第2章 プログラミングの前に知っておきたい…
パイプライン/SIMD/メモリ・アクセス/データ転送

CUDAを使うと並列演算が高速になる仕組み

鈴木 量三朗

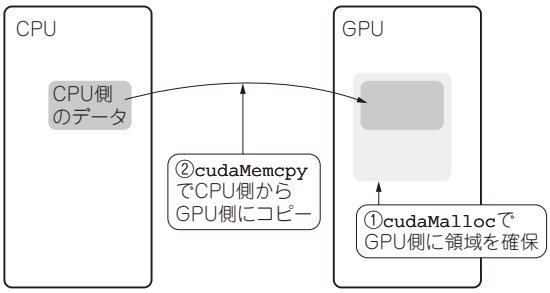

図1 GPUを使う大前提…GPUに計算を依頼するにはデータを転送する必要がある

● GPUに期待すること…並列演算による高速処理

ここでは，CUDAを使うと処理が高速になる仕組みをプログラマ視点で解説します．

特にエヌビディアのGPUを使うときは，PCI Express接続のグラフィックス・ボードを追加する場合が多いと思います．GPUには積和演算を並列にできる計算エンジンが多数搭載されており，CPUの逐次処理では時間のかかる計算を効率的に処理できます．GPUに計算を依頼することで，トータルとして高速に処理できることが期待できます．

● GPUを使う大前提…近くのDRAMへ転送が必要

CPUがGPUに計算を依頼するということは，計算の元となるデータをGPUでも扱えるようにしなくてはなりません．そのためCPUは自分の手元にあるデータをいったんGPUの近くにあるDRAMに転送します．

▶データの転送

例えば，CUDAでよく使うAPIの組み合わせは，cudaMallocとcudaMemcpyです．**図1**のように，cudaMallocでGPU側のDRAMに領域を確保し，cudaMemcpyでCPU側のデータをコピーします．

▶プログラムの転送

GPUが実際に走らせるプログラム自身も，CPUからGPUへデータとして転送されます．

リスト1 CPUからGPUで実行する関数を呼び出すコード

```
vectorAdd<<<blocksPerGrid, threadsPerBlock>>>(
                          d_A, d_B, d_C, numElements);
```

例えば**リスト1**のように，まるでCPUからGPUを呼び出しているように見えますが，実際には次の操作が裏で行われています．

(1) CPU側でCUDAのカーネル・プログラムをコンパイル
(2) GPU側のDRAMにプログラムを置く場所を確保
(3) コンパイルされたデータをGPUのDRAMへ転送
(4) GPUにプログラムの場所を教えて実行を指示

高速化の仕組み①…パイプラインとSIMD

● GPUでは意外と大変な足し算

単純な1234 + 5678の足し算を考えます．

```
  1234
+ 5678
```

この処理は，本質的に並列化が困難です．手計算する場合も，一の位の足し算4 + 8から始めるでしょう．千の位の1 + 5から始めようとすると，百の位の足し算の結果によって繰り上がりが発生する可能性があるので，千の位だけ見て単純に結果を6と決定できないためです．少し考えると，このケースでは百の位は2 + 6なので，繰り上がりが発生しないことが分かります．

幾つかの特殊なケースで並列化可能な箇所が見つかりますが，本質的に足し算は下の位から計算するという順序性があります．32ビットや52ビット(doubleの仮数部)の足し算をしようと思った場合，順序性があるため高速化するためには意外と工夫が必要です．

● 工夫1：パイプライン

足し算自身の並列化による高速化は困難ですが，ベルトコンベア方式で計算することで，多数の足し算の計算効率を上げることを考えます．