NVIDIA CUDA INSTALLATION GUIDE FOR MAC OS X

DU-05348-001_v9.1 | November 2017

Installation and Verification on Mac OS X

TABLE OF CONTENTS

Chapter 1. Introduction 1
1.1. System Requirements 1
1.2. About This Document 2
Chapter 2. Prerequisites 3
2.1. CUDA-capable GPU 3
2.2. Mac OS X Version 3
2.3. Xcode Version 3
2.4. Command-Line Tools 4
Chapter 3. Installation 5
3.1. Download 5
3.2. Install 5
3.3. Uninstall 7
Chapter 4. Verification 8
4.1. Driver 8
4.2. Compiler 8
4.3. Runtime 9
Chapter 5. Additional Considerations 11

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 need to have:

- a CUDA-capable GPU
- Mac OS X 10.12
- the *Clang* compiler and toolchain installed using Xcode
- the NVIDIA CUDA Toolkit (available from the CUDA Download page)

Table 1 Mac Operating System Support in CUDA 9.1

Toolchain		Mac OSX Version (native x86_64)
Xcode	Apple LLVM	10.12
8.3.3	8.1.0	YES

Note that the CUDA 9.1 driver supports macOS 10.13. This means that existing CUDA applications can be run on Mac systems with CUDA 9.1 installed, however new CUDA applications cannot be compiled with CUDA 9.1 on macOS 10.13.

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

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. PREREQUISITES

2.1. 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 on the CUDA-supported GPUs page, your GPU is CUDA-capable.

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

2.2. Mac OS X Version

The CUDA Development Tools require an Intel-based Mac running Mac OSX v. 10.12. To check which version you have, go to the **Apple** menu on the desktop and select **About This Mac**.

2.3. Xcode Version

A supported version of Xcode must be installed on your system. The list of supported Xcode versions can be found in the System Requirements section. The latest version of Xcode can be installed from the Mac App Store.

Older versions of Xcode can be downloaded from the Apple Developer Download Page. Once downloaded, the Xcode.app folder should be copied to a version-specific folder within **`/Applications`**. For example, Xcode 6.2 could be copied to **`/Applications/Xcode_6.2.app`**.

Once an older version of Xcode is installed, it can be selected for use by running the following command, replacing **`<Xcode_install_dir>`** with the path that you copied that version of Xcode to:

```
sudo xcode-select -s /Applications/<Xcode_install_dir>/Contents/Developer
```

2.4. Command-Line Tools

The CUDA Toolkit requires that the native command-line tools are already installed on the system. Xcode must be installed before these command-line tools can be installed.

The command-line tools can be installed by running the following command:

```
$ xcode-select --install
```

Note: It is recommended to re-run the above command if Xcode is upgraded, or an older version of Xcode is selected.

You can verify that the toolchain is installed by running the following command:

```
$ /usr/bin/cc --version
```

Chapter 3. INSTALLATION

Basic instructions can be found in the Quick Start Guide. Read on for more detailed instructions.

3.1. Download

Once you have verified that you have a supported NVIDIA GPU, a supported version the MAC OS, and clang, you need to download the NVIDIA CUDA Toolkit.

The NVIDIA CUDA Toolkit is available at no cost from the main CUDA Downloads page. The installer is available in two formats:

1. Network Installer: A minimal installer which later downloads packages required for installation. Only the packages selected during the selection phase of the installer are downloaded. This installer is useful for users who want to minimize download time.
2. Full Installer: An installer which contains all the components of the CUDA Toolkit and does not require any further download. This installer is useful for systems which lack network access.

Both installers install 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.

The download can be verified by comparing the posted MD5 checksum with that of the downloaded file. If either of the checksums differ, the downloaded file is corrupt and needs to be downloaded again.

To calculate the MD5 checksum of the downloaded file, run the following:

```
$ openssl md5 <file>
```

3.2. Install

Use the following procedure to successfully install the CUDA driver and the CUDA toolkit. The CUDA driver and the CUDA toolkit must be installed for CUDA to function.

If you have not installed a stand-alone driver, install the driver provided with the CUDA Toolkit.

