GPGPU による高速演算について

榎本 昌一

東京大学大学院工学系研究科システム創成学専攻

1. 概要

パーソナルコンピュータ用 CPU の高速化は日進月歩で進んでいるが，より高速な演算をさせる為には CPU 独自の能力では限界がある．そこで，海洋研究開発機構（JAMSTEC）の地球シミュレータなどの高速計算機システムでは複数の計算機よる分散処理行うクラスタリングによる高速化を実現している．だがここ数年，クラスタリングとはちょっと違った手法による高速演算が提案され実際に使用され始めた．それは GPGPU（General Purpose Graphics Processing Unit）と言われるもので，パーソナルコンピュータのグラフィックボード上のプロセッサ（GPU）に演算を行わせ，高速な演算処理を実現するものである．

本発表では，GPGPU プログラミングに必要な GPU とはどういうものか，また GPU による高速演算を実現するための開発環境，実際のプログラミングについて説明し，実際に高速演算が可能かを考察する．

2. GPU　とは

GPU（Graphics Processing Unit ）はパーソナルコンピュータやワークステーション等に於いて画像処理を担当する主要な部品のひとつで，CPU に負担を掛けずに３D の描画を行うソフトウェアルーチンがサポートされており，これにより高速な描画が可能となっている．図１（左）の写真は NVIDIA 社製のグラフィックボードで「GeForce 9500 GT」 と呼ばれる GPU と，高速にアクセスが可能な VRAM（video RAM）を搭載している．ファンの下に GPU（右）が実装されている．インターフェイスは PCI Express x16 である．廉価なパーソナルコンピュータの場合，マザーボードに GPU を直接載せている場合が多く，VRAM をメインメモリから振り分けることがある．そのためメモリアクセスの時間がかかり，描画速度が落ちることとなる．また，グラフィックボードはかなりの電力を必要とする．このボードでは最大消費電力は 59W であるが，この GPU の上位機種である「GeForce G TX 285」を搭載したものでは 200W を超えるため，バススロットからの電力供給では追いつかず，直接電源部から供給することになる．

図２に NVIDIA 社製 GPU を使用したグラフィックボードのアーキテクチャを示す．グラフィックボード自体には GPU と VRAM が搭載されており，GPU は 30 個の Streaming Multiproc essor（SM）で構成されている．さらにその SM は８個の Streaming Processor（SP）と 16KB の Shared Memory で構成されている．つまり，一つの GPU には 30×8＝240 個のプロセッサが集積されている．グラフィックボードはこれらのリソースを駆使し，3D-CG オブジェクトの移動・回転時の

