Getting Started

NVIDIA CUDA Development Tools 2.3

Installation and Verification on Linux

July 2009

Table of Contents

Chapter 1. Introduction 1
CUDA—Supercomputing on Desktop Systems 1
System Requirements 2
About This Document 2
Chapter 2. Installing the CUDA Development Tools 3
Verify You Have a CUDA-Capable System 3
Verify You Have a Supported Version of Linux 4
Verify That gcc Is Installed 4
Download the NVIDIA Driver and CUDA Software 4
Install the NVIDIA Driver 5
Install the CUDA Software 6
Verify the Installation 8
Compiling the Examples 8
Running the Binaries 8
Chapter 3. Additional Considerations 11
Compiling for Hardware Emulation 11
What's Next? 11

Chapter 1. Introduction

CUDA—Supercomputing on Desktop Systems

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, today, use C, one of the most widely used high-level programming languages, which can then be run at great performance on a CUDA-capable 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-capable 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-capable GPU
- 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.

Chapter 2. Installing the 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-capable 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-Capable System

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

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

An up-to-date list of CUDA-capable GPUs can be found on the NVIDIA CUDA Web site at http://www.nvidia.com/object/cuda_learn_products.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, as shown in Figure 1. 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.

Note: It is possible to develop CUDA software in the absence of a CUDA-capable GPU. You can test the software in an emulation mode described later in this document. Naturally, performance on this platform is far less than on the CUDA-capable processor, so the emulated hardware should not be used for release versions and performance tuning.

Verify You Have a Supported Version of Linux

The CUDA Development Tools 2.3 require an x86-based distribution of Linux. To determine which distribution and release number you're running, type the following at the command line:

```
uname -i && 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're 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.

As of version 2.3 of the development tools, your Linux distribution must be one of the following versions:

- Red Hat Enterprise Linux 4.3-4.7, 5.0-5.3
- SUSE Enterprise Desktop 10-SP2 or 11
- Open SUSE 11.1
- Fedora 10
- Ubuntu 8.10 or 9.04

Subsequent updates of the development tools will support other versions of Linux, so check the download page for the latest supported platforms.

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. Currently, CUDA development tools support gcc version 3.4 as well as versions 4.x. To verify the version of gcc installed on your system, type the following on the command line:

```
gcc --version
```

If an error message appears, 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 2.3 requires version 190 or later of the NVIDIA driver.

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`**. Figure 1 shows the resulting screen (based on Red Hat Enterprise Linux 4.x).

In addition, to run CUDA programs, you will need the following CUDA software:

- The CUDA Toolkit
- The CUDA SDK

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 CUDA 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

After you've downloaded the NVIDIA driver and software, you will need to install the driver. If you're in a GUI environment, exit the GUI (ctl-alt-backspace). At the command line, turn off X Windows via **`/sbin/init 3`**. Then run the driver package from the command line as a superuser. Restart the GUI environment (**`startx`** or **`init 5`**, or the equivalent command on your system). In your System Properties (or equivalent), verify that the driver is installed as shown in Figure 1.

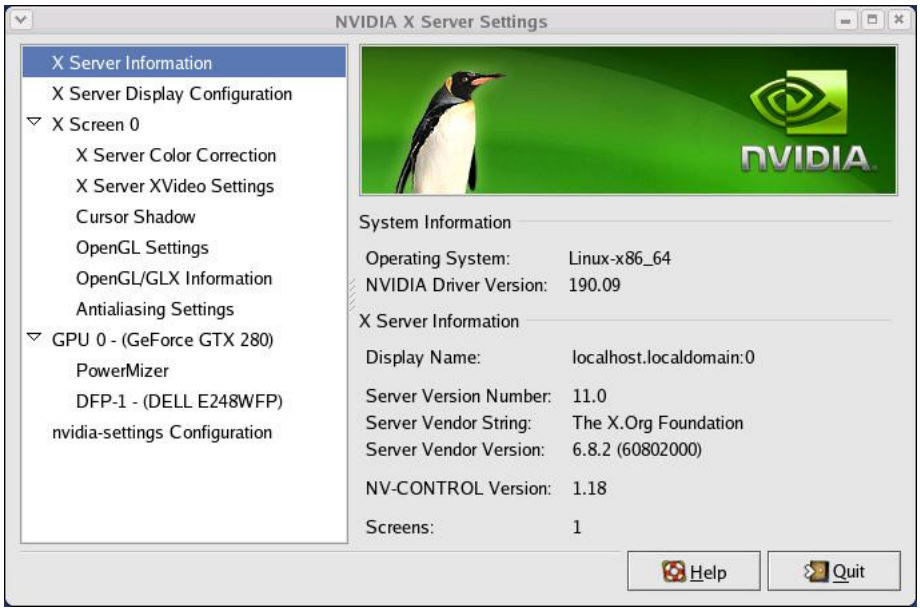

Figure 1. The NVIDIA Driver Information Window

More information on installing the driver is available at http://us.download.nvidia.com/XFree86/Linux-x86/1.0-9755/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 right release for the version of CUDA software you are using.

Install the CUDA Software

The following section describes the installation and configuration of the CUDA Toolkit and SDK, which you downloaded in a previous step.

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.

Uninstall any previous versions of the CUDA Toolkit and CUDA SDK if they have previously been installed. Do this by deleting the files from **`/usr/local/cuda`** and from **`$(HOME)/NVIDIA_CUDA_SDK/`**, the default installation locations. Adjust accordingly if you placed the files elsewhere. (If you wish to keep the files so you can compile for different versions of CUDA software, then rename the existing directories and modify your makefile accordingly.)

Install the CUDA Toolkit by running the downloaded .run file as a superuser. The CUDA Toolkit installation defaults to **`/usr/local/cuda`**. Several environment variables need to be defined in the installation. The **`PATH`** variable needs to include

/usr/local/cuda/bin. In addition, **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**.

The SDK is in the second .run file. It should be installed as a regular user (to avoid access issues by users) using the default location in the installation script, which is **$(HOME)/NVIDIA_GPU_Computing_SDK**. Note that this location is different than the default location in previous versions: **$(HOME)/NVIDIA_CUDA_SDK**.

