情報処理学会論文誌　コンピューティングシステム　Vol. 1　No. 2　250–260 (Aug. 2008)

プラズマ粒子シミュレーション電流計算のOpenMP並列化手法

臼井英之[†1]　杉崎由典[†2]　冨田清司[†2]
大村善治[†1]　三宅洋平[†1]　青木正樹[†2]

プラズマ粒子シミュレーションで電磁界成分を更新する際には電流値が必要であり，そのために，個々の粒子の運動量を各空間格子点に集める必要がある．しかし，粒子が空間的にランダムに分布しているため，電流計算の並列演算による高速化は容易ではなく工夫を要する．本論文では，粒子の位置情報を利用して各スレッドに粒子を明示的に割り当てるスレッド並列化アルゴリズムを新しく提案し，OpenMP を用いた実装によりその有効性を検証した．動作検証により，提案手法の CPU 台数効果はシミュレーション内の空間格子数の影響を受け，粒子数密度の影響はないことが分かった．特に，各スレッドに割り当てられた空間格子配列がキャッシュに収まりきる程度に細分化される場合，並列台数効果を得やすいことを明らかにした．特に並列台数 10 前後の場合，その台数効果はスーパリニアとなり，自動並列化コンパイラを用いた電流ルーチン実装に比べて高速になることを明らかにした．また，本提案手法は，各スレッドで全粒子を走査する冗長的な並列化方法であるため，従来アルゴリズムで用いられていた作業領域用配列は不要となり，シミュレーションに必要なメモリ容量を大幅に節約できることを示した．

OpenMP Parallelization Method for Current Calculation in Plasma Particle Simulation

Hideyuki Usui,[†1] Yoshinori Sugisaki,[†2]
Seiji Tomita,[†2] Yoshiharu Omura,[†1] Yohei Miyake[†1]
and Masaki Aoki[†2]

In Particle-In-Cell (PIC) plasma simulations, we calculate the current density to advance the electromagnetic fields. One of the ways to obtain the current density is to gather the velocity moment of each particle to the adjacent grid points. The current calculation is not basically parallelized because the particle positions, which are random in the simulation space, are independent of the array number of current density. In the present paper, we propose a new parallelization method which explicitly associates particles to threads by using OpenMP and evaluate the performance of the proposed method. We clarified that the scalability performance is affected by the number of spatial grid points and is independent of the number of particle per grid. In the proposed method, each thread is in charge of a part of the array of current density divided with the number of thread. When the memory size of the array allocated to each thread becomes small and close to the data cache size of CPU, we found that the scalability performance shows super-linear characteristics and the execution needs less time than the case of using the automatic parallelization compiler. In addition, each thread redundantly scans the particle array to obtain the information of the particle positions for assigning the corresponding particles in charge. Because of this redundant parallelization, we do not have to use work arrays and can save the memory consumed for simulations.

1. はじめに

宇宙プラズマ，核融合，プロセスプラズマ分野などにおけるプラズマ現象の定量解析では，数値シミュレーションが大きな役割を果たす．プラズマの運動論的効果を取り入れた解析を行うためには，プラズマを粒子として扱う必要があり，その方法の 1 つとして，PIC（Particle-In-Cell）手法[1] が広く利用されている．プラズマと電磁界の相互作用を解析対象とするプラズマ電磁粒子モデルのシミュレーションでは，FDTD 法でマックスウェル方程式を時間空間的に解き進めるとともに，得られる電磁界を使って各プラズマ粒子の運動方程式を解き，速度，位置更新を行う[2),3)]．**図 1** に電磁粒子シミュレーションの概念図を示す．

プラズマ粒子は，各格子点で定義されている電磁界成分とは違い，対象領域内のランダムな位置に分布している．そのため，各粒子の運動方程式を解く際には，粒子位置の隣接の空間格子に定義された電磁界成分を粒子位置に補間する必要がある．また，運動方程式によって更新された各粒子の速度値は，隣接する格子点に配分され電流値が計算される．この電流値をマックスウェル式内の電流項として用いることにより，プラズマ粒子と電磁場の相互作用を矛盾なく解き進めることができる．

近年の並列計算機の高性能化にともない，プラズマ粒子モデルの並列シミュレーションが

†1 京都大学生存圏研究所
Research Institute for Sustainable Humanosphere, Kyoto University
†2 富士通株式会社
FUJITSU LIMITED

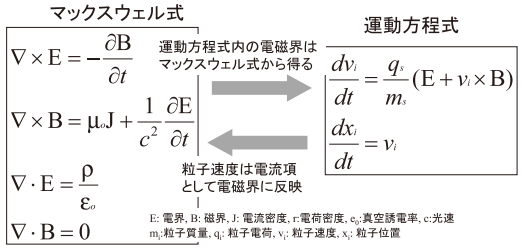

図 1　プラズマ電磁粒子シミュレーションで用いる基礎方程式
Fig. 1　Basic equations used in plasma electromagnetic Particle-In-Cell simulations.

様々な分野で試みられており[4)–6)]，その並列化手法についても議論されている[7)–13)]．特に，PC クラスタのように分散メモリ型の並列計算機を用いたプラズマ粒子シミュレーション手法については，負荷バランスの実現手法を含め，様々な検討がなされてきている[14)–17)]．

プラズマ粒子シミュレーション，特に電磁粒子モデルにおいて上述した電流値計算部が最も演算コストが高く，一般にその並列化は困難である．後で詳細を述べるが，並列化が困難な理由は，プラズマ粒子配列に格納された粒子位置がシミュレーション空間内においてランダムであり，場の空間配列番号との間に相関がないためである．また，このことに加え，粒子数が格子点数よりも格段に多いため，相対的に演算コストが高い．この電流値計算部の並列高速化については，すでにベクトルプロセッサを用いた手法が提案され実用化されている[12),13)]．

本論文では，スカラー並列機の中でも特に SMP マシンに着目し，OpenMP を用いて粒子位置とスレッドを明示的に関連付けることにより，この電流計算の並列化手法の開発を富士通のテクニカルチームとともに行った．本論文ではその具体的な手法を紹介するとともに，その性能を富士通社製の PRIMEPOWER HPC2500（SPARC64V 1.82 GHz，L1 データキャッシュ：128 KB，L2 キャッシュ：3 MB）を用いて検証した．

