ハイパフォーマンス
コンピューティング 82−32
（2000．8．5）

粒子コードの並列化における通信の隠ぺい

田上豊彦　坂上仁志　高橋　豊

姫路工業大学工学部情報工学科

近年，高い演算能力を持つ並列計算機が，実際の物理現象を解析するために必要不可欠な存在になってきている．一方，詳細でより高速な解析の必要性から，特別な超並列計算機での超大規模計算が必要とされている．超並列処理環境においては，並列処理における１％の並列化率の向上が大きな成果を得ることが分かっており，最適化により並列化率を向上させることが期待されている．

そこで，本論文では，２次元プラズマ粒子コードについて並列化を行い，さらにお互いに依存関係がない通信と計算を重ね合わせて実行する．この通信の隠ぺいにより，通信の占める割合を見かけ上減らし，並列化率を向上させることを目指す．

Hiding communication for paralleled particle code

Toyohiko TAGAMI, Hitoshi SAKAGAMI and Yutaka TAKAHASHI

Computer Engineering, Himeji Institute of Technology

It is necessary to use the parallel computers which have high performance for analyzing the real physical phenomenon. Analyzing the phenomenon more speedy and detail requires special large simulation with a super parallel computer. On the super parallel environment, only 1% efficient improvement make a good effort. So, a few performance up is expected by optimizing the code.

In this paper, we have paralleled 2-1/2 dimension plasma particle-in-cell code. In order to optimize this code, we hide the communication by over-wrapping calculation and communication which are executed indepently.

1. はじめに

近年，計算の大規模化・複雑化にともない，高い演算能力を持つ並列計算機が広く用いられている．特に，安価なワークステーションクラスタやSMP上での並列計算が注目され，実際の物理現象を解析するために必要不可欠な存在になっている [1]．一方，より詳細でより高速に解析するために，特別な超並列計算機での超大規模計算が必要とされている．多数のプロセッサを実装した並列計算機で，カタログ上高性能を達成することは容易であるが，1000台で実行しても1000倍の性能が単純に出るというわけではない [2]．超並列になればなるほどプログラムがどの程度並列に実行されるかが大きな問題となり，実際には，それほどの性能が出ない．これは，式(1)に示すアムダールの法則により明らかであり，全体の処理速度は低速部分に大きく影響される．

$$speedup = \frac{1}{(1-f)+\frac{f}{n}} \qquad (1)$$

ここで，fはプログラムの並列化率であり，speedupは全体の速度向上率，nはプロセッサ台数である．図1に並列化率が90％，99％，99.5％のときの，アムダールの法則による全体の速度向上率の変化を示す．

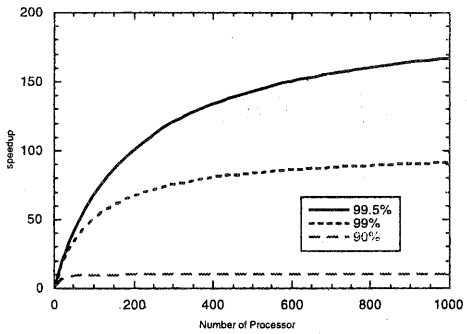

図 1: アムダールの法則

図 1 より，並列処理部分が 99 %でも，1000 台のプロセッサで得られるスピードアップは高々90 倍にしかならない．また，並列化率をわずか 0.5 %向上させて 99.5 %とすると，スピードアップは 170 倍となり，並列化率 99 %に比べて約 1.9 倍のスピードアップが得られる．

本論文では，2-1/2 次元プラズマ粒子コードの並列化を行う．超並列処理環境においては，図 1 から並列処理におけるわずかな並列化率の向上が大きな成果につながることは明らかである．このため，最適化によって少しでも並列化率を向上させると，超並列計算機の極めて高いパフォーマンスを引きだすことができる．そこで，お互いに依存関係がない通信と計算を重ね合わせて実行し通信時間の占める割合を見かけ上減らす通信の隠ぺいを行うことで最適化を行い，その性能を評価する．多くの並列計算機では，通信は専用の通信コントローラなどで CPU には依存せずに行われる．このため，計算と通信は独立して同時に実行でき，最適化による通信隠ぺいの顕著な効果が期待できる．

