Chapter 28. Configuring SLI and Multi-GPU FrameRendering

The NVIDIA Linux driver contains support for NVIDIA SLI FrameRendering and NVIDIA Multi-GPU FrameRendering. Both of these technologies allow an OpenGL application to take advantage of multiple GPUs to improve visual performance.

The distinction between SLI and Multi-GPU is straightforward. SLI is used to leverage the processing power of GPUs across two or more graphics cards, while Multi-GPU is used to leverage the processing power of two GPUs colocated on the same graphics card. If you want to link together separate graphics cards, you should use the "SLI" X config option. Likewise, if you want to link together GPUs on the same graphics card, you should use the "MultiGPU" X config option. If you have two cards, each with two GPUs, and you wish to link them all together, you should use the "SLI" option.

Rendering Modes

In Linux, with two GPUs SLI and Multi-GPU can both operate in one of three modes: Alternate Frame Rendering (AFR), Split Frame Rendering (SFR), and Antialiasing (AA). When AFR mode is active, one GPU draws the next frame while the other one works on the frame after that. In SFR mode, each frame is split horizontally into two pieces, with one GPU rendering each piece. The split line is adjusted to balance the load between the two GPUs. AA mode splits antialiasing work between the two GPUs. Both GPUs work on the same scene and the result is blended together to produce the final frame. This mode is useful for applications that spend most of their time processing with the CPU and cannot benefit from AFR.

With four GPUs, the same options are applicable. AFR mode cycles through all four GPUs, each GPU rendering a frame in turn. SFR mode splits the frame horizontally into four pieces. AA mode splits the work between the four GPUs, allowing antialiasing up to 64x. With four GPUs SLI can also operate in an additional mode, Alternate Frame Rendering of Antialiasing. (AFR of AA). With AFR of AA, pairs of GPUs render alternate frames, each GPU in a pair doing half of the antialiasing work. Note that these scenarios apply whether you have four separate cards or you have two cards, each with two GPUs.

With some GPU configurations, there is in addition a special SLI Mosaic Mode to extend a single X screen transparently across all of the available display outputs on each GPU. See below for the exact set of configurations which can be used with SLI Mosaic Mode.

Enabling Multi-GPU

Multi-GPU is enabled by setting the "MultiGPU" option in the X configuration file; see Appendix B, *X Config Options* for details about the "MultiGPU" option.

The nvidia-xconfig utility can be used to set the "MultiGPU" option, rather than modifying the X configuration file by hand. For example:

```
% nvidia-xconfig --multigpu=on
```

Enabling SLI

SLI is enabled by setting the "SLI" option in the X configuration file; see Appendix B, *X Config Options* for details about the SLI option.

The nvidia-xconfig utility can be used to set the SLI option, rather than modifying the X configuration file by hand. For example:

```
% nvidia-xconfig --sli=on
```

Enabling SLI Mosaic Mode

The simplest way to configure SLI Mosaic Mode using a grid of monitors is to use **nvidia-settings** (see Chapter 23, *Using the nvidia-settings Utility*). The steps to perform this configuration are as follows:

1. Connect each of the monitors you would like to use to any connector from any GPU used for SLI Mosaic Mode. If you are going to use fewer monitors than there are connectors, connect one monitor to each GPU before adding a second monitor to any GPUs.

2. Install the NVIDIA display driver set.

3. Configure an X screen to use the "nvidia" driver on at least one of the GPUs (see Chapter 6, *Configuring X for the NVIDIA Driver* for more information).

4. Start X.

5. Run **nvidia-settings**. You should see a tab in the left pane of nvidia-settings labeled "SLI Mosaic Mode Settings". Note that you may need to expand the entry for the X screen you configured earlier.

6. Check the "Use SLI Mosaic Mode" check box.

7. Select the monitor grid configuration you'd like to use from the "display configuration" dropdown.

8. Choose the resolution and refresh rate at which you would like to drive each individual monitor.

9. Set any overlap you would like between the displays.

10. Click the "Save to X Configuration File" button. NOTE: If you don't have permissions to write to your system's X configuration file, you will be prompted to choose a location to save the file. After doing so, you *must* copy the X configuration file into a location the X server will consider upon startup (usually `/etc/X11/xorg.conf`).

11. Exit nvidia-settings and restart your X server.

Alternatively, nvidia-xconfig can be used to configure SLI Mosaic Mode via a command like **nvidia-xconfig --sli=Mosaic --metamodes=METAMODES** where the METAMODES string specifies the desired grid configuration. For example:

```
nvidia-xconfig --sli=Mosaic --metamodes="GPU-0.DFP-0: 1920x1024+0+0, GPU-0.DFP-1: 1920x1024+1920+0, GPU-1.DFP-
```

will configure four DFPs in a 2x2 configuration, each running at 1920x1024, with the two DFPs on GPU-0 driving the top two monitors of the 2x2 configuration, and the two DFPs on GPU-1 driving the bottom two monitors of the 2x2 configuration.

See the MetaModes X configuration description in details in Chapter 12, *Configuring Multiple Display Devices on One X Screen*. See Appendix C, *Display Device Names* for further details on GPU and Display Device Names.

Hardware requirements

SLI functionality requires:

- Identical PCI Express graphics cards
- A supported motherboard (with the exception of Quadro Plex)
- In most cases, a video bridge connecting the two graphics cards
- SLI Mosaic Mode requires NVIDIA Quadro GPUs.

For the latest information on supported SLI and Multi-GPU configurations, including SLI- and Multi-GPU capable GPUs and SLI-capable motherboards, see http://www.geforce.com/hardware/technology/sli.

