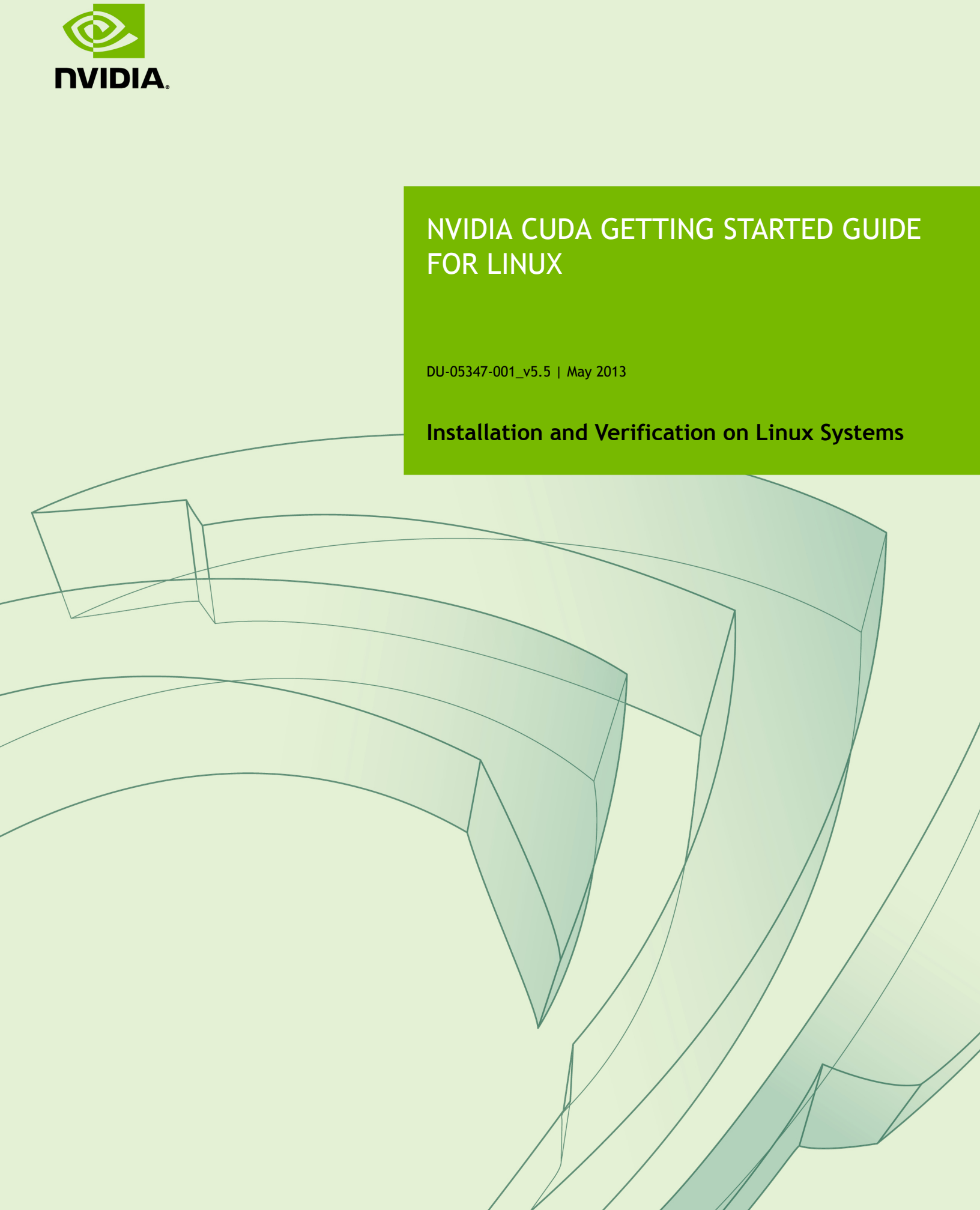

NVIDIA CUDA GETTING STARTED GUIDE FOR LINUX

DU-05347-001_v5.5 | May 2013

Installation and Verification on Linux Systems

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1. INTRODUCTION

CUDA™ is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

CUDA was developed with several design goals in mind:

- Provide a small set of extensions to standard programming languages, like C, that enable a straightforward implementation of parallel algorithms. With CUDA C/C++, programmers can focus on the task of parallelization of the algorithms rather than spending time on their implementation.
- Support heterogeneous computation where applications use both the CPU and GPU. Serial portions of applications are run on the CPU, and parallel portions are offloaded to the GPU. As such, CUDA can be incrementally applied to existing applications. The CPU and GPU are treated as separate devices that have their own memory spaces. This configuration also allows simultaneous computation on the CPU and GPU without contention for memory resources.

CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads. These cores have shared resources including a register file and a shared memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

This guide will show you how to install and check the correct operation of the CUDA development tools.

1.1. System Requirements

To use CUDA on your system, you will need the following installed:

- CUDA-capable GPU
- A supported version of Linux with a gcc compiler and toolchain
- NVIDIA CUDA Toolkit (available at no cost from http://www.nvidia.com/content/cuda/cuda-downloads.html)

1.2. About This Document

This document is intended for readers familiar with the Linux environment and the compilation of C programs from the command line. You do not need previous experience with CUDA or experience with parallel computation. Note: This guide covers installation only on systems running X Windows.

Many commands in this document might require *superuser* privileges. On most distributions of Linux, this will require you to log in as root. For systems that have enabled the sudo package, use the sudo prefix for all necessary commands. We will no longer remark on the matter of user privilege for the installation process except where critical to correct operation.

Chapter 2.
INSTALLING CUDA DEVELOPMENT TOOLS

The setup of CUDA development tools on a system running the appropriate version of Linux consists of a few simple steps:

- Verify the system has a CUDA-capable GPU.
- Verify the system has a supported version of Linux.
- Verify the system has gcc installed.
- Download the NVIDIA CUDA Toolkit.
- Install the NVIDIA CUDA Toolkit.
- Test that the installed software runs correctly and communicates with the hardware.

You can override the install-time prerequisite checks by running the installer with the -override flag. Remember that the prerequisites will still be required to use the NVIDIA CUDA Toolkit.

2.1. Verify You Have a CUDA-Capable GPU

To verify that your GPU is CUDA-capable, go to your distribution's equivalent of System Properties, or, from the command line, enter:

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

If you do not see any settings, update the PCI hardware database that Linux maintains by entering **`update-pciids`** (generally found in **`/sbin`**) at the command line and rerun the previous **`lspci`** command.

If your graphics card is from NVIDIA and it is listed in http://www.nvidia.com/object/cuda_gpus.html, your GPU is CUDA-capable.

The Release Notes for the CUDA Toolkit also contain a list of supported products.

2.2. Verify You Have a Supported Version of Linux

The CUDA Development Tools are only supported on some specific distributions of Linux. These are listed in the CUDA Toolkit release notes.

To determine which distribution and release number you're running, type the following at the command line:

```
uname -m && cat /etc/*release
```

You should see output similar to the following, modified for your particular system:

```
i386 Red Hat Enterprise Linux WS release 4 (Nahant Update 6)
```

The **i386** line indicates you are running on a 32-bit system. On 64-bit systems running in 64-bit mode, this line will generally read: **x86_64**. The second line gives the version number of the operating system.

2.3. Verify the System Has gcc Installed

The **gcc** compiler and toolchain generally are installed as part of the Linux installation, and in most cases the version of gcc installed with a supported version of Linux will work correctly.

To verify the version of gcc installed on your system, type the following on the command line:

```
gcc --version
```

If an error message displays, you need to install the *development tools* from your Linux distribution or obtain a version of **gcc** and its accompanying toolchain from the Web.

2.4. Download the NVIDIA CUDA Toolkit

Once you have verified that you have a supported NVIDIA GPU, a supported version of Linux, and gcc, download the NVIDIA CUDA Toolkit.

The NVIDIA CUDA Toolkit is available at no cost from the main CUDA download site at http://www.nvidia.com/content/cuda/cuda-downloads.html.

Look for the Linux distribution you are using and click the appropriate architecture for your system. The toolkit includes the CUDA driver, toolkit and samples.

