Vol.2015-HPC-151 No.25
2015/10/1

Particle-In-Cell プラズマ粒子コードの性能測定

梅田 隆行[†1]

Particle-In-Cell(PIC)プラズマ粒子コードは宇宙空間を満たす無衝突プラズマの第一原理シミュレーション手法である．PIC シミュレーションでは，電磁場による加速を記述する荷電粒子の運動方程式（ニュートン・ローレンツ式）を，電磁場の時間発展を記述するマックスウェル方程式と共に解き進めている．膨大な数の荷電粒子が，格子上に与えられた電磁場宙を自由に動き回ることから，PIC コードにおいて高い計算性能を出すためには様々な工夫が必要である．本論文では特に，単一ノード内の PIC コードに性能に注目し，Fujitsu FX10 及び Fujitsu CX400 において性能測定を行った．

Performance measurement of Particle-In-Cell plasma simulation code

TAKAYUKI UMEDA[†1]

Particle-In-Cell (PIC) plasma simulation code is a first-principle method for collisionless space plasma. The PIC code solves the Newton-Lorentz equation (equation of motion for charged particles) for each charged particle together with the Maxwell equations for electromagnetic fields. Since a huge number of charged particles move freely in the grid cells of electromagnetic fields, it is difficult to achieve a high performance of the PIC code on a massively-parallel scalar supercomputer. In the present study, the performance of PIC code on a single node is focused on and is measured on the Fujitsu FX10 and Fujitsu CX400.

1. はじめに

我々が住む宇宙の 99.99%以上の体積はプラズマと呼ばれる電離気体で占められている．宇宙空間に存在するプラズマの大部分は密度が非常に小さく無衝突状態にあり，宇宙プラズマ（無衝突プラズマ）を理解することは，宇宙の本質的な理解につながる．

我々が住む地球周辺の宇宙環境は，太陽から放出された高速のプラズマ流である太陽風及び太陽風が運ぶ惑星間空間磁場（太陽の固有磁場）と，地球の固有磁場との相互作用によって複雑な磁気圏構造を形成している．プラズマ放出現象をはじめとする太陽の様々な変動により，宇宙飛行士の被曝，人工衛星の故障や通信障害に繋がる地球磁気圏・電離圏の環境変動が引き起こされ，これを宇宙天気と呼ぶ．近年の国際宇宙ステーションでの活動や人工衛星の打ち上げなど，日本においても宇宙利用が現実的になってきており，宇宙天気の予報・予測に繋がる宇宙プラズマ研究は極めて重要である．

地球磁気圏内には，プラズマの密度や温度などの物理パラメータが異なる様々な領域が生じる．その領域間の境界層で現れる不安定性（平衡状態の破れ）は，磁気圏の変動に大きな影響を与えていると考えられている．グローバル磁気圏構造に対して，境界層不安定性は中間（メゾ）スケール現象と呼ばれる．これらのグローバル及び中間スケールの現象は，粒子運動論を扱う方程式であるブラソフ（無衝突ボルツマン）方程式の 0 次・1 次・2 次のモーメントを取ることによって求められる磁気流体力学（MHD）方程式によって記述される．しかし，近年の科学衛星による高精度な「その場」観測では，中間スケールの不安定性において MHD 方程式で記述できる物理過程と粒子の運動論方程式によって記述できる物理過程が結合していることを示唆している．これらのマルチスケールの磁気圏変動である宇宙天気を真に理解するためには，全てのスケールをシームレスに扱える運動論方程式（第一原理）によるシミュレーションが本質的である．

プラズマの運動論シミュレーションには 2 つの手法があるが，本研究では，プラズマ粒子であるイオンや電子などの個々の荷電粒子の運動をニュートン・ローレンツ方程式により解き進める PIC（Particle-In-Cell）法に注目する．格子点（Cell）上に定義された電磁場はマックスウェル方程式により解き進められ，Cell 中を粒子が自由に動きまわることから，古くから PIC 法と呼ばれている．宇宙空間に存在する膨大な数の荷電粒子を有限の計算機資源で扱うことは不可能であるため，PIC 法では，ある程度まとまった数の荷電粒子の集団を 1 つの“超”粒子（super-particle）として扱う．PIC 法はその数値解法の完成度が高く，プラズマ科学分野では 1960 年代より用いられており，また近年ではプラズマ科学以外の分野にも幅広く応用されている．しかし，プラズマを超粒子として扱うことにより熱雑音が大きくなることや，電荷密度や電流密度などの荷電粒子の運動に起因する場の量を格子上に割り振る際に生じる高波数モードが数値誤差として蓄積することなどの欠点がある．また PIC 法では，ラグランジュ変数である粒子の位置及び速度と，オイラー変数である電磁場が混在しており，スカ

