Vol.2013-ARC-207 No.5
Vol.2013-HPC-142 No.5
2013/12/16

並列言語XcalableMPによる核融合シミュレーションコードの開発

奴賀 秀男[1,a)]　朴 泰祐[1,2]　藤田 典久[2]　中尾 昌広[1]　佐藤 三久[1,2]　William Tang[3]

概要：GTC-P はトカマクプラズマ中の乱流現象を解析するために米国 Princeton 大学で開発された 3 次元 PIC（Particle In Cell）コードである．本研究では超並列計算環境に対応する核融合シミュレーションコードの高並列言語による開発を目的として，並列言語 XcalableMP を用いて GTC-P コードの並列化を行う．XcalableMP は分散メモリ環境におけるデータの管理に対し、グローバルビューとローカルビューという 2 種類の方法を提供しているが、本研究で扱うような PIC コードにおいてはこの両方の特性を用いることで柔軟かつ効率的な並列コードの記述が可能と考えられる。本稿では GTC-P の MPI による一対一通信を XcalableMP のローカルビュー機能を用いて置き換え，その性能を評価した．また，グローバルビュー機能を用いたさらなるコード開発を予定している．

1. はじめに

ITER を始めとする磁場閉じ込め型核融合装置では，閉じ込めプラズマ中に様々な時間・空間スケールの物理現象が発生する．時間スケールで 10^{-11}sec ～ 10^{0}sec，空間スケールで 10^{-5}m ～ 10^{0}m という幅広い領域で起こる様々な現象は互いに影響を及ぼしあうため，核融合プラズマの制御手法の開発や運転シナリオ作成には様々な物理現象を自己無撞着に解析する必要がある．また閉じ込めプラズマだけではなく，エッジプラズマと第一壁との相互作用も考慮する統合モデリングが求められる．このため，核融合プラズマのシミュレーションには膨大な計算機資源が必要であり，ITER[1] のシミュレーションではエクサスケール規模の計算機が求められる．

多国間研究協力事業 [2] の研究プロジェクトの一つである Nu-FuSE (Nuclear Fusion Simulations at Exascale) project[3] では ITER などの大規模実験を進めるうえで必要となる核融合シミュレーションコードのエクサスケールに向けた大規模化・高性能化を物理と計算機科学の協調と多国間協力関係の中で行う．その一環の研究として，次世代の超並列計算環境に対応する核融合シミュレーションコードの開発がある．

[1] 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba
[2] 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[3] Plasma Physics Laboratory, Princeton University
[a)] nuga@ccs.tsukuba.ac.jp

本研究では磁場閉じ込め核融合プラズマの乱流現象を解析するコード GTC-P[4] に並列プログラミング言語 XcalableMP[5]（以下 XMP）を用いた並列化を施す．XMP は高いスケーラビリティと生産性を求めて開発されたディレクティブベースの言語である．これを用いて，実際の物理計算に用いられる実アプリを容易に並列化できることを実証し，さらにこれを高い性能で実行可能であることを検証していく．

本稿の 2 節以降の構成は以下の通りである．2 節では本研究で用いた並列言語である XMP の概要について説明する．3 節では本研究で扱う核融合プラズマ中の微視的乱流現象解析コードである GTC-P について説明する．4 節では実際に行った実装について述べ，5 節ではその評価を行う．そして 6 節では本稿のまとめと今後の展望について述べる．

2. XcalableMP

現在，高性能計算機システムでの主要な並列言語として MPI[6] と OpenMP[7] が挙げられる．しかし，MPI はプログラマにとって複雑であり，ユーザーに対する教育やプログラミングにかかる時間などといったプログラムの生産性という面で問題がある．一方，OpenMP は逐次コードに簡単な指示文を挿入することで並列化を実現できるため高い生産性を期待できるが，使用できる環境が共有メモリシステムに限られ，結果としてその並列度に制限がある．これらに対し，XMP は分散メモリ型の並列プログラム開発を容易にするという目的で作られたディレクティブベースの

言語拡張である．

XMP は 2 つのプログラミングモデルをサポートしている [8]．一つはグローバルビューと呼ばれるもので，逐次コードのデータ分散やワークシェアリングのプロセスへのマッピングを XMP が提供する指示文を用いることで可能にするものである．これにより，分散メモリ型の並列プログラミングを OpenMP の様な指示文挿入によって実現することができる．もう一つはローカルビューと呼ばれる，PGAS モデル [9] を用いた Co-array Fortran[10] のような個別のプロセッサを意識した通信を可能とするプログラミングモデルである．PGAS モデル言語は片側通信と親和性が高く，MPI を用いた場合と比較して生産性だけではなく，性能の向上も期待できる場合がある．

2.1 PIC シミュレーションに対する XMP プログラミング方針

核融合プラズマ分野で適用される代表的な粒子シミュレーション手法として PIC（Particle In Cell）法とモンテカルロ法が挙げられる．このどちらもプラズマが荷電粒子の集合体であるという粒子的描像に基づくものであるが，PIC 法は主に外部場よりもプラズマ自身が作り出す電磁場の影響が強い現象に用いられ，一方モンテカルロ法はプラズマ粒子や不純物イオンなどの輸送現象解析に用いられる．本研究では PIC シミュレーションコードについて扱う．