2.5. Install the NVIDIA CUDA Toolkit

The CUDA Toolkit can be installed via 2 different methods: RPM/DEB packages, and a stand-alone installer. The RPM/DEB packages are available online in the NVIDIA CUDA

repositories. This is the recommended installation installation method. The alternate solution is to use the stand-alone installer.

RPM/DEB packages and repositories are not provided for Redhat 5 and Ubuntu 10.04. For those 2 Linux distributions, the stand-alone installer must be used.

2.5.1. Package Manager Installation

The installation is a two-step process. First the small repository configuration package must be downloaded from the NVIDIA CUDA download page, and installed manually. The package sets the package manager database to include the CUDA repository. Then the CUDA Toolkit is installed using the package manager software.

Prerequisites

- The package manager installations (RPM/DEB packages) and the stand-alone installer installations (.run file) of the NVIDIA driver are incompatible. Before using the RPM/DEB packages, uninstall the NVIDIA driver with the following command:

```
/usr/bin/nvidia-uninstall
```

- On Redhat, the NVIDIA driver RPM packages depend on other external packages, such as DKMS and libvdpau. Those packages are only available on third-party repositories, such as EPEL. Any such third-party repositories must be added to the package manager repository database before installing the NVIDIA driver RPM packages, or missing dependencies will prevent the installation from proceeding.
- On Ubuntu 12.04, to enable armhf as a foreign architecture, the following commands must be executed first:

```
$ echo "foreign-architecture armhf" >> /etc/dpkg/dpkg.cfg.d/multiarch
$ sudo apt-get update
```

Installation Instructions

- Redhat & Fedora

```
$ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
$ sudo yum clean expire-cache
$ sudo yum install cuda
```

- SLES & OpenSUSE

```
$ sudo rpm --install cuda-repo-<distro>-<version>.<architecture>.rpm
$ sudo zypper refresh
$ sudo zypper install cuda
```

- Ubuntu

```
$ sudo dpkg -i cuda-repo-<distro>_<version>_<architecture>.deb
$ sudo apt-get update
$ sudo apt-get install cuda
```

Available packages

The recommended installation packages are **cuda** and **cuda-cross**. Those two packages will install the full set of other CUDA packages required for development and should cover most scenarios

The **cuda** package installs all the available packages for native developments. That includes the compiler, the debugger, the profiler, NSight Eclipse Edition, the math libraries,... It also includes the NVIDIA driver package.

The **cuda-cross** package installs all the available packages for cross-platform developments, such as the 32-bit and 64-bit libraries. It does not include the NVIDIA driver package.

The packages installed by the packages above can also be installed individually by specifying their names explicitly. The list of available packages can obtained with:

```
$ yum --disablerepo="*" --enablerepo="cuda" list available     # RedHat & Fedora
$ zypper pa -r cuda                                           # OpenSUSE & SLES
$ cat /var/lib/apt/lists/cuda*Packages | grep "Package:"      # Ubuntu
```

Package Upgrades

The **cuda** and **cuda-cross** packages points to the latest stable release of the CUDA Toolkit. When a new version is available, the standard package managers upgrade command will install the latest CUDA Toolkit.

```
$ yum --disablerepo="*" --enablerepo="cuda" update       # RedHat & Fedora
$ zypper up -r cuda                                      # OpenSUSE & SLES
$ apt-get upgrade "cuda*" "nvidia*"                      # Ubuntu
```

Some desktop environments, such as GNOME or KDE, will display an notification alert when new packages are available.

To avoid any automatic upgrade, and lock down the installation to the X.Y release, install the **cuda-X-Y** or **cuda-cross-X-Y**.

Side-by-side installations are supported. For instance, to install both the X.Y CUDA Toolkit and the X.Y+1 CUDA Toolkit, install the **cuda-X.Y** and **cuda-X.Y+1** packages.

2.5.2. Runfile Installation

This section describes the installation and configuration of the CUDA Toolkit runfile, which you previously downloaded.

The toolkit installation can install any combination of the driver, toolkit and samples.

