Getting Started

NVIDIA CUDA Development Tools 2.3
Installation and Verification on Mac OS X
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 CUDA-Capable GPU ............................................................................................ 3
  Verify the Correct Version of Mac OS X .......................................................................... 4
  Verify That gcc Is Installed ............................................................................................ 4
  Download the CUDA Driver and Software ................................................................. 5
  Installing the CUDA Driver and Software ..................................................................... 5
  Verify the Installation .................................................................................................... 6
    Compiling the Examples .............................................................................................. 6
    Running the Binaries .................................................................................................. 6

Chapter 3. Additional Considerations ............................................................................. 9
  Compiling for Hardware Emulation ............................................................................... 9
  What’s Next? ............................................................................................................... 9
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 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-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 the CUDA Development Tools.
System Requirements

To use CUDA Development Tools 2.3 on your system, you will need the following installed:

- CUDA-enabled GPU
- Mac OS X v. 10.5.6 or later
- The gcc compiler and toolchain installed via Xcode
- CUDA driver and software (available at no cost from http://www.nvidia.com/cuda)

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.
Chapter 2.
Installing the CUDA Development Tools

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

- Verify the system has a CUDA-capable 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 CUDA-Capable GPU

Many NVIDIA products today contain CUDA-enabled 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-enabled 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 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.

Note: It is possible to develop CUDA software in the absence of a CUDA-enabled 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-enabled processor, so the emulated hardware should not be used for release versions and performance tuning.
Verify the Correct Version of Mac OS X

CUDA Development Tools 2.3 requires an Intel-based Mac running Mac OS X v. 10.5.6 or later. 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.

![About This Mac Dialog Box](image)

To use versions 2.3 of the CUDA Development Tools, you need to check for version 10.5.6 or later.

**Note:** New versions of CUDA Development Tools can require later versions of Mac OS X, so always verify that you are running the right release for the version of CUDA you are using.

Verify That gcc Is Installed

The gcc compiler and toolchain are installed via 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. One can check that gcc is installed via the 
/usr/bin/gcc --help command issue from a Terminal window.

Download the CUDA Driver and Software

The CUDA driver and software is available from the main CUDA download site at http://www.nvidia.com/object/cuda_get.html.

On this page, choose Mac OS from the Operating System menu, and download the CUDA Driver, CUDA Toolkit, and the CUDA SDK packages for version 2.3 of the Development Tools.

Note that in previous versions of the CUDA Development Tools on Mac OS, the driver was packaged with the toolkit. As of version 2.3, the CUDA driver is installed as a separate package. There are two options for the CUDA driver: for NVIDIA GeForce GPUs download cudadriver_2.3.0_macos.pkg, and for NVIDIA Quadro GPUs download cudadriver_2.3.1_macos.pkg.

The CUDA Toolkit contains the tools needed to compile and build a CUDA application in conjunction with the nvcc 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.

Installing the CUDA Driver and Software

Follow these few steps for a successful installation. For information not listed here, see the documentation in /usr/local/cuda/doc.

Download the CUDA software from http://www.nvidia.com/object/cuda_get.html and save the .pkg files as directed in the previous section.

Uninstall 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, their default installation location. Adjust accordingly if you placed the files elsewhere. (If you wish to keep the files so you can compile for different versions of CUDA, then rename the existing directories and modify your makefile accordingly.)

Install the CUDA Driver by installing either the cudadriver_2.3.0_macos.pkg or cudadriver_2.3.1_macos.pkg file. The former is for use with NVIDIA GeForce GPUs, the latter with NVIDIA Quadro GPUs.

Install the CUDA Toolkit by installing the cudatoolkit_2.3_macos.pkg file. The CUDA Toolkit installation defaults to /usr/local/cuda. Several environment variables need to be defined in the installation:
PATH needs to add /usr/local/cuda/bin. In addition, 
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:

```bash
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.
The CUDA SDK is in the cudasdk_2.3_macos.pkg file. It should be installed. The
installation process will place the files in /Developer/GPU Computing. Note that
this differs from the default location of the previous version of the SDK, which was
placed in /Developer/CUDA.

**Verify the Installation**

Before proceeding, it’s important to verify that the CUDA programs can find and
communicate correctly with the CUDA-enabled 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 /Developer/GPU Computing/C and typing **make**.
The resulting binaries will be installed in the directory /Developer/GPU
Computing/C/bin/darwin/release. If the compilation fails because it doesn’t
find gcc, the compiler should be installed via an Xcode installation.

**Running the Binaries**

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

In addition, the executables need to find **libutil.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 earlier, 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.

![Terminal output showing deviceQuery results]

There is 1 device supporting CUDA.

Device: "GeForce 9800 GT"

Cuda driver version: 2.10
Cuda runtime version: 2.10
Cuda capability major/ minor number: 1 1
Total amount of global memory: 268239840 bytes
Number of multiprocessors: 1
Number of cores: 12
Total amount ofshared memory per block: 49152 bytes
Total number oftransfers available per block (RPP): 166Mbytes/s

Maximum number of threads per block: 512
Maximum size of each dimension of a grid: 16384 x 16384 x 4
Minimum number of threads per block: 32
Max Threads per Grid: 278576 bytes
Error flags: 1
Cuda initial: 1.8, 7.6
Enforcement policy and suspension: yes
Max number of active threads: yes
Integrated: no
Support for host page locked memory mapping: no
Compute mode: Default (multiple host threads can use this device simultaneously)

Test PASSED

Press ENTER to exit...

---

**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.

Running the `bandwidthTest` program ensures that the system and the CUDA device are able to communicate correctly. Its output is shown in Figure 3.
Figure 3. Valid Results from Sample CUDA bandwidthTest Program

Note that the measurements for your CUDA 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 an NVIDIA GPU on your system that supports CUDA and make sure it is properly installed.

To see a graphical representation of what CUDA can do, run the sample particles executable.

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 /Developer/GPU Computing/C directory, type `make`, and the resulting binaries will be installed in /Developer/GPU Computing/C/bin/darwin/release.

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. CUDA enables the creation of binaries for an emulated hardware environment by using `make emu=1` from the command line in /Developer/GPU Computing/C. The resulting binaries will be placed in /Developer/GPU Computing/C/bin/darwin/emurelease.

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 CUDA Programming Guide, located in /usr/local/cuda/doc.

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.