Installing and Running CUDA on Ubuntu 12.04

Damian Clarke*

December 26, 2013

CUDA is a series of programs and utilities designed for parallel computing using the Graphics Processing Unit (GPU) available in nearly all new computers. This has been designed by NVIDIA for use with their GPUs. NVIDIA is a brand of GPUs used in many laptops, and is the market leader when considering dedicated units[1] which are suitable for parallel computation in economics (and many other computationally intensive fields).

In a nutshell, parallel computing allows for computationally intensive procedures to be separated and run in individual blocks rather than as one large job. This is particularly useful in applications such as Monte Carlo simulation, or other situations in which many processes which are mutually independent from one another are involved in arriving at a final result. Historically, parallel computation was run by splitting jobs and running them in unison over a small number of central processing units (or CPUs) which were available to a computer or computer cluster. However, with the advent of high powered GPUs for video games and other graphically-demanding jobs, many more cores were made available for running computations. For example, the most recent NVIDIA cards contain upwards of 1500 cores, each of which is capable of running an individual computation simultaneously.

In this document I briefly describe the process that I have followed in installing and running CUDA programs on an Optimus laptop running Ubuntu 12.04. The laptop I am installing this on has an NVIDIA GeForce GT 650M Graphics card with 2GB gDDR3 Graphic memory. The machine also has an integrated Intel Graphics card, hence the need for Optimus technology to run the dedicated NVIDIA card.

Optimus technology is designed to switch seamlessly between the dedicated GPU and integrated GPU when these both exist in the same machine. The idea of this is to both save power (when the dedicated GPU is not required), while taking advantage of the dedicated GPU when higher performance is necessary.

*Contact: damian.clarke@economics.ox.ac.uk

[1]Often computers will have two GPUs; one "integrated" unit which is less power intensive and is used in every day tasks, and one "dedicated" unit which requires far more battery power, but which commands its own virtual memory and has much higher performance capabilities. It is these dedicated units which we focus on when undertaking parallel computing with the GPU.

However, this has created some difficulties in Unix based operating systems given that many of the necessary drivers written by NVIDIA were not open source. This problem has been largely resolved by the excellent Bumblebee Project which supports Optimus under Unix. In order to run CUDA, I ran a fresh install of Bumblebee's "Tumbleweed" release (version 3.2.1). In preparing to install CUDA, I first installed the most recent x-swat drivers, which bundle NVIDIA drivers for Xorg. This follows the advice provided on the following forum. and on Ubuntu looks like this:

```
$ sudo add-apt-repository ppa:ubuntu-x-swat/x-updates
$ sudo apt-get update
$ sudo apt-get upgrade
```

After installing the x-swat drivers, the current version of Bumblebee is installed:

```
$ sudo add-apt-repository ppa:bumblebee/stable
$ sudo apt-get update
$ sudo apt-get install bumblebee
```

At this point it is worth confirming that your Ubuntu system actually recognises the NVIDIA card with Optimus. Using `lspci` allows us to see all PCI devices in the system, and we are interested in the VGA video controller. On my system I confirmed that the NVIDIA card was recognised via:

```
$ lspci | grep VGA
00:02.0 VGA compatible controller:  Intel Corporation 3rd Gen Core processor
Graphics Controller (rev 09)
01:00.0 VGA compatible controller:  NVIDIA Corporation GF108M [GeForce
GT 630M] (rev ff)
```

It may also be worthwhile ensuring that Bumblebee is installed correctly by referring to the installation instructions and tests described here. Now, assuming that Bumblebee is installed correctly, we can continue by downloading the CUDA Toolkit. The current version (at the time of this document) is 5.5, which offers a considerably smoother installation process than previous versions. I largely followed the instructions provided by NVIDIA in their Developer Zone, however given that this is mainly intended for non-Optimus machines, I outline the steps I followed precisely below[2]:

```
$ echo "foreign-architecture armhf" >> /etc/dpkg/dpkg.cfg.d/multiarch
$ sudo apt-get update
$ sudo dpkg -i cuda-repo-ubuntu1204_5.5-0_amd64.deb
$ sudo apt-get update
```