2. 電流計算アルゴリズムと自動並列化コンパイラによる実装

プラズマ粒子シミュレーションでは，多数のプラズマ粒子を代表する，いわゆる代表粒子を用い，これをシミュレーション空間内に多数配置することによりプラズマの集団的振舞いを模擬する[1)]．**図 2** に代表粒子の 1 次元モデル例を示す．代表粒子は空間関数によって決められる形状を持ち，その関数に従って，各粒子の運動量は粒子位置の近傍の格子点に配分される．本論文では，以後，代表粒子のことを単に粒子またはプラズマ粒子と呼ぶことに

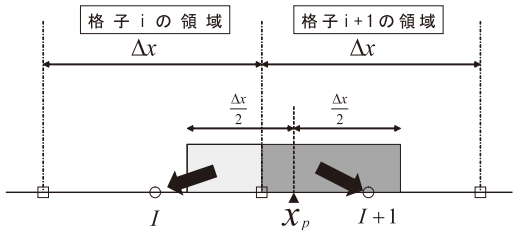

図 2　代表プラズマ粒子の概念と粒子情報の空間格子点への配分
Fig. 2　Concept of a superparticle and the distribution of the particle information such as momentum to adjacent grid points, I and I+1.

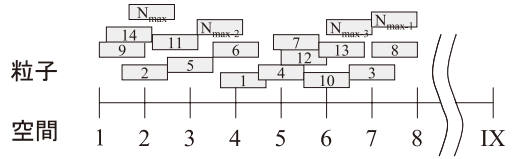

図 3　粒子群の位置とその配列番号，および場の空間配列との関係の一例
Fig. 3　Example of relation between array for plasma particle positions and those for electromagnetic fields including current density.

する．**図 3** に，粒子群の位置とその配列番号，および場の空間配列との関係の概念図を示す．粒子位置がシミュレーション空間内においてランダムであり，場の空間配列番号との間に相関がないことが分かる．また，粒子位置は時間とともに変化する．一方，プラズマ粒子シミュレーションでは，数値的なノイズを抑えるため，空間格子点あたり数百個以上の粒子数が望ましいとされており，最低でも数十個は必要である．すなわち，全空間格子数に比べてプラズマ粒子数は格段に大きく，プラズマシミュレーションの高速化は粒子計算部の効率化に依存しているといっても過言ではない．

次に，具体的な電流計算手法の基本をプログラム 1 に沿って示す．まず，粒子配列 X から粒子の中心位置（図 3 の X_p）を読み出し，それに一番近い空間格子点番号を変数 I に入れる．別途用意されている電流配列 Flx の要素 $Flx(I)$ とその隣の $Flx(I+1)$ に粒子の速度量 V をある重み関数 $ShapeFunc1$ および $ShapeFunc2$ に従ってそれぞれ配分する．この操作を粒子数回（$Nmax$）行う．問題は，Do ループの変数である N（粒子データが格納されている配列番号）と粒子位置から得られる空間格子位置 I と $I+1$ とは相関がないこ

とであり，そのため，この計算は単純に並列計算を行うことができない．

```
プログラム 1：電流計算基本ルーチン
!N:粒子の背番号，Nmax:総粒子数
!X:粒子位置，V:粒子速度
!Flx:電流
!ShapeFunc1&2: 重み関数（粒子位置に依存）
!
Do N=1, Nmax
 I=int(X(N))
 S1=ShapeFunc1(X(N))
 S2=ShapeFunc2(X(N))
 Flx(I   )=Flx(I   )+S1*V(N)
 Flx(I+1 )=Flx(I+1 )+S2*V(N)
Enddo
```

この電流計算の並列化手法アルゴリズムとしては，(1) 粒子配列番号により各スレッドの担当粒子を均等に割り当てる手法，(2) 粒子配列に格納された位置情報により各スレッドの担当粒子を明示的に割り当てる手法，の 2 つが考えられる．これらのアルゴリズムを実装する手段としては，自動並列化コンパイラを用いる方法と OpenMP を用いる方法がある．自動並列化コンパイラを用いる場合，基本的には Do 文の分割により並列化を行う．そのため，粒子配列を Do 文により分割するアルゴリズム (1) の並列化手法に適合性がある．

一方，アルゴリズム (2) では，各スレッドで全粒子の位置を走査し，ある空間範囲にある粒子だけを処理する．この冗長な処理の並列化を簡単に実現するには OpenMP が適している．自動並列化コンパイラを用いたアルゴリズム (2) の実装も可能ではあるが，OpenMP 利用の場合と比較して複雑となるため，本論文では OpenMP を用いた実装およびその評価を行うことにした．アルゴリズム (2) について実装方法による違いは少ないと推定されるが，今後検証が必要である．次章では，OpenMP を用いたアルゴリズム (2) の実装方法についてその詳細を述べる．

また，本論文では，簡単のため，シミュレーション内の粒子分布は一様と仮定する．上述した (1) の手法は，粒子位置に依存せず，粒子が各スレッドに均等に割り当てられるため，粒子分布が空間的に非一様な場合でもスレッド間の負荷バランスは保たれる．一方，(2) の手法では，粒子分布に空間的な偏りがある場合，各スレッドに割り当てられる粒子数は必ずしも一様ではない．その場合，スレッド間での負荷の不均一が生じる可能性がある．実際の

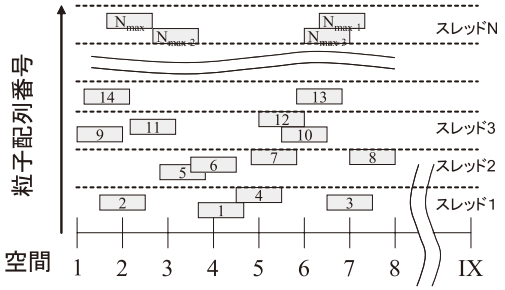

図 4　アルゴリズム (1) を自動並列化コンパイラを用いて実装する場合の各スレッドと粒子配列の関係
Fig. 4　Relation between threads and array elements of particles for the algorithm(1) with automatic parallelization compiler.

