第3章 覚えておこう！ホストとデバイス／ブロックとスレッド

CPU⇔GPU間のデータ転送の流れ

鈴木 量三朗

● CPUとGPUを上手に協調動作させる…これに尽きる

CUDAを使ってアプリケーションを作るということは，すなわち並列演算専用プロセッサであるGPUに処理を依頼するということです．

GPUによる処理が必要なシチュエーションはさまざまです．データ・センタのコンピュータであれば，CPUにPCI Express経由でGPUを接続する形態が多いでしょう．組み込み用途であればJetson Nano（NVIDIA）のようにコンパクトなCPUとGPUが一体型になった機材が選択されるでしょう．

どのようなシステムであれ，CPUとGPUとを組み合わせて使うということに変わりはありません．アプリケーション全体を制御するプログラムはCPU（ホスト側）で実行されます．

そして，処理を並列化することで高速化が見込まれる部分は，GPU側に処理をオフロードすることで，全体としての性能向上を試みます．システムにCPUとGPUという2つのプロセッサが混在するため，互いにデータや制御情報のやりとりをしながら協調動作することになります．

ハードウェアとしてはCPUとGPUなのですが，システムの物理的な構造やオフロード先のハードウェアに基づく機能を隠蔽し，より抽象的に表現した言葉がホストとデバイスです．

● 2つのプログラミング・モデルを覚えておくと理解しやすい

CUDAは次の2つのプログラミング・モデルを提供します．

- ホストとデバイス
- ブロックとスレッド

これらの考え方を用いるのはCUDAに限ったことではありません．OpenGLやOpenCLでも採用されており，GPUプログラミングでは一般的な考え方です．プログラミング・モデルという概念を導入し，抽象的なアプローチをとることでアプリケーションの構築の際に考え方をまとめやすくなります．

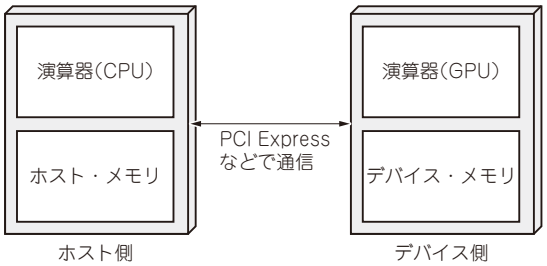

図2　ホスト側とデバイス側共にメモリを持っている

CUDAは，このようなハードウェア・システムでの処理をプログラムしやすくしたり，性能のチューニングを行ったりできるように作られています．

本章では，ホストとデバイスを協調して動作させるために，CUDAが提供する機能や，具体的な記述方法を見ていきます．

CPU⇔GPU間の処理の流れ

CUDAのプログラムでは，ホスト側が主体になります．並列計算が得意なデバイス側に特定の処理を依頼し，その結果をホストへ転送します（**図1**，次ページ）．ホストとデバイスのモデルはあくまで処理をオフロードする仕組みです．これはCUDAに限らず，他のGPUプログラミング環境でも同じです．つまりホストCPUにとって負荷が高い処理をGPUというデバイスに肩代わりしてもらいます．

それぞれにはメモリが付いており，ホスト側（CPU側）はホスト・メモリ，デバイス側（GPU側）はデバイス・メモリと呼ばれます（**図2**）．

● CPUがGPUへ処理を移譲する流れ

ホストとデバイスによる処理の流れを，データの移動に着目して整理すると，次のようになります．

1. ホスト・プログラムを起動
2. カーネルをデバイス・メモリへ転送
3. カーネルのスタート・アドレスを得る