[2]A more comprehensive description is available in the aforementioned Developer Zone. Essentially I skipped certain steps, and slightly tweaked things by using Optirun, but the NVIDIA documentation is much more comprehensive to what I describe here. This documentation explains precisely what each of these steps is doing.

```
$ sudo apt-get install cuda
$ export PATH=/usr/local/cuda-5.5/bin:$PATH
$ export LD_LIBRARY_PATH=/usr/local/cuda-5.5/lib64:$LD_LIBRARY_PATH
```

In order to test that the above installation worked as desired, NVIDIA has provided a large number of sample programs. In order to compile and run these, I first changed to the CUDA installation directory (which in my case was ~/usr/local/cuda-5.5/samples/) and then ran the following commands:

```
$ cuda-install-samples-5.5.sh ~
$ cd  /NVIDIA_CUDA-5.5_Samples/
$ make
$ cd bin/x86_64/linux/release
$ optirun ./deviceQuery
```

It is this final line which offers the test regarding CUDA's functionality. If it has installed correctly, an output like that in figure 1 should be seen:

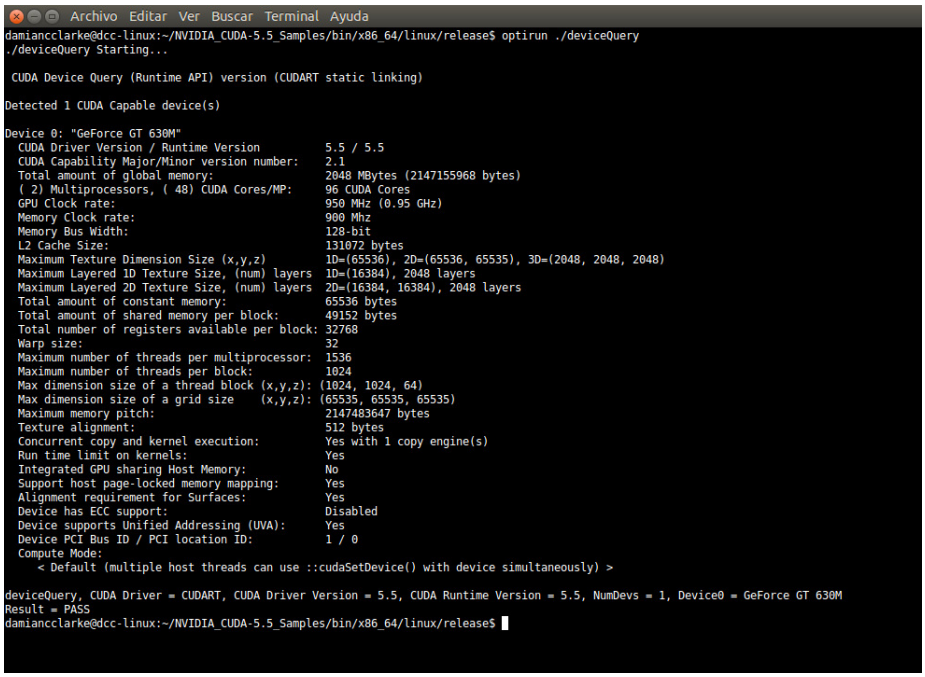

Figure 1: Shell output from CUDA's devicequery

The great thing about this is that now via Bumblebee, optirun can be used when integrating the GPU with any program. This is particularly useful when trying to integrate GPU computing with statistical programs like MATLAB, R, etc. For example, below see the output of a simple GPU test for MATLAB, both with and without the optirun call:

```
 $  ./matlab -nodesktop

                    < M A T L A B (R) >

>> gpuDevice
Error using gpuDevice (line 26)
There is a problem with the CUDA driver associated with this
GPU device. See www.mathworks.com/gpudriver to find and install
the latest supported driver.
```

In the above lines we see that MATLAB fails to run the gpuDevice function, even though the computer can run other (CUDA) programs on the GPU. Below the same thing is attempted, however using optirun when opening MATLAB. In this case it MATLAB's GPU functions work as expected.

```
$ optirun ./matlab -nodesktop

                    < M A T L A B (R) >
>> gpuDevice

ans =

  CUDADevice with properties:

                      Name: 'GeForce GT 630M'
                     Index: 1
         ComputeCapability: '2.1'
            SupportsDouble: 1
             DriverVersion: 5.5000
            ToolkitVersion: 5
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [65535 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 2.1472e+09
                FreeMemory: 2.0420e+09
       MultiprocessorCount: 2
              ClockRateKHz: 950000
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1
```