プラズマシミュレーションでは，様々な要因によるプラズマ不安定性とそれによるプラズマ波動と粒子の相互作用が存在するため，局所的に見ると粒子分布が必ずしも一様ではない．この粒子分布の不均一性によるアルゴリズム (2) への影響については後述する．

本章では，アルゴリズム (1) を自動並列化コンパイラを用いて実装する方法について述べる．プログラム 2 にその詳細を示す．

まず，並列スレッド数分の配列を電流配列 Flx に 1 次元増加させた作業配列 $FlxW$ を用意する．この作業配列は，スレッドごとに電流配列を用意することと等価なので，自動並列化コンパイラを用いた場合，これらの同時アクセスは発生せず自由に書き込むことができ，並列化が可能となる．その概念を**図 4** に示す．

各スレッドが担当する範囲は，粒子数をスレッド数で割った数 $iLen$ である．Do ループの変数 $iPara$ が各スレッドに対応し，それに相当する電流作業配列が呼び出され，そこに粒子の速度量 V が重み関数に従って書き込まれることになる．このようにして作業配列を用いて各スレッドにおいて計算された結果を総和（reduction）演算により総計し各格子点での電流値 Flx を得る．ただし，この手法では，問題規模が大きくなると，追加した作業配列のサイズが大きくなり，全体としてメモリ使用量が増大することになることが欠点である．

プログラム 2：アルゴリズム (1) の自動並列化コンパイラによる実装

```
! iPara  : スレッド番号
! Nthread: 全スレッド数
! Nx     : 空間格子数
! FlxW   : 電流計算用作業配列
! iLen   : 1 スレッドが担当する粒子数
!
---- 作業配列の初期化
do iPara = 1,Nthread
  do i = 1,Nx
     FlxW(i, iPara) = 0.0
  end do
end do
---- 粒子ループ
   iLen = Nmax / Nthread
do iPara = 1, Nthread
  ms = (iPara-1)*iLen+1
  me = min(iPara*iLen,Nmax)
do i = ms, me
  ii = X(i)
  S1 = ShapeFunc1(X(i))
  S2 = ShapeFunc2(X(i))
  FlxW(ii,   iPara)
     =FlxW(ii,   iPara)+S1*V(i)
  FlxW(ii+1, iPara)
     =FlxW(ii+1, iPara)+S2*V(i)
end do
end do
---- 作業配列値を reduction によって総計
do iPara = 1,Nthread
do i = 1,Nx
  Flx(i)= Flx(i)+ FlxW(i, iPara)
end do
end do
```

3. OpenMP を用いた電流ルーチンの実装

本章では，前章で述べたアルゴリズム (2) を OpenMP を用いて実装する方法を述べる．OpenMP を用いた実装では，作業配列 $FlxW$ を用いることなく，粒子の位置情報とスレッド番号を明示的に関連付けることができる．概念図を**図 5** に示す．

まず，粒子位置情報によりその粒子を処理するスレッド番号を割り付けることにより，全粒子を各スレッドへ振り分ける．各スレッドに振り分けられた粒子はそれぞれ並列に処理される．実際例をプログラム 3 に示す．

プログラム 3：アルゴリズム (2) の OpenMP による実装

```
! iNum      : スレッド数
! iThread   : スレッド番号
! iWidth    : 1 スレッドが担当する粒子数

!$omp parallel
!$omp&private(m, iThread, I1, I2)
!$omp&private(S1,S2)
!$omp&private(iNum, iWidth, iStt, iEnd)
!$omp&private(TheadTbl1, TheadTbl2)
   iThread  = OMP_GET_THREAD_NUM()
   iNum     = OMP_GET_NUM_THREADS()
   iWidth   = Nmax/iNum

   iStt     = 1 + ( iWidth * iThread)
   iEnd     = iStt +1
   if( iEnd < 1 )  iEnd = Nmax
    m       = iStt

Do while (m  /=  iEnd)
   I1 = INT(X(m))
   I2 = I1+1
   TheadTbl1  =   mod(I1, iNum)
   TheadTbl2  =   mod(I2, iNum)
if(TheadTbl1 = iThread) then
   S1     = ShapeFunc1(X(m))
   Flx(I1)= Flx(I1)+S1*V(m)
endif
if(TheadTbl2 = iThread) then
   S2     = ShapeFunc2(X(m))
   Flx(I2)= Flx(I2)+S2*V(m)
endif
   m = m +1
   if( m > Nmax ) m = 1
Enddo
!$omp end parallel
```

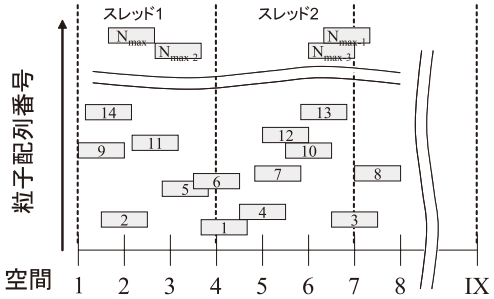

図 5　アルゴリズム (2) を OpenMP を用いて実装する場合の各スレッドと粒子配列の関係
Fig. 5　Relation between threads and array elements of particles for the algorithm(2) with OpenMP.

