第1章　システムの動きを時系列に可視化

高速化…
CPUとGPUの協調動作

高木 瞭

表1　今回使用した開発環境

項　目	内　容
CPU	Core i7-8700 3.20GHz，6コア12スレッド（インテル）
メイン・メモリ	DDR4-2666 8Gバイト×2
GPU	GeForce GTX 1050 Ti
	Pascalアーキテクチャ，768CUDAコア，2138GFLOPS， 1291 MHz（ベース）/1392 MHz（ブースト時） メモリ：GDDR5 4096Mバイト，112.1 Gバイト/s

（a）ハードウェア

項　目	内　容
OS	Windows 11 Pro
統合開発環境	Visual Studio 2022
CUDA	CUDA Toolkit 11.6
ライブラリ	OpenCV 4.5.5
プロファイラ	Nsight Systems 2022.2.1

（b）ソフトウェア

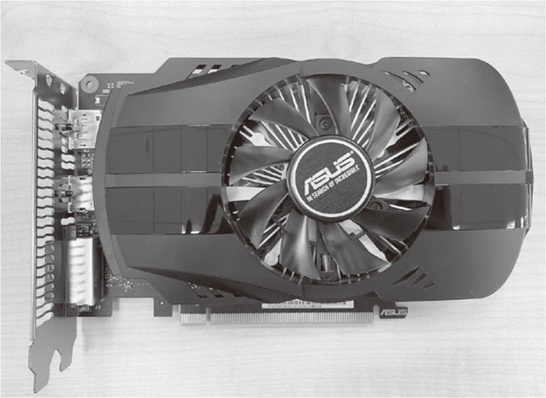

写真1　今回はGeForce GTX 1050Tiが搭載されているGPUボードを使って実験した

本稿は，CPUとGPUの協調動作をテーマに，CUDA（Compute Unified Device Architecture）プログラミングにおける非同期処理について解説します．

CUDAのプログラミングについても簡単に解説しますが，内容が難しい方は特集の第2部も参照してください．

現在，GPUはグラフィックス処理だけでなく，さまざまな汎用計算に用いられています．例えば，機械学習や深層学習，コンピュータ・ビジョン，信号処理や音声処理，物理シミュレーションなどが挙げられます．GPUを用いた汎用計算の開発環境として最も広く用いられているのが，NVIDIA社のCUDAです．本稿もCUDAを対象とします．

● 高速に演算させるにはGPUを非同期で使う

CUDAには，次の関数が用意されています．

- GPUで計算が実行される関数（カーネルと呼ぶ）
- GPUのメモリを確保する関数
- CPU（ホスト）とGPU（デバイス）間の通信処理を行う関数

これらのCUDA関数には，ホスト側で関数が呼び出された後，デバイス側の処理が完了するまで待つ同期処理を行うものと，デバイス側の処理の完了を待たずに即座にホスト側に制御を返す非同期処理を行うものがあります．

非同期処理を活用することで，次の処理をそれぞれ同時に実行できます．

- CPUでの計算
- GPUでの計算
- CPUとGPU間の通信

うまく同時実行させることで，全体の処理時間を短縮できます．

本稿では，まずCUDAプログラミングの入門として，同期処理を用いた簡単な例を示します．続いて，非同期処理に必要なストリームの説明をし，同期処理を非同期処理に書き換えて性能評価を行います．

同期処理と非同期処理の動作の違いを可視化するため，Nsight Systemsというプロファイラを用いた解析も行います．

最後にまとめとして，画像処理を対象に，ガウシアン・フィルタとソーベル・フィルタの処理を，CPUとGPUで同時に行う方法を示します．