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-9.1`** by default. Symlinks are created in **`/usr/local/cuda/`** pointing to their respective files in **`/Developer/NVIDIA/CUDA-9.1/`**. Previous installations of the toolkit will be moved to **`/Developer/NVIDIA/CUDA-#.#`** to better support side-by-side installations.
- **CUDA Samples (read-only)**: A read-only copy of the CUDA Samples is installed in **`/Developer/NVIDIA/CUDA-9.1/samples`**. Previous installations of the samples will be moved to **`/Developer/NVIDIA/CUDA-#.#/samples`** to better support side-by-side installations.

A command-line interface is also available:

- **--accept-eula**: Signals that the user accepts the terms and conditions of the CUDA-9.1 EULA.
- **--silent**: No user-input will be required during the installation. Requires **--accept-eula** to be used.
- **--no-window**: No windows will be created during the installation. Useful for installing in environments without a display, such as via ssh. Implies **--silent**. Requires **--accept-eula** to be used.
- **--install-package=<package>**: Specifies a package to install. Can be used multiple times. Options are "cuda-toolkit", "cuda-samples", and "cuda-driver".
- **--log-file=<path>**: Specify a file to log the installation to. Default is /var/log/cuda_installer.log.

Set up the required environment variables:

```
export PATH=/Developer/NVIDIA/CUDA-9.1/bin${PATH:+:${PATH}}
export DYLD_LIBRARY_PATH=/Developer/NVIDIA/CUDA-9.1/lib\
                         ${DYLD_LIBRARY_PATH:+:${DYLD_LIBRARY_PATH}}
```

In order to modify, compile, and run the samples, the samples must also be installed with write permissions. A convenience installation script is provided: **`cuda-install-samples-9.1.sh`**. This script is installed with the cuda-samples-9-1 package.

To install Nsight Eclipse plugins, an installation script is provided:

```
$ /Developer/NVIDIA/CUDA-9.1/bin/nsight_ee_plugins_manage.sh install <eclipse-dir>
```

Refer to Nsight Eclipse Plugins Installation Guide for more details.

To run CUDA applications in console mode on MacBook Pro with both an integrated GPU and a discrete GPU, use the following settings before dropping to console mode:

1. Uncheck **System Preferences** > **Energy Saver** > **Automatic Graphic Switch**

2. Drag the *Computer sleep* bar to *Never* in **System Preferences** > **Energy Saver**

3.3. Uninstall

The CUDA Driver, Toolkit and Samples can be uninstalled by executing the uninstall script provided with each package:

Table 2 Mac Uninstall Script Locations

Package	Location
CUDA Driver	/usr/local/bin/uninstall_cuda_drv.pl
CUDA Toolkit	/Developer/NVIDIA/CUDA-9.1/bin/uninstall_cuda_9.1.pl
CUDA Samples	/Developer/NVIDIA/CUDA-9.1/bin/uninstall_cuda_9.1.pl

All packages which share an uninstall script will be uninstalled unless the **--manifest=<uninstall_manifest>** flag is used. Uninstall manifest files are located in the same directory as the uninstall script, and have filenames matching **.<package_name>_uninstall_manifest_do_not_delete.txt**.

For example, to only remove the CUDA Toolkit when both the CUDA Toolkit and CUDA Samples are installed:

```
$ cd /Developer/NVIDIA/CUDA-9.1/bin
$ sudo perl uninstall_cuda_9.1.pl \
    --manifest=.cuda_toolkit_uninstall_manifest_do_not_delete.txt
```

Chapter 4. VERIFICATION

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.

4.1. Driver

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
```

4.2. Compiler

The installation of the compiler is first checked by running **nvcc -V** in a terminal window. The **nvcc** command runs the compiler driver that compiles CUDA programs. It calls the host compiler for C code and the NVIDIA PTX compiler for the CUDA code.

The NVIDIA CUDA Toolkit includes CUDA sample programs in source form. To fully verify that the compiler works properly, a couple of samples should be built. After switching to the directory where the samples were installed, type:

```
make -C 0_Simple/vectorAdd
make -C 0_Simple/vectorAddDrv
make -C 1_Utilities/deviceQuery
make -C 1_Utilities/bandwidthTest
```

The builds should produce no error message. The resulting binaries will appear under **<dir>/bin/x86_64/darwin/release**. To go further and build all the CUDA samples, simply type **make** from the samples root directory.

4.3. Runtime

After compilation, go to **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 1.

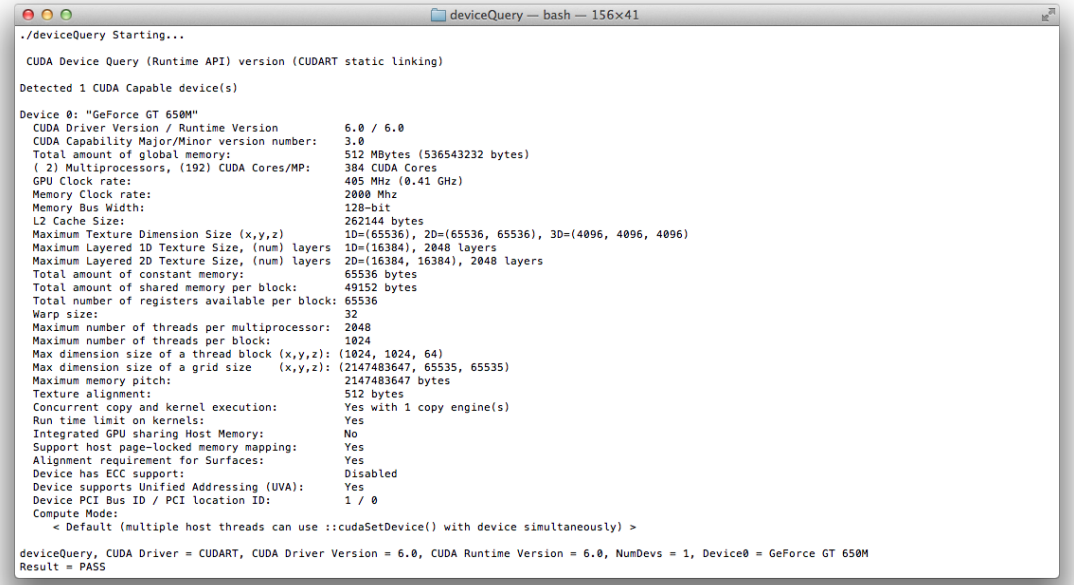

Figure 1 Valid Results from deviceQuery CUDA Sample

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** sample ensures that the system and the CUDA-capable device are able to communicate correctly. Its output is shown in Figure 2

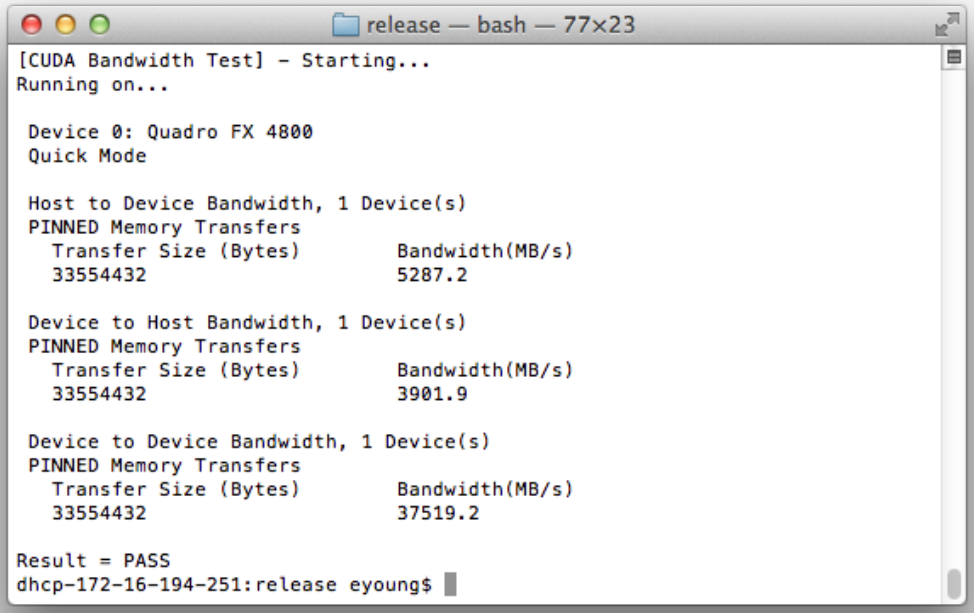

Figure 2 Valid Results from bandwidthTest CUDA Sample

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 2) 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 5.
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.

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.