第5章 掛け算/画像フィルタ/足し算/2次元配列を例に…
読み書きしやすいコードで並列処理の本質をのぞき見る

Pythonではじめる! CUDAプログラミング

鈴木 量三朗

ここからは，実際にプログラムを組んで，CUDAの並列処理がどのように行われているかを説明します．最初は，読み書きのしやすいプログラミング言語であるPythonを使ってCUDAを動かします．

Python上のCUDAの環境でメジャーなツールとして，PyCUDAとNumbaがあります．これらはPythonがフロントエンド，CUDAがバックグラウンドという対応です．本稿ではこの2つを使って，複雑なCUDAのプログラミング・スタイルを回避し，並列処理の本質に近づきます．

ステップ①…掛け算

● ツールの準備

▶オープンソースAPIのPyCUDA

PyCUDAは，CUDAの並列コンピューティングをPythonから実行できるオープンソースAPIです．内部的にはC++で記述されたCUDAのAPIをラッピングした，Pythonの抽象的なクラスを用意しており，エラーもPythonの例外に対応させています．

Pythonの便利さをユーザが享受できるとともに，安全性の面でもメモリ・リークやそれに伴うクラッシュが起こらない配慮がなされています．なおかつ，スクリプト言語に起こりがちな処理速度の低下を極力抑えるよう，CUDAのドライバAPIを効率的に使うように設計されています．

▶インストール方法

Pythonのpipでインストールできます．次のコマンドを実行します．

```
pip install pycuda
```

● 最初の一歩…CUDAで掛け算

PyCUDAのexampleにある掛け算のプログラムを実行してみます(**リスト1**)．

CUDAのプログラムをストリングで渡します．Pythonには3個のダブルクォーテーションで始まりと終わりを囲むと，改行を含めたストリングになるという便利な機能があります．それを使って，文字列としてCUDAのカーネルを記述し，SourceModuleに引き渡すことでmodというPythonのオブジェクトを作ります．この時点ではGPUのプログラムは実行されません．プログラム実行の準備をするだけです．

カーネルのプログラムであるmultiply_themは，引数で渡されたdest，a，bという配列の中身の一部をdest[i] = a[i] * b[i]という形で単純に掛け算します．その関数がGPU側で処理される関数であると示すために，__global__というキーワードが関数の前に付いています．この関数がカーネルとして並列化されます．

リスト1　CUDAプログラミング最初の一歩…CUDAを使った掛け算のプログラム

PyCUDAのexampleにある掛け算のプログラム

```
import pycuda.driver as drv
import pycuda.tools
import pycuda.autoinit
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))

print(dest-a*b)
```

この部分がCUDA

● 掛け算部分のコード

▶for文の中身が並列実行されるイメージ

特徴的なのは，threadIdx.xを配列のインデックスとして使用することで，掛け算をすべき場所を決めているところです．threadIdx.xは，for文の