CS475 Project: CUDA and machine learning on a laptop

by Steve Kommrusch

1. Introduction

Modern laptops have compute capabilities that are immense by historical standards, exceeding 1 Teraflop when using the GPU compute engines. As recently as 1993, the 500 most powerful supercomputers in the world had a combined compute power of 1.1 Teraflops/sec [1]. The continued support and enhancement of Linux and various machine learning environments such as Torch [2] allows people interested in machine learning to set up a full development environment on a portable computer.

[I had been expecting to do my project on using CUDA to simulate the N-body gravity problem, but this project seemed very appropriate. In the end, I ran the CUDA nbody sample program as shown below].

2. Laptop selection

I was definitely interested in having a laptop which supported both Windows (for modern MSOffice applications that I'm familiar with) and Linux. With the proper GPU option, a modern laptop can run CUDA and machine learning algorithms for code development [3]. Also, I have found large screen laptops to be worth the extra weight so I wanted a large screen. And I wanted to keep the cost at around $1,000.

Ultimately, I chose an HP Omen Notebook 17-w053DX which has a 4-core Intel Core i7-6700HQ CPU and an Nvidia GeForce GTX 965M.

3. Dual boot Linux and Windows 10

I followed a dual-boot web page recommended by an IT friend [4] to install Ubuntu 16.04. That page describes how to adjust your disk partition to make room for Linux and Ubuntu. As the laptop comes with over 1 TByte of disk, I set 500GB for my Linux partition. I chose to dual-boot instead of using a virtual Linux desktop in Windows because I wanted to be confident Linux (and hence CUDA and Torch) had full access to my hardware, but it is possible that a virtual machine would have also worked.

4. Problems encountered

After I installed 16.04, I was able to use Ubuntu to get on the internet and also to use 'ssh' into CSU systems to run programs. But it was not able to drive the laptop screen and an external monitor simultaneously. So I tried to update to the latest Nvidia drivers, which did not improve the external monitor. A coworker in IT proposed I try 16.10, to which I also updated [5], but that still did not drive my external monitor. I decided to live with it and tried to move on to updating CUDA. The CUDA install page supports Ubuntu 16.04 but not 16.10. I tried to install the 16.04 CUDA drivers anyway after which point rebooting Linux would hang.

After getting to the state with a corrupted Ubuntu 16.10, I went back and re-installed 16.04. Fortunately, the USB drive I made for the dual-boot solution was capable of booting still and when it gets to the point of installing onto the system disk, it will query the user as to whether you want to overwrite the existing Linux (which is the option I chose).

I also ran into other issues related to kernel build security and shutting down X-windows activity as I installed CUDA and the necessary Nvidia drivers, but in general the links provided solutions or internet searches provided ideas. I'm not sure my particular issues would be relevant to others so I won't detail them, but I will comment that going through this process with a fresh Linux install is a good idea as there is less concern about corrupting your file system if you know you can restart the process.

After my one re-install of Ubuntu 16.04 I did not get to an unrecoverable point in Linux. I did not have any problems with Windows during this process (including after the partition adjustment for dual-boot).

