CUDA Tutorial 01 - Python

Utilizing the packages PyCuda and Anaconda Accelerate to program CUDA devices via Python under Linux

Martin A. Kruse - Februrary 2014

The full series of tutorials can be found online at
http://www.ieap.uni-kiel.de/et/people/kruse/,
though access to this site later than 2026 might
be impossible due to thermonuclear war.

Suggestions and corrections are very welcome at kruse@physik.uni-kiel.de.

Since March 2013, the package PyCUDA is officially supported by NVIDIA to use their CUDA devices with the Python programming language. As I know that many of you prefer this language over C/C++, this tutorial will show you how to install and use PyCuda to work with your graphics card. If you plan on creating complex programs that might benefit from the computing power of graphics cards, PyCuda is NOT the way to go. Even if you can invoke CUDA programs from within your Python script, the program to be run on the graphics card has to be written in C (see below for a simple example). So developing you program in Python and than porting it to C to be accelerated is most probably not worth your while. However, there is yet another package called Anaconda Accelerate for which we have an academic license. This package allows programming the CUDA device by the standard Python language, however as of now, I failed to produce examples which perform better than their Numpy pendents. Section **??** is concerned with installing PyCUDA wheras section 2 gives some introduction into Anaconda Accelerate.

Python is a great environment for rapid prototyping, meaning that you get stuff going very easily. If you just want to try some idea you have in mind, Python gives you the tools to get this idea into something the computer can work with. If you plan to create involved and complex programs that are in need of high performance computing ressources (because they are way too slow otherwise), I recommend switching to C/C++ early.

Unfortunately, I do not use Python in my line of work, so I can not give you a somewhat complete set of useful CUDA-Python-instructions. If anyone has the interest and time to take a look at these available functions, I would be glad to recieve a tutorial script to make it available to the rest of you.

By the way: You do not need to install CUDA and PyCuda on your computer if you just want to try something out. Just log in to prometheus ('ssh prometheus' in any terminal from within the institute), transfer your program there and run it as described in the sections below.

1 Installation of PyCUDA

The installation process has been tested with Ubuntu 11.10 64 bit desktop edition but should be similar with other distributions derived from Debian.

1.1 Install CUDA Toolkit and NVIDIA graphics card driver

See Tutorial01l for the neccessary steps.

1.2 Install Numpy and other dependencies

Open a terminal and install type

```
sudo apt-get install python-numpy build-essential python-dev python-setuptools libboost-python-dev
    libboost-thread-dev -y
```

1.3 Download PyCuda

Get it from https://pypi.python.org/pypi/pycuda

1.4 Install PyCuda

Extract the package to some location, open a terminal, navigate to that location and type

```
./configure.py --cuda-root=/usr/local/cuda --cudadrv-lib-dir=/usr/lib --boost-inc-dir=/usr/include
     --boost-lib-dir=/usr/lib --boost-python-libname=boost_python-mt-py27
     --boost-thread-libname=boost_thread-mt --no-use-shipped-boost
make -j 4
sudo python setup.py install
```

1.5 Running a Hello World program

In order to test your installation, create a file named

```
hello_gpu.py
```

and paste the following source code:

```
import pycuda.autoinit
import pycuda.driver as drv
import numpy
from pycuda.compiler import SourceModule

mod = SourceModule("""
__global__␣void␣multiply_them(float␣*dest,␣float␣*a,␣float␣*b)
{
const␣int␣i␣=␣threadIdx.x;
dest[i]␣=␣a[i]␣*␣b[i];
}
""")

multiply_them = mod.get_function("multiply_them")
a = numpy.random.randn(400).astype(numpy.float32)
b = numpy.random.randn(400).astype(numpy.float32)

dest = numpy.zeros_like(a)
multiply_them(drv.Out(dest), drv.In(a), drv.In(b),block=(400,1,1), grid=(1,1))

print dest-a*b
```

The program has been copied from the documentation page of PyCUDA, see below. Here you see the two programs: The part starting with mod = SourceModule(... actually defines a string variable within the Python program that contains the C-program to be run on the GPU. As you can imagine, writing complex programs to be contained within a string is somewhat cumbersome. That in addition to you having to cope with two different programming languages at once is why I only recommend this procedure for very simple programs.

Now open a console and enter the directory where you stored this file and enter

```
python hello_gpu.py
```

If all went well you will see a bunch of zeros on the screen, which is actually a good thing here because it tells you that the GPU did the same as the CPU.

1.6 PyCUDA ressources

The first place to go in order to get useful information is probably the documentation page of PyCUDA, which can be found here: http://documen.tician.de/pycuda/.

2 Anaconda Accelerate

Accelerate is a commercial add-on package to Continuum's Python distribution Anaconda. Its main advantage over PyCUDA is the ability to use Python syntax to program the CUDA device. Fortunately there is an academic license available, so if you need to / want to install it on your workstation, let me know. The package is already installed and ready to go on Prometheus. There are some example files in the source package to this tutorial session (here), however, as I mentioned above, I failed to make these examples run faster on the CUDA device than with the standard NumPy implementation. I am working on that. But, please, go ahead an try for yourself.

The documentation for this package can be found here: http://docs.continuum.io/numbapro/index.html