The samples install to two directories: a pristine directory, and a writable directory. The writable directory is where you will build and run the samples. The pristine directory is used by the Samples Browser, and can be used to replace the writable directory, should it become corrupt.

Before installing the CUDA software packages, you should read the bundled *Release Notes*, as the notes provide important details on installation and software functionality.

Then, follow these few steps for a successful installation.

If you wish to have a version of the Cuda Toolkit installed earlier than v5.5, it must be installed prior to installing a Cuda Toolkit v5.5 or greater.

If you've already installed a stand-alone driver, you need to make sure it meets the minimum version requirement for the toolkit. This requirement can be found in the CUDA Toolkit release notes.

On many distributions, the driver version number can be found in the graphical interface menus under **Applications > System Tools > NVIDIA X Server Settings.** Or, from the command line, run:

```
/usr/bin/nvidia-settings
```

You can extract individual installers for each component by running (xx in 5.5.xx is the minor version of the installation package):

```
sh cuda_5.5.xx_linux_32_rhel5.x.run -extract=/path/to/extract/dir/
```

Keep in mind that the extraction path must be an absolute path.

1. Exit the GUI if you are in a GUI environment by pressing **Ctrl-Alt-Backspace**.
   Some distributions require you to press this sequence twice in a row; others have disabled it altogether in favor of a command such as

   ```
   sudo /etc/init.d/gdm stop
   ```

   Still others require changing the system runlevel using a command such as

   ```
   /sbin/init 3
   ```

   Consult your distribution's documentation to find out how to properly exit the GUI.
2. If you are running an Ubuntu distro which installs the nvidia-current Debian package by default, and want to install the NVIDIA Display Driver, remove this package by running:

   ```
   sudo apt-get --purge remove nvidia-current
   ```

3. The driver and toolkit must be installed for CUDA to function. If you have not installed a stand-alone driver, install the driver from the NVIDIA CUDA Toolkit.

   If the CUDA Samples are installed as a different user, copy the pristine installation of the CUDA Samples (/usr/local/cuda-5.5/samples) to the desired user directory and change the ownership of the copied directory and subdirectories to the user.

Install the CUDA Toolkit (xx in 5.5.xx is the minor version of the installation package) by running the downloaded **.run** file as a *superuser*.

```
sudo sh cuda_5.5.xx_linux_32_rhel5.x.run
```

If you are using an Optimus system and are installing the driver, you must pass the --optimus option to the CUDA Toolkit installer. If you are instead installing a stand alone driver on an optimus system, you must pass --no-opengl-files to the installer and decline the xorg.conf update at the end of the installation.

If the Display Driver component fails to install, you may have the Nouveau drivers worked into your root filesystem (initramfs). To remove this, you will have to rebuild your initramfs image:

```
sudo mv /boot/initramfs-$(uname -r).img /boot/initramfs-$(uname -r)-
nouveau.img
```

```
sudo dracut /boot/initramfs-$(uname -r).img $(uname -r)
```

If you are using Grub2 as a bootloader, you may also need to edit its config to prevent Nouveau from loading. Add "rdblacklist=nouveau nouveau.modeset=0" to the end of the GRUB_CMDLINE_LINUX entry in /etc/default/grub. Then, remake your Grub configuration by running:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

. Once this is done, reboot your machine and attempt the install again.

You can select which packages you wish to install at the start of the installation.

The CUDA Toolkit installation defaults to **/usr/local/cuda-5.5**. In addition, a symbolic link is created from **/usr/local/cuda** to **/usr/local/cuda-5.5** in order for existing projects to make use of the new CUDA Toolkit.

The CUDA Samples are installed to a user folder **$(HOME)/NVIDIA_CUDA-5.5_Samples**. A pristine copy of the CUDA Samples also resides in **/usr/local/cuda-5.5/samples**.

