NVIDIA CUDA GETTING STARTED GUIDE FOR LINUX
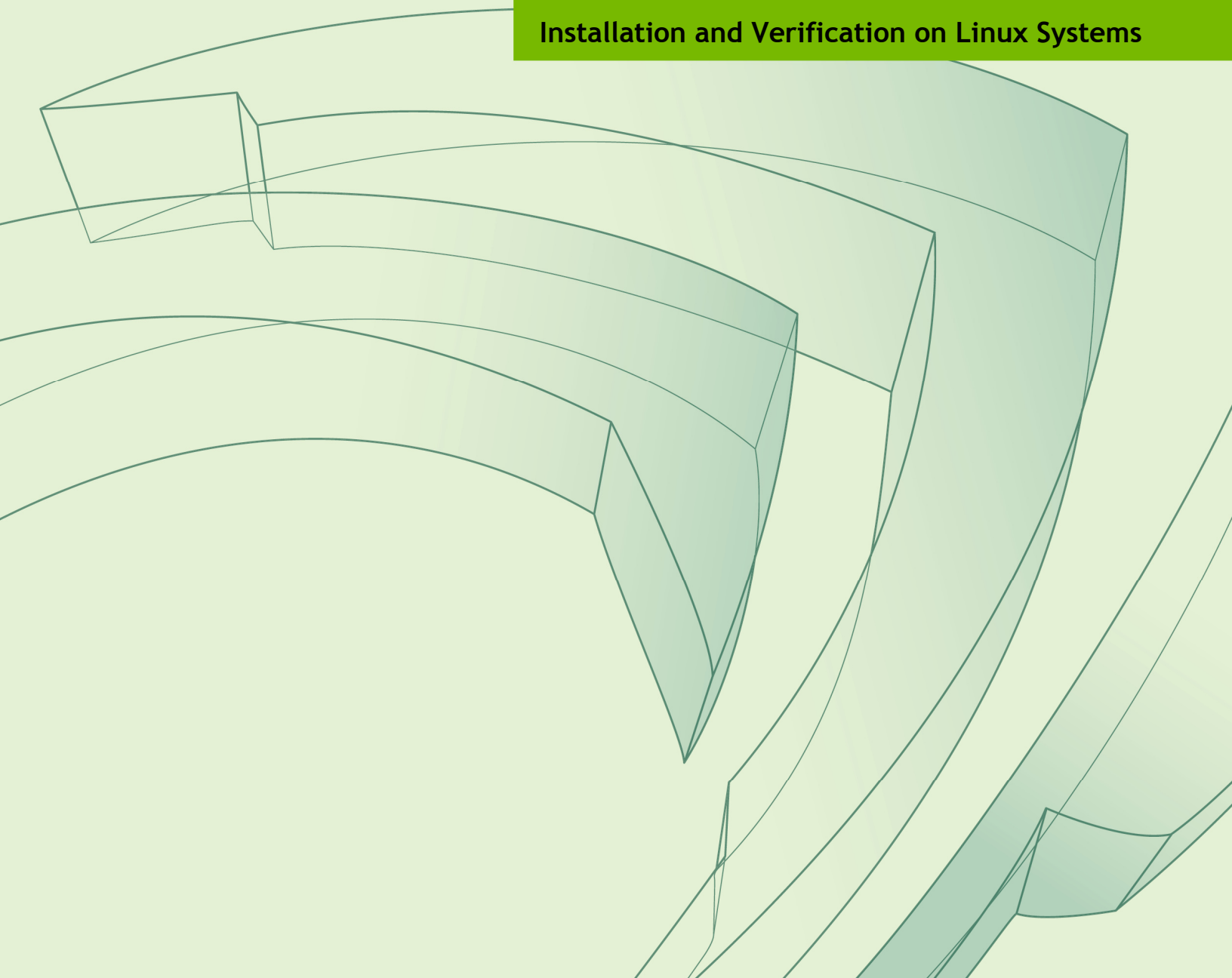
Installation and Verification on Linux Systems

DOCUMENT CHANGE HISTORY

Version	Date	Authors	Description of Change
01	April 20, 2010	CW, TS	Release
02	August 19, 2010	CW	Updated for CUDA Toolkit 3.2
03	March 3, 2011	CW	Updated for CUDA Toolkit 4.0

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

NVIDIA® CUDA™ is a general purpose parallel computing architecture introduced by NVIDIA. It includes the CUDA Instruction Set Architecture (ISA) and the parallel compute engine in the GPU. To program to the CUDA architecture, developers can use C, one of the most widely used high-level programming languages, which can then be run at great performance on a CUDA-enabled processor.

The CUDA architecture and its associated software were 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 and C for CUDA, 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 both the CPU and GPU without contention for memory resources.

CUDA-enabled GPUs have hundreds of cores that can collectively run thousands of computing threads. Each core has shared resources, including registers and 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.

SYSTEM REQUIREMENTS

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

▶ CUDA-enabled GPU
▶ Device driver
▶ A supported version of Linux with a gcc compiler and toolchain
▶ CUDA software (available at no cost from http://www.nvidia.com/cuda)

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.

> **Note:** 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.

INSTALLING CUDA DEVELOPMENT TOOLS

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

▶ Verify the system has a CUDA-enabled GPU and a supported version of Linux.
▶ Download the NVIDIA driver and the CUDA software.
▶ Install the NVIDIA driver.
▶ Install the CUDA software.

Test your installation by compiling and running one of the sample programs in the CUDA software to validate that the hardware and software are running correctly and communicating with each other.

