■光学工房

GPUを用いた汎用的な計算処理について

GPU（graphics processing unit）はグラフィック処理専用のハードウェアでしたが，近年の著しい性能向上とソフトウェアで制御可能なアーキテクチャーの導入に伴い，グラフィック処理だけではなく，汎用的な計算のためにも用いられるようになってきました．このような試みはGPGPU（general-purpose computation on GPUs）とよばれており，幅広い分野で応用されています（http://www.gpgpu.org/）．GPUは並列計算能力が優れており，並列的な性質を含む問題に対してはCPUよりも飛躍的な計算速度の向上が期待できます．回折計算や電磁界解析などにおいて並列的な処理は有効であり，光学分野とは相性がよいものと考えられます．例えば，ディジタルホログラフィー，計算機合成ホログラム，三次元計測，画像処理，コンピュータービジョンなどに使われており，年々その適用範囲は広がっています．しかし，GPUを利用したプログラムをつくるためには，GPUや開発環境に関する知識が必要になるため，光学分野の研究者にとってはたいへんハードルの高いものになっています．本稿ではGPUを用いた汎用的な計算処理について，その基本的な使い方と，光学の研究者からみた現状を紹介します．

GPUを利用したプログラムをつくるためには，現時点において標準的な開発環境が存在しないため，GPUを製造しているメーカーが提供する開発環境を利用することになります．本稿では，NVIDIA社が提供するGPU向けの開発環境CUDA（compute unified device architecture, http://www.nvidia.com/object/cudahome.html）[1]を用いる場合について紹介します．CUDAはC言語を拡張した言語で，一般に利用されているほとんどのOSに対応しています．ソフトウェア自体は無償でダウンロードできます．C/C＋＋言語の開発環境は指定されていますが，こちらも無償の開発環境が利用できます．ただし，ハードウェアは同社のGPUを搭載するビデオカード，または専用ハードウェアを用いる必要があります．一般的にはビデオカードで十分と思いますので，本稿では通常のビデオカードの使用を前提にしています．

CUDAを用いたGPUプログラミングについて簡単に紹介します．CUDAのマニュアルではパソコン本体をホスト，ビデオカードをデバイスとして区別しており，それにならって説明します．基本的には以下の手順で計算を行います．

① ホスト側にメモリーを確保してデータを用意する
② デバイス側にメモリーを確保する
③ ホスト側のメモリーをデバイス側のメモリーにコピーする
④ GPUで計算を行う
⑤ 計算結果をホスト側のメモリーにコピーする

GPUで計算するためには，あらかじめデバイス側のメモリーを必要な分だけ確保しておき，ホスト側のメモリーの内容をデバイス側のメモリーにコピーして使う必要があります．ここで問題となるのが，コピーに費やされる時間がGPUでの計算時間に比べて相対的に長いことです．これはGPUの問題というよりも，現在のビデオカードのインターフェースとして利用されているPCI-Express×16の転送速度に関係する問題です．データ数が大きい場合，GPUでの計算時間よりもデータ転送時間のほうが長くなることもあり，単純にGPUを使えば高速計算ができるというわけではないのです．GPUのもつ並列性の高い計算能力を効果的に利用するためには，計算はデバイス側で行い，データ転送はできるだけ抑えるというスタイルが必然的に要求されます．したがって，計算時間は手順④のGPU単体での計算時間で考えるだけではなく，手順③〜⑤の転送時間を含めた計算時間でも考える必要があります．

CUDAを用いてプログラムをつくるためには，ビデオカードのメモリー構成に関する知識が必要になります[1]．GPUは複数の種類のメモリーを利用

できるようになっています．一般に使うのはグローバルメモリーです．適切な並列アルゴリズムで実装することにより，十分な高速処理が得られます．ほかに代表的なものとして，共有メモリーがあります．こちらはグローバルメモリーよりも高速アクセスが可能であり，さらに高速化が可能になります．しかし容量が少なく，使用法が独特なので注意が必要です．

CUDA のプログラミングスタイルと並列アルゴリズムの開発には GPU に合わせた独自仕様があり，多少の慣れが必要になると思います．そして，パフォーマンスの向上のためには最適化が必須です．最適化にはメモリーや命令の最適化，または並列アルゴリズムの改良などがありますが，いずれも GPU やメモリー構成に関する専門知識が必要です．CUDA はマニュアルやサンプルプログラムが多数用意されていて，詳しい情報はそれらから得ることができます．サンプルプログラムにはグローバルメモリーを使うプログラムを，共有メモリーを使ってさらに数倍高速化する例などがありますので，たいへん参考になります．

実際の計算例として，二次元高速フーリエ変換（FFT）の計算を CPU と GPU で行ったときの計算時間の比較を図 1 に示します（CPU は Intel Core2Duo E8400，GPU は NVIDA GeForce GTX280 を使用）．FFT の計算には，CPU 用には FFT 計算で有名なライブラリーである FFTW（http://www.fftw.org/）を，GPU 用には CUDA の FFT 用ライブラリーである CUFFT[2] を使いました．いずれも $N \times N$（N は 2 のべき乗）の単精度実数の二次元配列データに対して二次元フーリエ変換と二次元フーリエ逆変換をそれぞれ 1000 回行い，その平均の計算時間を求めたものです．図 1 では $N=64$ 以下のときは CPU のほうが高速であり，$N=128$ を超えると GPU が高速になりました．$N=4096$ のときは GPU が CPU よりも約 14 倍高速でした．GPU は並列性の高い演算処理を得意としており，データ数が大きい場合の FFT に対して十分な並列演算性能が発揮されます．光学分野では FFT が必要な計算が多いので，GPU の利用はかなり有効です．しかし，前述したとおり，ホスト側メモリーとデバイス側メモリー間のデータ転送時間があり，この時間を加えると，$N=4096$ の場合は GPU が CPU よりも約 5.7 倍になり，データ数が増えるとともに転送時間の影響は大きくなります．

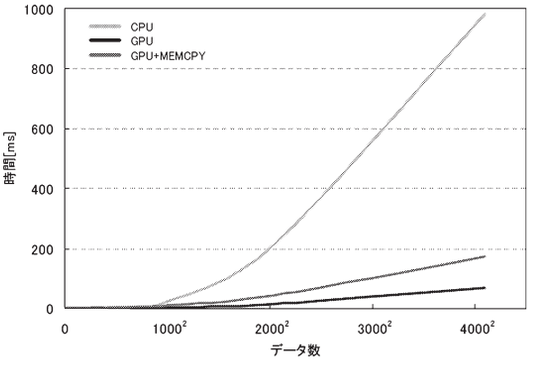

図 1　二次元 FFT の計算時間の比較．

今回は GPU プログラミングの現状について紹介しました．GPU は並列処理による高い演算性能とともに，市販のビデオカードに搭載されているため導入コストが低いという利点もあるため，その利用範囲は急速に拡大しています．現在では，GPU を用いた並列コンピューティングのためのオープンなプログラミング環境が開発されています．また，数値計算用アプリケーションソフトでも GPU が利用されはじめています．近い将来には，わかりやすいインターフェースで GPU の計算能力を利用できるようになるかもしれません．GPU はハードウェア，ソフトウェアともに急速に発展しており，光学分野においても多岐にわたる応用が期待されます．

（埼玉大学　吉川宣一）

文　献

1）CUDA Programming Guide Version 2.0, NVIDIA.
2）CUDA CUFFT Library, NVIDIA.