4. Define the environment variables.

   - The **PATH** variable needs to include **/usr/local/cuda-5.5/bin**
   - **LD_LIBRARY_PATH** needs to contain **/usr/local/cuda-5.5/lib** on a 32-bit system, and both **/usr/local/cuda-5.5/lib** and **/usr/local/cuda-5.5/lib64** on a 64-bit system

   The typical way to place these values in your environment is with the following commands:

   ```
   export PATH=/usr/local/cuda-5.5/bin:$PATH
   ```

   and

   ```
   export LD_LIBRARY_PATH=/usr/local/cuda-5.5/lib:$LD_LIBRARY_PATH
   ```

   for 32-bit operating systems, or

   ```
   export LD_LIBRARY_PATH=/usr/local/cuda-5.5/lib:/usr/local/cuda-5.5/lib64:
   $LD_LIBRARY_PATH
   ```

for 64-bit operating systems.

To make such settings permanent, place them in **~/.bash_profile**.

5. If you wish to build *all* of the samples, including those with graphical rather than command-line interfaces, you may need to install additional system libraries or headers if you have not done so before.

   While every Linux distribution is slightly different with respect to package names and package installation procedures, the libraries and headers most likely to be necessary are OpenGL (e.g., Mesa), GLU, GLUT, and X11 (including Xi, Xmu, and GLX). These can be installed on Ubuntu as follows, for example:

```
sudo apt-get install freeglut3-dev build-essential libx11-dev libxmu-dev
 libxi-dev libgl1-mesa-glx libglu1-mesa libglu1-mesa-dev
```

6. If you do not use a GUI environment, or use an Optimus system, ensure that the device files**/dev/nvidia*** exist and have the correct file permissions.

   (This would be done automatically when initializing a GUI environment on a discrete GPU.) This can be done by creating a startup script like the following to load the driver kernel module and create the entries as a *superuser* at boot time:

```
#!/bin/bash

/sbin/modprobe nvidia

if [ "$?" -eq 0 ]; then
  # Count the number of NVIDIA controllers found.
  NVDEVS=`lspci | grep -i NVIDIA`
  N3D=`echo "$NVDEVS" | grep "3D controller" | wc -l`
  NVGA=`echo "$NVDEVS" | grep "VGA compatible controller" | wc -l`

  N=`expr $N3D + $NVGA - 1`
  for i in `seq 0 $N`; do
    mknod -m 666 /dev/nvidia$i c 195 $i
  done

  mknod -m 666 /dev/nvidiactl c 195 255

else
  exit 1
fi
```

7. Restart the GUI environment (using the command **startx** or **init 5** or **sudo /etc/init.d/gdm start** or the equivalent command on your system).

More information on installing the driver is available at http://us.download.nvidia.com/XFree86/Linux-x86/295.59/README/index.html.

New versions of CUDA software can require later versions of Linux and of the NVIDIA driver, so always verify that you are running the correct driver release and version of Linux for the version of the CUDA toolkit you are using.

Installing Mesa may overwrite the **/usr/lib/libGL.so** that was previously installed by the NVIDIA driver, so a reinstallation of the NVIDIA driver might be required after installing these libraries.

2.6. Verify the Installation

Before continuing, it is important to verify that the CUDA toolkit can find and communicate correctly with the CUDA-capable hardware. To do this, you need to compile and run some of the included sample programs.

Ensure the PATH and LD_LIBRARY_PATH variables are set correctly as described in step 4 of section 2.5.

2.6.1. Verify the Driver Version

If you installed the driver, verify that the correct version of it is installed.

This can be done through your System Properties (or equivalent) or by executing the command

```
cat /proc/driver/nvidia/version
```

Note that this command will not work on an Optimus system.

2.6.2. Compiling the Examples

The version of the CUDA Toolkit can be checked by running **nvcc -V** in a terminal window. The **nvcc** command runs the compiler driver that compiles CUDA programs. It calls the **gcc** compiler for C code and the NVIDIA PTX compiler for the CUDA code.

The NVIDIA CUDA Toolkit includes sample programs in source form. You should compile them by changing to **~/NVIDIA_CUDA-5.5_Samples** and typing **make**. The resulting binaries will be placed in **~/NVIDIA_CUDA-5.5_Samples/bin/linux/release**.

2.6.3. Running the Binaries

After compilation, go to **~/NVIDIA_CUDA-5.5_Samples/bin/linux/release** and run **deviceQuery**. If the CUDA software is installed and configured correctly, the output for **deviceQuery** should look similar to that shown in Figure 1.

