1A-01

Particle-In-Cell 法の GPU への移植

宮島　敬明†　　張　科寅†　　藤田　直行†

†宇宙航空研究開発機構　航空技術部門　数値解析技術研究ユニット

あらまし　ホールスラスタは、従来の化学推進エンジンに比べ 1 桁程度高い比推力と搭載推進剤重量の削減による軽量化により、宇宙における輸送コストの低減を実現する技術として期待されている。その設計には、コスト面での成約により計算機を用いた数値シミュレーションによる試作が行われているが、第一原理計算に近い処理のため、膨大な計算時間が必要なことが知られている。我々は、JAXA が研究開発を行っているシミュレーション用コード”NSRU-Full-PIC”のアクセラレータを用いた高速化について研究を行っている。本稿では、巨大なループの一部の GPU への移植について述べる。

1. Particle-In-Cell 法によるホールスラスタのシミュレーション

ホールスラスタはプラズマの放電を推進力としている。その支配方程式は、個々の荷電粒子の運動を支配するボルツマン方程式(式 1)と、電磁場の時間と空間の変化を支配するマクスウェル方程式となり、確率解法である Particle-In-Cell (PIC) 法を用いて解くことができる[1]。

$$\frac{\partial f}{\partial t}+v\cdot\frac{\partial f}{\partial r}+\frac{F}{m}\cdot\frac{\partial f}{\partial v}=\iint((f'f_1'-ff_1))gd\Omega v_1 \quad (1)$$

ここで、f は速度分布関数、v は 3 次元方向の速度、r は 3 次元座標、m は原子の質量、g は衝突する 2 個の粒子の相対速度 (g = v1 − v) の大きさ、dΩ は衝突の微分断面積を表す。なお、粒子衝突は速度がそれぞれ v, v1 である 2 粒子が衝突し、それぞれ v′, v′ 1 になった際の速度分布関数をそれぞれ f ≡ f (v, r, t), f1 ≡ f (v1 , r, t), f′ ≡ f (v′ , r, t), f1 ′ ≡′ f (v1, r, t) とした。ボルツマン方程式の左辺と右辺は、時間ステップが十分に小さいとすれば、独立した過程として取り扱うことができ、それぞれを以下のように扱っている。左辺(速度分布関数の時間発展)を各粒子に代表させて運動方程式で解くことでラグランジュ的に扱う。すなわち、中性ガス、イオン、電子それぞれのシミュレーション粒子の軌道を運動方程式とし、4 次のルンゲクッタ法で解く。以上を 1 タイムステップとして粒子と場を交互に更新することにより、プラズマの流れと静電場の時間発展を解く。一般的に、1 タイムステップは実時間の 1x10^-12 [sec]に相当し、格子間隔 0.2 [mm]程度の正方格子、計算に必要な粒子の数は数千万から数億に達する。

"Porting a Particle-In-Cell method for Hall-thruster to GPU"
†Japan Aerospace Exploration Agency (JAXA)

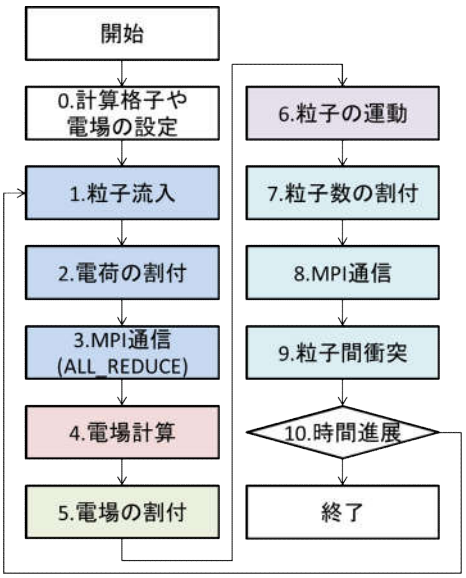

図 1：NSRU-Full-PIC の処理手順

2. 既存 CPU コードの解析

JAXA で研究開発を行っている、ホールスラスタ用 Particle-In-Cell 法プログラム”NSRU-Full-PIC”のフローチャートを図 5 に示す。各ステップの括弧に主要な処理を示した。PIC 法では各粒子はモデル上を自由に動きまわるが、電場を計算するためのグリッド(固定座標上の格子)が用意されている。それに伴い、粒子を格納する配列とグリッドを示す配列の大まかに 2 種類の配列を元に計算を行う。NSRU-Full-PIC は CPU 用のコードで約 7000 行の Fortran 90 で記述されており、MPI を用いた粒子分割（マルチプロセス化）が行われているが、