2. プラズマ粒子コード

2.1 概要

このプラズマ粒子コードは，日本原子力研究所関西研究所光量子グループによって開発され，レーザー・プラズマ相互作用の研究ため，大規模なプラズマ粒子シュミレーションに用いられている [3]．レーザーの強い電場のため原子は直接光電離し，またプラズマの加熱の時間スケールが電子の緩和時間より短いため，加熱された電子の速度分布関数は熱平衡分布（Maxwell 分布）から大きくずれている．このため，加熱されたプラズマの熱輸送や制動放射は，通常の公式には従わず非線形性の極めて強いものとなっている．さらに，相対論的な強度のレーザーを考えているため，レーザーにより誘起される電磁場は極めて強いので，電荷密度による静電場は無視できる．通常，静電場の計算には FFT が用いられるが，FFT は並列処理のために多くの通信が必要であり，並列効率を下げる要因となっている．しかし，このコードでは FFT が必要ないので，並列化に適している．また，粒子は 2 次元の空間 (x,y)，3 次元の速度空間に配置し，電磁場は 3 次元の空間を用いてシミュレーションを行っている．

このコードでは，物理量の時間変化を求めるにあたりリープフロッグ法を用いており，磁場 B，電流密度 J，運動量 P と電場 E，粒子の位置 x が，それぞれ時間ステップ 1/2 ずつで入れ子となっている．この様子を図 2 に示す．磁場，電流密度，運動量は，

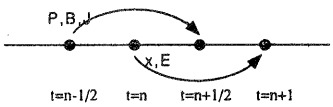

図 2: 各物理量の時間の定義点

時間ステップ $(t = n + 1/2)$ で定義され，電場と粒子の位置は，時間ステップ $(t = n + 1)$ で定義されている．空間については，スタガードメッシュを用いており，図 3 に示すように空間メッシュ1/2 ずつで入れ子となっている．電場の x 成分 E_x と磁場の

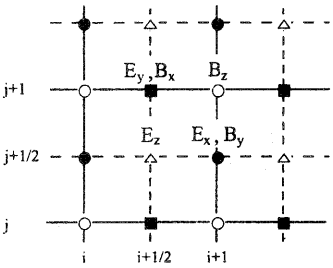

図 3: 各物理量の空間の定義点

y 成分 B_y は，空間 $(i+1, j+1/2)$ に，電場の y 成分 E_y と磁場の x 成分 B_x は空間 $(i+1/2, j+1)$ に，電場の z 成分 E_z は空間 $(i+1/2, j+1/2)$ に，そして，磁場の z 成分 B_z は空間 $(i+1, j+1)$ に定義されている．

このコードの主要部分は次のような流れとなっている．

1. 磁場の時間発展 (ハーフステップ)
   $[\ B^n \leftarrow B^{n-\frac{1}{2}}, E^n\ ]$
2. 運動量の時間発展
   $[\ P^{n+\frac{1}{2}} \leftarrow P^{n-\frac{1}{2}}, E^n, B^n\ ]$
3. 磁場の時間発展 (ハーフステップ)
   $[\ B^{n+\frac{1}{2}} \leftarrow B^{n-\frac{1}{2}}, E^n\ ]$
4. 粒子の時間発展 (ハーフステップ)
   $[\ x^{n+\frac{1}{2}} \leftarrow x^n, P^{n+\frac{1}{2}}\ ]$
5. 電流密度の計算
   $[\ J^{n+\frac{1}{2}} \leftarrow x^{n+\frac{1}{2}}, P^{n+\frac{1}{2}}\ ]$
6. 粒子の時間発展 (ハーフステップ)
   $[\ x^{n+1} \leftarrow x^n, P^{n+\frac{1}{2}}\ ]$
7. 電場の時間発展
   $[\ E^{n+1} \leftarrow E^n, B^{n+\frac{1}{2}}, J^{n+\frac{1}{2}}\ ]$
   (1 から 7 を繰り返す)

磁場の時間発展では，電場をもとに磁場を求めている．磁場の計算は運動量の計算の前後でハーフステップの計算を 2 回行っている．運動量の時間発展では，電場と磁場から粒子の運動量を求めている．粒子の時間発展では，運動量をもとに粒子の座標 (x, y) の計算を行う．粒子の計算は電流密度の計算の前後でハーフステップの計算を 2 回行っている．電流密度の計算では，各粒子の位置から電流密度を求めている．電場の時間発展では，磁場と電流密度をもとに電場を求めている．このため，磁場と位置はハーフステップの計算を 2 回行っており，完全なリープフロッグ法にはなっていない．境界条件は，x 方向，y 方向ともに周期的としている．

2.2 並列化手法

並列処理のためのデータ分割方法には，図 4 のようなブロック領域分割を用いる．本粒子コードでは，レーザーとの相互関係を考えているため，粒子の両側に真空領域を確保しなければならず，プラズマ粒子を x 方向の一部の領域に集中して配置している．