簡単化のために x 方向の 1 次元モデルを考える．まず，各スレッドにおいてすべての粒子の位置情報を走査する必要があるため，メモリアクセス競合が生じないように各スレッドが参照開始する粒子配列番号をスレッドごとにずらす（$iStt$ から $iEnd$ までの $Nmax$ を走査）．次に，実行ループ内では，粒子位置情報から得られる空間格子番号を全スレッド数（$iNum$）で割って得られる余り（mod 関数利用）によって粒子の各スレッドへの振り分けを行う．その値が，$TheadTbl1$ および $TheadTbl2$ に入る．注意すべき点は，このフィルタリングは各スレッドが行う処理であり冗長性を持つ．次に $TheadTbl1$ および $TheadTbl2$ の値が，それぞれのスレッド番号（$iThread$）と一致すれば，Flx 配列への速度情報（V）の足し込みを直接行う．ここで，自動並列方法で用いたような作業配列（$FlxW$）を使っていないことに注意されたい．この際，各スレッドにおいて各粒子がアクセスする Flx 配列要素が違うためメモリアクセス競合がない．実行ループは，各スレッドにおいて全粒子数分のループ回数が必要であり冗長であるが，前述した自動並列方法のように余分な作業配列およびその内容の足し込み計算は不要であるため，高効率演算が期待される．

4. 評価テスト

3 次元シミュレーションモデルを用いて，上述の OpenMP による電流計算性能を評価した．評価に用いたコンパイラは，ParallelNavi 2.4.0 Fujitsu Fortran Version 5.6 であり，コンパイルオプションは-Kfast_GP2=3 -Klargepage=2 -KV9 -Khardbarrier -Kparallel=3 -X9 とした．粒子の位置情報は z 方向成分のみに着目し，その値を用いたスレッドへのフィルタリングを行った．なお，x 方向，y 方向の空間格子数はそれぞれ 32 と固定し，z 方向

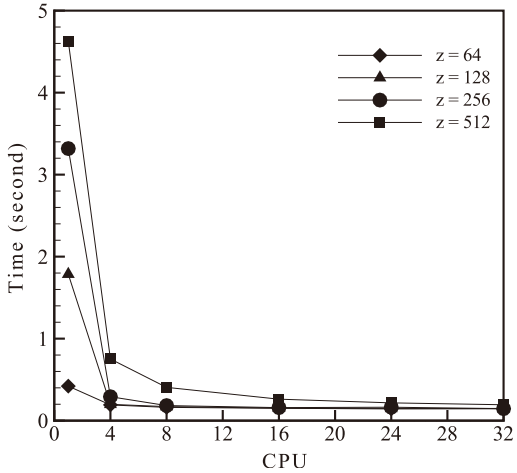

図 6　OpenMP を用いた電流並列化ルーチンの特性評価：格子あたりの粒子数を 1 に固定し，z 方向格子数を変化させた場合における演算時間の CPU 依存性
Fig. 6　Performance of the current density routine parallelized with OpenMP: Dependence of execution time on the number of CPU for different numbers of grid along z direction. In all cases, the number of particle per grid is fixed as one.

の格子数をパラメータとした．3 次元空間では粒子分布は一様であるとする．**図 6** は，粒子密度すなわち格子あたりの粒子数を 1 個に固定した場合において，全粒子の処理にかかる時間（秒）の CPU 数依存性を示したものである．z 方向の格子点数が 64，128，256，512 の 4 通りについて，粒子処理時間の CPU 数依存性を調べた．

まず，全体の傾向として，いずれの場合においても，CPU 数が増加すると，1CPU 数が担当する格子数が少なくなるとともに処理すべき粒子数も少なくなり，結果として全体の処理時間が短くなる．当然ながら，z 軸格子点数が多い場合は，演算時間が多くかかる．特筆すべき点は，いずれの場合においても，CPU 数が 16 までの範囲で CPU 数が増えると演算時間が短縮されるが，それ以上 CPU 数が増加しても，演算時間の大幅な短縮は見られず，一定値に収束する傾向があることである．$z = 512$，256，64 の場合について図 6 の拡大図を**図 7** に示す．また，図 7 で示した各処理時間を**表 1** に示す．

図から明らかなように，z 軸の格子数が大きくなると処理時間の収束点は右にずれる，すなわち，処理時間が最小になるためには，より多くの CPU 数を必要とするが，図 6 からも分かるように，処理時間の収束値は，いずれの場合もほぼ同じになる．CPU 数が多くな

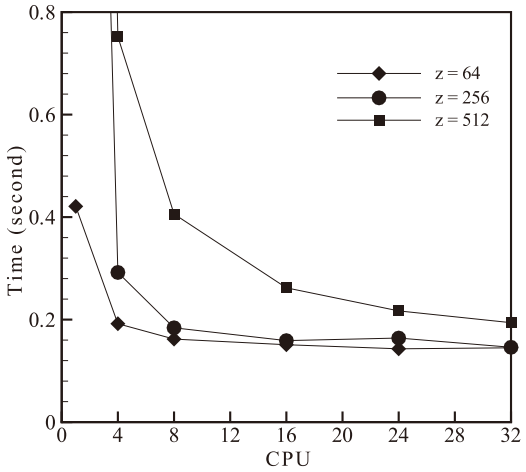

図 7　OpenMP を用いた電流並列化ルーチンの特性評価：格子あたりの粒子数を 1 に固定し，z 方向格子数を変化させた場合における演算時間の CPU 依存性（図 6 の拡大図）

Fig. 7　Performance of the current density routine parallelized with OpenMP: Dependence of execution time on the number of CPU for different numbers of grid along z direction. In all cases, the number of particle per grid is fixed as one (Enlarged figure of Fig. 6).

表 1　図 7 に示した演算時間の詳細（秒）

Table 1　Details of execution time shown in 図 7.

	並列数				
	4	8	16	24	32
z=64	0.19	0.16	0.15	0.14	0.14
z=256	0.29	0.18	0.15	0.15	0.15
z=512	0.75	0.40	0.26	0.22	0.19

り，1CPU が担当する処理量が各 CPU のキャッシュを効率良く使える程度少なくなってくると，メモリアクセス性能の向上により 1CPU の演算効率は急激に上がり，全体としての演算の高速化はさらに加速される．ただし，いったん，キャッシュ内での処理が可能になると，CPU 数を増やして 1CPU あたりの処理量を小さくしても，演算速度は収束すると考えられる．プログラム中において具体的にどの部分が処理速度の収束に関係しているかについては後ほど議論する．

次に，z 方向の格子数を 512 と固定した場合，上述の手法が粒子数密度の変化にどのように影響されるかを調べた．**図 8** には，各格子に 4，16，64 個の粒子を入れた場合それぞれについて CPU 数を変化させて調べた台数効果を示す．縦軸の性能は，それぞれの粒子密度の場合において，1CPU 利用時の速度を基準に示してある．多少性能に揺らぎはあるが，図から明らかなように，z 方向格子数一定の場合，台数効果は粒子数密度には依存しないことが分かる．

