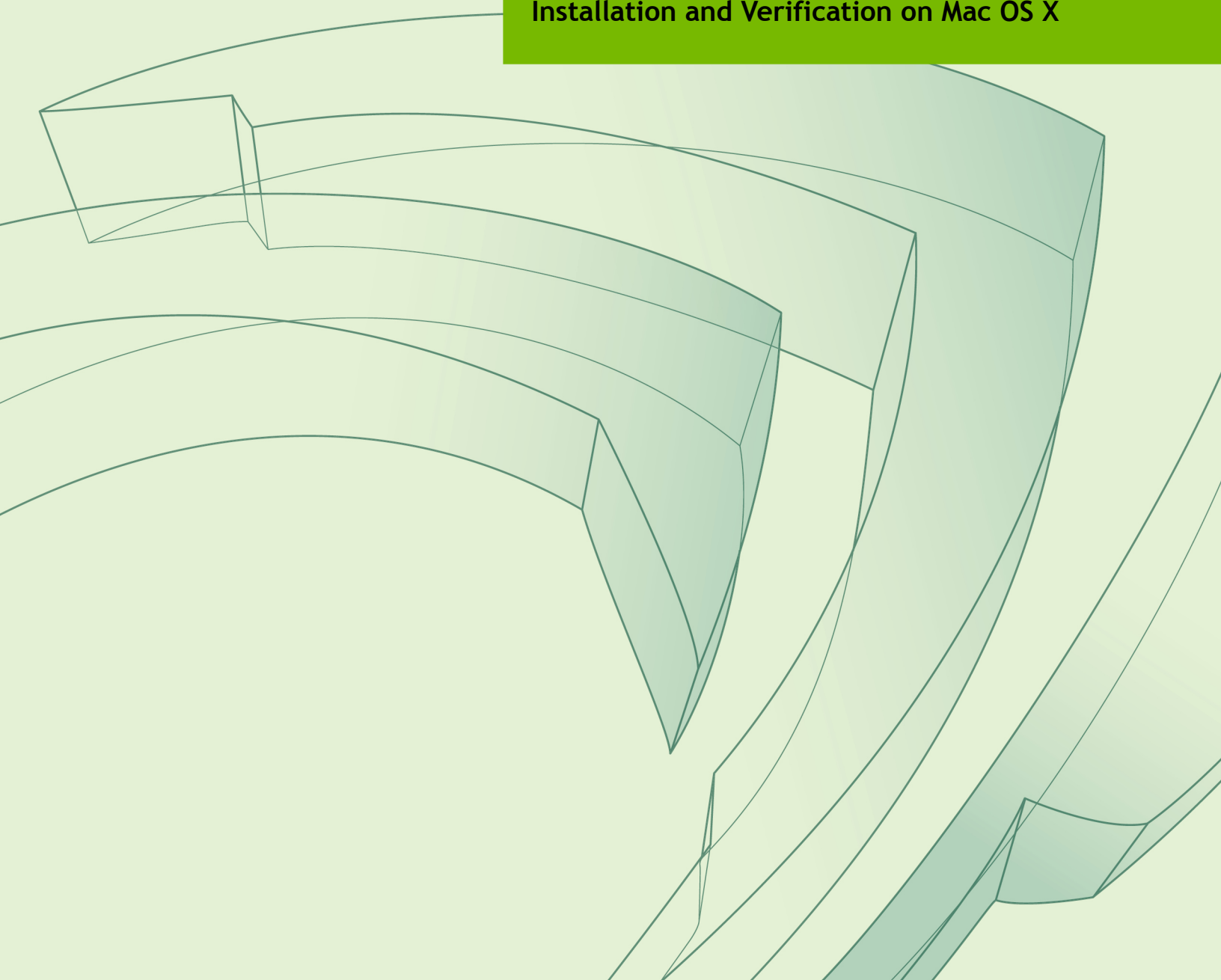




# NVIDIA CUDA GETTING STARTED GUIDE FOR MAC OS X

DU-05348-001\_v5.5 | July 2013

**Installation and Verification on Mac OS X**



# TABLE OF CONTENTS

|  |          |
|--|----------|
| <b>Chapter 1. Introduction.....</b>                      | <b>1</b> |
| 1.1. System Requirements.....                            | 1        |
| 1.2. About This Document.....                            | 2        |
| <b>Chapter 2. Installing CUDA Development Tools.....</b> | <b>3</b> |
| 2.1. Verify You Have a CUDA-Capable GPU.....             | 3        |
| 2.2. Verify the Correct Version of Mac OS X.....         | 3        |
| 2.2.1. Verify the System Has gcc Installed.....          | 4        |
| 2.3. Download the CUDA Software.....                     | 4        |
| 2.4. Install the CUDA Driver and Software.....           | 5        |
| 2.5. Verify the Installation.....                        | 6        |
| 2.5.1. Verify the Driver Installation.....               | 6        |
| 2.5.2. Compiling the Examples.....                       | 6        |
| 2.5.3. Running Binaries.....                             | 6        |
| <b>Chapter 3. Additional Considerations.....</b>         | <b>9</b> |

