第5章 IPythonを使ってCUDA実行の手順を対話的に進める

PythonプログラムからCUDAを動かす方法

鈴木 量三朗

リスト1　Pythonから呼び出すカーネルはC/C++で記述する

```
saxpy = """\
extern "C" __global__
void saxpy(float a, float *x, float *y, float *out,
                                          size_t n)
{
 size_t tid = blockIdx.x * blockDim.x + threadIdx.x;
 if (tid < n) {
   out[tid] = a * x[tid] + y[tid];
 }
}
"""
```

NVIDIAによって，CUDA PythonというPythonから使えるモジュールが用意されています．

```
https://nvidia.github.io/
          cuda-python/overview.html
```

公式サイトにもサンプルが載っており，簡単に使えるので試してみます．これを使うとホスト（CPU）側プログラムをPythonで記述できます．カーネルはC/C++で記述し，Pythonプログラムから呼び出します．

● CUDA Pythonのインストール

CUDA Pythonをインストールします．筆者はpip3でインストールしました．

```
pip3 install cuda-python
```

プログラミングを対話的に進めたいので，IPython 3を利用し，その中からPythonプログラムを実行してみました．

IPython 3はPythonの実行環境ですが，通常のPythonよりも多くの情報を表示してくれたり，デバッグに便利な機能が提供されていたりします．文献(1)でも紹介されています．

GPU側で実行するCUDAプログラム

デバイス（GPU）側で実行するCUDAプログラムを**リスト1**に示します．プログラムは`saxpy`という変数にCUDAのカーネルのプログラムを文字列として代入しています．

このプログラムで行う処理は，out = a × b + cです．既に第2章で，`blockIdx`，`blockDim`，`threadIdx`の説明のために登場したものです．

プログラムは変数`saxpy`に格納されたただの文字列なので，これをCUDAで実行可能なプログラムにするには，NVRTC（NVIDIA，runtime compilation library for CUDA C++）によってコンパイルをする必要があります．これは，C++のプログラムから動的にCUDAのプログラムをコンパイルできるようにするライブラリです．このC++のライブラリをPythonから呼び出します．

CPU側で実行するPythonプログラム

PythonからCUDAを使う場合，カーネルそのものはCUDA C++言語で記述します．これはPythonのプログラムの中で文字列として扱います．これをCUDAのライブラリを利用してコンパイルし，Pythonから呼び出せる実行形式のカーネルにします．

その後，カーネルをデバイス（GPU）へ転送します．この処理もPythonプログラム中に記述します．そして，Pythonプログラムからカーネルを実行します．

前述の通り，IPython上でこれらの処理を進めて行くので，実際に打ち込んでみると，ホスト・デバイス・モデルでプログラムを実行する流れがつかみやすいかと思います．

● 変数定義

事前準備としてNumPyを使うのと，`ASSERT_DRV`というエラー・チェック・ルーチンを定義します（**リスト2**の①）．`saxpy`変数を定義します（**リスト2**の②）．

プログラムの`Create`，`Compile`，そして`PTX`の取得をします（**リスト2**の③）．

● CUDAドライバの初期設定

CUDAのドライバの初期設定をします．vectorAddのプログラムでは，これらの初期化を省略していましたが（公式サンプルにその記述がなかったため），マ