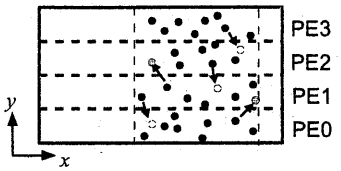

図 4: 1 次元ブロック領域分割法

このため，x 方向の分割を，粒子が均一になるようにすれば，場が不均一になり，逆に，場が均一になるようにすれば粒子が均一にならない．サイクリック分割を用いれば，粒子も場も均一にはできるが通信が非常に多くなってしまう．そこで，x 方向には分割を行わず，y 方向のみの 1 次元ブロック分割を行う．それぞれの領域を対応するプロセッサに割り当てて計算を行う．

また，通信は必要な場合に行う形式ではなく，計算が済んだ時点で領域の境界値を通信する方式をとっている．

この粒子コードが，プロセッサ間で行う通信は，粒子が他のプロセッサの領域に移動するときと電場や磁場や電流密度の計算で領域境界を処理するときに発生する．

粒子が他のプロセッサの領域に移動する場合は，移動した粒子の情報（運動量，位置）を移動先である上方と下方のプロセッサと交換する必要がある．このとき，周期的境界条件であるので，上端と下端の領域間でも情報の交換を行う．交換の方法は，移動した粒子情報をバッファにため，移動した粒子数と粒子情報をそれぞれ上方と下方のプロセッサに通信し，受け取った新たな粒子情報を追加することで行う．つまり，通信は隣り合うプロセッサ間とのシフト通信となっている．このような通信では，すべてのプロセッサで送信動作を行い，その後受信動作を行うと，デッドロックが起こる．これを回避するためには，例えば奇数番号のプロセッサが送信し偶数番号のプロセッサが受信を行ってから逆方向の送受信を行う方法が一般的である．また，デッドロックは，送信操作をノンブロッキングにすることで，送信待ち動作を無くすことでも回避できる．この 2 つについて性能を比較したところノンブロッキング送信を用いるほうが高速に実行できており，ここで

は，ノンブロッキング送信を利用してシフト通信を実現している．

電場や磁場や電流密度の計算では，領域の境界値を利用して計算を行っており，そのために，領域の境界値をお互いに通信する必要がある．

まず，電場と磁場について考える．図5に磁場の計算のプログラムを，図6に電場の計算のプログラムを示す．

```
do j=....
  do i=....
    Bx(i,j) = Bx(i,j) - c*dt*(Ez(i,j)-Ez(i,j-1))/dlt_yg
  enddo
enddo
do j=....
  do i=....
    By(i,j) = By(i,j) + c*dt*(Ez(i,j)-Ez(i-1,j))/dlt_xg
  enddo
enddo
do j=....
  do i=....
    Bz(i,j) = Bz(i,j) - c*dt*(Ey(i+1,j)-Ey(i,j))/dlt_xg
                      + c*dt*(Ex(i,j+1)-Ex(i,j))/dlt_yg
  enddo
enddo
```

図 5: 磁場の計算

```
do j=....
  do i=....
    Ex(i,j) = Ex(i,j) + cdlt_toy*(Bz(i,j)-Bz(i,j-1))
                      - dlt_t*rJx(i,j)
  enddo
enddo
do j=....
  do i=....
    Ey(i,j) = Ey(i,j) - cdlt_tox*(Bz(i,j)-Bz(i-1,j))
                      - dlt_t*rJy(i,j)
  enddo
enddo
do j=....
  do i=....
    Ez(i,j) = Ez(i,j) + cdlt_tox*(By(i+1,j)-By(i,j))
                      - cdlt_toy*(Bx(i,j+1)-Bx(i,j))
                      - dlt_t*rJz(i,j)
  enddo
enddo
```

図 6: 電場の計算

y 方向のブロック分割であるため，単純に分割すると，図5より，磁場の計算では，B_x を求めるために，下方向から E_z の領域境界値が必要であり，B_z を求めるために，上方向から E_x の領域境界値が必要である．また，図6より，電場の計算では，E_x を求めるために，下方向から B_z の領域境界値が必要であり，E_z を求めるために，上方向から B_x の領域境界値が必要である．このため，B_x, E_x（x 成分）の領域境界値を上方のプロセッサに，B_z, E_z（z 成分）の領域境界値を下方のプロセッサに通信している．

次に，電流密度の通信について述べる．領域付近にいる粒子の影響は隣接する領域にも及んでいる．たとえば，図7のように，粒子が領域の境界付近にいた場合，この粒子の影響は (i,j), $(i+1,j)$, $(i,j+1)$, $(i+1,j+1)$ の4点に及んでいる．