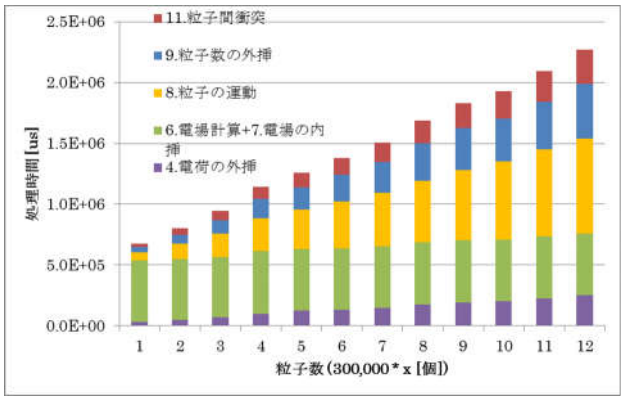

図 2：粒子数と処理時間の関係

OpenMP などのマルチスレッド化は行われていない。図 1 内のステップ 4 と 9 では、各粒子の物理量をグリッドの四隅に割りつける処理が発生する。この処理は、Read-After-Write(RAW)ハザードを引き起こす典型的な処理で、スレッド並列化を阻害し、グリッド内部の粒子の数（数百個）に応じて計算負荷が爆発的に高くなる。加えて、粒子の持つ物理量には浮動小数点型が用いられており、計算順序によって結果が異なってしまうことも並列化を行う上で問題である。また、ステップ 5 と 10 では粒子の情報を全プロセスで共有するため、MPI の ALL REDUCE 通信を複数回行う。ステップ 6 では、電場の計算（グリッド数*グリッド数の粗行列計算）には PETSc[2]を用いている。

図 2 に 1 プロセスで粒子数を増やした際の処理時間の推移を積層グラフで示す。評価には、Xeon E5-2697v2(2.7GHz), DDR3-1600 128GB を用いた。x 軸は正規化した粒子数を示す。なお、基準値(左端の 1)は電子とイオンが 30 万個で中性子が 200 万個であり、右端の 12 は電子とイオンが 30*12 万個で中性子が 12*200 万個である。処理時間は粒子数に比例して伸びている。詳細な評価では、4.電化の外挿に含まれる particle_att_{ion,ele}や 8.粒子の運動内の particle_move_{ion,ele,neu}、11.粒子間衝突の際にソートを行う collision_sort_{ion,ele,neu}のサブルーチンが粒子数に比例している。

3. GPU への移植の検討

手始めに、NSRU-Full-PIC の中で最も計算負荷の高い「4.電荷の外挿処理」のシングル GPU 実装について述べる。 4. 電荷の外挿処理は大きく 2 つの処理に分かれる。 1 つ目は各粒子（イオンと電荷）の割り付けを行う処理

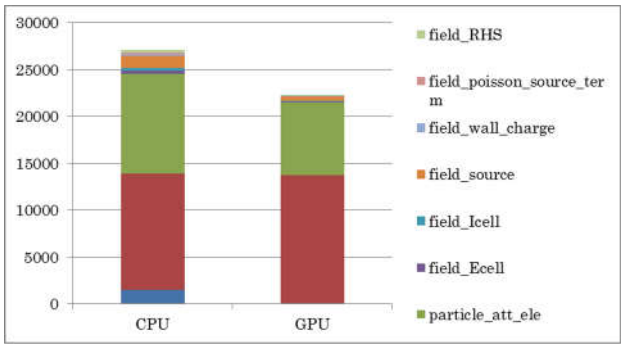

図 3：CPU 実装と GPU 実装の処理時間比較

(particle_att_{ion,ele})、2 つ目はイオンと電荷の状態を元にポアソン方程式を解くための前処理(field_poisson) である。データ構造は、CPU 版をそのまま引き継ぎ SoA 形式とした。また、互換性のため、CPU 版の各 Fortran サブルーチンをそのまま GPU の CUDA カーネルとした。評価には NVIDIA 社の Tesla K20c と CUDA 7.5 を利用した。

図 3 に「4.電荷の外挿処理」の CPU 実装と GPU 実装の処理時間の比較を示す。割り付け処理は、間接参照を原因とするメモリへのランダムアクセスに加え、RAW ハザードが多発しており、そのままでは高速化が困難であった。1.1 倍程度の高速化は充分とは言えないため、更なる研究が必要である。なお、ポアソン方程式の前処理は 2x2 ステンシル計算を主としており、3.5 倍程度の高速化を達成した。GPU 版の関数と CPU 版 NSRU-Full-PIC とを結合した全体評価では、データ転送がボトルネックとなり 1 桁程度遅くなってしまった。大規模なループを持つ本プログラムは一部だけの GPU 化では高速化が不可能であった。

4. まとめと今後の課題

本稿では、Full-PIC 法を用いたホールスラスタ用シミュレーションコード”NSRU-Full-PIC”の割り付け処理の GPU を用いた高速化について検討を行った。今後は、粒子のソートを利用した割り付け処理の高速化と更なるプログラムの解析を行い、全体の処理時間を短縮していく。

文献

[1] Shinatora Cho, Kimiya Komurasaki, Yoshihiro Arakawa : ” Kinetic particle simulation of discharge and wall erosion of a Hall thruster ”, (2013)

[2] “Portable, Extensible Toolkit for Scientific Computation ”, http://www.mcs.anl.gov/petsc.