PIC シミュレーションは場の計算を行う計算格子と格子によらない粒子軌道計算の組み合わせによって構成される．一般的に PIC シミュレーションは以下の 3 つのステップからなる [11]．

1: 電荷分配 各粒子が持つ電荷を近傍の格子点に分配し，それらを重ねあわせる．

2: 場の計算 格子点上の電荷密度から格子点上の静電ポテンシャル，ならびに静電場を計算する．

3: 粒子加速 粒子近傍の格子点の電場から粒子位置での電場，粒子に働く力を計算し，シミュレーションを 1 ステップ進める．

PIC シミュレーションではこのように離散量である格子座標 $\vec{r_g}$ と連続量である粒子座標 $\vec{r_p}$ を結びつけている．

この一連の過程を並列化する場合，以下の様な手順が必要になると考えられる．

0: 計算格子を複数のプロセス[*1] で分割する．

1'-1: 各プロセスが担当する格子内の粒子が持つ電荷を近傍の格子点に分配する．

1'-2: 近傍の格子点が別のプロセスにある場合，電荷を分配するための通信を行う．（図 1(A)）

2'-1: 場に対する Poisson 方程式を解く．

[*1] MPI のプロセスに相当するものを XMP ではノードと呼ぶが本稿では MPI，XMP の両方を扱う関係上，簡略化のためどちらもプロセスと呼称する．

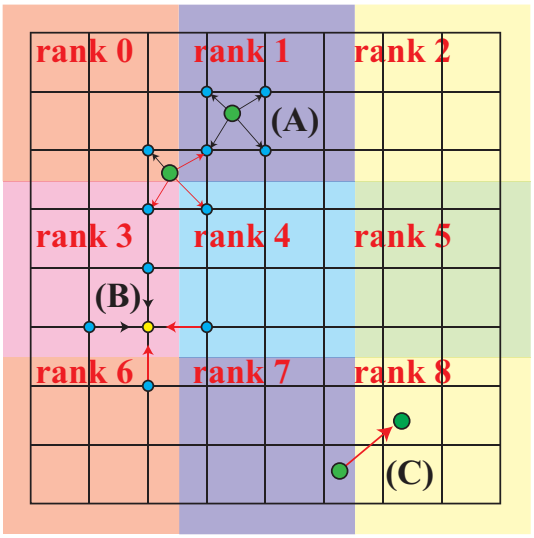

図 1 2 次元空間を 2 次元領域分割した図．赤矢印が通信が必要となる場合．(A) 粒子が近傍の格子点に電荷を分配する．(B) 隣接格子点のデータから静電ポテンシャルを計算する．(C) 粒子を 1 ステップ分移動させる．

2'-2: 陽解法でのステンシル計算では近接格子間で通信が発生する．（図 1(B)）

3'-1: 粒子近傍の格子点上の電場から粒子に働く力を計算する．

3'-2: 近傍の格子点が別のプロセスにある場合，通信を行う．

3'-3: 粒子を 1 ステップ分移動させる．

3'-4: 移動先が別のプロセスである場合，通信を行う．（図 1(C)）

並列化を行った場合に 1'-2, 2'-2, 3'-2, 3'-4 で通信を行う必要が発生するが，このうち 1'-2, 2'-2, 3'-2 の 3 つは格子点座標に関する通信であり，3'-4 は粒子座標についての通信である．

これを XMP を用いて実装する場合，計算格子のように要素数の変化しないものを分割するにはグローバルビューによる分割が適しており，一方，粒子運動のようなプロセスが担当する要素数が動的に変化するものは，ローカルビューによる通信で記述することが適している．従って，0, 1'-2, 2'-2, 3'-2 はグローバルビューの領域分割，ならびに袖領域通信で実装される．XMP の template 指示文，shadow, reflect，reduction 指示文などがそれに対応する．一方，3'-4 はローカルビューの Co-array 通信で実装される．

