DRIVER PERSISTENCE

vR331 | March 2014

Driver Persistence

TABLE OF CONTENTS

Chapter 1. OVERVIEW

The NVIDIA kernel mode driver must be running and connected to a target GPU device before any user interactions with that device can take place. The driver behavior differs depending on the OS. Generally, if the kernel mode driver is not already running or connected to a target GPU, the invocation of any program that attempts to interact with that GPU will transparently cause the driver to load and/or initialize the GPU. When all GPU clients terminate the driver will then deinitialize the GPU. Driver load behavior is important for end users in two ways:

- Application start latency

  Applications that trigger GPU initilization may incur a short (order of 1-3 second) startup cost per GPU due to ECC scrubbing behavior. If the GPU is already initialized this scrubbing does not take place.
- Preservation of driver state

  If the driver deinitializes a GPU some non-persistent state associated with that GPU will be lost and revert back to defaults the next time the GPU is initialized. See Data Persistence. To avoid this the GPU should be kept initialized.

Default driver behavior differs between operating systems:

1.1. Windows

On Windows the kernel mode driver is loaded at Windows startup and kept loaded until Windows shutdown. Consequently Windows users can mostly ignore the driver persistence implications described in this document.

Driver reload events, e.g. due to TDR or new driver installation, will result in reset of non-persistent state.

1.2. Linux

Under Linux systems where X runs by default on the target GPU the kernel mode driver will generally be initalized and kept alive from machine startup to shutdown, courtesy of the X process. On headless systems or situations where no long-lived X-like client maintains a handle to the target GPU, the kernel mode driver will initilize and deinitialize the target GPU each time a target GPU application starts and stops. In HPC environments this situation is quite common. Since it is often desireable to keep the GPU initialized in these cases, NVIDIA provides two options for changing driver behavior: Persistence Mode (Legacy) and the Persistence Daemon.

Chapter 2. DATA PERSISTENCE

Different classes of driver state have different lifetime durations. It can be important to understand the differences, as this can affect the behavior of GPU management features like clock settings, ECC mode, etc. Generally, driver state falls into the following categories. This is not intended to be an exhaustive list, but will cover common cases:

2.1. GPU Initialization Lifecycle

State of this type lasts from the time the driver initializes a GPU until the time the GPU is unititialized. This is the narrowest lifecycle, as the kernel driver itself is still loaded and may be managing other GPUs. The GPU typically initializes a GPU if a client application tries to access the GPU. The GPU is typically deinitialized after the last client exits.

State:

- Compute Mode, Accounting Mode, Persistence Mode
- Application Clocks, Application Clocks Permission Settings
- SW-Based Power Capping Limit
- Volatile ECC errors, Pending Retired Pages

2.2. Kernel Driver Lifecycle

State of this type lasts from the time the driver loads until the time the driver unloads (e.g. rmmod). In most environments this is the entire machine boot cycle. Exceptions include GPU reset events and driver installs.

State:

- Accounting process data

2.3. GPU Board Lifecycle

State of this type lasts across boot cycles, as it is stored in the board's persistent inforom. In some cases such state can be explicitly cleared, but in general this state is deemed to be persistent for the entire life of the board -- or until next changed by the user.

State:

- ECC Mode, Aggregate ECC errors, Retired Pages
- GPU Operation Mode, Driver Model

Chapter 3. BACKGROUND

The NVIDIA GPU driver has historically followed Unix design philosophies by only initializing software and hardware state when the user has configured the system to do so. Traditionally, this configuration was done via the X Server and the GPUs were only initialized when the X Server (on behalf of the user) requested that they be enabled. This is very important for the ability to reconfigure the GPUs without a reboot (for example, changing SLI mode or bus settings, especially in the AGP days).

More recently, this has proven to be a problem within compute-only environments, where X is not used and the GPUs are accessed via transient instantiations of the Cuda library. This results in the GPU state being initialized and deinitialized more often than the user truly wants and leads to long load times for each Cuda job, on the order of seconds.

NVIDIA previously provided Persistence Mode to solve this issue. This is a kernel-level solution that can be configured using nvidia-smi. This approach would prevent the kernel module from fully unloading software and hardware state when no user software was using the GPU. However, this approach creates subtle interaction problems with the rest of the system that have made maintenance difficult.