図１ グラフィックボードと GPU

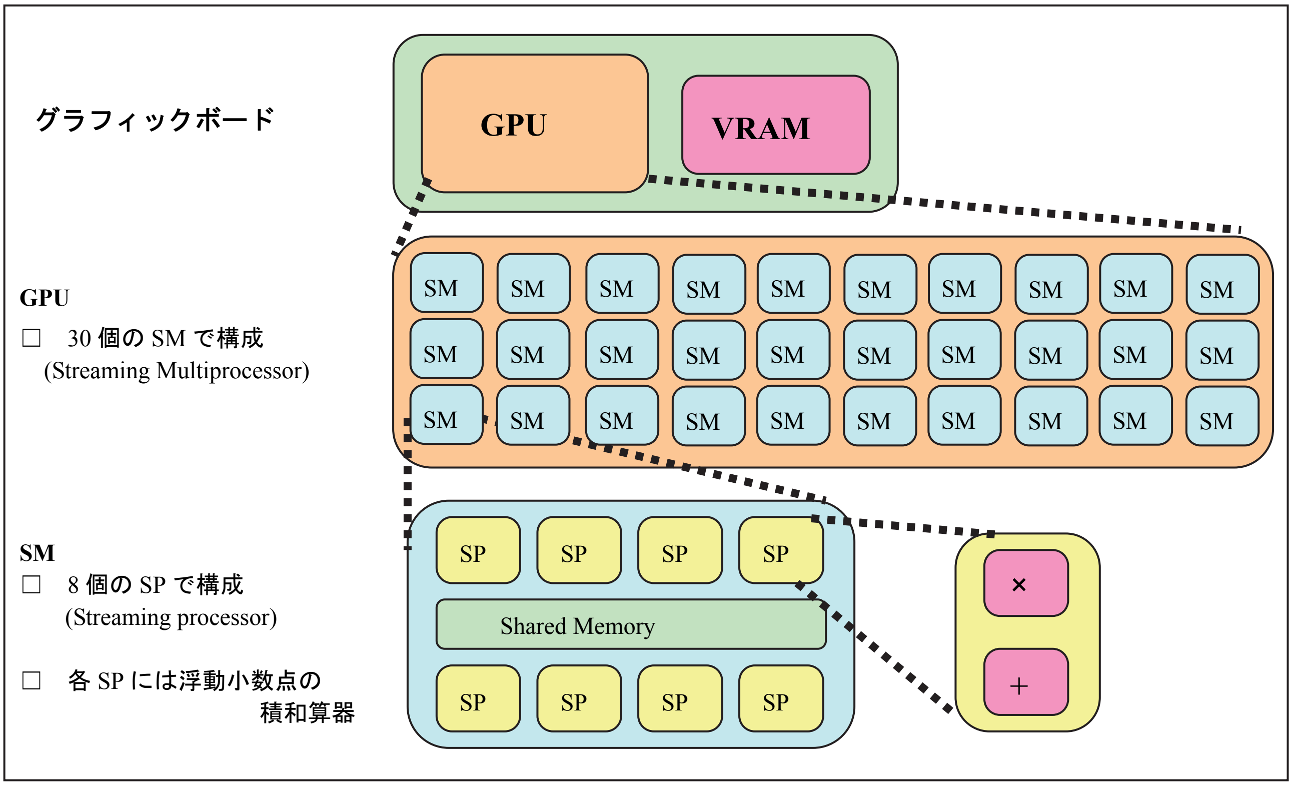

図２ NVIDIA 社のグラフィックボードのアーキテクチャ

座標変換を行う為の行列演算（アフィン座標変換），3D-CG オブジェクトを生成するポリゴンメッシュ，3D-CG オブジェクトの面に色や模様を貼り付けるテクスチャマッピング，3D-CG オブジェクトに陰影付けを行うシェーダ機能を高速で実現している．

3. GPGPU　開発環境　CUDA

240 個のプロセッサを持つ GPU をグラフィックだけではなく数値演算にも使えないか，つまり，グラフィックス専用のプログラムだけではなく，データ処理等の一般的なプログラムを動作できないかと考える研究者が現れ，2006 年 12 月，NVIDIA 社のチーフサイエンティストである David kirk 博士により開発環境 CUDA（Compute Unified Device Architecture）が発表された．CUDA は NVIDIA 社の GeForce に特化した開発環境である．現在では同じグラフィックボードメーカーの AMD 社から「ATI Stream」という開発環境も出ており，こちらは同社の GPU である Radeon に特化している．今回は，その熟成度，世界での利用度を考慮し，CUDA を使用してみた．また，CUDA は基本的に C,C++言語であり，その上に GPGPU を実現する為の CUDA ランタイム API，ユーティリティ関数，GPU を実際に動かすカーネル関数が統合されている．先頃 Fortran コンパイラも提供された．これは，計算流体力学（気候および海洋モデリングなど），有限要素分析，分子力学，量子化学などの高速計算の必要な分野では Fortran が使われている現状がある為である．

3-1.　CUDA のインストール

CUDA は Linux 版，Windows 版，MAC-OS 版が有り，NVIDIA 公式サイト (http:/ /www.nvidia.co.jp/object/cuda_get_jp.html) からダウンロードが出来る．OS 環境に合わせたファイルをダウンロードし，インストールすればよい．