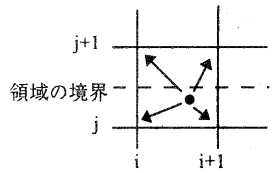

図 7: 粒子の境界における影響

$(i,j+1)$, $(i+1,j+1)$ は，隣接するプロセッサの担当範囲であるため，これらの影響を通信する必要がある．本粒子コードでは，電流密度は粒子の影響を足し込んでで求めている．そこで，まず粒子が配置しているプロセッサ内でのこれらの粒子の影響による電流密度を求め，その電流密度を隣接するプロセッサとの間で通信し，対応する電流密度にさらに足し込むことで実現する．

これらの並列化手法を MPI を用いて開発し，粒子コードを並列化する [4]．

2.3 性能評価

本学 UNIX 室のワークステーション群に MPI を実装させ，性能評価を行う．このワークステーションは，CPU が UltraSparc-IIi(270MHz) であり，ネットワークは 100BaseTX のスイッチを利用した fast イーサネットである [5]．それらのワークステーションを1～32台用いて実験を行う．今回は総粒子数を，2097152 個，メッシュサイズを 512×128 に固定し，プロセッサ台数を変化させて並列性能を計測する．実行時間を計測しそのスピードアップを図8に示す．なお，この実行時間の中には，初期化などの準備部分は含まず，主要な計算部分だけを利用している．

図8から計測を行った32台までは高い並列効率を得ており，台数倍以上のスーパースカラとなって

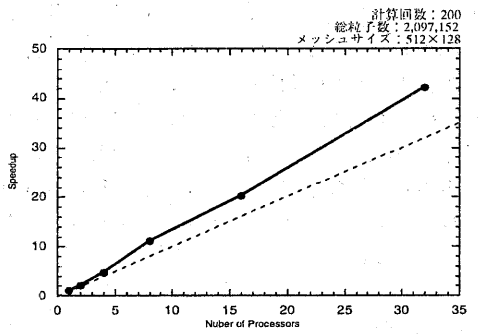

図 8: 粒子コードのスピードアップ

いる．これは，計算領域の減少によるキャッシュのヒット率の向上のためである．

この結果，単純な並列化だけでも良い性能が出ており，このままでも十分実用的である．

3. 通信の隠ぺい

単純な並列化だけでも良い性能は出ているが，ここでは，さらなる性能の向上を目指し，通信の隠ぺいを行う．一般に通信の隠ぺいは，ノンブロッキング通信を用いて実現する．これは，通信は起動するがプログラムが扱えるデータとメッセージバッファ間のコピーが完了する前に，呼び出し元に処理が戻ってくる操作である．ノンブロッキング通信により通信を起動し，通信が完了するまでの間に計算を行うことで通信を隠ぺいできる．ただし，データのコピーが完了する前に，そのデータに対して参照，代入を行うと，不正な実行結果となるので，注意が必要である．本論文でも，ノンブロッキング送信を利用することで隠ぺいを行っている．なお，受信操作は通常のブロッキング受信を用いている．受信操作もノンブロッキングにするとより隠ぺいを行うことができるが，受信操作では直後で受信データを利用していることが多く，ノンブロッキングにする効果がないために行っていない．次に，各部分について隠ぺい方法を説明する．

3.1 電場・磁場の領域の境界処理

図 5，6 より，磁場・電場では，x 成分と z 成分について通信する必要がある．しかし，磁場・電場とも，直後の計算で参照されているため，ルーチン内で処理する必要がある．このため，x 成分については，y 成分と z 成分の計算の重ね合わせることができるが，z 成分については，このままでは重ね合わせることはできない．y 成分の計算と順番を変えることにより，y 成分の計算と重ね合わせることはできるが，y 成分のみであり，十分な隠ぺいは行えない．そこで，本来は通信すべき z 成分の領域の境界値の通信を行わず，プロセッサ内でその値を重複して持つようにし，x 成分の通信のみで済むようにする．プロセッサ内で z 成分の領域の境界値を持つと，z 成分の計算に必要な x 成分の領域の境界値を通信する必要がある．これにより，通信データ量には変化はないが，x 成分のみの通信で済むようになる．この様子を図 9 に示す．

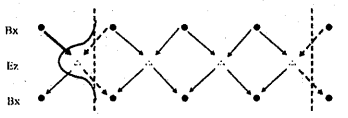

図 9: 電磁場の相関性

●は磁場の x 成分であり，△は電場の z 成分である．元々は，点線の部分で分割していたのを，実線の部分に変えて z 成分を隣のプロセッサと重複して持たすことで x 成分のみの通信となるようにしている．これにより，y 成分と z 成分の計算と重ね合わせることができ，隠ぺいの効果を得ることができる．