Other Notes and Requirements

The following other requirements apply to SLI and Multi-GPU:

- Mobile GPUs are NOT supported
- GPUs with ECC enabled may not be used in an SLI configuration
- SLI on Quadro-based graphics cards always requires a video bridge
- TwinView is also not supported with SLI or Multi-GPU. Only one display can be used when SLI or Multi-GPU is enabled, with the exception of Mosaic.
- If X is configured to use multiple screens and screen 0 has SLI or Multi-GPU enabled, the other screens configured to use the nvidia driver will be disabled. Note that if SLI or Multi-GPU is enabled, the GPUs used by that configuration will be unavailable for single GPU rendering.

28.1. Frequently Asked SLI and Multi-GPU Questions

Why is glxgears slower when SLI or Multi-GPU is enabled?

When SLI or Multi-GPU is enabled, the NVIDIA driver must coordinate the operations of all GPUs when each new frame is swapped (made visible). For most applications, this GPU synchronization overhead is negligible. However, because glxgears renders so many frames per second, the GPU synchronization overhead consumes a significant portion of the total time, and the framerate is reduced.

Why is Doom 3 slower when SLI or Multi-GPU is enabled?

The NVIDIA Accelerated Linux Graphics Driver does not automatically detect the optimal SLI or Multi-GPU settings for games such as Doom 3 and Quake 4. To work around this issue, the environment variable __GL_DOOM3 can be set to tell OpenGL that Doom 3's optimal settings should be used. In Bash, this can be done in the same command that launches Doom 3 so the environment variable does not remain set for other OpenGL applications started in the same session:

```
% __GL_DOOM3=1 doom3
```

Doom 3's startup script can also be modified to set this environment variable:

```
#!/bin/sh
# Needed to make symlinks/shortcuts work.
# the binaries must run with correct working directory
cd "/usr/local/games/doom3/"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
export __GL_DOOM3=1
```

```
exec ./doom.x86 "$@"
```

This environment variable is temporary and will be removed in the future.

Why does SLI or MultiGPU fail to initialize?

There are several reasons why SLI or MultiGPU may fail to initialize. Most of these should be clear from the warning message in the X log file; e.g.:

- `Unsupported bus type`
- `The video link was not detected`
- `GPUs do not match`
- `Unsupported GPU video BIOS`
- `Insufficient PCIe link width`

The warning message `'Unsupported PCI topology'` is likely due to problems with your Linux kernel. The NVIDIA driver must have access to the PCI Bridge (often called the Root Bridge) that each NVIDIA GPU is connected to in order to configure SLI or MultiGPU correctly. There are many kernels that do not properly recognize this bridge and, as a result, do not allow the NVIDIA driver to access this bridge. See the below "How can I determine if my kernel correctly detects my PCI Bridge?" FAQ for details.

Below are some specific troubleshooting steps to help deal with SLI and MultiGPU initialization failures.

- Make sure that ACPI is enabled in your kernel. NVIDIA's experience has been that ACPI is needed for the kernel to correctly recognize the Root Bridge. Note that in some cases, the kernel's version of ACPI may still have problems and require an update to a newer kernel.
- Run **lspci** to check that multiple NVIDIA GPUs can be identified by the operating system; e.g:

  ```
  % /sbin/lspci | grep -i nvidia
  ```

  If **lspci** does not report all the GPUs that are in your system, then this is a problem with your Linux kernel, and it is recommended that you use a different kernel.

  Please note: the **lspci** utility may be installed in a location other than `/sbin` on your system. If the above command fails with the error: `'/sbin/lspci: No such file or directory'`, please try:

  ```
  % lspci | grep -i nvidia
  ```

  , instead. You may also need to install your distribution's `pciutils` package.

- Make sure you have the most recent SBIOS available for your motherboard.
- The PCI Express slots on the motherboard must provide a minimum link width. Please make sure that the PCI Express slot(s) on your motherboard meet the following requirements and that you have connected the graphics board to the correct PCI Express slot(s):
  - A dual-GPU board needs a minimum of 8 lanes (i.e. x8 or x16)
  - A pair of single-GPU boards requires one of the following supported link width combinations:
    - x16 + x16
    - x16 + x8
    - x16 + x4
    - x8 + x8

How can I determine if my kernel correctly detects my PCI Bridge?

As discussed above, the NVIDIA driver must have access to the PCI Bridge that each NVIDIA GPU is connected to in order to configure SLI or MultiGPU correctly. The following steps will identify whether the kernel correctly recognizes the PCI Bridge:

- Identify both NVIDIA GPUs:

  ```
  % /sbin/lspci | grep -i vga

  0a:00.0 VGA compatible controller: nVidia Corporation [...]
  81:00.0 VGA compatible controller: nVidia Corporation [...]
  ```

- Verify that each GPU is connected to a bus connected to the Root Bridge (note that the GPUs in the above example are on buses 0a and 81):

  ```
  % /sbin/lspci -t
  ```

good:

```
-+-[0000:80]-+-00.0
 |           +-01.0
 |           \-0e.0-[0000:81]----00.0
...
 \-[0000:00]-+-00.0
             +-01.0
             +-01.1
             +-0e.0-[0000:0a]----00.0
```

bad:

```
-+-[0000:81]---00.0
...
 \-[0000:00]-+-00.0
             +-01.0
             +-01.1
             +-0e.0-[0000:0a]----00.0
```

Note that in the first example, bus 81 is connected to Root Bridge 80, but that in the second example there is no Root Bridge 80 and bus 81 is incorrectly connected at the base of the device tree. In the bad case, the only solution is to upgrade your kernel to one that properly detects your PCI bus layout.