†1 名古屋大学太陽地球環境研究所
Solar-Terrestrial Environment Laboratory, Nagoya University

ラ型超並列計算機において高い性能を得るのは困難である．

PIC 法の性能の引き出すためには，大きく分けて２つのアプローチがある．１つは，プロセス並列率を高める手法であり，近年特に精力的に研究されている点である．PIC コードを単純に領域分割で並列化した場合，分割された領域の粒子数が必ずしも均一であるとは限らない．各計算領域(計算のプロセスに相当)の負荷のバランスを均一保つには，各プロセス内の粒子数を均一に保つ必要があり，様々な方法が提案されている[1,2]．２つ目は，ノード内の演算性能を上げるためのアプローチであり，スレッド並列化やキャッシュチューニングなどが挙げられるが，これらの点に注目した研究はあまりおこなわれていない．本研究では特に，２つ目の PIC コードのノード内の演算性能に注目し，SPARC プロセッサ及び x86-64 プロセッサを有するマルチコア計算機の単一ノードを用いて性能の測定を行った．

2. 計算手法の概要

2.1 基礎方程式

無衝突プラズマの運動は，以下の荷電粒子の運動（ニュートン・ローレンツ）方程式によって記述される．

$$\frac{\mathrm{d}\vec{r}_n}{\mathrm{d}t} = \vec{v}_n \tag{1}$$

$$\frac{\mathrm{d}\vec{v}_n}{\mathrm{d}t} = \frac{q_n}{m_n}\left(\vec{E} + \vec{v}_n \times \vec{B}\right) \tag{2}$$

ここで $\vec{E}$，$\vec{B}$，$\vec{r}$ と $\vec{v}$ はそれぞれ電場，磁場，位置，速度を表す．また，q と m はそれぞれ電荷と質量を表し，添え字は n 番目の粒子であることを示す．

プラズマ粒子の分布関数は，電磁場によって変形する．電磁場の時空間発展は以下のマックスウェル方程式によって記述される．

$$\nabla \times \vec{B} = \mu_0 \vec{J} + \frac{1}{c^2}\frac{\partial \vec{E}}{\partial t} \tag{3.1}$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{3.2}$$

$$\nabla \cdot \vec{E} = \frac{\rho}{\varepsilon_0} \tag{3.3}$$

$$\nabla \cdot \vec{B} = 0 \tag{3.4}$$

ここで $\vec{J}$ は電流密度，ρ は電荷密度，μ_0 は真空中の透磁率，ε_0 は真空中の誘電率，c は光速を示す．式(2.1)の発散をとり，式(2.3)を代入すると，以下の電荷保存則が得られる．

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \vec{J} = 0 \tag{4}$$

マックスウェル方程式(3.1)に含まれる電流密度 $\vec{J}$ はプラズマの運動によって生じ，これにより電磁場が変化する．電流密度 $\vec{J}$ が電荷保存則(4)を満足する限り，ポアソン方程式(3.3)は自動的に満たされる．

以上の方程式は，PIC コードにおいて解いている基礎方程式であり，無衝突プラズマの第一原理と呼ぶ．

2.2 数値解法の概要

PIC コードの計算は大きく分けて，式(3)の電磁場の更新(以下，Maxwell という)，式(2)の粒子の速度更新(以下，Velocity という)及び，式(1)の粒子の位置更新と式(4)による電流密度の計算(以下，Current という)の３つに分類できる．電流は荷電粒子の移動によって生じるものなので，式(4)の電流密度は粒子の位置更新に基づいて計算される．

Maxwell 部は，FDTD（Finite Difference Time Domain）法と呼ばれる電磁場解析法を用いて解く．FDTD 法では，Yee 格子[3]と呼ばれる staggered 格子を用いており，式(2.4)が自動的に満たされるように物理量が配置されている．また leap-frog アルゴリズムに基づいて電場と磁場を半タイムステップずらしており，時空間精度は 2 次である．

Velocity 部は，Boris 法[4]と呼ばれる手法を用いており，leap-frog アルゴリズムに基づいて電磁場と速度が半タイムステップずれた，2 次の時間精度になっている．以下は 2 次元コードにおけるプログラムの概要である．

```
DO n=1,np
  i = nint(x(n))
  j = nint(y(n))
  wx1 = ...
  wx2 = ...
  wx3 = ...
  wy1 = ...
  wy2 = ...
  wy3 = ...
  pex + ex(i-1,j-1)*wx1*wy1 &
      + ex(i  ,j-1)*wx2*wy1 &
      + ex(i+1,j-1)*wx3*wy1 &
      + ex(i-1,j  )*wx1*wy2 &
      + ex(i  ,j  )*wx2*wy2 &
      + ex(i+1,j  )*wx3*wy2 &
      + ex(i-1,j+1)*wx1*wy3&
      + ex(i  ,j+1)*wx2*wy3 &
      + ex(i+1,j+1)*wx3*wy3
  :
  vx(n) = vx(n) + ...
  vy(n) = vy(n) + ...
  vz(n) = vz(n) + ...
END DO
```