3.2 粒子の移動

本粒子コードでは 1 ステップの計算の中で粒子の移動ルーチンは電流密度の計算の前後に 2 度呼ばれる．1 度目に呼ばれるルーチンについては，直後の電流密度の計算と重ね合わせることで隠ぺいを行う．しかし，電流密度の計算では，粒子から電流密度を計算しており，粒子の移動と完全には独立していない．すなわち，単純に重ね合わせたら移動した粒子の分だけ電流密度に影響が出てしまう．ところで，電流密度は粒子の影響を足し込んで求めているということに注目すると，移動していない粒子の計算と移動した粒子の計算は別々に行うことができる．そこで，まず，移動していない粒子の計算を先に行い，これと粒子の移動による通信を重ね合わせて通

信し，その後，移動した粒子の計算を行い足し込むことで隠ぺいを実現させている．2度目に呼ばれるルーチンについても，直後の電場の計算と重ね合わせて通信を行っている．電場の計算は，粒子の移動とは独立しているため，単純に隠ぺいを行える．ただし，電場ではx成分の隠ぺいが行われており，そのために，2重の通信を行うこととなる．

3.3 性能評価

次のパターンについて評価を行う．

- 何も隠ぺいしない（non-hide 版）
- 電磁場の境界条件のみ隠ぺい（eb-hide 版）
- 粒子の移動のみ隠ぺい（m-hide 版）
- 電磁場，移動両方とも隠ぺい（ebm-hide 版）

それぞれのパターンをにおいて，同様に粒子数を，2097152 個，メッシュサイズを 512×128 に固定し，プロセッサ数が 8 台と 32 台の場合について各部分の実行時間を計測する．表 1 に 8 台の，表 2 に 32 台における電磁場と粒子の移動の部分の実行時間を示す．

表 1: 実行時間 (8 プロセッサ)　[sec]

パターン	電場	磁場	粒子の移動
non-hide	0.9867	1.1296	27.3855
m-hide	0.9919	1.1289	15.3856
eb-hide	0.9500	1.1264	25.1891
ebm-hide	0.9435	1.1345	15.4508

表 2: 実行時間 (32 プロセッサ)　[sec]

パターン	電場	磁場	粒子の移動
non-hide	0.7326	0.8107	15.5037
m-hide	0.7394	0.8049	9.2580
eb-hide	0.6277	0.6993	12.4913
ebm-hide	0.5409	0.6921	9.6850

この結果から通信を隠ぺいすることによりほぼ性能が向上していることが分かる．電場の境界処理を隠ぺいすることで 32 台で 1.26 倍程度，8 台でも 1.04 倍程度の実行時間の向上がみられ，また，磁場の境界処理を隠ぺいすることで 32 台で 1.16 倍程度の向上がみられた．しかし，8 台では 0.99 倍程度と逆に遅くなってしまっている．また，粒子の移動の部分を隠ぺいすることで 32 台で 1.47 倍程度，8 台では 1.70 倍程度の実行時間の向上がみられた．すなわち，8 台の磁場を除き，各ルーチンとも非常にわずかではあるが性能の向上が得られた．32 台程度では，これらの隠ぺいの差がほとんど性能に影響しないが，プロセッサ台数が増えるにつれて影響が大きくなっていくものと考える．

4. おわりに

超並列処理環境では，わずかな並列化率の向上で大きな成果を上げることができる．ここでは，通信の隠ぺいを行うことで並列化率の向上を目指した．2-1/2 次元プラズマ粒子コードについて並列化を行い，さらに電磁場の境界条件と粒子の移動について通信の隠ぺいを行った．その結果，通信が隠ぺいされた部分では実行時間がほとんどの場合短縮し，約 1.40 倍程度向上した．実験環境の都合上，32 台以上の環境での詳細は分からないが，超並列計算機などのプロセッサ台数が多い環境ではこの影響が顕著に現れ大きな成果を得ることができると考えられる．

参考文献

[1] 三木光範：“並列処理入門”，超並列計算研究会 (1999)

[2] 超並列コンピューティング，http://www.fuji-ric.co.jp/ccse/project/ngcse/parallel/report/

[3] 上島豊，荒川拓也，佐々木明，横田恒：“Paragon 上でのスカラー超並列プログラム開発ガイド”，日本原子力研究所 (1998)

[4] “MPI:メッセージ通信インターフェース標準（日本語訳ドラフト）”，MPI フォーラム MPI 日本語訳プロジェクト (1996)

[5] 姫路工業大学情報処理センター，“UNIX 室の利用手引き”，http://www.cnte.himeji-tech.ac.jp/guide/guide1.htm