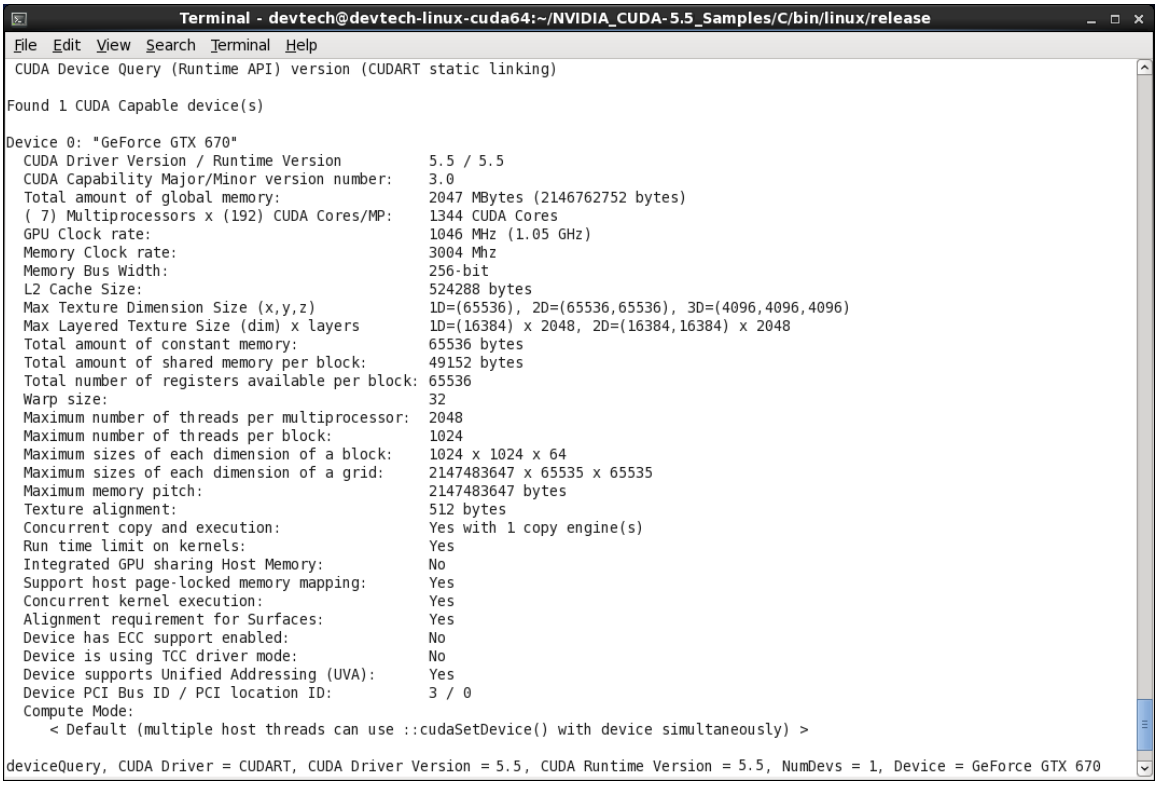

Figure 1 Valid Results from SDK deviceQuery Program

The exact appearance and the output lines might be different on your system. The important outcomes are that a device was found (the first highlighted line), that the device matches the one on your system (the second highlighted line), and that the test passed (the final highlighted line).

If a CUDA-capable device and the CUDA Driver are installed but **deviceQuery** reports that no CUDA-capable devices are present, this likely means that the **/dev/nvidia*** files are missing or have the wrong permissions.

On systems where **SELinux** is capable, you might need to temporarily disable this security feature to run **deviceQuery**. To do this, type:

```
#setenforce 0
```

from the command line as the *superuser*.

Running the **bandwidthTest** program ensures that the system and the CUDA-capable device are able to communicate correctly. Its output is shown in Figure 2.