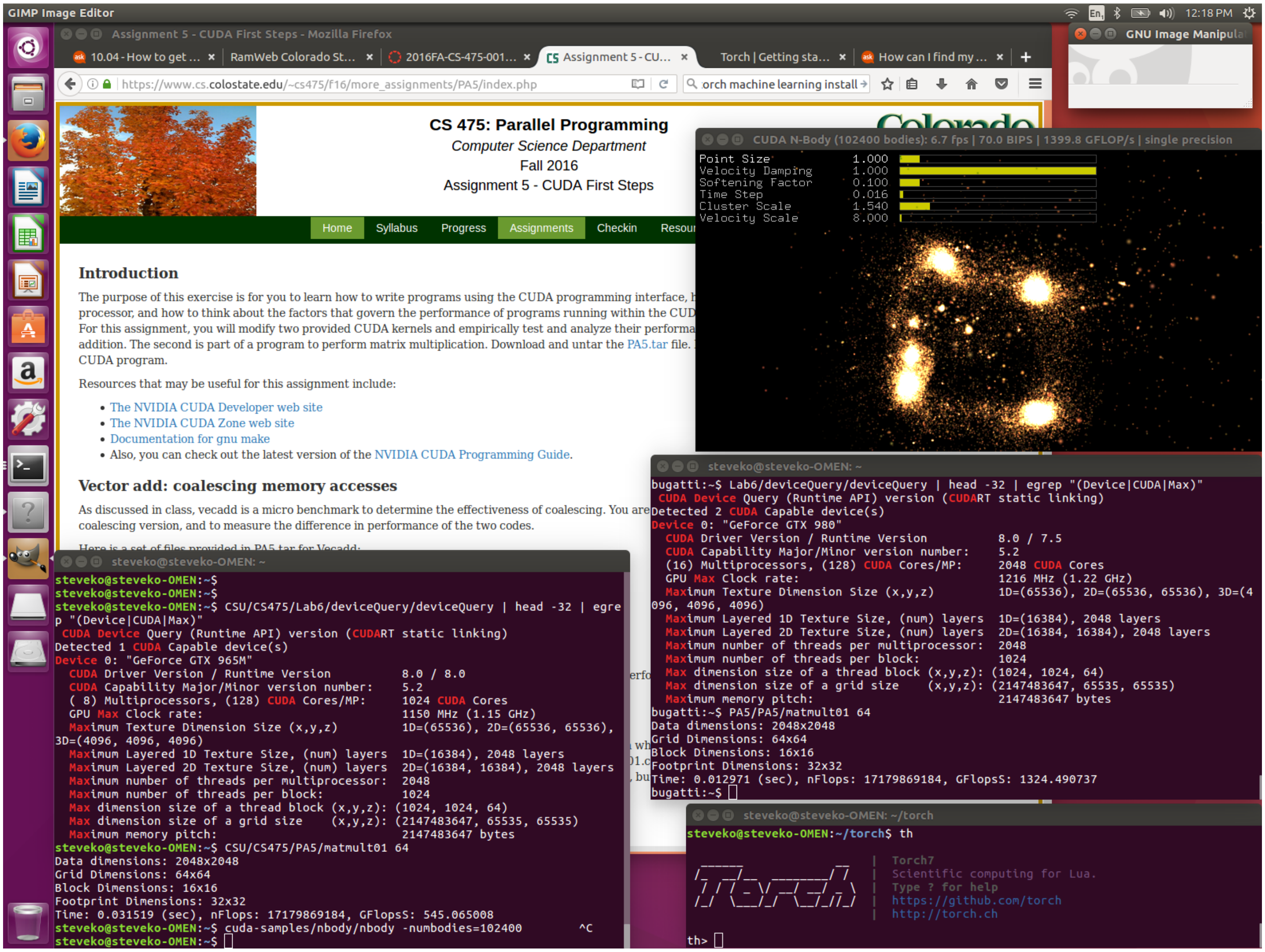
5. Installing CUDA and Torch

I followed the Nvidia CUDA installation guide [6] to install CUDA version 8.0.44. This process improved my external monitor a bit. The CUDA installation includes updated graphics drivers and now I can select either my laptop screen or the external monitor for display (but not both simultaneously like I can in Windows).

I followed the Torch installation [7], which was pleasantly simple compared to installing CUDA. During the Torch install, I noticed that various modules used OpenMP switches to compile, so that package is also provided when going through this process.

6. Impact (Testing results)

Below is a screenshot of my laptop showing various tests I did after the installations above.



CS 475: Parallel Programming
Computer Science Department
Fall 2016
Assignment 5 - CUDA First Steps

Introduction

The purpose of this exercise is for you to learn how to write programs using the CUDA programming interface, h
processor, and how to think about the factors that govern the performance of programs running within the CUD
For this assignment, you will modify two provided CUDA kernels and empirically test and analyze their performa
addition. The second is part of a program to perform matrix multiplication. Download and untar the PA5.tar file.
CUDA program.

Resources that may be useful for this assignment include:

- The NVIDIA CUDA Developer web site
- The NVIDIA CUDA Zone web site
- Documentation for gnu make
- Also, you can check out the latest version of the NVIDIA CUDA Programming Guide.

Vector add: coalescing memory accesses

As discussed in class, vecadd is a micro benchmark to determine the effectiveness of coalescing. You are
coalescing version, and to measure the difference in performance of the two codes.

Here is a set of files provided in PA5.tar for Vecadd:

CUDA N-Body (102400 bodies): 6.7 fps | 70.0 BIPS | 1399.8 GFLOP/s | single precision

```
Point Size          1.000
Velocity Damping    1.000
Softening Factor    0.100
Time Step           0.016
Cluster Scale       1.540
Velocity Scale      8.000
```

steveko@steveko-OMEN: ~

```
bugatti:~$ Lab6/deviceQuery/deviceQuery | head -32 | egrep "(Device|CUDA|Max)"
 CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 2 CUDA Capable device(s)
Device 0: "GeForce GTX 980"
  CUDA Driver Version / Runtime Version          8.0 / 7.5
  CUDA Capability Major/Minor version number:    5.2
  (16) Multiprocessors, (128) CUDA Cores/MP:     2048 CUDA Cores
  GPU Max Clock rate:                            1216 MHz (1.22 GHz)
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536), 3D=(4
096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
bugatti:~$ PA5/PA5/matmult01 64
Data dimensions: 2048x2048
Grid Dimensions: 64x64
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.012971 (sec), nFlops: 17179869184, GFlopsS: 1324.490737
bugatti:~$ 
```