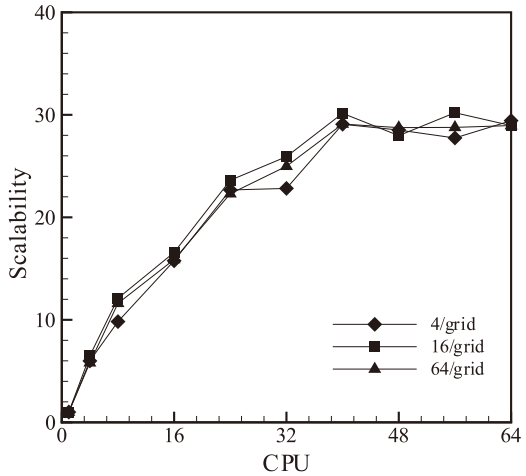

図 8　OpenMP を用いた電流並列化ルーチンの特性評価：z 方向格子数を 512 に固定し，粒子数密度を変化させた場合に場合における CPU 台数効果特性．各ケースにおいて，縦軸は CPU 数が 1 の場合の値を基準にした速度比

Fig. 8　Performance of the current density routine parallelized with OpenMP: Dependance of scalability on the number of CPU for different particle density. In all cases, the nubmer of grid along z direction is fixed as 512.

粒子数が多い場合，それに比例して演算量は増加することは明らかだが，1CPU 利用時の速度を基準にすると，CPU 数が増えることによって 1CPU が担当する格子点範囲が小さくなる割合はどの場合も同じであり，結果として台数効果の傾向はどの場合もほぼ等しくなる．

一方，自動並列手法では，プログラム 2 に示すようにスレッド数分に相当する作業配列を必要とするが，基本的には作業配列を用いた処理は台数効果が見られ，それによる高速化が見込まれる．空間格子数を固定し，1 格子あたりの粒子数を 8 個とした場合と 128 個について自動並列手法を用いて調べた台数効果を**図 9** に示す．メインループは作業配列を用

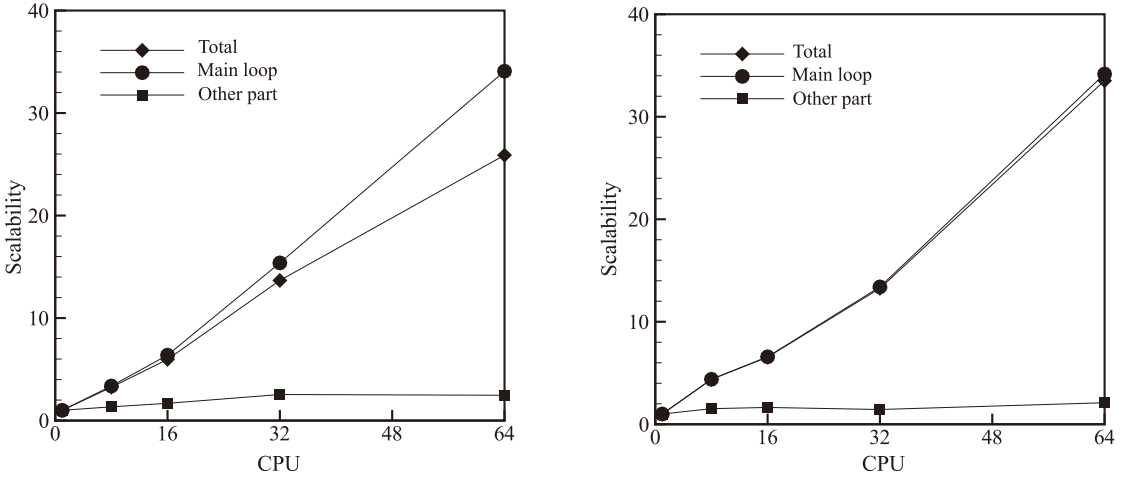

図 9　自動並列化コンパイラを用いた電流並列化ルーチンにおける計算量の内訳：1 グリッドあたりの粒子数が 8 個（左），128 個（右）
Fig. 9　Scalability performance of each part of the current routine parallelized with automatic parallelization compiler for cases of 8 (left) and 128 (right) particles per cell, respectively.

いた電流計算そのものの部分を示すが，メインループ以外は，作業配列の初期化，そして，その配列内容の総和（reduction）計算部を示す．メインループは CPU 増加にともない台数効果が顕著に見られる．しかし，作業配列成分はスレッド数に比例するため，メインループ以外の部分は CPU 数を増やしても実行時間の短縮は見込めない．全体として考えると，粒子数がスレッド数に比べて膨大に多い場合，作業配列を用いたメインループ処理の計算時間が，作業配列の初期化や reduction に費やされる演算時間よりもはるかに長くなり，右図に示すように，結果として全体として台数効果が得られる．一方，粒子数が比較的少ない場合，スレッド数を多くすると，作業配列の初期化や reduction という，CPU 数を増やしても実行時間の短縮が見込めない処理が目立つようになり，左図の Total 曲線で示されたように，結果として全体計算の台数効果が望めない．

自動並列手法では，スレッド数分に相当する作業配列を必要とする．x，y，z 方向それぞれの格子数を Nx，Ny，Nz，格子あたりの粒子数を $Npdns$，スレッド数を $Nthread$ とすると，演算に必要な電流用配列と粒子用配列のメモリ量の総和は $(Nx \times Ny \times Nz \times 8\,\text{bytes} \times 3$ 成分$)+(Nx \times Ny \times Nz \times Npdns \times 8\,\text{bytes} \times 6$ 成分$)$ であり，これに加えて作業配列として (電流用配列サイズ$\times\ Nthread$) が必要となる．具体的には，$Nx = Ny = 32$，$Nz = 64$，$Npdns = 32$，$Nthread = 16$ の場合は，作業配列サイズは 24 MB 必要となり，全体の 122 MB に対して 20% の割合を占める．$Nthead = 128$ の場合は，作業配列サイズは 192 MB に増加し，全体の 290 MB に対して 66% もの割合を占めることになる．本提案手法では，これらの作業配列を用いないため，その分の配列メモリ量を節約することができる．