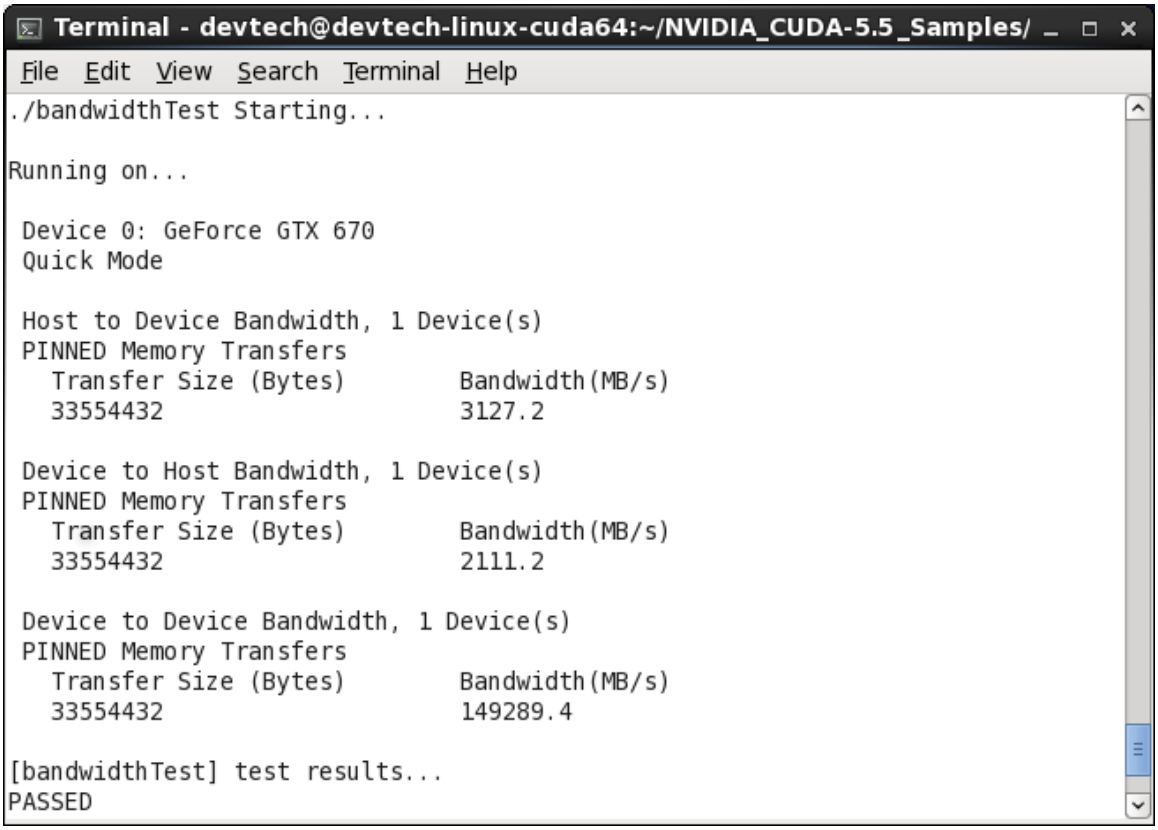

Figure 2 Valid Results from SDK bandwidthTest Program

Note that the measurements for your CUDA-capable device description will vary from system to system. The important point is that you obtain measurements, and that the second-to-last line (in Figure 2) confirms that all necessary tests passed.

Should the tests not pass, make sure you have a CUDA-capable NVIDIA GPU on your system and make sure it is properly installed.

If you run into difficulties with the link step (such as libraries not being found), consult the *Linux Release Notes* found in the **doc** folder in the samples directory.

Chapter 3. ADDITIONAL CONSIDERATIONS

Now that you have CUDA-capable hardware and the NVIDIA CUDA Toolkit installed, you can examine and enjoy the numerous included programs. To begin using CUDA to accelerate the performance of your own applications, consult the *CUDA C Programming Guide*, located in **`/usr/local/cuda-5.5/doc`**.

A number of helpful development tools are included in the CUDA Toolkit to assist you as you develop your CUDA programs, such as NVIDIA® Nsight™ Eclipse Edition, NVIDIA Visual Profiler, cuda-gdb, and cuda-memcheck.

For technical support on programming questions, consult and participate in the developer forums at http://developer.nvidia.com/cuda/.