粒子がどの格子点の近傍に居るか(i,j)を計算し，その隣

接格子への粒子の重み(wx1,wx2,...,wy3)を計算する(具体的な重み計算については省略)．次に，各格子点の電磁場(ex,ey,ez,bx,by,bz)に対してそれぞれ重みを掛け，粒子の位置での電磁場を計算する．最後に，粒子の位置での電磁場より加速度を計算し，速度を更新する(具体的な加速度の計算については省略)．以上の作業をnp個の粒子に対して行っている．粒子の番号nと粒子が居る格子点の番号(i,j)は無関係であるため，格子点データへのアクセス（ロード）はランダムになる．

Current部のうち，式(1)は，leap-frogアルゴリズムに基づいて速度と位置が半タイムステップずれた，2次の時間精度になっている．また式(4)に基づいて電流密度を計算する際に，電荷保存と呼ばれる手法[5]を用いている．以下は2次元コードにおけるプログラムの概要である．

```
DO n=1,np
  i = nint(x(n))
  j = nint(y(n))
  wx1 = ...
  wx2 = ...
  wx3 = ...
  wy1 = ...
  wy2 = ...
  wy3 = ...
  jx(i-1,j-1) = jx(i-1,j-1) + q(n)*wx1*wy1
  jx(i  ,j-1) = jx(i  ,j-1) + q(n)*wx2*wy1
  jx(i+1,j-1) = jx(i+1,j-1) + q(n)*wx3*wy1
  jx(i-1,j  ) = jx(i-1,j  ) + q(n)*wx1*wy2
  jx(i  ,j  ) = jx(i  ,j  ) + q(n)*wx2*wy2
  jx(i+1,j  ) = jx(i+1,j  ) + q(n)*wx3*wy2
  jx(i-1,j+1) = jx(i-1,j+1) + q(n)*wx1*wy3
  jx(i  ,j+1) = jx(i  ,j+1) + q(n)*wx2*wy3
  jx(i+1,j+1) = jx(i+1,j+1) + q(n)*wx3*wy3
  :
END DO
```

Velocityと同様に，まず粒子がどの格子点の近傍に居るか(i,j)を計算し，その隣接格子への粒子の重み(wx1,wx2,...,wy3)を計算する(具体的な重み計算については複雑なため省略)．次に，各格子点の電流密度(jx,jy,jz)に対して，粒子の位置での電流密度に重みを掛けて足し込む．以上の作業をnp個の粒子に対して行っている．Velocityとは異なり，Currentの格子点データへのアクセスはランダムロード・ランダムストアになる．

3. 性能測定

単一ノードにおける各システム／コンパイラのPICコード性能を表1に示す． FX10はノードあたり16コアであり，CX400はノードあたり12コアのCPUが2つ（計24コア）である．ノードあたりの粒子数を144,000,000個に固定した．またCX400では富士通コンパイラとインテルコンパイラの両方を用いた．

Velocityは，富士通コンパイラを用いたCX400がFX10の約4倍，インテルコンパイラを用いたCX400がFX10の約2.7倍高速であった．Currentは，富士通コンパイラを用いたCX400がFX10の約2.5倍，インテルコンパイラを用いたCX400がFX10の約1.2倍高速であった．またMaxwellの計算時間はほとんど無視できることが分かるため，以降の計測ではVelocityとCurrentの2つにのみ注目する．

図1及び2に，スレッド数を変化させた時の，Velocity及びCurrentサブルーチンの経過時間をそれぞれ示す．Velocityは，スレッド数に対して比較的よくスケールしており，最大スレッド数のスケーラビリティは，FX10では約80%, 富士通コンパイラを用いたCX400ではほぼ100%, インテルコンパイラを用いたCX400では約60%であった．1スレッドの場合はインテルコンパイラを用いたCX400のほうが約1割高速であったが，スレッド数が増えた場合には富士通コンパイラを用いたほうが高い性能となった．

	FX10 - 16t	CX400 - 24t + Fujitsu コンパイラ	CX400 - 24t + Intel コンパイラ
Velocity	37.89 sec	9.517 sec	14.29 sec
Current	99.69 sec	40.93 sec	80.09 sec
Maxwell	3.02E-02 sec	1.99E-02 sec	3.02E-02 sec

表1：144,000,000個の粒子を10時間ステップ解いた場合の各サブルーチンの経過時間.