VERIFY YOU HAVE A CUDA-ENABLED SYSTEM

Many NVIDIA products today contain CUDA-enabled GPUs. These include:

▶ NVIDIA GeForce® 8, 9, 200, 400, and 500 series GPUs
▶ NVIDIA Tesla™ computing solutions
▶ Many of the NVIDIA Quadro® products

An up-to-date list of CUDA-enabled GPUs can be found on the NVIDIA CUDA Web site at http://www.nvidia.com/object/cuda_gpus.html

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

To verify which video adapter your system uses, find the model number by going 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.

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.

Verify That gcc Is 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.

DOWNLOAD THE NVIDIA DRIVER AND CUDA SOFTWARE

Once you have verified that you have a supported NVIDIA processor and a supported version of Linux, you need to make sure you have a recent version of the NVIDIA driver. The CUDA Toolkit release notes specify which minimum version of the NVIDIA driver is required.

On many distributions, the driver release 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
```

The following CUDA software is required to run CUDA programs:

▶ The CUDA Toolkit
   The CUDA Toolkit contains the tools needed to compile and build a CUDA application in conjunction with the compilation driver. It includes tools, libraries, header files, and other resources.

▶ The GPU Computing SDK
   The GPU Computing SDK includes sample projects that provide source code and other resources for constructing CUDA programs.

The NVIDIA driver and CUDA software are available at no cost from the main CUDA download site at http://www.nvidia.com/object/cuda_get.html .

Choose the Linux distribution you are using, click **Search**, and download the NVIDIA driver. Save the driver file on your local system. Likewise, download and save the SDK and Toolkit.

INSTALL THE NVIDIA DRIVER

With the NVIDIA driver and software downloaded, you need to install the driver. Use the following procedure to install the driver:

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` to exit the GUI.

2. Run the driver installation package from the command line as a *superuser*.

3. Verify that the correct version of the driver is installed. This can be done through your System Properties (or equivalent) or by executing the command **cat /proc/driver/nvidia/version** .

4. If you do not use a GUI environment, ensure that the device files **/dev/nvidia*** exist and have the correct file permissions. (This would be done automatically when initializing a GUI environment.) This can be done 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
```

5. 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-x86/256.35/README/index.html.

> 💬 **Note:** 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 release for the version of CUDA software you are using.

INSTALL THE CUDA SOFTWARE

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

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

Then, follow these few steps for a successful installation.

1. Uninstall any previous versions of the CUDA Toolkit and the GPU Computing SDK. Do this by deleting the files from **/usr/local/cuda** and from **~/NVIDIA_GPU_Computing_SDK**, the default installation locations. (Note that older versions of the SDK installed into **~/NVIDIA_CUDA_SDK** by default rather than **~/NVIDIA_GPU_Computing_SDK**.) Adjust accordingly if you placed the files in non-default directories. (If you wish to keep the files so you can compile for different versions of CUDA software, then rename the existing directories before installing the new version and modify your Makefile accordingly.)

2. Install the CUDA Toolkit by running the downloaded **.run** file as a *superuser*. The CUDA Toolkit installation defaults to **/usr/local/cuda**.

3. Define the environment variables.
   - The PATH variable needs to include **/usr/local/cuda/bin**.
   - LD_LIBRARY_PATH needs to contain either **/usr/local/cuda/lib** or **/usr/local/cuda/lib64** for 32- or 64-bit operating systems, respectively.

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

   ```
   export PATH=/usr/local/cuda/bin:$PATH
   export LD_LIBRARY_PATH=/usr/local/cuda/lib:$LD_LIBRARY_PATH
   ```

   for 32-bit operating systems, with **lib64** replacing **lib** for 64-bit operating systems as mentioned above. To make such settings permanent, place them in **~/.bash_profile**.

4. Install the SDK (located in the second **.run** file) as a *regular user* in the default location, **$(HOME)/NVIDIA_GPU_Computing_SDK**.
   Installing as a regular user avoids access issues.

5. If you wish to build *all* of the SDK 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
```

Note: 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.

> **! Important:** Best practice for a multiuser Linux system is to also install a copy of the GPU Computing SDK as root that is accessible to users on a read-only basis. This pristine copy can then be copied to a user directory in the event users corrupt their copy of the source code.

VERIFY THE INSTALLATION

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

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.

NVIDIA includes sample programs in source form in the GPU Computing SDK. You should compile them all by changing to **~/NVIDIA_GPU_Computing_SDK/C** and typing **make**. The resulting binaries will be installed in **~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release**.

Running the Binaries

The sample projects use libraries pointed to by LD_LIBRARY_PATH, as described earlier, so make sure it points to the right directory.

After compilation, go to **~/NVIDIA_GPU_Computing_SDK/C/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. 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-enabled 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 enabled, 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-enabled device are able to communicate correctly. Its output is shown in Figure 2.

Figure 1.   Valid Results from SDK deviceQuery Program

```
gruetsch@localhost:~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release
File  Edit  View  Terminal  Tabs  Help
Running on......
        device 0:GeForce GTX 280
Quick Mode
Host to Device Bandwidth for Pageable memory
.
Transfer Size (Bytes)    Bandwidth(MB/s)
 33554432                1714.5

Quick Mode
Device to Host Bandwidth for Pageable memory
.
Transfer Size (Bytes)    Bandwidth(MB/s)
 33554432                1374.2

Quick Mode
Device to Device Bandwidth
.
Transfer Size (Bytes)    Bandwidth(MB/s)
 33554432                114816.5

&&&& Test PASSED

Press ENTER to exit...
```

Figure 2.   Valid Results from SDK bandwidthTest Program

Note that the measurements for your CUDA-enabled 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-enabled 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 SDK directory.

ADDITIONAL CONSIDERATIONS

Now that you have CUDA-enabled hardware and the software 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/doc**.

For technical support on programming questions, consult and participate in the bulletin board and mailing list at http://forums.nvidia.com/index.php?showforum=71.