3J-06

GPUを用いた格子ボルツマン法のループ展開を利用したメモリアクセスの局所性向上による高速化

鈴木 宏明† 富永 浩文† 佐藤 一馬† 中村 あすか† 前川 仁孝†

†千葉工業大学情報科学部情報工学科

1 はじめに

GPUを用いた格子ボルツマン法は，格子点上の粒子分布関数を計算するために，解析領域をブロック分割し，時間ステップごとにスレッドブロックに割り当てることで，流体の流れを解析する[1]．スレッドブロックは，解析領域をメモリアクセスコストが低いシェアードメモリに格納して計算する．このため本手法は，スレッドブロックを生成するたびにグローバルメモリへのアクセスが発生し，メモリアクセスの時間的局所性が得られない．そこで本研究では，格子ボルツマン法のメモリアクセスの時間的局所性を向上するために，ループ展開を用いてグローバルメモリへのアクセス回数を削減する手法を提案する．

2 GPUを用いた格子ボルツマン法

GPUを用いた格子ボルツマン法は，時間ステップごとに衝突，並進計算を繰り返すことによって時間発展方程式を求解する[1]．以下に，位置 x，時刻 t で i 方向の速度 c_i をもつ粒子分布関数の時間発展方程式を示す．

$$f_i(x+c_i,t+1) = (1-\frac{1}{\tau})f_i(x,t) + \frac{1}{\tau}f_i^{eq}(x,t)$$

$$f_i^{eq}(x,t) = \omega_i\rho(x,t)(1-1.5u(x,t)^2+3c_iu(x,t)+4.5(c_iu(x,t))^2)$$

$$\rho(x,t) = \sum_i f_i(x,t), \qquad u(x,t) = \frac{1}{\rho}\sum_i c_i f_i(x,t)$$

ここで，τ は緩和時間係数，$f_i^{eq}(x,t)$ は局所平衡状態における粒子分布関数，$\rho(x,t)$ は密度，$u(x,t)$ は流速，ω_i は i 方向への重み係数を表す．本手法は，前の時間ステップの周囲9方向の格子点を参照して ρ，u を更新するため，同一時間ステップの各格子点の計算を並列にできる．このためGPUを用いて計算する場合，グローバルメモリに格納した解析領域をブロック分割して複数のスレッドブロックに割り当て，各スレッドで格子点を更新する[2]．この時，各スレッドブロックは，割り当てられた解析領域へのメモリアクセスコストを削減するために，図1のように割当て領域だけでなく割当て領域に隣接する格子点もシェアードメモリに格納して計算する．このため，時間ステップごとにカーネルを起動し，グローバルメモリにアクセスすることでシェアードメモリに格納されている格子点を更新する．

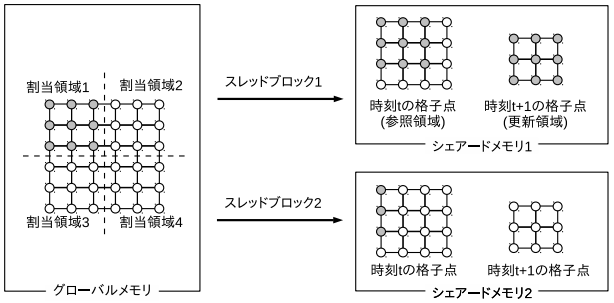

図1: 格子点の格納例

3 ループ展開を用いた格子ボルツマン法

GPUを用いた格子ボルツマン法は，時間ステップごとにカーネルを起動するため，前の時間ステップで格納されたシェアードメモリの格子点を参照できず，時間ステップ間のメモリアクセスの時間的局所性が得られない．時間ステップ間のメモリアクセスの時間的局所性を向上するためには，複数ステップの計算を1つのカーネルに割り当てる必要がある．そこで本研究では，ループ展開を用い，複数ステップの計算を1つのカーネルに割り当てることで，時間ステップ間のメモリアクセスの時間的局所性を向上する．ただし，単純なループ展開により複数の時間ステップの時間発展方程式を計算するカーネルを生成すると，異なるスレッドブロックが更新した解析結果を参照するため，スレッドブロック間の同期が必要となり，高速化が難しい．このため，単純にループ展開するだけでなく時刻 t の計算結果のみ参照するように，時間発展方程式を変形する．以下に，2ステップ前の計算結果から格子点を計算する時間発展方程式を示す．

$$f_i(x+2c_i,t+2) = (1-\frac{1}{\tau})\left\{\alpha + f_i^{eq}(x,t)\right\} + f_i^{eq}(x+c_i,t+1)$$

Speedup of the Lattice Boltzmann Method Using Loop Transformation for Locality Improvement of Memory Access on GPU

Hiroaki SUZUKI† and Hirobumi TOMINAGA† and Kazuma SATOU† and Asuka NAKAMURA† and Yoshitaka MAEKAWA†

