

Finally fixing Multi-GPU on Linux

Who are we?

- Victoria Brekenfeld
 - I work at System76
 - on COSMIC DE
 - mostly on the Wayland Compositor `cosmic-comp`
 - Also member of wayland-protocols
 - active in various FreeDesktop spaces
- Sebastian Wick
 - I work at Red Hat
 - on mutter (GNOME Shell compositor)
 - on flatpak
 - also active in various FreeDesktop spaces

The issue

- Multi-GPU has a bad reputation on Linux
 - ... of being slow
 - ... of being an unnecessary power draw
 - ... known as being difficult to get working
 - ... people expect to have to manually select the GPU, to possibly log out/in or even reboot
 - ... forums are full of random env-variables without a lot of explanation
 - ... various “prime”-utilities and magic-solutions exist, few work consistently or provide any actual benefit
- We argue none of this has to be true

Outline

- Problems
- Prior attempts
- Ongoing efforts
- Our solution

Problems

- Client-side GPU selection
- Containers add to the selection and configuration issue
- Client-side offloading
- Unnecessary buffer copies and migrations on the server-side
- GPUs are kept active / don't go to sleep
- Lack of transparency
- Lack of configurability

Prior Art

- switcheroo-control
 - Incorporated by most bigger desktop environments
 - Rarely used in smaller standalone launchers
 - Doesn't really "fix" GPU selection, but gives launchers a way to provide some selection
 - Outsources transparency and configuration
 - Typically goes hand-in-hand with `X-PreferNonDefaultGPU`
- https://gitlab.freedesktop.org/glvnd/libglvnd/-/merge_requests/228
 - Kyle Brennemann has worked for quite some time on a configuration format for gpu offloading
 - Hooks into libglvnd (for both glx and egl) and vulkan-loader
 - Establishes a common static configuration format to provide profiles, device overrides/filters, etc

Ongoing efforts

- [linux-dmabuf v6](#)
 - Allows compositors to advertise multiple GPUs it can sample from
 - Fixes client-side offloading, by making it possible to move the responsibility to the compositor
 - Allows compositors to avoid unnecessary copies, but doing so requires more complex buffer management and multiple graphics contexts.

Problems

- Client-side GPU selection
- Containers add to the selection and configuration issue
- Client-side offloading
- Unnecessary buffer copies and migrations on the server-side
- GPUs are kept active / don't go to sleep
- Lack of transparency
- Lack of configurability

Our solution:
gpu-daemon

(todo: name bikeshedding)

gpu-daemon

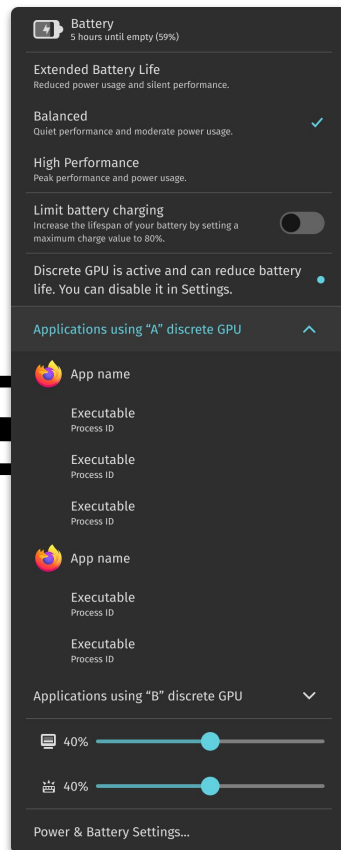
- Small varlink service
- Can be bind mounted into containers (could also be nested)
- Provides dynamic, virtualized view and access to the GPUs in the system
- Provides GPU preferences
- Provides configuration (settings, quirks)
- Allow integrators to dynamically configure based on
 - available GPUs, app id, system state, user actions, ...
- API to query and thus visualize the state of the system

gpu-daemon: Endpoints

- Graphics API Loaders
 - Query Preferred GPU and potentially “force” a particular gpu via filtering
 - hotplug... if they start supporting it
- Mesa / proprietary drivers
 - Use the service to obtain file descriptors to the necessary devices for sandboxes
- Integrators (e.g Desktop Environments)
 - Provide runtime configuration (change default, power off GPU)
 - Provide app specific configuration (this app should always start on this GPU)
 - May use provided APIs to visualize the current state

gpu-daemon: Why varlink?

- just a bit of JSON over a unix socket
- can send updates
- can be bind mounted into containers
- the service can look up properties about the connected process to use in the dynamic configuration
- no broker, but direct connection



Power & Battery Settings...

Xwayland

- <https://gitlab.freedesktop.org/xorg/xserver/-/issues/1380>

xwayland: Create a xwl_gbm context for each available GPU

 Offen  Ticket erstellt 13.09.2022 von Austin Shafer

Xwayland currently only creates one `xwl_gbm` instance corresponding to the primary GPU. This is then used in `glamor_pixmap_from_fds` to import buffers as part of creating a pixmap for a buffer. This is an issue because in multi-GPU PRIME environments we are starting to consider scenarios where we want to directly scan out clients running on the dGPU. Trying to import dGPU-specific buffers with scanout modifiers fails since the `xwl_gbm` object is not created on the dGPU.

What most likely needs to happen is we need to create a `xwl_gbm` (and associated context) for each GPU in the system. We could then probably use something like the new `DRMSetDeviceInUse` request from DRI3 to look up the correct gbm context to import a buffer into while creating a pixmap.

This issue exists to document this problem and start a discussion on how to improve it. This is related to [!818 \(merged\)](#)

Problems

- Client-side GPU selection
- Client-side offloading
- Unnecessary buffer copies and migrations on the server-side
- GPUs are kept active / don't go to sleep
- Lack of transparency
- Lack of configurability

Help is needed

- We need reviews on linux-dmabuf v6
 - (And compositors to do the right thing.. ™)
- We need people wanting to work on gpu-daemon
- We need feedback from mesa devs and people working on vulkan-loader / glvnd
- We need sandbox integrations (and wanna hear about their requirements!)
- We need desktop integrations!

Thank you for listening!

Q/A