NVIDIA CUDA GETTING STARTED GUIDE FOR LINUX

DU-05347-001_v3.0 | May 2012

Installation and Verification on Linux Systems

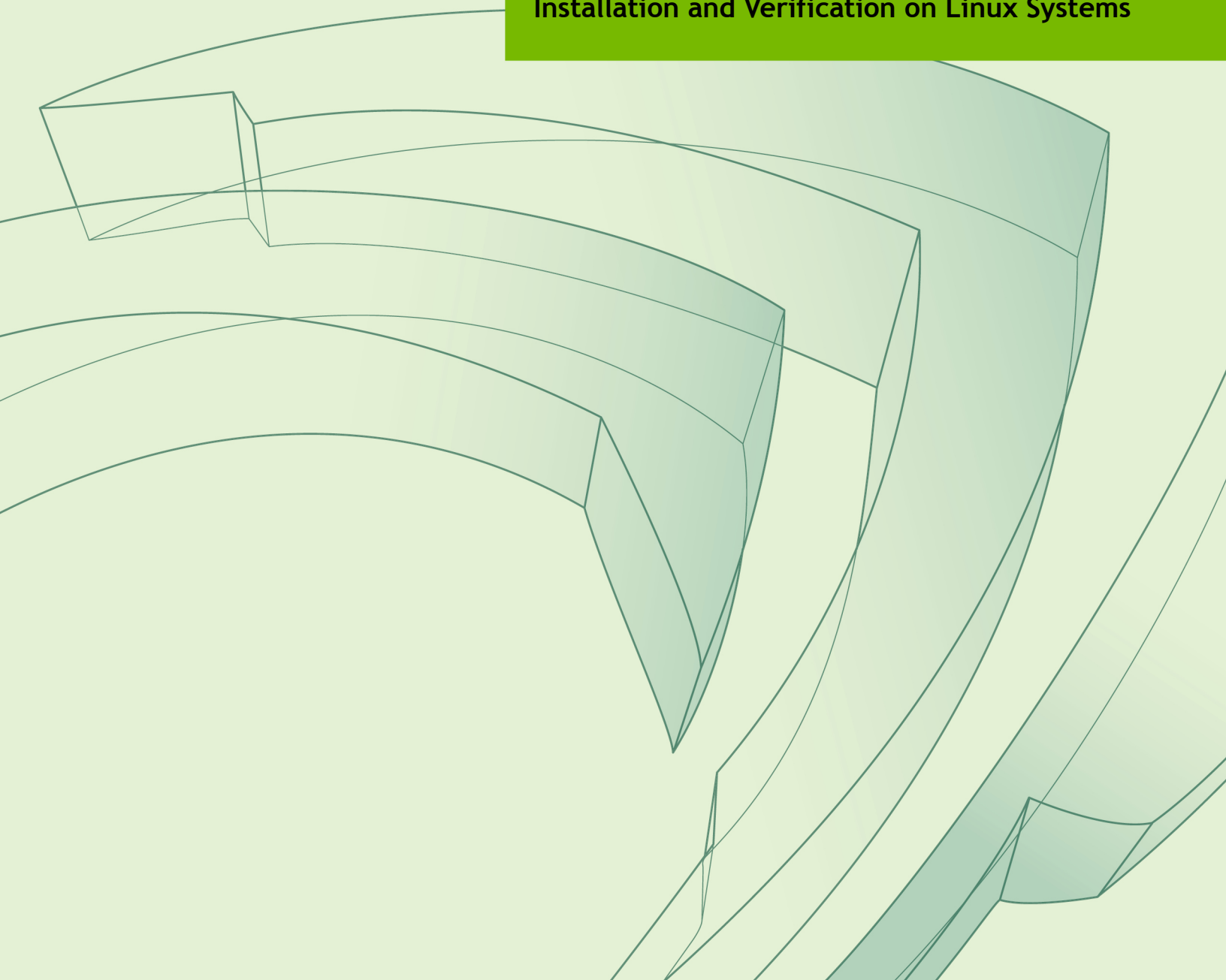

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1. 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-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.

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.

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 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.

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

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

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

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 v5.0, it must be installed prior to installing a Cuda Toolkit v5.0 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
```

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
   ```

2. 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.

   Install the CUDA Toolkit by running the downloaded `.run` file as a *superuser*.

   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 along 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.

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

   The CUDA Toolkit installation defaults to `/usr/local/cuda-5.0`. In addition, a symbolic link is created from `/usr/local/cuda` to `/usr/local/cuda-5.0` 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-5.0_Samples`. A pristine copy of the CUDA Samples also resides in `/usr/local/cuda-5.0/samples`.

