Translation(s): English - español - Polski

This page describes how to identify, configure and troubleshoot NVIDIA Optimus enabled systems for Debian. NVIDIA Optimus is a technology that enables dynamic, switchable graphics between the central processing unit's (CPU) embedded graphics capability and the discrete graphics processing unit (GPU) card. Due to the nature of this technology, various software components must be aware of, and configured for, the proper output of the display based on the user's desired configuration. The situation is further complicated by the fact that some laptop models have all external display connectors hardwired to the onboard GPU the same way the internal screen is, some have all external display connectors wired to the discrete GPU, and some have a mix. In order to take advantage of Optimus and also have full use of external displays, different configuration is required depending on the hardware variant. There appears to be no way to query the system for this information, which leaves trial and error: try the configurations for each hardware variant until one works.

Tabla de Contenidos

Identification

The quickest method to determine if your device uses an Optimus card is to search against the documented list on NVIDIA's website. Obtain the NVIDIA GPU identifier of your card with:

```
$ lspci | grep -E "VGA|3D"
```

Compare the identifier (`i.e. GeForce 7XXM, 8XXM, 9XXM`) with the list, here: http://www.geforce.com/hardware/technology/optimus/supported-gpus

Also note the PCI identifier of the card, which is the 5-digit ID at the beginning of the previous command's output, `XX:XX.X`.

Methods

Configuration of an NVIDIA Optimus enabled system can be somewhat complex, depending upon the desired end state. This section mentions and details a few of the more common configuration scenarios and how to adjust your Debian installation, accordingly.

To summarize the different approaches you can take that are supported in Debian:

- Using PRIME Render Offload - Theoretically, with Debian 11 (or newer), special configuration shouldn't be needed and offloading should be available as soon as you've installed the proprietary drivers, at least as far as the internal screen and any display outputs wired to the onboard GPU are concerned. Making this work with

display outputs that are wired to the discrete GPU requires some manual configuration. See the relevant section for more details.

- Using only the integrated GPU - Uses little battery power but is also low performance, as the NVIDIA GPU will be completely powered off. Still, this is suitable for some use-cases.
- Using only the NVIDIA GPU - High performance but is also a massive drain on battery life. There's almost always a better solution than this, but it's useful if you expect to use your laptop as a desktop (i.e. always plugged in).
- Using Bumblebee/Primus - An alternative solution to PRIME Render Offload that works similarly, with support for both Nouveau and the proprietary drivers. Up to Debian 11, this has had severe performance issues for some. While the situation is much better in Debian 11, this should really only be used if the official PRIME Render Offload functionality in the proprietary drivers does not work for you.
- Using nvidia-xrun - Runs a completely separate desktop session with the NVIDIA card, achieving maximum performance from the GPU at the cost of extra complexity.
- Using Nouveau - Nouveau is the open-source driver for NVIDIA GPUs. It also supports PRIME Render Offload via Mesa without the need to use the proprietary drivers. Unfortunately, it does not support Vulkan and has severe performance issues, particularly with modern cards.

Using NVIDIA PRIME Render Offload

This method makes the most effective use of Optimus-capable hardware and is the recommended approach for any user who is willing to rely on the proprietary NVIDIA driver.

As of X.Org Server 1.20.6 (with more patches enabling automatic configuration in version 1.20.8), official PRIME Render Offload functionality from NVIDIA should be available and working out-of-the-box as soon as you install the proprietary drivers. Debian 11 and later versions support everything required for this. This method supports both OpenGL and Vulkan. Bumblebee/Primus must be uninstalled before this can be used. Also make sure that the outdated `xserver-xorg-video-intel` package is not installed. The "modesetting" xorg driver has superseded it and will be used in this configuration.

The only requirements are to install the proprietary drivers (As per the NvidiaGraphicsDrivers page) and then run your application with the `__NV_PRIME_RENDER_OFFLOAD=1` environment variable set, and in some cases (e.g. for GLX applications), the `__GLX_VENDOR_LIBRARY_NAME=nvidia` environment variable set (or the `__GLX_VENDOR_LIBRARY_NAME=nvidia-current` environment variable, if it's installed an older driver).