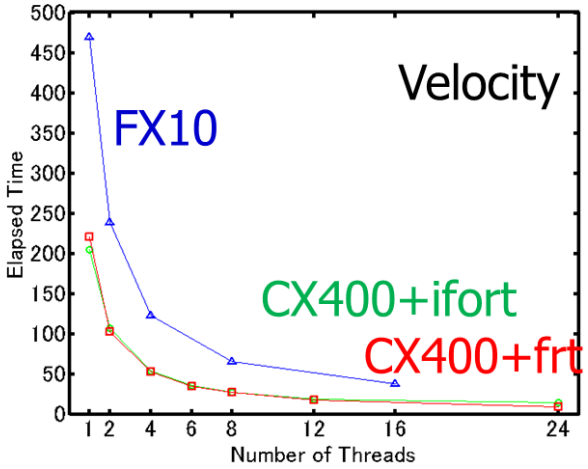

図1：144,000,000個の粒子を10時間ステップ解いた場合のVelocityサブルーチンの経過時間.

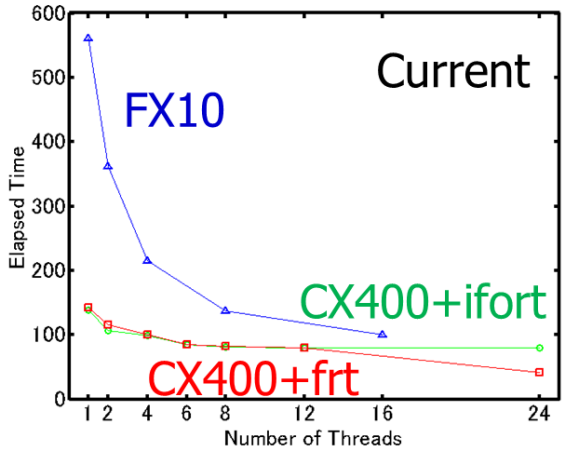

図 2：144,000,000 個の粒子を 10 時間ステップ解いた場合の Current サブルーチンの経過時間.

一方で，Current にはランダムストアがあるため，最大スレッド数のスケーラビリティは， FX10 では約 35%，富士通コンパイラを用いた CX400 では約 15%, インテルコンパイラを用いた CX400 では約 10%であった．特に CX400 では，6 スレッドあたりで性能が飽和した．

以上の結果は，ランダムストアがある場合には，スレッド並列の性能が極端に落ちることを示唆している．粒子データを並べ替えて(ソートして)ランダムアクセスを無くすことにより，スカラ型 CPU における PIC コードの性能が向上することは知られている[6]．しかし，スレッド性能に対するソーティングの効果についてはあまり知られていない．表 2 に，ソートした粒子データを用いた場合の単一ノードにおける各システム／コンパイラの PIC コード性能を示す．粒子データのソーティングにより，Velocity は 1.5 倍から約 4 倍，Current は約 8 倍以上の性能向上が見られることが分かる．

図 3 及び 4 に，ソートした粒子データを用いてスレッド数を変化させた時の，Velocity 及び Current サブルーチンの経過時間をそれぞれ示す．

	FX10 - 16t	CX400 - 24t + Fujitsu コンパイラ	CX400 - 24t + Intel コンパイラ
Velocity	10.91 sec (x3.7)	6.823 sec (x1.4)	9.369 sec (x1.5)
Current	13.29 sec (x7.7)	5.193 sec (x7.9)	7.985 sec (x10)

表 2：ソートした 144,000,000 個の粒子を 10 時間ステップ解いた場合の各サブルーチンの経過時間及び，ソート無しの場合（表 1）に対する倍率.

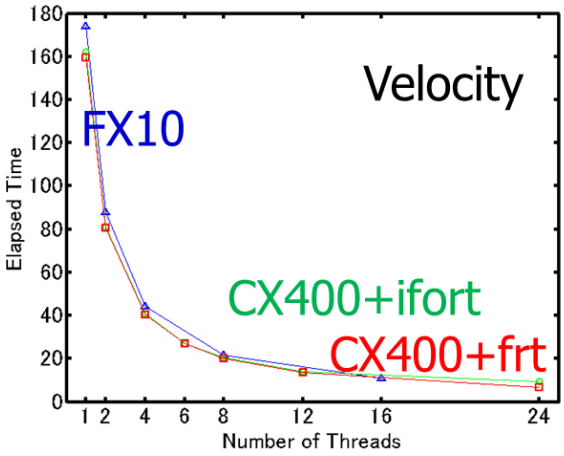

図 3：ソートした 144,000,000 個の粒子を 10 時間ステップ解いた場合の Velocity サブルーチンの経過時間.