   Important: This pristine copy can then be copied to a user directory in the event users corrupt their copy of the source code.

3. Define the environment variables.

   - The `PATH` variable needs to include `/usr/local/cuda-5.0/bin`
   - `LD_LIBRARY_PATH` needs to contain either `/usr/local/cuda-5.0/lib` or `/usr/local/cuda-5.0/lib64` for 32-bit 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-5.0/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-5.0/lib:$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. 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
   ```

5. 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
   ```

6. 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 3 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-5.0_Samples/C` and typing `make`. The resulting binaries will be placed in `~/NVIDIA_CUDA-5.0_Samples/C/bin/linux/release`.

2.6.3 Running the Binaries

After compilation, go to `~/NVIDIA_CUDA-5.0_Samples/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 Valid Results from SDK deviceQuery Program.

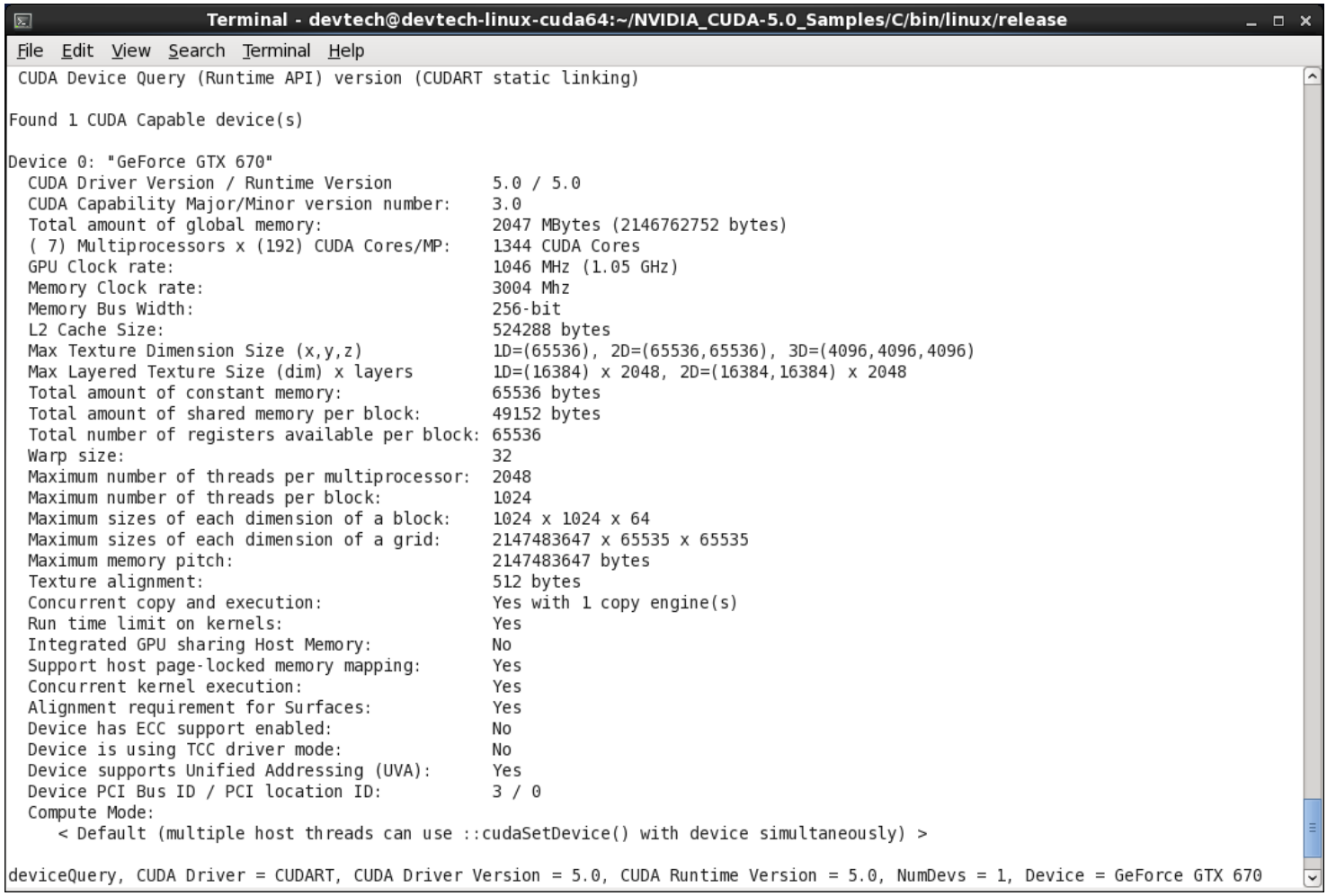

Figure 1 Valid Results from SDK deviceQuery Program

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 Valid Results from SDK bandwidthTest Program.