When running an application from the terminal, an example of this would look like:

```
__NV_PRIME_RENDER_OFFLOAD=1 __GLX_VENDOR_LIBRARY_NAME=nvidia supertuxkart
```

This method is also supported out-of-the-box in GNOME 3.36+ with a "Start on secondary GPU" context menu item for applications.

Steam games can be launched on your NVIDIA GPU by right-clicking their entry to open the context menu, opening the "Properties" panel, clicking the "Set Launch Options" button in the window that appears, and then setting the contents of the resulting text field to be:

```
__NV_PRIME_RENDER_OFFLOAD=1 __GLX_VENDOR_LIBRARY_NAME=nvidia %command%
```

The above configuration uses the Nvidia discrete GPU as an "output source provider", commonly referred to simply as "Prime", and should be all that's needed in the case where all display connectors are wired to the onboard GPU (or if external display connectors are not needed). If your display connectors are wired to the discrete GPU, they will not work under this configuration. In that case, the Nvidia discrete GPU must be configured as an "output sink provider", commonly referred to as "Reverse Prime". The "Reverse Prime" setup requires some manual configuration, which is described in the Nvidia driver documentation linked below.

More information, including troubleshooting tips, can be found in the official NVIDIA documentation for this feature: https://us.download.nvidia.com/XFree86/Linux-x86_64/450.57/README/primerenderoffload.html

If the NVIDIA driver won't load with something like the following in `/var/log/boot.log`:

```
[FAILED] Failed to start Load Kernel Modules.
See 'systemctl status systemd-modules-load.service' for details.
```

consider disabling Secure Boot in the UEFI.

PRIME Render Offload with an External Display

Some systems require the use of the NVIDIA driver as the output source provider when the EFI graphics configuration is set to 'Hybrid' mode. This can easily be configured with:

```
sudo nvidia-xconfig --prime
```

If after detaching the external display Xorg fails to start, simply rename the Xorg.conf that was generated and restart the display manager:

```
sudo mv /etc/X11/Xorg.conf /etc/X11/Xorg.conf-external-display
sudo systemctl restart <display manager>
```

Using nvidia-xrun

See nvidia-xrun.

Using Bumblebee

The proprietary NVIDIA graphics driver can also be used to enable dynamic graphics switching between the embedded and discrete graphics providers through the use of Bumblebee. This method takes advantage of Optimus' power saving features, but can be more complex to successfully enable offloaded 3D applications.

See Bumblebee.

Using Nouveau

See the Nouveau wiki page on Optimus setups: https://nouveau.freedesktop.org/wiki/Optimus/

Using only one GPU

Rather than enabling the power saving features of dynamic, "switchable" graphics, one can simply configure the system to output to the local display using only a single graphics provider.

The simplest method of setting the graphics provider, if your hardware supports it, is to manually select the embedded CPU or NVIDIA GPU as the display provider in the system's BIOS. Of course, this is vendor specific and you must consult your hardware manufacturer's documentation for further detail.

Using NVIDIA GPU as the primary GPU

You can use PRIME to render an X screen using an NVIDIA GPU while display it on monitors connected to an Intel integrated GPU. While this configuration does not take advantage of Optimus' power saving features, it provides maximum performance.

1. Install the NVIDIA driver and `xrandr`:

```
# apt install x11-xserver-utils
```

2. Place the following in /etc/X11/xorg.conf:

```
Section "ServerLayout"
    Identifier "layout"
    Screen 0 "nvidia"
    Inactive "intel"
EndSection

Section "Device"
    Identifier "nvidia"
    Driver "nvidia"
    BusID "<BusID for NVIDIA device here>" # e.g. PCI:1:0:0
EndSection

Section "Screen"
    Identifier "nvidia"
    Device "nvidia"
    Option "AllowEmptyInitialConfiguration"
EndSection

Section "Device"
    Identifier "intel"
    Driver "modesetting"
    BusID "<BusID for Intel device here>" # e.g. PCI:0:2:0
    #Option "AccelMethod" "none"
EndSection

Section "Screen"
    Identifier "intel"
    Device "intel"
EndSection
```

> You can find the BusID for your graphic devices by running the lspci command. For example, if the output of the command was "01:00.0", you would set BusID to "PCI:1:0:0".

