland compositor running on the dedicated card
  - Copy done in the embedded compositor.
  - But induces small lag for input/output, and more cpu comsumption.
  - Glitches only if input lag $>$ (1/refresh rate)ms
- Do the copy in Mesa
  - Glitches if we don't glFinish
  - But glFinish induces a loss of performance

In both cases

- You can rull full desktop on the card you want
- No tearings !
- Vsync working

Several cards displaying

OK, but what about the following case:

- Two displays, A and B.
- Two cards, "1" connected to A, "2" connected to B.

X DRI2

- Server controls the devices
- DDX for each device
- Copy tiled buffer $\rightarrow$ linear buffer done server side
- Clients authenticate to the server
- Special server code to handle rendering on a different card

Wayland

- Server doesn't need to do anything
- Rely on render-nodes
- Client knows it uses a different card than the server and handles this case differently.
- Copy tiled buffer $\rightarrow$ linear buffer done client side (or with an embed compositor)

What has been done

- Render nodes
- DRI_PRIME inside Mesa (rendering in a linear buffer if needed)
- We can choose the device to use with ID_PATH_TAG
- Shutdown the dedicated GPU when unneeded

What needs to be done

- Dma-buf fences
- Mesa: rendering to a tiled buffer, and doing a copy to a linear buffer
- Use driconf to remember which device we want to use for an application
- Remaining applications using Gem Names must be ported to use Prime (ex: vaapi)
- Handle displays connected to multiple GPUs

XWayland: wlglamor

wlglamor: XWayland DDX using Glamor to support Xrender and DRI2/DRI3.

XWayland: Xserver linked to a Wayland compositor.
Glamor: don't care of the GPU. OpenGL based.

$\rightarrow$ No need to support X GPU offloading.

Problem: DRI2 doesn't work with render-nodes.

Hopefully DRI3 can work with render-nodes. And DRI3 GPU offloading support could be similar.

DRI3 still not entirely ready. Fixes coming.

Thanks!