Best practice for a multiuser Linux system is to also install a version 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 proceeding, it's important to verify that the CUDA programs can find and communicate correctly with the CUDA-capable hardware. To do this, you will 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. **`nvcc`** is the command to run 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 CUDA SDK. You should compile them all by changing to **`NVIDIA_GPU_Computing_SDK/C`** in the user's home directory and typing **`make`**. The resulting binaries will be installed under the home directory 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`** in the user's home directory and run **`deviceQuery`**. If the CUDA software is installed and configured correctly, the output for **`deviceQuery`** should look similar to Figure 2. 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).

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.

Note: On multiuser systems, access to NVIDIA devices must be enabled for remote users. To do this, enable read-write privileges for all users on /dev/nv* devices.

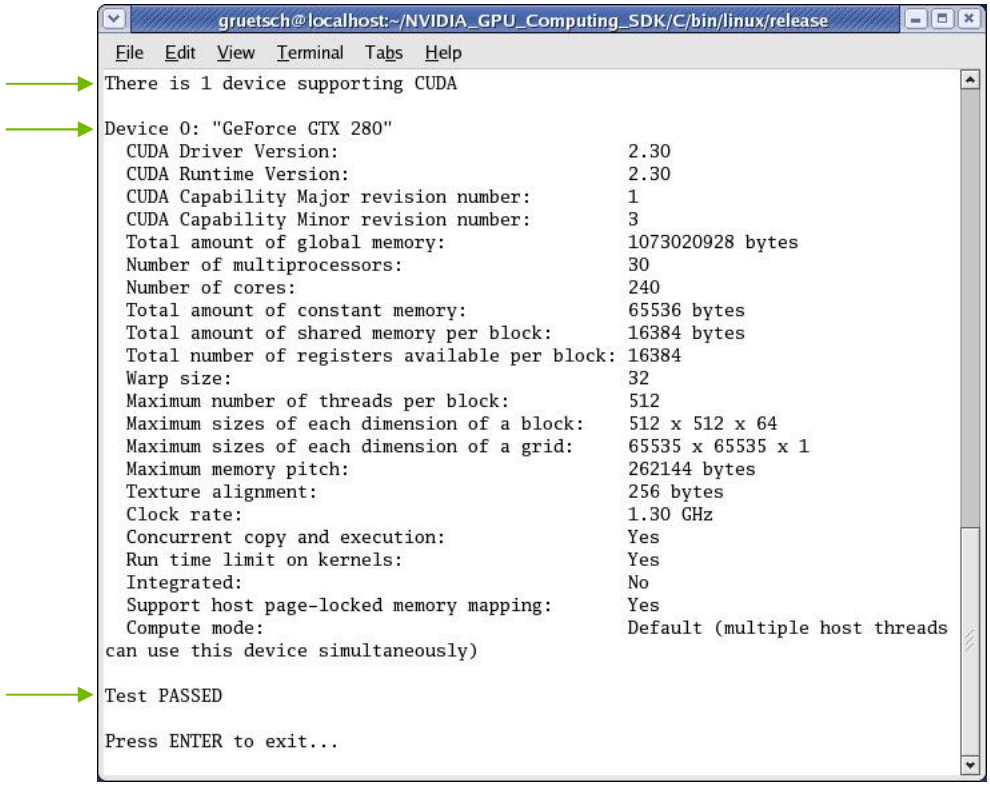

Figure 2. Valid Results from the SDK **`deviceQuery`** Program

Running the **`bandwidthTest`** program ensures that the system and the CUDA-capable device are able to communicate correctly. Its output is shown in Figure 3.

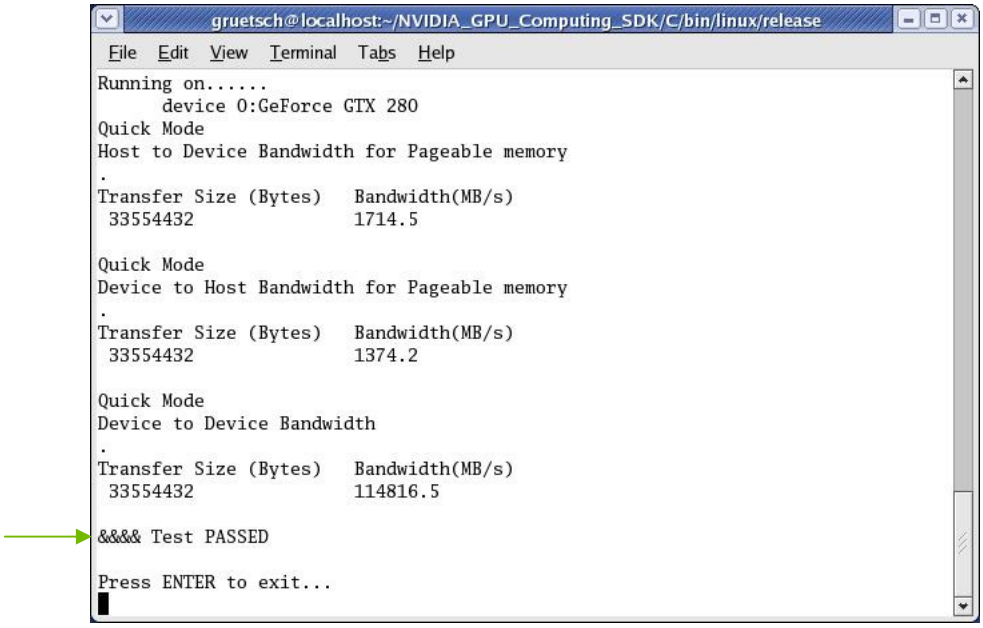

Figure 3. Valid Results from SDK **`bandwidthTest`** Program

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 (highlighted) 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 SDK directory.

To compile programs for emulation (on systems that have no CUDA-capable graphics hardware available), see the next section.

Chapter 3. Additional Considerations

Compiling for Hardware Emulation

The previous section explained how to compile and build the included files. To recap, simply go to the SDK installation directory, type **`make`**, and the resulting binaries will be installed in **`bin/linux/release`** under the SDK installation directory.

To see the individual steps in the build process, be sure to enable the verbose option on make.

On systems without a CUDA-capable GPU, it will be necessary to use an emulated GPU. The SDK enables the creation of binaries for an emulated hardware environment by using **`make emu=1`** from the command line. The resulting binaries will be placed in **`bin/linux/emurelease`** under the SDK installation directory.

What's Next?

Now that you have CUDA-capable 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 *NVIDIA CUDA Programming Guide*, located in **`/usr/local/cuda/doc`**.

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

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, CUDA, GeForce, NVIDIA Quadro, and Tesla are trademarks or registered trademarks of NVIDIA Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2009 NVIDIA Corporation. All rights reserved.

NVIDIA Corporation | 2701 San Tomas Expressway | Santa Clara, CA 95050 | **www.nvidia.com**