3. Place the following commands in ~/.xsessionrc:

```
xrandr --setprovideroutputsource modesetting NVIDIA-0
xrandr --auto
xrandr --dpi 96
```

and make the script executable:

```
$ chmod +x ~/.xsessionrc
```

> The DPI setting, (--dpi) should be fine for most screens; however, this may need to be adjusted for newer high density pixel screens. Systems with HiDPI screens likely will want to set this to 192. Consult your hardware manufacturer's specification for the appropriate setting.

Display managers

If you are using a display manager then you will also need to configure it to run above xrandr commands during display setup.

LightDM

1. Create a display setup script /etc/lightdm/display_setup.sh:

```
xrandr --setprovideroutputsource modesetting NVIDIA-0
xrandr --auto
xrandr --dpi 96
```

and make it executable:

```
# chmod +x /etc/lightdm/display_setup.sh
```

2. Configure lightdm to run the script by editing (or adding) the `SeatDefaults` section in /etc/lightdm/lightdm.conf:

```
[SeatDefaults]
display-setup-script=/etc/lightdm/display_setup.sh
```

3. Restart lightdm:

```
# systemctl restart lightdm.service
```

Simple Desktop Display Manager (SDDM)

1. Place the following commands to /usr/share/sddm/scripts/Xsetup:

```
# Xsetup - run as root before the login dialog appears
xrandr --setprovideroutputsource modesetting NVIDIA-0
xrandr --auto
xrandr --dpi 96
```

2. Restart SDDM:

```
# systemctl restart sddm
```

GNOME Display Manager (GDM)

1. Create /usr/share/gdm/greeter/autostart/optimus.desktop file with the following content:

```
[Desktop Entry]
Type=Application
Name=Optimus
Exec=sh -c "xrandr --setprovideroutputsource modesetting NVIDIA-0; xrandr --auto"
NoDisplay=true
X-GNOME-Autostart-Phase=DisplayServer
```

2. Reboot.

PRIME synchronization

You can enable PRIME synchronization to prevent screen tearing. It requires:

- Linux kernel 4.5 or higher;
- X server 1.19 or higher;
- NVIDIA driver 370.23 or higher.

PRIME synchronization will be enabled automatically after you enable KMS for `nvidia-drm` module.

1. Add this line to the end of /etc/modprobe.d/nvidia.conf:

```
options nvidia-drm modeset=1
```

2. Regenerate your initramfs image by running:

```
# update-initramfs -u
```

3. The `modesetting` driver for an Intel GPU loads a module called `glamor`, which conflicts with the NVIDIA GLX implementation. To disable `glamor` uncomment this line in /etc/X11/xorg.conf:

```
Option "AccelMethod" "none"
```

4. Reboot.

Switching back to Nouveau

You can easily switch back to using the open source Nouveau driver, while keeping the proprietary NVIDIA driver installed.

1. Disable `/etc/X11/xorg.conf` by renaming it, e.g., to `/etc/X11/xorg.conf.disabled`.

2. Run the following command:

```
# update-alternatives --config glx
```

You will likely see that there are two alternative GLX providers available in your system: the free MESA implementation and the proprietary NVIDIA one which is currently being used. Switch to MESA by selecting `/usr/lib/mesa-diverted`.

3. Regenerate your initramfs image by running:

```
# update-initramfs -u
```

4. Reboot.

Checking drivers

You can check if installed drivers support 3D OpenGL graphics by executing following command:

```
$ glxinfo | grep OpenGL
```

Hybrid GPUs

If you have a hybrid GPU and installed bumblebee driver you can check it for Intel:

```
$ glxinfo | grep OpenGL
```

And for NVIDIA:

```
$ optirun glxinfo | grep OpenGL
```

Also very recommended is checking displaying 3D OpenGL graphics by running *glxgears* program.

See also

- NVIDIA Optimus - ArchWiki

CategoryHardware | GraphicsCard | NvidiaGraphicsDrivers