The purpose of the NVIDIA Persistence Daemon is to replace this kernel-level solution with a more robust user-space solution. This enables compute-only environments to more closely resemble the historically typical graphics environments that the NVIDIA GPU driver was designed around.

Chapter 4.
PERSISTENCE MODE (LEGACY)

Persistence Mode is the term for a user-settable driver property that keeps a target GPU initialized even when no clients are connected to it. This solution is near end-of-life and will be eventually deprecated in favor the Persistence Daemon

Persistence mode can be set using nvidia-smi or programmaticaly via the NVML API.

To enable persistence mode using nvidia-smi (as root):

```
nvidia-smi -i <target gpu> -pm ENABLED
    Enabled persistence mode for GPU <target gpu>.
    All done.
```

To view current persistence mode using nvidia-smi:

```
nvidia-smi -i <target gpu> - q
    ==============NVSMI LOG==============

    Timestamp                           : ----
    Driver Version                      : ----

    Attached GPUs                       : ----
    GPU 0000:01:00.0
        Product Name                    : ----
        Display Mode                    : ----
        Display Active                  : ----
        Persistence Mode                : Enabled
        Accounting Mode                 : ----
        ...
```

4.1. Supported Environments

- Drivers: All shipping driver versions
- OSes: All standard driver-supported Linux platforms
- GPUs: All shipping Tesla, Quadro and GRID products

Chapter 5. PERSISTENCE DAEMON

NVIDIA is providing a user-space daemon on Linux to support persistence of driver state across Cuda job runs. The daemon approach provides a more elegant and robust solution to this problem than persistence mode.

NVIDIA will support both solutions for the near future (likely through Cuda 6.0), but will focus all future development and bug fixes on the daemon.

The daemon is installed in /usr/bin, while sample installation and init scripts are included with the driver in the documentation directory. The scripts are provided as a guide for installing the daemon to run on system startup for some common init systems; they may require some changes for certain distributions, due to the wide variety of init system configurations.

NVIDIA encourages customers to shift to this daemon approach at their earliest availability.

5.1. Supported Environments

- Drivers: R319 and higher
- OSes: All standard driver-supported Linux platforms
- GPUs: All shipping Tesla, Quadro and GRID products

5.2. Implementation Details

On Linux systems running the NVIDIA GPU driver, clients attach a GPU by opening its device file. Conversely, the GPU is detached by closing the device file. The GPU state remains loaded in the driver whenever one or more clients have the device file open. Once all clients have closed the device file, the GPU state will be unloaded unless persistence mode is enabled.

To simulate graphics environments without incurring the overhead of user-space graphics drivers, we have implemented the NVIDIA Persistence Daemon, which essentially runs in the background and sleeps with the device files open. The daemon uses libnvidia-cfg to open and close the correct device files based on its PCI bus address,

and provides an RPC interface to control the persistence mode of each GPU individually. Thus, while the daemon holds the device files open, at least one client, the daemon, has the GPU attached and the driver will not unload the GPU state. Once the daemon starts running, it remains in the background until it is killed, even if persistence mode is disabled for all devices.

Because of the nature of the solution, the daemon can be used as a drop-in replacement for what we are now calling "legacy persistence mode" as implemented in the NVIDIA kernel-mode driver. NVIDIA SMI has been updated in driver version 319 to use the daemon's RPC interface to set the persistence mode using the daemon if the daemon is running, and will fall back to setting the legacy persistence mode in the kernel-mode driver if the daemon is not running. This is all handled transparently by NVIDIA SMI, so there should be no change in how persistence mode is configured. Eventually, the legacy persistence mode will be deprecated and removed in favor of the NVIDIA Persistence Daemon, once it has achieved wide adoption in the relevant use cases.

5.3. Permissions and Security

The NVIDIA Persistence Daemon provides a more robust implementation of persistence mode on Linux, since it simply mimics an external client of the GPU but does not actually use the GPU for any work. In this way, it causes the NVIDIA GPU driver to operate within the assumptions of its original design.

Once the daemon is running, there is minimal overhead for keeping persistence mode enabled. The daemon will simply sleep waiting for a command.

