GETTING STARTED GUIDE FOR cuda-gdb PREVIEW RELEASE

DU-05637-001_v01 | December 2010

User Guide

DOCUMENT CHANGE HISTORY

DU-05637-001_v01

Version	Date	Authors	Description of Change
01	December 21, 2010	CW, TS, SS, VS	Initial release

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.

Note: This is special release of cuda-gdb preview on Mac OS X 10.6.5. This release should be used specifically for previewing cuda-gdb on Mac OS X.

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 shows you how to install and check the correct operation of the CUDA Development Tools with emphasis on the cuda-gdb preview release on Mac OS X 10.6.5.

SYSTEM REQUIREMENTS

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

- CUDA-enabled GPU
- The gcc compiler and toolchain installed using Xcode
- cuda-gdb for Mac OS packages available at: http://developer.nvidia.com/object/cuda-gdb.html
- Mac OS X v.10.6.5 - 32 bit support only
- Tesla GPU

Note: This preview release does not support Fermi.

ABOUT THIS DOCUMENT

This document is intended for readers familiar with the Mac OS X environment and the compilation of C programs from the command line. You do not need previous experience with CUDA or experience with parallel computation.

INSTALLING CUDA DEVELOPMENT TOOLS

The installation of CUDA development tools on a system running Mac OS X consists of three simple steps:

- Verify the system has a CUDA-enabled GPU, a supported version of Mac OS X, and that gcc has been installed via Xcode.
- Download the CUDA driver and software.
- Install the CUDA driver and 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, and 400 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 Mac OS X system uses, under the **Apple** menu select **About This Mac**, click the **More Info ...** button, and then select **Graphics/Displays** under the **Hardware** list.

VERIFY THE CORRECT VERSION OF MAC OS X

The cuda-gdb on Mac requires an Intel based Mac running 32-bit Mac OS X 10.6.5. To check which version you have, go to the Apple menu on the desktop and select **About This Mac**. You should see a dialog box similar to Figure 1.

Figure 1. About This Mac Dialog Box

Verify that gcc is Installed

The **gcc** compiler and toolchain are installed using the installation of Xcode. The Xcode development environment is found on the **Xcode Developer Tools DVD** that ships with new Mac systems and with Leopard, if you buy the operating-system upgrade. When installing Xcode, the package that contains **gcc** and the necessary tools is called *Developer Tools Essentials*. You can verify that **gcc** is installed entering the command **`/usr/bin/gcc --help`** from a Terminal window.

DOWNLOAD the cuda-gdb PREVIEW SOFTWARE

Before installing the CUDA software packages, read the Release Notes bundled with each package. The release notes provide important details on installation and software functionality.

Download the following special cuda-gdb preview packages from http://developer.nvidia.com/object/cuda-gdb.html:

- MacOS Preview: **Getting Started Guide for cuda-gdb on MacOS**
- MacOS Preview: **User Guide for cuda-gdb on Mac OS**
- MacOS Preview: **CUDA Driver with cuda-gdb support**
- MacOS Preview: **CUDA Toolkit with cuda-gdb included**
- MacOS Standard: **GPU Computing SDK 3.2 code samples**

INSTALL THE CUDA DRIVER AND SOFTWARE

Use the procedure described here to successfully install the CUDA driver and software. For information not listed here, refer to the documentation under **`/usr/local/cuda/doc`** in the download location. Before installing the CUDA software packages, you should read the **Release Notes** bundled with each, the release notes provide important details on installation and software functionality.

To install successfully, do the following steps:

1. **Uninstall any previous versions of the CUDA Toolkit and GPU Computing SDK.**
   Do this by deleting the files from **`/usr/local/cuda`** and from **`/Developer/GPU Computing`**, the default installation locations. (Note that older versions of the SDK installed into **`/Developer/CUDA`** by default rather than **`/Developer/GPU Computing`**.) 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 Driver.**
   Install the CUDA driver package by executing the installer and following the on-screen prompts. This will install **`/Library/Frameworks/CUDA.framework`** and the UNIX-compatibility stub **`/usr/local/cuda/lib/libcuda.dylib`** that refers to it.
3. **Install the CUDA Toolkit.**
   Install the CUDA Toolkit by executing the Toolkit installer package and following the on-screen prompts. The CUDA Toolkit supplements the CUDA Driver with compilers and additional libraries and header files that are installed into **`/usr/local/cuda`** by default.
4. **Define the environment variables.**
   - The **`PATH`** variable needs to include **`/usr/local/cuda/bin`**.
   - **`DYLD_LIBRARY_PATH`** needs to contain **`/usr/local/cuda/lib`**.