3-2.　CUDA プログラム

CUDA のプログラムは，PC の CPU に関係した部分と GPU を搭載したグラフィックボードの部分に分かれて動作する．CPU 側を「ホスト」，GPU 側を「デバイス」と呼び，デバイス上で動作するプログラムをカーネルプログラムという．図３に CUDA プログラムの基本的な流れを示す．

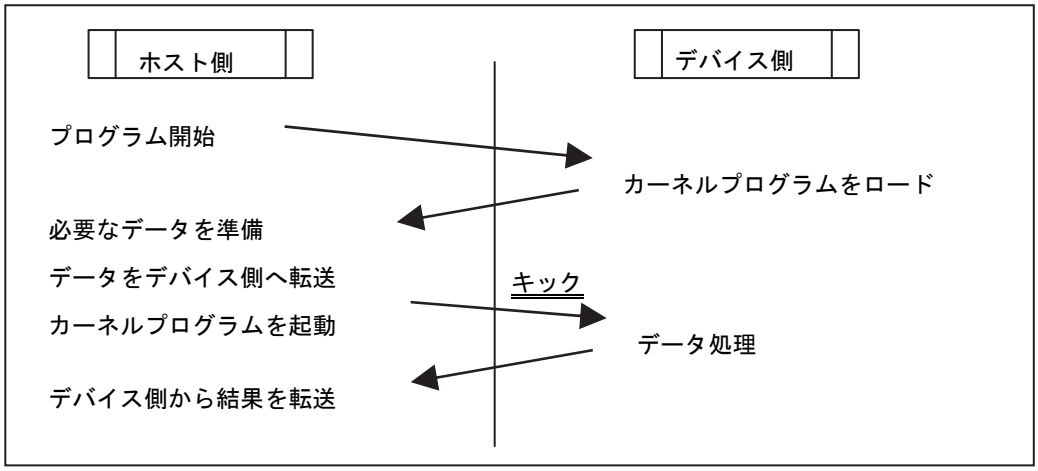

図３　CUDA プログラムの動作

① 通常のプログラムのようにホスト側でプログラムを起動し，カーネルプログラムをデバイス側にロードする

② ホスト側で必要なデータを作成し，デバイス側のメモリに転送する

③ ホスト側からカーネルプログラムを起動させる（「キックする」という）

④ カーネルプログラムでの処理が終わったら，結果をホスト側へ転送する

これが一連の流れである．図４にプログラム例を示す．

```
// ホスト側プログラム
int main()
{
int n = 500;
int nb = sizeof(float) * n;
float *x = (float*) malloc(nb);
float *y;
cudaMalloc( (void**) &y, nb);
for(int i=0; i<n; i++) x[i] = i;
cudaMemcpy( y, x, nb, cudaMemcpyHostToDevice); //      データ転送（ホスト → GPUメモリ）
calc_on_gpu <<<1, 500 >>> ( y );                //      カーネル関数(500個のスレッドで処理)
cudaMemcpy(x, y, nb, cudaMemcpyDeviceToHost);     //  データ転送（GPU → ホストメモリ）
return 0;
}

// デバイス側プログラム（カーネルプログラム）
__global__    //                                      カーネルプログラム宣言
void calc_on_gpu (float *y)
{
        int i = threadIdx.x;                          // 各スレッド毎の番号
 y[i]         = sqrt(y[i]);          //                 各スレッドで計算
}
```

図５　CUDA プログラム例

4.　CUD A による高速演算

大量のプロセッサを使って並列データ処理が行える CUDA だが，演算の高速化を行うには GPU のハードウェアを理解し，数学的なプログラミング手法を身につける必要がある．

4-1.　プロセッサ群とメモリモデル

CUDA では 240 個のプロセッサをマルチスレッドとして使用でき，最大スレッド数は 65535×65535×512 個となっている．このように大量のスレッドを管理する為，グリッドとブロックという概念を導入している（図５）．スレッドはブ

