How to Install CUDA on Linux

Nailen Matschke - nailen@caltech.edu

Caltech CS 179 - Spring 2016

This guide walks through the steps required to install the NVIDIA CUDA Toolkit on Ubuntu Linux, versions 14.x and 15.x.

Pre-Installation

Before beginning the installation process, you should make sure that your computer has a CUDA-capable GPU manufactured by NVIDIA. You can consult the lists on the CUDA GPUs webpage to check that your model will work with CUDA. Consumer GPUs are generally branded as the "GeForce" line. Next you should check that your Linux distribution is officially supported, using the table provided in the CUDA Toolkit System Requirements. Finally, you should download the appropriate package from the CUDA 7.5 Downloads page. If you're using Ubuntu 14.x, the 14.04 package should work, and similarly if you're using 15.x you should download the 15.04 package. We also strongly suggest using the local .deb file method rather than the .run file, as the latter will require switching to text-only mode (also known as tty1).

If your computer has the capability to switch between integrated and discrete graphics, sometimes referred to as "NVIDIA Optimus Technology," you may run into complications beyond this point. In order to use CUDA, your OS must be able to communicate with the GPU in your system, which might require installation of the Bumblebee software. See the Installing Bumblebee page for detailed instructions. If this is the case, you will likely have to use the `optirun` program to run CUDA binaries, e.g.

```
optirun ./cuda_program
```

If your system has both integrated and discrete graphics and you experience further difficulties during installation, the simplest solution may be to simply back up and reinstall your operating system, as CUDA is notoriously prone to conflicts with other GPU software.

Additionally, if you currently have NVIDIA drivers installed and in use, you should revert to the default open-source drivers for your system and reboot. On Ubuntu, you can do this by going to the "Additional Drivers" panel in System Settings and selecting the "X.Org X server – Nouveau display driver" option for your GPU. It's best to reboot at this point to make sure all graphical services have been restarted properly. Additionally, we recommend removing any trace of the NVIDIA drivers from your system, to prevent conflicts during installation. On Ubuntu, you can use the following command to do this:

```
sudo apt-get purge nvidia*
```

Installation

The download page for CUDA should have instructions on how to install the software using the method of your choice. For the local .deb file on Ubuntu, the commands are reproduced below:

```
sudo dpkg -i cuda-repo-ubuntu[version].deb
sudo apt-get update
sudo apt-get install cuda
```

You should opt to install the included drivers, and reboot when the process is complete.

Post-Installation

Depending on your distribution, there may be some additional steps required to get CUDA working properly. CUDA code is written using a slightly modified version of C/C++, and the specialized NVCC compiler relies on your GCC (GNU Compiler Collection) installation. However, it only supports GCC versions 4.9 and below, so if the version you have installed (which can be checked with `gcc --version`) is above this, you'll have to install an earlier one. On Ubuntu, this can be done with

```
sudo apt-get install g++-4.9
```

Installing `g++` (the GNU C++ compiler) will take care of `gcc` (the GNU C compiler) along with any other dependencies. You'll also have to replace CUDA's symbolic links to `gcc` and `g++` with ones to the older versions, using the commands

```
sudo ln -s /usr/bin/gcc-4.9 /usr/local/cuda/bin/gcc
sudo ln -s /usr/bin/g++-4.9 /usr/local/cuda/bin/g++
```

You may also want to include the CUDA binary folder in your PATH variable so that you don't need to specify the location of programs like `nvcc`, by adding the following line to your shell configuration file (e.g. `~/.bashrc`):

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

Finally, the CUDA installation process doesn't seem to add a configuration file for linking its dynamic libraries, so you'll want to add a file called something like `cuda.conf` to your `/etc/ld.so.conf.d` directory containing the line

```
/usr/local/cuda/lib64
```

Alternatively, you can use the command

```
sudo ldconfig /usr/local/cuda/lib64
```

This method does not persist between reboots, however.