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

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

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

5. **Install the CUDA SDK.**
   The installation process places the files in **`/Developer/GPU Computing`**.

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.

Verify the Driver Installation

If the CUDA Driver is installed correctly, the CUDA kernel extension (`/System/Library/Extensions/CUDA.kext`) should be loaded automatically at boot time. To verify that it is loaded, use the command **`kextstat | grep -i cuda`**.

Verify Toolkit Installation

If the CUDA toolkit is correctly installed and the path is correctly setup then you should be able to execute "which nvcc" to verify that the correct CUDA compiler is being used.

Compile 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 **`/Developer/GPU Computing/C`** and type **`make`**. The resulting binaries will be installed under the home directory in: **`/Developer/GPU Computing/C/bin/darwin/release`**.

Note: This release is a 32 bit only release so boot your OS in 32bit mode by pressing 3 and 2 during boot or if you prefer to run 64bit then build the samples by explicitly passing the i386 flag as follows:
"**make i386=1**"

Note: If you want to run the samples in the debugger, first edit /Developer/GPU Computing/C/common/common.mk file to include –G in addition to –g debug option as follows: COMMONFLAGS += -g -G. Then run **make** command with debug symbols as follows: **"make i386=1 dbg=1"**

Run the Binaries

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

The executables need to find **libcutil.a** in **/Developer/GPU Computing/C/lib** and the corresponding include file **in /Developer/GPU Computing/C/common/inc**. They also need to access graphics libraries in **/Developer/GPU Computing/C/common/lib/darwin**. If you have installed the CUDA software as explained in this document, these locations are defaults and the programs should run without difficulty.

Once these required files are in place, go to **/Developer/GPU Computing/C/bin/darwin/release** and run **deviceQuery**. If CUDA is installed and configured correctly, the output for **deviceQuery** should look similar to Figure 2.