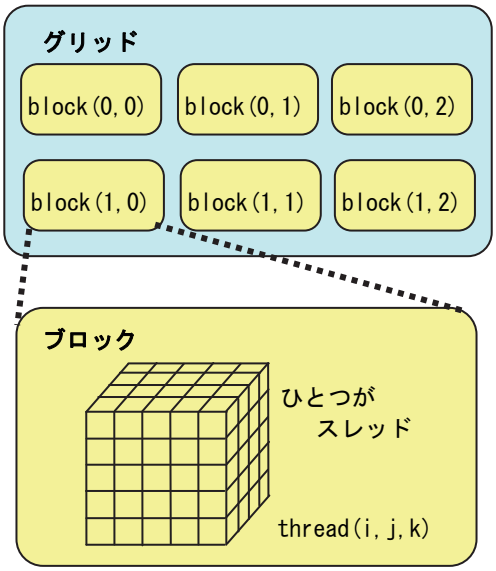

図５ グリッドとブロック

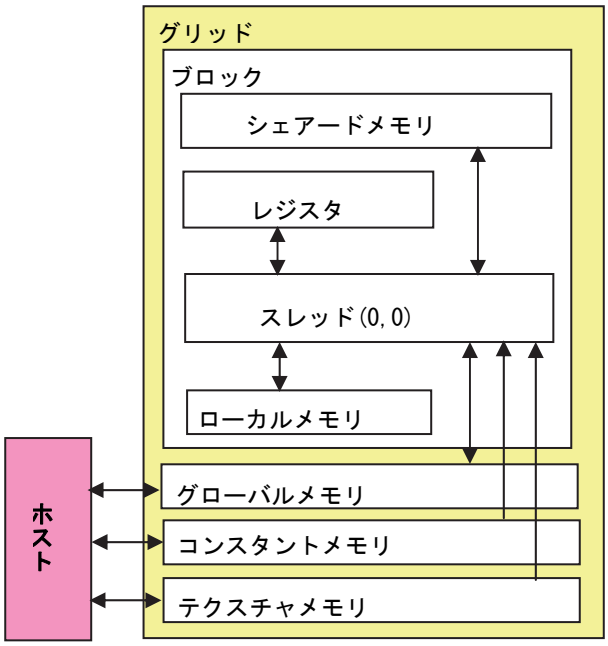

図６ メモリモデル

ロックでまとめられており，１ブロックで最大 512 スレッドを管理できる．ブロック内のスレッドは，１次元（512 個），２次元（16×16 個），３次元（8×8×8 個）で表現することが出来る．

GPU のメモリモデルを図６に示す．特別な場合を除き CUDA で使用するメモリはグローバルメモリとシェアードメモリである．グローバルメモリはホスト側と入出力に使われ，シェアードメモリはレジスタ並みの高速内部メモリである．

4-2. プログラミングと実行

CUDA を使って 512×512 の行列積演算プログラムを作成した．①CPU のみの演算，②GPU のグローバルメモリを使った演算，③GPU のシェアードメモリを使った演算についてそれらの演算時間を出力した（表１）．計算結果③のように，CPU を使った演算に比べ，GPU を使った演算はかなりの高速化が期待できるが，②のように，GPU のメモリの使用方法により，高速化が期待できないこともあることがわかった．

表１ 計算時間

計算方法	計算時間（msec）
① CPU のみの演算	266.8
② GPU のグローバルメモリを使った演算	212.0
③ GPU のシェアードメモリを使った演算	18.0

5. 終わりに

GPGPU を実現する開発環境 CUDA を使ってみた．条件判定等の制御系には向かないが，大量の計算を並列で行うことが出来るため，高速演算を実現できることがわかった．ただし，プログラミングには線形代数的なプロセッサ群の管理やメモリアクセスについてのかなりのスキルが必要である．

今後は，３次元レーザスキャナの計測データのような，３次元データを持つ数千万個の点群データの解析に応用したいと考えている．

参考文献

はじめての CUDA プログラミング　青木尊之・額田彰　工学社　ISBN978-4-7775-1477-9

CUDA 高速 GPU プログラミング入門　小山田耕二・岡田賢治　秀和システム　ISBN978-4-7980-2578-0

CUDA ZONE　http://www.nvidia.co.jp/object/cuda_home_new_jp.html