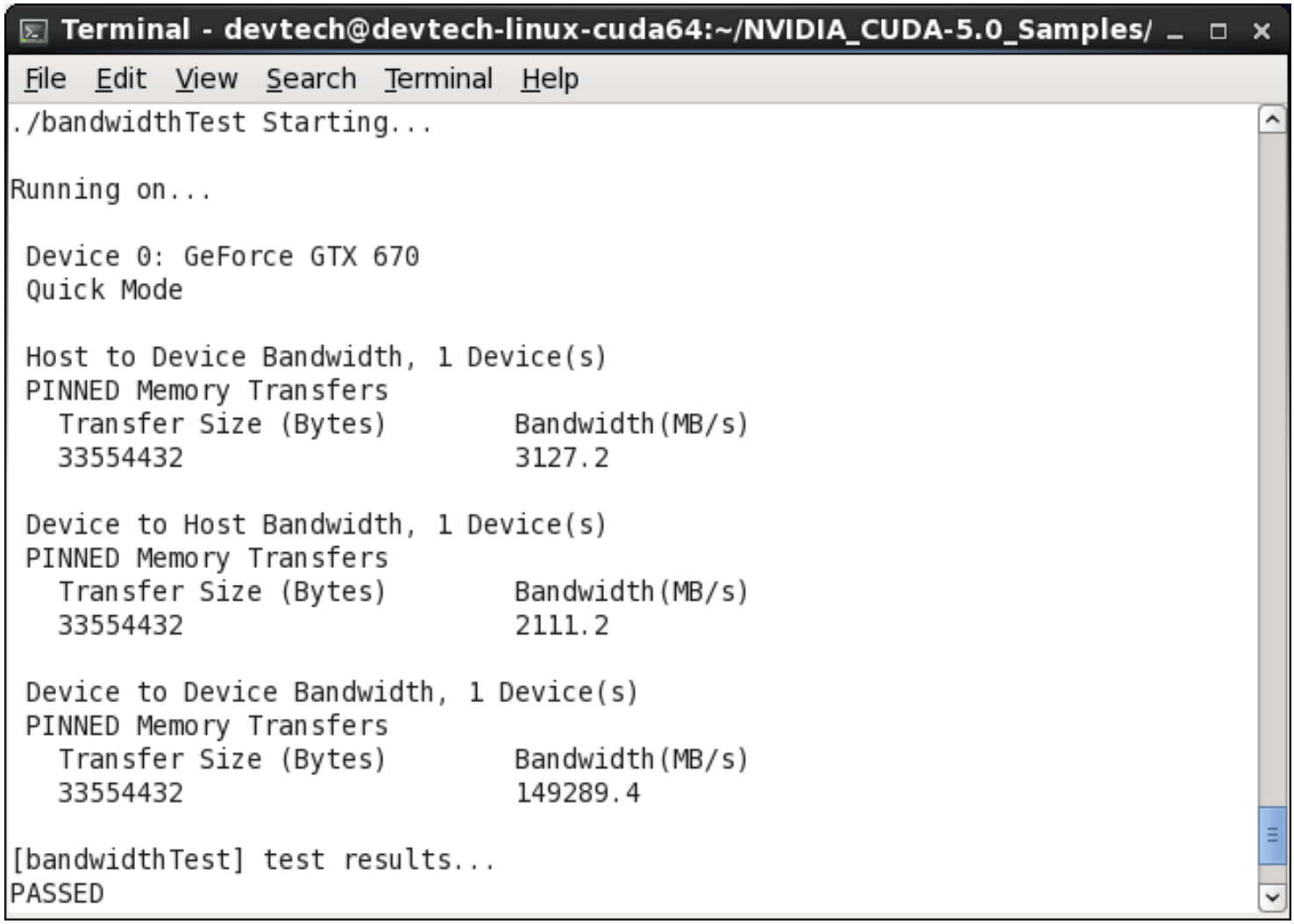

Figure 2 Valid Results from SDK bandwidthTest Program

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 Valid Results from SDK bandwidthTest Program) 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-5.0/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.