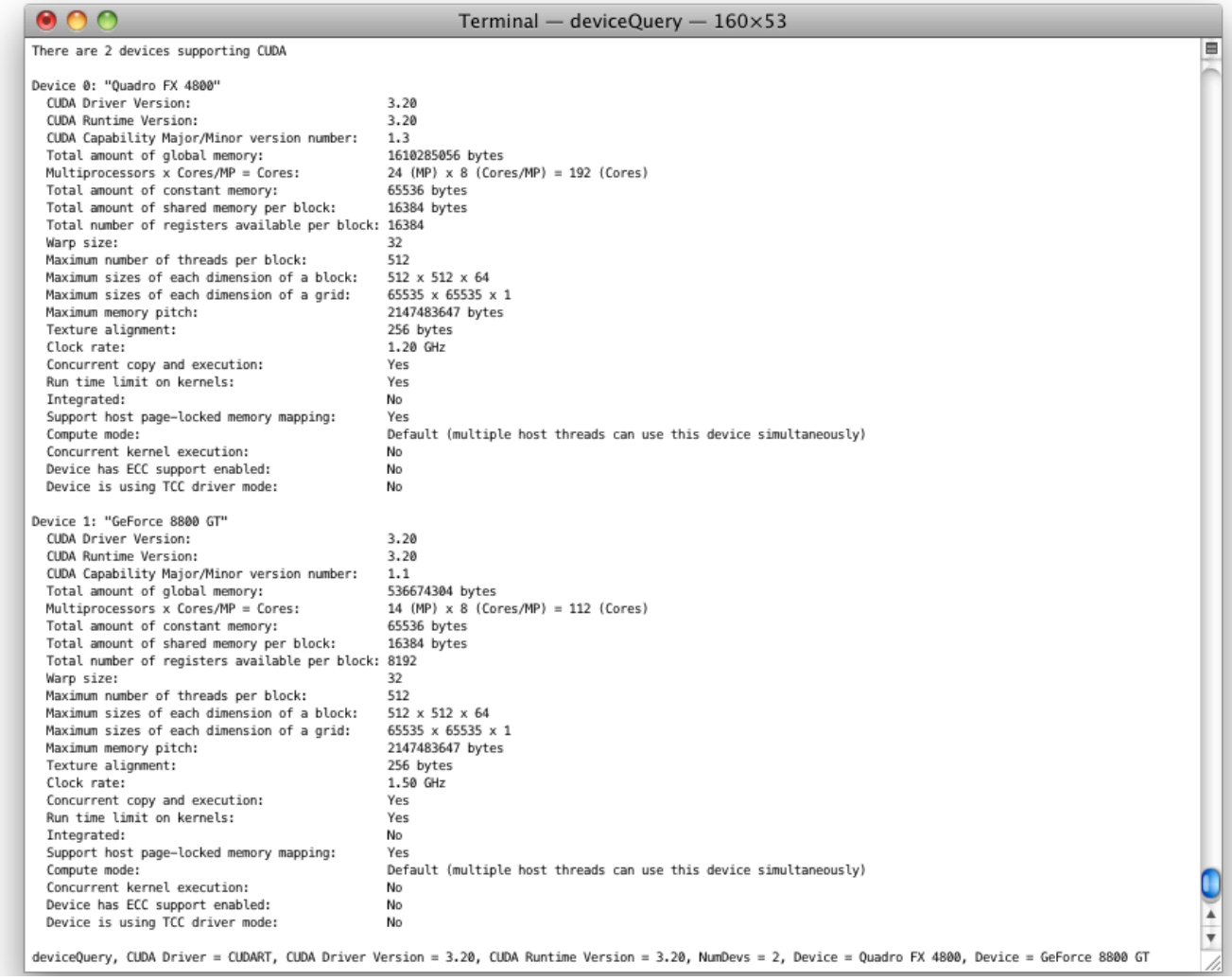

Figure 2. Valid Results from Sample CUDA deviceQuery Program

Note that the parameters for your CUDA device will vary. The key lines are the first and second ones that confirm a device was found and what model it is. Also, the next-to-last line, as indicated, should show that the test passed.

EVALUATING cuda-gdb on Mac OS

INSTALLING cuda-gdb

Binaries for cuda-gdb get installed as part of the cuda-gdb preview toolkit install package. Verify that cuda-gdb is installed properly by executing "which cuda-gdb" to make sure cuda-gdb is located. Adjust the path if cuda-gdb is not located. If it is in place run cuda-gdb; if it loads quit cuda-gdb at this point. Before running any apps in the debugger, please review the next section entitled, “Debugger System Usage Scenarios.” If you are unable to run cuda-gdb then execute the following commands which will setup the correct permissions:

```
sudo chgrp procmod /usr/local/cuda/bin/cuda-binary-gdb
sudo chmod 2755 /usr/local/cuda/bin/cuda-binary-gdb
sudo chmod 755 /usr/local/cuda/bin/cuda-gdb
```

Make sure cuda-gdb launches at this point; if it does not launch, reinstall the cuda-gdb preview package for Mac OS X.

COMPILING APPS TO RUN WITH cuda-gdb

CUDA apps should be compiled using nvcc with -g -G flags to include the debug information. Since this preview build supports only 32bit, use the -m32 flag during compilation of the .cu file if the boot environment is 64bit. Once the 32 bit binary is generated it can be run with cuda-gdb <app name>. Please refer to *cuda-gdb Mac Preview User Guide* for additional information.

DEBUGGER SYSTEM USAGE SCENARIOS

When the graphics desktop manager is running on the same GPU as CUDA then debugging that GPU results in pausing of the GPU which also pauses the desktop and thus freezes the GUI which makes the desktop unusable. To avoid desktop hangs cuda-gdb can be used in the following system configurations.

Single System Debugging where the host and the target are the same system

Single GPU console mode: To debug CUDA applications, cuda-gdb can be used only if Aqua desktop manager is **not** running which requires you to be in console mode with GPU initialized.