5. 議　　論

粒子数を一定にしてスレッド数を増加させた場合，図 6 に示したように，OpenMP 手法を利用すると処理速度が収束する．その理由について検討を加えたので以下に示す．

プログラム 3 を見て分かるように，処理内において，配列 I の処理と並列に $I+1$ の処理も行う．$I+1$ の配列内容を参照することにより，多並列時に隣り合う CPU が同じ領域を参照する可能性があり，そのためキャッシュ競合が起きて性能向上を妨げているのではないか，という点をまず確かめた．

テストとして，$I+1$ の配列内容を参照し処理する部分を削除した場合について速度性能を測定した．具体的な処理速度は記さないが，メインルーチン内の if 文処理が半分になった

ためその分実行時間は短縮されることは確認できたが，結局は，図 6 で示したような処理速度の収束が見られた．この結果から，処理速度の収束は，隣り合う CPU が同じ領域を参照することによるキャッシュ競合ではなく，それ以外の別の要因があることが考えられる．そこで次に以下のようなテストを行った．

プログラム 3 内の全粒子数（変数 m）で回すループ内において，粒子位置から粒子を処理するスレッドを決める部分 (a) を

```
I1 = INT(X(m))
I2 = I1+1
TheadTbl1  =   mod(I1, iNum)
TheadTbl2  =   mod(I2, iNum)
```

if 文でスレッドごとに実行する/しないを設定している部分 (b) を

```
if(TheadTbl1. eq. iThread) then
    S1=ShapeFunc1(X(m))
    Flx(I1   ) =   Flx(I1  )+S1*V(m)
endif
```

と定義し，スレッド数を変化させてそれぞれについての実行時間を測定し，その台数効果を調べた．具体的には，3 次元モデルで x，y，z 方向の格子数をそれぞれ 32，32，128 とし，格子あたりの粒子数を 8 とした場合，1CPU で実行した 1 ステップあたりの実行時間を測定した．その結果，(a)，(b) それぞれ約 0.06 秒，0.76 秒かかり，1CPU 利用時は実行時間比はほぼ 1 : 10 となった．

(a) の部分は，各スレッドにおいてすべての粒子について処理を行うため，スレッド増による並列高速は望めない．一方，(b) の部分については，if 文判定そのものは全粒子数分必要であるものの，スレッド数の増加にともない，if 文内の処理を受ける粒子数は減少し，結果として並列高速化が見込まれる．

すなわち，スレッド数増加により (b) の実行時間は短縮されるが (a) の実行時間はほとんど変化がないため，あるスレッド数以上になると，(a) の部分の実行時間が (b) の部分の実行時間よりも長くなり，全体として，台数効果が出にくい結果となる．このことが，処理速度の収束に直接関係していると考えられる．

また，CPU 数を増やし，処理速度の収束に至る前段階として，1CPU のキャッシュにその CPU が担当する範囲の電流用配列が収まりきる状況が発生すると，演算効率の最大化が見込まれる．今回用いたパラメータでそれを検証してみる．

表 2　電流配列のメモリサイズを CPU 数で割った値．単位は MBytes
Table 2　Memory size of the current density array divided by the number of CPU.

	並列数		
	1	2	4
z=64	1.5	0.75	0.375
z=256	6.0	3.0	1.5
z=512	12.0	6.0	3.0

格子点数は x，y は固定でどちらも 32 とし，z 方向の格子点数を変数とする場合，倍精度で宣言されている電流配列全体が占める大きさは $32 \times 32 \times z \times 8$ bytes $\times$ 3 成分であり，各スレッドが担当する配列要素分はこれを CPU 数で割ったサイズである．**表 2** に $z = 64$，256，512 の場合について，1CPU が担当する電流配列メモリサイズを示す． このサイズがキャッシュサイズに近づけば，電流配列の一部はキャッシュに収まることになり，演算効率の上昇につながる．

PRIMEPOWER HPC2500 の場合，キャッシュサイズは 3 MBytes であり，$z = 512$ を例にとると，CPU 数が 4 のときにキャッシュサイズとほぼ同じサイズになり，それ以上の CPU 数になると，1CPU あたりが担当する電流配列要素はキャッシュサイズより小さくなる． **図 10** の $z = 256$ および 512 の場合に顕著に見えるが，CPU 数が 4，8 あたりでは，演算性能は CPU 数の増加度合い以上に上昇し，いわゆるスーパリニアな台数効果性能が確認できる．これは，電流配列がキャッシュ上に収まる効果によるものである．$z = 64$ の場合は，1CPU 時の場合からすでに電流配列がキャッシュに収まっている状態でありキャッシュ効果は含まれているため，CPU 数を多くしてもキャッシュの効果による台数効果は見られない．

また，図 10 に示されたスーパリニアな台数効果特性について，その実行時間を**表 3** に示す．なお，表の括弧内の値は，アルゴリズム (1) を自動並列化コンパイラにより実装して得られた実行時間を示す．表から分かるように，1 スレッドのみの場合は，アルゴリズム (1) を自動並列化コンパイラで実装した方が高速であるが，複数スレッドによる並列演算の場合，スーパリニアな特性を示すアルゴリズム (2) の OpenMP 実装方式の方が高速になっている．すなわち，並列台数が 10 前後の小規模スレッド数の場合では，アルゴリズム (2) の OpenMP 実装方式が有効であるといえる．

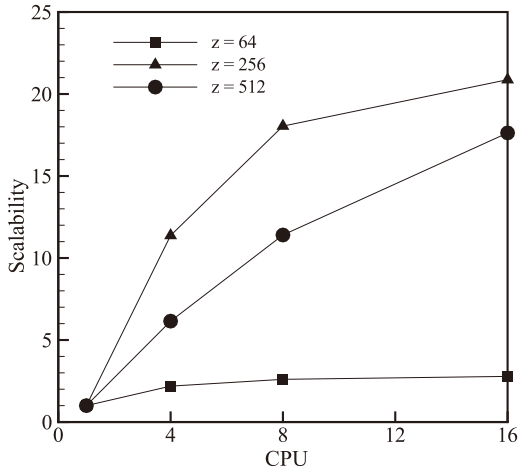

図 10　OpenMP を用いた電流並列化ルーチンの特性評価：格子あたりの粒子数を 1 に固定し，z 方向格子数を変化させた場合における CPU 台数効果特性．各ケースにおいて，縦軸は CPU 数が 1 の場合の値を基準にした速度比

Fig. 10　Performance of the current density routine parallelized with OpenMP: Dependance of scalability on the number of CPU for for different numbers of grid along z direction. In all cases, the number of particle per grid is fixed as one.

