PIC 法の GPU による高速化

鈴木 淳也（電気通信大学情報システム基盤学専攻）、 島津 浩哲（情報通信研究機構けいはんな研究所）、
深沢 圭一郎（九州大学大学院理学研究院地球惑星科学部門）、 田 光江（情報通信研究機構）

1. はじめに

PIC(particle-in-cell)法はプラズマ数値シミュレーション手法の１つとして、現在幅広く利用されている。PIC 法では、粒子数分の評価点があり、さらに、精度良く計算をするためには、1000 のオーダーの粒子が必要となり、粒子数の増加により膨大な計算時間を要する。そのため、シミュレーションの高速化は必要不可欠である。本研究では、PIC 法によるプラズマの数値シミュレーションコードである KEMPO1[1]の GPU(Graphics processing units) による高速化を試みる。

2. KEMPO1 の計算

PIC 法とは、粒子法の一種で、電子、イオンを粒子として取り扱い、電磁場は格子として取り扱う手法である。本研究では、PIC 法を用いた 1 次元のプラズマの数値シミュレーションコードである KEMPO1（Kyoto university's Electro-Magnetic Particle cOde）[1]を扱う。KEMPO1 で解く方程式を以下に示す。まず、Maxwell 方程式であり、

$$\nabla \times B = \mu_0 J + \frac{1}{c^2}\frac{\partial E}{\partial t},\quad \nabla \times E = -\frac{\partial B}{\partial t}$$

ここで B は磁場、E は電場、J は電流密度、μ_0 は透磁率、c は光速である。次に、粒子の運動方程式であり、

$$\frac{d\upsilon}{dt} = \frac{q}{m}(E + \upsilon \times B)$$

ここで q、m、v はそれぞれ粒子の電荷、質量、速度である。これらの方程式を解くことで、シミュレーションを進める。KEMPO1 の 1 回の時間ステップは主に 5 つの関数、velcty（粒子の速度を計算）positn（粒子の位置を更新）、currnt（電流を計算）、efield（電場の計算）、bfield（磁場の計算）により構成される。

3. KEMPO1 の GPU 上の実装

PIC 法では上記のように粒子と格子の相互作用により計算を行なう。その中で特に問題となるのは、currnt で電流を求める計算である。それは、格子の情報にアクセスする時のランダムなメモリアクセスと複数スレッドによるデータの競合を防ぐための頻繁な同期処理により GPU 上での実行は CPU に比べて非常に遅くなるためである。また、共有メモリは小さすぎるので格子の情報が入りきらないので活用が制限される。

そこで、我々は、領域分割法を導入することにした。領域を分割し計算に必要な格子の情報を少なくすることで、GPU の共有メモリを有効に活用できる。まず、計算する 1 次元を領域分割する。各粒子もどこの領域に存在するかにより分割を行う。次に、GPU 上で割り当てられた領域ごとに逐次計算を行い、それぞれの領域の電流の値を求める。そして、分割した領域分の電流の計算を行い、最終的に領域全体の電流を求める。割り当てられた領域内の電流を求める計算は、ブロック内で各粒子による電流の和を共有メモリ上で計算をする。また、粒子を分割する処理は、CPU 上で実行した。

4. 実験

実験環境について CPU は Intel Xeon X5550、GPU は Nvidia Tesla C1060 を用いて行なった。理論性能値はそれぞれ 42.56×4GFLOPS、933GFLOPS である。CPU は 4 コアを使用、コンパイラは gcc 4.3 で、Open MP を用いて並列実行した。GPU は CUDA[2]で実装し、コンパイラは nvcc 3.1 を用いた。ここで KEMPO1 のコードの主要な関数の実行時間の比較を行なった。領域サイズは格子数 2048 とし、1 格子あたりの粒子数は 1 万個で実行した。

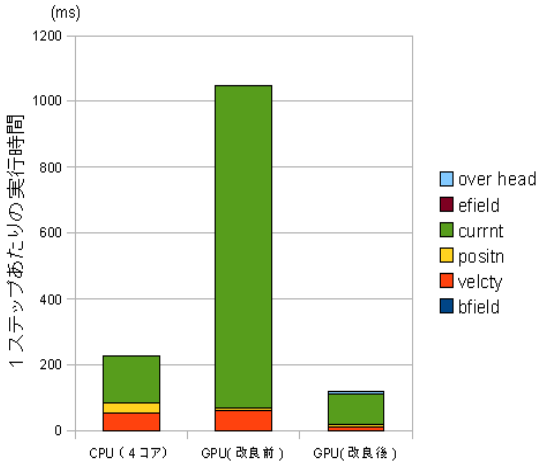

Fig.1 格子数 2048 の場合の実行時間の結果

その結果、GPU の currnt の計算が改善され、GPU が CPU に比べ 2 倍程度のスピードアップを確認できた。ここで、overhead は GPU のみによる場合の overhead は GPU の領域分割にかかる 1 ステップあたりの計算時間である。

5. まとめ

今回、PIC 法によるプラズマの数値シミュレーションコードである KEMPO1 の高速化を目指し、GPU へ移植を行い、最適化を行った。その結果 CPU での実行に比べて、GPU では 2 倍程度の高速化を確認できた。1 次元の PIC コードでは GPU よる高速化は可能であるが、GPU の不向きな計算が遅すぎるためボトルネックとなる問題がある。そこで、GPU に合わせた最適化は必要不可欠であり、高速化を行うためにアルゴリズムを大きく変更する必要がある。

References

[1] H. Matsumoto and Y. Omura, KEMPO1, Computer Space Plasma Physics, Chapter2, pp.21-65 , 1985.

[2] NDVIA CUDA programing guide 1.1, HTTP://developer.download.nvidia.com

author's e-mail: junya@hpc.is.uec.ac.jp