This can be achieved as follows:
In the desktop UI login prompt type ">console" as the user name to login into the console. This allows CUDA apps to be executed and debugged in a single GPU configuration.

Multi-GPU console mode

Simultaneous debugging of applications running CUDA kernels on multiple GPUs can be achieved by using cuda-gdb. In console mode cuda-gdb can be used to pause and debug every GPU in the system. You can enable console mode as described above for the single GPU console mode.

Multi-GPU with desktop manager running

This can be achieved by running Aqua on one GPU and CUDA on the other GPU to avoid hanging the desktop GUI. To determine which GPU should be used for CUDA run the deviceQuery app from the SDK sample. The output of deviceQuery as shown in Figure 2 indicates all the GPUs in the system. For example if you have two GPUs you will see Device0: "GeForce xxxx" and Device1: "GeForce xxxx". Choose the Device<index> that is not rendering the desktop on your connected monitor. For example if Device0 is rendering the desktop you choose Device1 for running and debugging CUDA application. This exclusion of the desktop can be achieved by executing this command: export CUDA_VISIBLE_DEVICES=1, where 1 corresponds to Device1 exposed to CUDA and Device0 dedicated to rendering desktop.

Figure 3 below shows an **nbody** CUDA application with UI getting debugged in this scenario.

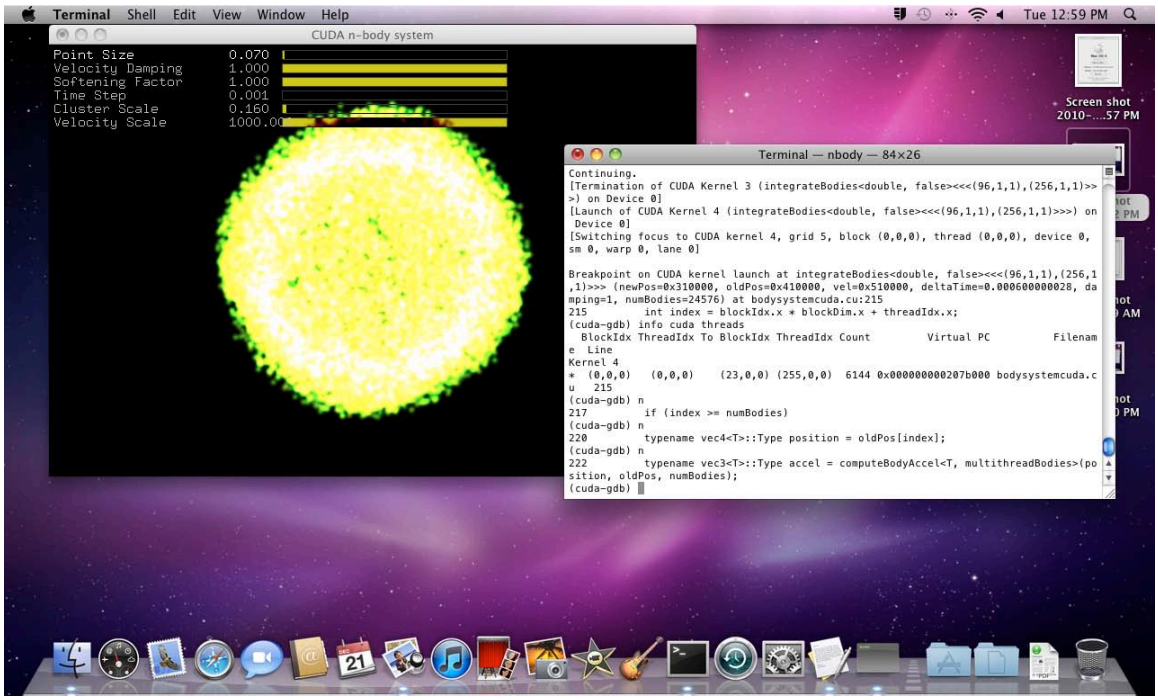

Figure 3. Multi-GPU System Setup

Remote Debugging where the host and target are different systems

From the host system use SSH or VNC to connect to the target system to have full control of the target system.

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.

For any **cuda-gdb** related bugs please email: CUDA-debugger-bugs@nvidia.com

Notice

Trademarks

Copyright