プラズマシミュレーションは，ビーム成分などの非マックスウェル速度成分やプラズマの温度異方性などが要因となって生じるプラズマ不安定性現象の定量解析に主として用いられる．シミュレーションでは，プラズマ不安定性により生じる波動粒子相互作用により，最大成長率を持つプラズマ波動の波長スケールのプラズマ密度変動が生じると予想される．このプラズマ密度変動のスケールが，1 スレッドが担当する空間領域より小さい場合は，密度変動の効果は平均化されるため各スレッドが担当する粒子数に大きな変動がなく，スレッド間の負荷バランスは大きくくずれないと考えられる．ただし，波動粒子相互作用が非線形の領域に達し，密度変動が初期の密度に対して無視できない場合や，変動スケールが 1 スレッド担当分の空間領域より大きい場合は，各スレッドが担当する粒子数が不均一となる可能性があり，スレッド間の負荷バランスはくずれる．この場合，本提案手法による並列演算の実行は可能ではあるが，スレッド間負荷バランスが悪くなるため演算効率の低下が予想される．すなわち，スレッド数を多くして各スレッドの担当空間領域を狭くしすぎると，スレッド間の負荷バランスが悪くなる可能性があるため注意する必要がある．負荷バランスを保つためには，1 スレッドの担当領域が，最大成長率を持つプラズマ波動の波長スケール以上になるように設定することが考えられる．

表 3　OpenMP を用いた電流並列化ルーチンの特性評価：格子あたりの粒子数を 1 に固定し，z 方向の格子数を変化させた場合における実行時間（秒）ただし，括弧内の時間はアルゴリズム (1) を自動並列化コンパイラにより実装して得られた実行時間

Table 3　Performance of the current density routine parallelized with OpenMP: Dependence of execution time on the number of CPU for for different numbers of grid along z direction. In all cases, the number of particle per grid is fixed as one. Numbers in the parentheses imply execution times for the perfomances of the algorithm(1) with automatic parallelization compiler.

	並列数		
	1	4	8
z=256	3.32	0.29	0.18
	(2.77)	(0.66)	(0.42)
z=512	4.63	0.75	0.41
	(3.11)	(0.98)	(0.56)

6. ま と め

本論文では，プラズマ粒子シミュレーションにおいて並列高速化の妨げとなっている電流計算ルーチンについて，粒子の位置情報により各スレッドに担当粒子を明示的に割り当てるアルゴリズムを提案し，OpenMP を用いた実装によりその性能評価を行った．

動作検証の結果，提案手法は，各 CPU に割り当てられた電流値配列がキャッシュサイズ程度に細分化される場合，並列台数効果を得やすいことを明らかにした．特に今回のモデルでは，並列台数 10 前後の比較的小規模の場合，その台数効果はスーパリニアとなり，自動並列化コンパイラを用いた電流ルーチン実装に比べて高速になることを明らかにした．ただ，本提案手法は，粒子位置取得のために各スレッドで全粒子を走査する冗長な並列化方法であるため，その部分が台数効果には寄与しない．そのため，電流値配列がいったんキャッシュに収まった状況で，さらにスレッド数を増加させても処理速度は増加せず，全体として台数効果が出にくくなる点には注意する必要がある．また，格子数を一定とした場合，台数効果は粒子数密度には依存しないことが分かった．

また，粒子配列番号により各スレッドの担当粒子を均等に割り当てる従来法の実装には，各スレッドに作業配列を用意する必要があるが，本提案手法の OpenMP を用いた実装には，その作業配列は不要となる．すなわち，シミュレーションに必要なメモリ容量を大幅に節約

できる．この利点により，大規模シミュレーションのモデリングを行う際に配列の利用において自由度が増えるとともに，計算機メモリ資源の有効利用にも寄与することができる．

プラズマ粒子シミュレーションにおいても，他の数値流体モデル同様，全体の空間領域をノード数に分割する領域分割による並列化が一般的になってきている．本提案手法を並列化に用いることにより，各ノードに従来定義していた作業配列を使うことなくノード内のスレッド並列の効率化を最大限に得ることができる．これによって全体の演算パフォーマンスの向上も大いに期待できる．