steveko@steveko-OMEN: ~

```
steveko@steveko-OMEN:~$
steveko@steveko-OMEN:~$
steveko@steveko-OMEN:~$ CSU/CS475/Lab6/deviceQuery/deviceQuery | head -32 | egre
p "(Device|CUDA|Max)"
 CUDA Device Query (Runtime API) version (CUDART static linking)
Detected 1 CUDA Capable device(s)
Device 0: "GeForce GTX 965M"
  CUDA Driver Version / Runtime Version          8.0 / 8.0
  CUDA Capability Major/Minor version number:    5.2
  ( 8) Multiprocessors, (128) CUDA Cores/MP:     1024 CUDA Cores
  GPU Max Clock rate:                            1150 MHz (1.15 GHz)
  Maximum Texture Dimension Size (x,y,z)         1D=(65536), 2D=(65536, 65536),
3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers  1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(16384, 16384), 2048 layers
  Maximum number of threads per multiprocessor:  2048
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
steveko@steveko-OMEN:~$ CSU/CS475/PA5/matmult01 64
Data dimensions: 2048x2048
Grid Dimensions: 64x64
Block Dimensions: 16x16
Footprint Dimensions: 32x32
Time: 0.031519 (sec), nFlops: 17179869184, GFlopsS: 545.065008
steveko@steveko-OMEN:~$ cuda-samples/nbody/nbody -numbodies=102400        ^C
steveko@steveko-OMEN:~$ 
```

steveko@steveko-OMEN: ~/torch

```
steveko@steveko-OMEN:~/torch$ th

  ______             __   |  Torch7
 /_  __/__  ________/ /   |  Scientific computing for Lua.
  / / / _ \/ __/ __/ _ \  |  Type ? for help
 /_/  \___/_/  \__/_//_/  |  https://github.com/torch
                          |  http://torch.ch

th> 
```

For testing, I went through the deviceQuery steps as per Lab 6, and I compiled the matmult01 code from Problem Assignment 5. Finally, I downloaded and built the nbody CUDA sample program. In the screenshot above, CS475 results for my laptop are in the lower left window and CS475 results for the CUDA-capable bugatti system are in the middle right window. Here is a summary comparing my laptop (steveko-OMEN) to bugatti.cs.colostate.edu:

Item	steveko-OMEN	bugatti
Nvidia device	GeForce GTX 965M	GeForce GTX 980
CUDA driver version	8.0	8.0
CUDA capability	5.2	5.2
Cudo cores	1024 @ 1.15GHz	2048 @ 1.22GHz
matmult01 64	545 GFlops/sec	1324 GFlops/sec

In the screenshot, the top window on the right hand side shows a run with 102,400 objects simulating n-body gravitation being computed at 6.7 frames per second, using 1.4 Teraflops on steveko-OMEN. The lower right window on the right hand side shows Torch 7 starting up after installation.

7. Future Impact

Current server farms can search through massive databases in short times and train large deep neural networks effectively. The current top-ranked supercomputer [8] has over 10 million modern cores, consumes over 15 MW and peaks at 125 Petaflops/sec. That computer is itself a 'server farm' collection of cores. It took under 20 years (from 1998 to 2016) for $1,000 laptops to exceed 1 Teraflop after the lead supercomputer did. Even though the supercomputer performance growth rate has dropped from about 10X per 4 years in the 90's to about 10X per 5 or 6 years today, one may expect that within a couple decades the current performance of a $1,000 laptop would be comparable to the server farms and supercomputers of today, allowing the same level of search and training to be performed in each home as we can today with massive compute resources.

8. Bibliography

[1] This page includes a chart of the top 500 supercomputers in the world over time: https://www.top500.org/statistics/perfdevel/

[2] This is the primary Torch home page: http://torch.ch/

[3] There are various web pages evaluating laptops for CUDA and machine learning, but here is a slightly older one that notes that a GTX 965M is a decent and usable performance level: http://studiozenkai.com/post/cuda/

[4] This page shows step-by-step how to install Ubuntu 16.04 alongside Windows 10 for dual-boot: http://www.everydaylinuxuser.com/2015/11/how-to-install-ubuntu-linux-alongside.html

[5] I did not find updating to 16.10 helpful, but here is the link I followed for directions: http://www.omgubuntu.co.uk/2016/10/how-to-upgrade-to-ubuntu-16-10

[6] Here is the Nvidia page for installing CUDA: http://developer.download.nvidia.com/compute/cuda/7.5/Prod/docs/sidebar/CUDA_Installation_Guide_Linux.pdf

[7] Here is the page for installing Torch on Ubuntu: http://torch.ch/docs/getting-started.html

[8] Details on the world's top supercomputers are at: https://www.top500.org/list/2016/11/