†Department of Computer Science, Chiba Institute of Technology

$$f_i^{eq}(x+c_i,t+1) = \omega_i \rho'(x+c_i,t+1)$$
$$\left\{1 - 1.5u'(x+c_i,t+1)^2 + 3c_i u'(x+c_i,t+1)\right.$$
$$\left.+4.5(c_i u'(x+c_i,t+1))^2\right\}$$
$$\rho'(x+c_i,t+1) = \sum_i (\alpha + \frac{1}{\tau} f_i^{eq}(x,t))$$
$$u'(x+c_i,t+1) = \frac{c_i}{\rho'}(\alpha + \frac{1}{\tau} f_i^{eq}(x,t))$$

ただし，$\alpha = (1-\frac{1}{\tau})f_i(x,t)$ である．本手法によるスレッドブロックは，ループ展開前よりも広い領域を参照して計算するため，ループ展開前よりも広い領域をシェアードメモリに格納する．図 2(a) にループ展開しないで 2 ステップ計算する例を，図 2(b) にループ展開して 2 ステップ計算する例を示す．図 2 よりループ展開する手法では，スレッドブロックは，1 つのカーネルで 2 ステップ分計算するために，図 2(a) より広い領域をシェアードメモリに格納して計算する．これにより，1 回の時間ステップでシェアードメモリに格納する格子点数は多くなるが，同期回数を増やすことなくグローバルメモリへのアクセス回数を削減し，メモリアクセスの時間的局所性を向上できる．

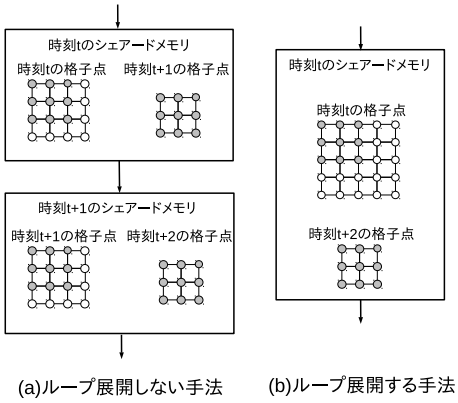

図 2: 2 ステップ分の格子点の格納例

4 評価

提案する手法の有効性を示すために，ポアズイユ流れを解析し，解析時間を測定する．評価で使用する環境は，CPU が AMD Phenom II X4 965，メモリが 8GB，GPU が GeForce GTX TITAN Black であり，API に CUDA6.5 を用いる．また，スレッドブロックのサイズは，全ての解析領域サイズで x 方向 128，y 方向 2 とする．

まず，表 1 にループ展開の有無による解析時間を示す．表 1 より，全解析領域サイズにおいてループ展開する手法は，しない手法に対して解析時間が約 30%削減することが確認できた．これは，ループ展開により，グローバルメモリへのアクセス回数が削減し，時間ステップ間のメモリアクセスの時間的局所性が向上したためと考えられる．次に，グローバルメモリへのアクセス回数が解析時間の削減率に対する影響を確認するために，CUDA Tool Kit の NVIDIA Visual Profiler を用いてグローバルメモリへのアクセス回数を測定する．表 2 にループ展開の有無によるグローバルメモリへのアクセス回数を示す．表 2 より，ループ展開する手法は，グローバルメモリへのアクセス回数の削減率が全解析領域サイズで約 31%であることが確認できた．このことからループ展開する手法は，グローバルメモリへのアクセス回数の削減率が全解析領域サイズで一定であるため，解析時間の削減率も一定になったと考えられる．また，解析時間のほとんどをグローバルメモリへのアクセスが占めているため，高い削減率が得られたと考えられる．

表 1: ポアズイユ流れの解析時間

解析領域サイズ	256	512	1024	2048
ループ展開無 [ms]	16.79	56.71	218.05	835.14
ループ展開有 [ms]	11.67	39.43	146.92	574.94
削減率 [%]	30.49	30.47	32.62	31.16

表 2: グローバルメモリへのアクセス回数

解析領域サイズ	256	512	1024	2048
ループ展開無 [$\times 10^5$ 回]	1.25	5.00	19.99	79.95
ループ展開有 [$\times 10^5$ 回]	0.86	3.44	13.76	55.05
削減率 [%]	31.2	31.2	31.2	31.1

5 おわりに

本研究では，GPU を用いた格子ボルツマン法を高速化するために，ループ展開を用いてメモリアクセスの時間的局所性を向上する手法を提案した．評価の結果，提案手法は，従来手法に対して約 1.4 倍の高速化率を得ることが確認できた．また，時間ステップごとのグローバルメモリへのアクセス回数を減らすことで，時間ステップ間のメモリアクセスの時間的局所性が向上し，解析時間を削減できることが確認できた．

参考文献

[1] Astorino, M., Becerra Sagredo, J. and Quarteroni, A.:A modular lattice Boltzmann solver for GPU computing processors, SeMA Journal, Vol.59, pp.53-78(2012).

[2] Mawson, M. and Revell, A.:Memory transfer optimization for a lattice Boltzmann solver on Kepler architecture nVidia GPUs, CoRR, Vol.1983, pp.53-78(2013).