謝辞　本論文の手法開発およびその特性評価には，京都大学生存圏研究所の全国共同利用装置である電波科学計算機実験装置（KDK）を用いて行われました．

参 考 文 献

1) Birdsall, C.K and Langdon, A.B.: *Plasma Physics Via Computer Simulation*, Adam Hilger, Bistol (1991)
2) 内藤裕志：プラズマ計算機シミュレーション入門 I，2. 粒子シミュレーションの基礎，プラズマ・核融合学会誌，Vol.74, No.5, pp.470–478 (1998).
3) Omura, Y. and Matsumoto, H.: KEMPO1: Techinical Guide to One-Dimensional Electromagnetic Particle Code, *Computer Space Plasma Physics: Simulation Techniques and Software*, Matsumoto, H. and Omura, Y.(Eds.), pp.21–65, Terra Scientific Publishing Company (1993).
4) Candel, A.E., Dehler, M.M., and Troyer M.: A massively parallel particle-in-cell code for the simulation of field-emitter based electron sources, *Nuclear Instruments and Methods in Physics Research*, A558, pp.154–158 (2006).
5) Qiang, J., Ryne, R.D., Blind, B., Billen, J.H., Bhatia, T., Garnett, R.W., Neuschaefer, G., and Takeda, H.: High-resolution parallel particle-in-cell simulations of beam dynamics in the spallation neutron source linac, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Vol.457, No.1, 11, pp.1–11 (2001).
6) 西原功修，新宮　哲，川口正仁，岸本泰明，谷　啓二：高密度プラズマ粒子コード「SCOPE」の並列処理，プラズマ・核融合学会誌，Vol.72, No.9, pp.926–934 (1996).
7) 内藤裕志，徳田伸二：プラズマの粒子シミュレーションコードの並列化，ブラズマ・核融合学会誌，Vol.72, No.8, pp.737–743 (1996).
8) Wang, J., Liewer, P.C. and Decyk, V.K.: 3D electromagnetic plasma particle simulations on a MIMD parallel computer, *Comput. Phys. Commun.*, Vol.87, pp.35–53 (1995).
9) Messmer, P.: A Parallel Relativistic Fully 3D Electromagnetic Particle-in-Cell Code, *Applied Parallel Computing. New Paradigms for HPC in Industry and Academia*, Vol.1947/2001, DOI: 10.1007/3-540-70734-4-41, pp.350–355, Springer Berlin/Heidelberg, (2001).
10) 秋山　泰，三十尾潔高，大村善治，松本　紘，斎藤　稔，野口　保，鬼塚健太郎：宇宙プラズマ粒子シミュレーションの並列化，情報処理学会研究報告ハイパフォーマンスコンピューティング，IPSJ SIG Notes, Vol.97, No.37 (19970509), pp.1–6 (1997).
11) 上岡功治，村田健史，上田裕子，岡田雅樹，大村善治，松本　紘：プラズマ粒子シミュレーションコードの並列化モデルと速度向上率の評価，情報処理学会研究報告 MPS，数理モデル化と問題解決研究報告，IPSJ SIG Notes, Vol.2001, No.91 (20010917), pp.9–12 (2001).
12) 杉山　徹，寺田直樹，村田健史，大村善治，臼井英之，松本　紘：LISTVEC 指示行を使った多粒子シミュレーションの大規模化—主メモリを節約し，かつ高速化を可能にする 1 つの方法，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG 6 (ACS 6), pp.171–175 (2004).
13) Okada, M., Usui, H., Omura, Y., Ueda, H.O., Murata, T. and Sugiyama T.: Spacecraft Plasma Environment Analysis via Large Scale 3D Plasma Particle Simulation, *High-Performance Computing, Lecture Notes in Computer Science (LNCS)* Vol.4759/2008 DOI 10.1007/978-3-540-77704-5, pp.383–392, Springer-Verlag (2008).
14) Liewer, P.C. and Decyk, V.K.: A general concurrent algorithm for plasma particle-in-cell simulation codes, *Journal of Computational Physics*, Vol.85, pp.302–322 (1989).
15) Walker, D.W.: Characterizing the parallel performance of a large-scale particle-in-cell plasma simulation code, *Concurrency*, Vol.2, pp.257–288 (1990).
16) Plimpton, S.J., Seidel, D.B., Pasik, M.F., Coats, R.S. and Montry, G.R.: A Load-Balancing Algorithm for a Parallel Electromagnetic Particle-in-Cell Code, *Comp. Phys. Comm.*, Vol.152, pp.227–241 (2003).
17) Nakashima, H., Usui, H. and Omura, Y.: OhHelp: A simple but efficient space-partitioned dynamic load balancing for particle simulations, 情報処理学会研究報告ハイパフォーマンスコンピューティング，IPSJ SIG Notes, Vol.2007, No.122 (20071207) pp.25–30 (2007).

(平成 20 年 1 月 25 日受付)
(平成 20 年 5 月 5 日採録)

臼井　英之

昭和 39 年生．平成 4 年京都大学大学院工学研究科電子工学専攻博士後期課程研究指導認定退学．同年京都大学超高層電波研究センター（現，生存圏研究所）助手に着任．平成 11 年より京都大学超高層電波研究センター助教授．改組にともない，京都大学宙空電波科学研究センター助教授．改組にともない，京都大学生存圏研究所準教授．宇宙プラズマ環境のプラズマ粒子シミュレーション研究に従事．京都大学博士（工学）．地球電磁気・地球惑星圏学会（SGEPSS），米国地球物理学会連合（AGU），電子情報通信学会各会員．

杉崎　由典

昭和 37 年生．静岡大学工学部情報工学科卒業，富士通静岡エンジニアリング入社．現在，富士通（株）．prolog，VPPFortran ライブラリ，HPF ライブラリ開発．プログラムチューニングに従事．

冨田　清司

昭和 51 年生．中部大学経営情報学部経営情報学科卒業．(株）インタープロジェクト（現, (株）シーテック）入社．富士通（株）の協力会社メンバとして Fortran ライブラリ開発，プログラムチューニングに従事．

大村　善治

昭和 60 年京都大学大学院工学研究科博士後期課程電気工学第二専攻研究指導認定退学．同年京都大学工学部助手に着任．昭和 63 年京都大学超高層電波研究センター助手．平成元年京都大学超高層電波研究センター助教授．平成 12 年京都大学宙空電波科学研究センター教授．平成 16 年改組にともない京都大学生存圏研究所教授．宇宙プラズマ波動粒子相互作用のシミュレーション研究に従事．京都大学博士（工学）．地球電磁気・地球惑星圏学会（SGEPSS），米国地球物理学会連合（AGU），電子情報通信学会各会員．

三宅　洋平

昭和 57 年生．平成 18 年京都大学大学院工学研究科電気工学専攻修士課程修了．同年より京都大学大学院工学研究科電気工学専攻後期博士課程在学．衛星搭載用電界センサ特性の計算機シミュレーション研究に従事．日本学術振興会特別研究員．地球電磁気・地球惑星圏学会（SGEPSS），米国地球物理学会連合（AGU）各会員．

青木　正樹

1977 年富士通株式会社入社．科学技術計算分野の言語処理系の研究開発に従事．