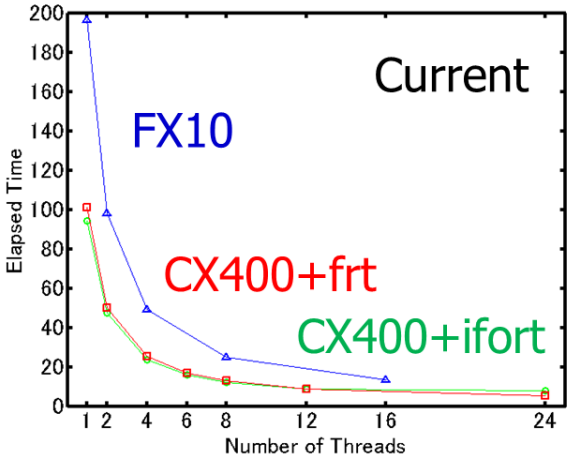

図 4：ソートした 144,000,000 個の粒子を 10 時間ステップ解いた場合の Current サブルーチンの経過時間.

Velocity の最大スレッド数のスケーラビリティは，FX10 及び富士通コンパイラを用いた CX400 ではほぼ 100%となり，インテルコンパイラを用いた CX400 では約 70%に向上した．Current の最大スレッド数のスケーラビリティは，FX10 では約 95%，富士通コンパイラを用いた CX400 では約 80%，インテルコンパイラを用いた CX400 では約 50%にそれぞれ向上した．また単一スレッドの性能が FX10 で約 3 倍，CX400 で約 1.5 倍向上しており，粒子データのソーティングによりスカラ性能のみならずスレッド性能も向上することが示された．

4. おわりに

PIC コードは，宇宙空間に広く存在する無衝突プラズマの第一原理シミュレーション手法であるのみならず，プラ

ズマ科学分野外でも広く用いられている．プラズマは通常の荷電粒子の集団をまとめた超粒子として定義される．PIC シミュレーションはラグランジュ変数とオイラー変数が混在しており，計算性能の向上が困難である．本研究では特に，PIC コードのスカラ性能及びスレッド性能に着目した性能評価を行った．その結果，格子データへのランダムアクセスを無くすように粒子データをソートするより，スカラ性能のみならずスレッド性能も向上することが分かった．また粒子データのソーティングの効果は，ランダムロードよりもランダムストアに対してより有効であることが分かった．

今後の課題として, 平成 27 年度より名古屋大学において稼働する FX100 及び CX400/2550 における性能測定が挙げられる．また，高速かつスレッド化が容易なソーティングアルゴリズムの開発も，今後の重要な課題として挙げられる．

謝辞 本研究は，科学研究費補助金・基盤研究(B) No. 26287041 によりサポートを受けている．ベンチマークテストに使用したスーパーコンピュータシステム(Fujitsu FX10 及び CX400)の計算リソースは, 名古屋大学太陽地球環境研究所計算機利用共同研究及び九州大学情報基盤研究開発センター先端的計算科学研究プロジェクトにより提供された．

参考文献

1. Nakashima, H., Miyake, Y., Usui, H., Omura, Y. : OhHelp: A Scalable Domain-Decomposing Dynamic Load Balancing for Particle-in-Cell Simulations., *Proc. Intl. Conf. Supercomputing*, 90—99 (2009).
2. Fujimoto, K.: A new electromagnetic particle-in-cell model with adaptive mesh refinement for high-performance parallel computation, *J. Comput. Phys*., Vol.230, 8508—8526 (2011).
3. Yee, K. S., Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Antenn. Propagat.*, Vol.AP-14, 302—307 (1966).
4. Boris, J. P.: Relativistic plasma simulation-optimization of a hybrid code, *Proc. Fourth Conf. Num. Sim. Plasmas*, ed. by J. P. Boris and R. A. Shanny, pp.3—67, Naval Research Laboratory, Washington D. C. (Nov. 1970).
5. Umeda, T., Omura, Y., Tominaga, T., Matsumoto, H. : A new charge conservation method in electromagnetic particle-in-cell simulations, *Comput.Phys.Commun*., Vol.156, 73—85 (2003).
6. Bowers, K. : Accelerating a particle-in-cell simulation using a hybrid counting sort, *J. Comput.Phys*, Vol.173, 393—411 (2001).