## LIST OF FIGURES

|   |   |
|---|---|
| Figure 1 About This Mac Dialog Box .....                    | 4 |
| Figure 2 Valid Results from deviceQuery CUDA Sample .....   | 7 |
| Figure 3 Valid Results from bandwidthTest CUDA Sample ..... | 8 |



# Chapter 1.

## INTRODUCTION

CUDA™ is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

CUDA was 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 C/C++, 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 the CPU and GPU without contention for memory resources.

CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads. These cores have shared resources including a register file and a shared 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
- ▶ Mac OSX v. 10.7.5 or later
- ▶ The *gcc* or *Clang* compiler and toolchain installed using Xcode
- ▶ NVIDIA CUDA Toolkit (available at <http://developer.nvidia.com/cuda-downloads>)

## 1.2. 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 CUDA DEVELOPMENT TOOLS

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

- ▶ Verify the system has a CUDA-capable GPU.
- ▶ Verify the system is running a supported version of Mac OS X.
- ▶ Verify the system has gcc or Clang installed via Xcode.
- ▶ 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 system is CUDA-capable, under the **Apple** menu select **About This Mac**, click the **More Info ...** button, and then select **Graphics/Displays** under the **Hardware** list. There you will find the vendor name and model of your graphics card. If it is an NVIDIA card that is listed in <http://developer.nvidia.com/cuda-gpus>, your GPU is CUDA-capable.

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

### 2.2. Verify the Correct Version of Mac OS X

The CUDA Development Tools require an Intel-based Mac running Mac OSX v. 10.7.5 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](#).



Figure 1 About This Mac Dialog Box

### 2.2.1. Verify the System Has gcc 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.

### 2.3. Download the CUDA Software

Once you have verified that you have a supported NVIDIA GPU, a supported version the MAC OS, and `gcc`, you need to 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>. It contains the driver and tools needed to create, build and run a CUDA application as well as libraries, header files, CUDA samples source code, and other resources.




## 2.4. Install the CUDA Driver and Software

Use the following procedure to successfully install the CUDA driver and software. For information not listed here, see the documentation under `/Developer/NVIDIA/CUDA-5.5/doc` in the download location.

Before installing the CUDA Toolkit, you should read the *Release Notes*, as they provide important details on installation and software functionality.

Then, follow these few steps for a successful installation.

1.  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.

Install the CUDA toolkit by executing the installer and following the on-screen prompts. You will be able to choose which packages you wish to install. The packages are:

- ▶ **CUDA Driver**

This will install `/Library/Frameworks/CUDA.framework` and the UNIX-compatibility stub `/usr/local/cuda/lib/libcuda.dylib` that refers to it.

- ▶ **CUDA Toolkit**

The CUDA Toolkit supplements the CUDA Driver with compilers and additional libraries and header files that are installed into `/Developer/NVIDIA/CUDA-5.5` by default. Symlinks are created in `/usr/local/cuda/` pointing to their respective files in `/Developer/NVIDIA/CUDA-5.5/`.

Previous installations of the toolkit will be moved to `/Developer/NVIDIA/CUDA-#. #` to better support side-by-side installations.

- ▶ **CUDA Samples**

The CUDA Samples are installed to `/Developer/NVIDIA/CUDA-5.5/samples`.

Previous installations of the samples will be moved to `/Developer/NVIDIA/CUDA-#. #/samples` to better support side-by-side installations.

