GPU を用いた直交格子法による流体の移流計算の高速化

奥田遼介[†,☆] 高橋大樹[†] 鈴木貴樹[†]

GPU Challenge 2010 の規定課題部門において，直交格子法による流体の移流計算を GPU を用いて高速化するという課題に取り組んだ．本大会では，NVIDIA 社の CUDA を用いて同社の GPU(Tesla C1060) に対して最適化を図った．GPU は CPU と比べ非常に高い演算能力があり，VRAM の転送速度も高速であるが，その性能を発揮するにはプログラミング上の制限を満たさなければならない．本開発では，共有メモリ，テクスチャメモリの活用や 1 スレッドでの計算量を最適化することにより，実行委員会の用意した CPU のプログラムに対して 240 倍程度，GPU のプログラムに対しても 16 倍程度の高速化を達成した．

1. はじめに

まず，GPU Challenge 2010 の規定課題について述べる．二次元領域を大きさ $\Delta x \times \Delta y$ の $nx \times ny$ 個の直交格子に分割する．各格子点 (jx, jy) についてのインクの濃度 $f(jx, jy)$ と流体の X 方向の速度成分 $u(jx, jy)$，Y 方向の速度成分 $v(jx, jy)$ が与えられる．一時間ステップ Δt のときの，nt ステップ後のインク濃度を求めるというのが今回の課題である．移流の計算は，Cubic セミ・ラグランジュ法により補完して計算する．この方法では，1 点の計算には最大で 2 つ隣りの点の情報が必要となるため，本課題では，四辺の幅 2 の境界部分は時間ステップ 0 における値のまま不変とされている．また，問題には，nx, ny はそれぞれ 32768 以下の 64 の倍数で，$nx \times ny$ は $134,217,728(2^{27})$ 以下という制約が課されている．

2. プログラム概要

今回のプログラム制作は与えられた toolkit を改良することで行った．基本的なアルゴリズムは toolkit と同様のものを用い，計算式の形を変形し演算量を減らした．また，共有メモリ，テクスチャメモリの活用などにより，データの転送を最適化した．

3. GPU に対するプログラムの最適化

3.1 メモリ帯域の最適化

1 点を計算するには， 周囲の 16 点の情報が必要である．仮に 1 スレッドで 1 点を計算した場合，各スレッドで周囲 16 点についてグローバルメモリから読み込む必要がある．共有メモリを利用することで，各スレッドブロックとその周囲を読み込むことで済む．

一方で，グローバルメモリへのアクセスはコアレスアクセスと呼ばれる一定の規則に一致した場合に高速に転送することが可能となる．しかしながら，コアレスアクセスとならない場合は転送速度が遅くなる．そこで，各点のインク濃度の読み出しにテクスチャメモリを利用した．テクスチャメモリへの参照はキャッシュされるので，コアレスアクセスとならない場合でも転送速度の低下を防ぐ．

3.2 SP の演算量の削減

今回制作したプログラムでは，SP(Streaming Processor) がボトルネックとなっている．

Tesla C1060 は 1 秒間に 933G 回の単精度浮動小数点演算が可能である．しかしこの数値は，SP で FMAD（FMUL ＋ FADD），SFU(Special Function Unit) で FMUL を実行できる場合の値である．

SP が担当する演算は FMAD，FADD，FMUL，整数演算，型変換である．これらの命令を GPU 内の全 SP で 1 秒間に 312G 回実行できる．プログラムによっては，単精度計算以外の命令によって SP がボトルネックになる．

今回のプログラムにおいて，SP の負担を減らすためには「単精度補完演算」か「アドレス計算」のいずれかを減らさなければならない．特にアドレス計算は 64bit 整数加算と，32bit 整数掛け算，型変換を必要とするため命令数が多い．

今回は nx を 32K，16K，8K，4K の 4 種類の値に切り上げてデータをメモリ上に配置した．これによって，nx の値をコンパイル時に確定できることとなり，アドレス計算が最初の部分を除き定数加算 1 命令で行えるようになった．しかし，このままでは最初の 1 回のオーバーヘットが占める割合が多いため，1 スレッドが 8 セル分の演算を担当している．

† 一関工業高等専門学校
☆ 現在，東北大学

4. 結　　果

nx 4096, ny 2048, nt 10240 において実行委員会の用意したプログラムでは 184.440 seconds かかったが，本プログラムでは 16.8 倍の 10.983 seconds になった．

さらに，小さいサイズ（nx 256, ny 256, nt 1280）では，1.523 seconds が 0.076 seconds になり，20.0 倍となった．

しかし，1 セル 1 単位時間あたりの演算量で考えると $1069.541 \times 10^6 updatespersecond$ ，$7757.958 \times 10^6 updatespersecond$ と開きがある．

5. ま と め

テクスチャメモリの使用による転送のボトルネックの解消と SP に着目した命令数の削減によって，実行委員会の用意した CPU のプログラムに対して 240 倍程度，GPU のプログラムに対しても 16 倍程度の高速化を達成した．

参 考 文 献

1) GPU チャレンジ 2010 実行委員会: GPU Challenge 2010.http://www.hpcc.jp/sacsis/2010/gpu/.
2) NVIDIA: CUDA Reference Manual Version 2.3
3) NVIDIA: CUDA Programming Guide Version 2.3.1