The daemon does not require super-user privileges to run - however, it does require super-user privileges to set up some runtime data in /var/run. The daemon allows for two mechanisms to run as a user without super-user privileges:

- An administrator (or script run with super-user privileges) may create the /var/run/nvidia-persistenced directory and `chown` it to the user the daemon will run as. The daemon can then be run as the intended user using `su` or similar. In this case, the /var/run/nvidia-persistenced directory will not be removed when the daemon is killed.
- The daemon may be started with super-user privileges and use the `--user` option. This will force the daemon to drop its super-user privileges as soon as possible after creating the /var/run/nvidia-persistenced directory and run as the specified user. Note that with this mechanism, the daemon may not be able to remove the /var/run/nvidia-persistenced directory when it is killed, since the user may not have write permissions to /var/run.

Note that in both cases, the daemon may not be able to remove its runtime data directory when it is killed, so this task should typically be handled by the init script or service for the daemon.

The daemon may also be run with perpetual super-user privileges by simply omitting the `--user` option, but this is not recommended and is not necessary for functionality.

The daemon also provides a `--verbose` option, which increases its logging output to syslog for debugging purposes.

The source code for the daemon is also available under the MIT license, to allow for second- and third-party security auditing.

5.4. Usage

To run the NVIDIA Persistence Daemon, simply run (as root):

```
# nvidia-persistenced --user foo
```

After doing a minimal amount of setup tasks that require super-user privileges, the daemon will drop super-user privileges and run as user 'foo'.

You may also use the `--persistence-mode` flag to indicate that the daemon should enable persistence mode for all devices by default:

```
# nvidia-persistenced --user foo --persistence-mode
```

You may then use NVIDIA SMI to change the persistence mode setting. For example, to disable persistence mode on all GPUs, simply run (again, as root):

```
# nvidia-smi -pm 0
```

Please see the nvidia-persistenced(1) manpage, which is installed by the NVIDIA GPU driver installer, or the output of:

```
% nvidia-persistenced --help
```

for detailed usage information.

Please see Installation Caveats for details about installing the daemon to always run on system startup.

5.5. Installation Caveats

The reason why we cannot immediately deprecate the legacy persistence mode and switch transparently to the NVIDIA Persistence Daemon is because at this time, we cannot guarantee that the NVIDIA Persistence Daemon will be running. This would be a feature regression as persistence mode might not be available out-of- the-box.

The NVIDIA Persistence Daemon ships with the NVIDIA Linux GPU driver starting in driver version 319 and is installed by the installer as /usr/bin/nvidia-persistenced. Ideally, the daemon would start on system initialization according to the Linux distribution's init system, transparently to the user, and exit on system shutdown. Unfortunately, there is no single standard for installing an application to start on system initialization on Linux, so we cannot reliably do so on the wide range of systems the NVIDIA GPU driver supports.

Therefore, we want to encourage individual distributions, who typically re-package the NVIDIA GPU driver for installation via their package manager, to install the NVIDIA Persistence Daemon to start on system initialization, which is a nearly trivial task once the init system is known. To this end, we are providing sample "init scripts" in the driver package to aid in this installation. These scripts attempt to cover three of the

most prevalent init systems found in Linux distributions today: SystemV, systemd, and Upstart. The sample scripts also come with an installer script that attempts to detect the init system and install the appropriate script for the user. The sample scripts and installer script are installed to /usr/share/doc/NVIDIA_GLX-1.0/sample/nvidia-persistenced-init.tar.bz2 by the NVIDIA GPU driver installer. They are not unpacked or run by the driver installer since we cannot guarantee that they will work correctly on all supported systems out-of-the-box.

By default, the installer scripts attempt to create a new system user for the daemon to run as, and the sample init scripts demonstrate the second option described in Permissions and Security for running the daemon without super-user privileges.

5.6. Customer Visibility

The daemon is visible to end customers, as it will typically require some sort of manual installation into the init system. However, after initial installation steps are taken, the daemon should operate transparently in the background, with NVIDIA SMI handling the necessary switching to determine if the daemon persistence mode can be used. Ideally, the eventual deprecation and removal of the legacy persistence mode will be transparent to customers using the daemon.