2. Define the environment variables.

- ▶ The **PATH** variable needs to include `/Developer/NVIDIA/CUDA-5.5/bin`
- ▶ **DYLD\_LIBRARY\_PATH** needs to contain `/Developer/NVIDIA/CUDA-5.5/lib`

To change the environment variables for 32-bit operating systems:

```
export PATH=/Developer/NVIDIA/CUDA-5.5/bin:$PATH
export DYLD_LIBRARY_PATH=/Developer/NVIDIA/CUDA-5.5/lib:$DYLD_LIBRARY_PATH
```

To change the environment variables for 64-bit operating systems:

```
export PATH=/Developer/NVIDIA/CUDA-5.5/bin:$PATH
export DYLD_LIBRARY_PATH=/Developer/NVIDIA/CUDA-5.5/lib:$DYLD_LIBRARY_PATH
```

### 3. (Optional) Install a writable copy of the samples.

In order to modify, compile, and run the samples, the samples must be installed with write permissions. A convenience installation script is provided:

```
cuda-install-samples-5.5.sh <dir>
```

This script is installed with the `cuda-samples-5-5` package. The `cuda-samples-5-5` package installs only a read-only copy in `/Developer/NVIDIA/CUDA-5.5/samples`.

## 2.5. 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 `DYLD_LIBRARY_PATH` variables are [set correctly](#).

### 2.5.1. 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
```

### 2.5.2. 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 the directory where you installed the writable samples, `<dir>`, and typing `make`. The resulting binaries will be placed under `<dir>/bin/x86_64/darwin/release`.

### 2.5.3. Running Binaries

After compilation, go to `<dir>/bin/x86_64/darwin/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 2](#).

```

release — bash — 131x41
./deviceQuery Starting...

CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Quadro FX 4800"
  CUDA Driver Version / Runtime Version      5.5 / 5.5
  CUDA Capability Major/Minor version number: 1.3
  Total amount of global memory:             1536 MBytes (1610285056 bytes)
  (24) Multiprocessors, ( 8) CUDA Cores/MP: 192 CUDA Cores
  GPU Clock rate:                            1204 MHz (1.20 GHz)
  Memory Clock rate:                         800 Mhz
  Memory Bus Width:                          384-bit
  Maximum Texture Dimension Size (x,y,z)     1D=(8192), 2D=(65536, 32768), 3D=(2048, 2048, 2048)
  Maximum Layered 3D Texture Size, (num) layers 1D=(8192), 512 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(8192, 8192), 512 layers
  Total amount of constant memory:           65536 bytes
  Total amount of shared memory per block:    16384 bytes
  Total number of registers available per block: 16384
  Warp size:                                 32
  Maximum number of threads per multiprocessor: 1024
  Maximum number of threads per block:       512
  Max dimension size of a thread block (x,y,z): (512, 512, 64)
  Max dimension size of a grid size (x,y,z): (65535, 65535, 1)
  Maximum memory pitch:                      2147483647 bytes
  Texture alignment:                         256 bytes
  Concurrent copy and kernel execution:      Yes with 1 copy engine(s)
  Run time limit on kernels:                  Yes
  Integrated GPU sharing Host Memory:         No
  Support host page-locked memory mapping:    Yes
  Alignment requirement for Surfaces:         Yes
  Device has ECC support:                     Disabled
  Device supports Unified Addressing (UVA):   No
  Device PCI Bus ID / PCI location ID:       2 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 5.5, CUDA Runtime Version = 5.5, NumDevs = 1, Device0 = Quadro FX 4800
Result = PASS
dhcp-172-16-194-251:release eyoung$

```

Figure 2 Valid Results from deviceQuery CUDA Sample

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-capable device are able to communicate correctly. Its output is shown in [Figure 3](#).

```

[CUDA Bandwidth Test] - Starting...
Running on...

Device 0: Quadro FX 4800
Quick Mode

Host to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                   5287.2

Device to Host Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                   3901.9

Device to Device Bandwidth, 1 Device(s)
PINNED Memory Transfers
  Transfer Size (Bytes)      Bandwidth(MB/s)
  33554432                   37519.2

Result = PASS
dhcp-172-16-194-251:release eyoung$

```

Figure 3 Valid Results from bandwidthTest CUDA Sample

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 3](#)) 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 *Release Notes* found in the `doc` folder in the CUDA Samples directory.

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

## 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 `/Developer/NVIDIA/CUDA-5.5/doc`.

A number of helpful development tools are included in the CUDA Toolkit to assist you as you develop your CUDA programs, such as NVIDIA® Nsight™ Eclipse Edition, NVIDIA Visual Profiler, `cuda-gdb`, and `cuda-memcheck`.

For technical support on programming questions, consult and participate in the developer forums at <http://developer.nvidia.com/cuda/>.

## **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 of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## **Trademarks**

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## **Copyright**

© 2009-2013 NVIDIA Corporation. All rights reserved.