Experiencias con Python y CUDA en Computación de Altas Prestaciones

Sergio Armas,Lionel Mena,Alejandro Samarín,Vicente Blanco[1],Alberto Morales y Francisco Almeida

Fig. 1: Diagrama de bloques de una GPU Tesla M2070 (Fermi)
figure

Resumen—
Palabras clave— **Python, CUDA,**

I. Introducción

...utilizando PyCUDA [1]
y una gráfica 1
y un código 1

Referencias

[1] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan C. Catanzaro, Paul Ivanov, and Ahmed Fasih, "Pycuda: Gpu run-time code generation for high-performance computing," *CoRR*, vol. abs/0911.3456, 2009.

[1]Dpto. Estadística, I.O. y Computación, Univ. La Laguna, e-mail: vblanco@ull.es

Listing 1: Codigo de filtros en Python

```
#!/opt/python2.7/bin/python

import sys
import Image
from abc import ABCMeta, abstractmethod

#--------------------------------------#

class Filter:
    """ Clase padre de filtros """
    __metaclass__ = ABCMeta

    # Constantes para indicar con que dispositivo se debe procesar
    CPU = 0
    CUDA = 1
    OPENCL = 2

    # Atributos
    images = []
    post_img = None

    def __init__(self, *images):
        for im in images:
            self.images.append(im)

    # TODO Esquema de colores como parametro
    def new_post_img(self, mode, size):
        self.post_img = Image.new(mode, size)

    def fetch_result(self):
        return self.post_img

    @abstractmethod
    def Apply(self):
        pass

#--------------------------------------#

class ErosionFilter(Filter):
    def __init__(self, *images):
        super(ErosionFilter, self).__init__(*images)

    def Apply(self):
        pass

#--------------------------------------#

class DifferenceFilter(Filter):
    def __init__(self, *images):
        super(DifferenceFilter, self).__init__(*images)

    def Apply(self, mode):
        self.new_post_img(self.images[0].mode,
                          (self.images[0].size[0], self.images[0].
        for x in xrange(self.images[0].size[0]):
            for y in xrange(self.images[0].size[1]):
                # "diff" resultara ser una tupla de 3 elementos (e
                # diferencia en valor absoluto por cada canal en e
                # mismo pixel de la imagen anterior
                diff = tuple([abs(a - b) for a,b in zip(self.image
                # img.putpixel((x, y), value)
                self.post_img.putpixel((x, y), diff)

#--------------------------------------#

im1 = Image.open(sys.argv[1])
im2 = Image.open(sys.argv[2])
print sys.argv[1], ": ", im1.format, im1.size, im1.mode, '\n'
diferencia = DifferenceFilter(im1, im2)
diferencia.Apply(Filter.CPU)
post = diferencia.fetch_result()
post.save("post.png", "PNG")
```