図 2 は PIC シミュレーションに対し，XMP によって空間 2 次元分割を行った場合のイメージである．図 2 では Co-array 通信を 1 次元的に扱い X 方向，Y 方向を別々に処理するようにしてあるが，これは後述する GTC-P がそのようになっているからである．2 次元方向へ纏めて通信しても問題はない．また送信バッファ，受信バッファの配列サイズは 1 プロセスが担当する粒子数としている．

```
1  #pragma xmp nodes p2(NX,NY)
2          // 総プロセス数はNX*NY
3  double f[NX][NY]; // electric field data
4  double myp[2,2*NP]; // 自プロセスが持つ粒子の座標
5          // NP は 1 プロセスが担当する粒子数（初期値）
6          // 粒子移動があるのでやや大きめにとる
7  double sendr[NP], sendl[NP]; // sendbuf
8  double recvr[NP], recvl[NP]; // recvbuf
9          // 1step ごとの通信量が NP より十分小さいとする
10 #pragma xmp distribute t(block,block) onto p2
11 #pragma xmp align f[i][j] with t(i,j)
12 #pragma xmp shadow f[1:1][1:1]
13 #pragma xmp coarray recvl:[*]
14 #pragma xmp coarray recvr:[*]
15 ...
16 for(istep=0; istep<TIME; istep++){ // 時間発展
17 .../* 1'−1, 1'−2の計算 */
18 #pragma xmp reduction(sum:rho) //rho は電荷密度
19 .../* 2'−1, 2'−2の計算 */
20 #pragma xmp reflect(f)
21 .../* 3'−1, 3'−2の計算 */
22 #pragma xmp reduction(sum:qE)
23 .../* 3'−3の計算 */
24 //以下 3'−4
25 .../* myp のうち，X 方向の隣接プロセスへ出て行く粒子を
26      判定しsendl, sendr に詰め込む.
27      送信要素数icountr, icountl を求める
28      送信先プロセスleft_pe, right_pe を求める */
29 recvl[0:icountr]:[left_pe] = sendr[0:icountr];
30 recvr[0:icountl]:[right_pe] = sendl[0:icountl];
31 #pragma xmp sync_all
32 .../* Y 方向も同様に行う */
33 //ここまで 3'−4
34 }
```

図 2 PIC シミュレーションを XMP で実装した場合のイメージ.

各部分の記述に関しては 4 節に詳述するが，上記のように，PIC シミュレーションの並列化は XMP のグローバルビュー，ローカルビューの両方を用いたプログラミングの典型例となりうる.

3. 核融合プラズマシミュレーション

トカマクを始めとする磁場閉じ込め核融合装置において，プラズマの閉じ込め性能に重大な影響を及ぼす異常輸送や不安定性といった現象は，非常に時間・空間スケールの小さい現象である微視的乱流現象によって駆動される．これらの現象は一般的にジャイロ運動論 [12] によって記述されるが，この計算には高い次元数（5 次元位相空間）と高い空間解像度が必要となる．このため，核融合プラズマ中の乱流現象のシミュレーションには，膨大な計算機資源が必要とされる．また，実験装置の大型化に応じて必要な計算機資源量も増大する.[*2] 従って，核融合プラズマ中の乱流現象シミュレーションには高い並列性，高いスケーラビリティが要求される．そして，これらの目的と要求を満たす数多くのコードがこれまで開発されてきた（[4], [13], [16] など）.

[*2] 5 次元ジャイロ運動論コードである GT5D[13] に対して，JT-60U[14] のサイズで 750TFlops×1day，ITER[1] のサイズで 7PFlops×1day の計算量が必要であると試算されている [15].

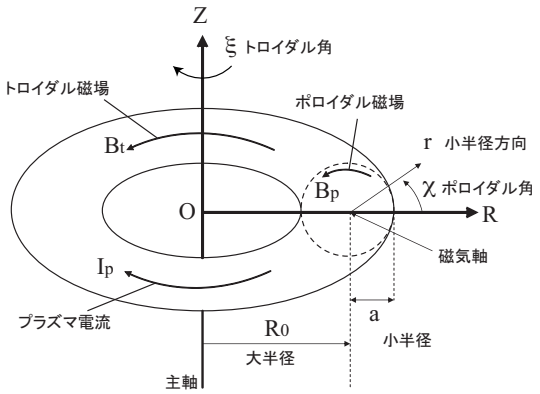

図 3 トロイダル座標系の概念図 [17]．主軸 Z を回る方向をトロイダル方向，磁気軸を回る方向をポロイダル方向，磁気軸からの距離の方向を径方向と呼ぶ．また，トーラスを縦に切った断面（図の破線円）をポロイダル断面と呼ぶ.

3.1 GTC ならびに GTC-P コード

GTC (Gyrokinetic Toroidal Code) コード [16] は，磁場閉じ込め装置における核融合プラズマ中の微視的乱流現象の解析を目的として米国 Princeton 大学で開発された，三次元ジャイロ運動論的 PIC コードである．言語は Fortran である．GTC では多数の荷電粒子を一つの粒子とみなした超粒子の粒子軌道計算と，格子点上のデータとして与えられる静電ポテンシャルとの相互作用を，自己無撞着に計算することで，荷電粒子の速度分布関数を求めている．計算手順は (1) 超粒子の電荷密度を格子点上に分配する (2) Poisson 方程式から格子点上での静電ポテンシャルを求める (3) 超粒子が静電場から受ける力を計算する (4) 超粒子を移動させる，の繰り返しとなる.

本研究で扱う GTC-P[4] コードは GTC コードの発展版であり，C 言語版と Fortran 版が存在する．本研究では C 言語版を用いる．GTC では MPI による領域分割が 1 次元だけであったが，GTC-P では 2 次元で分割され，さらに各領域内の粒子数も分割される.

3.2 格子点と領域分割

図 3 に一般的なトロイダル座標系を示す．GTC-P では MPI によって計算領域がトロイダル方向に n_t 分割，径方向に対して n_r 分割され，さらに 2 次元分割された各領域内の粒子数を n_{rp} に分割している．従って，総プロセス数 n_p は $n_p = n_t \times n_r \times n_{rp}$ として与えられる．このうち n_t は，トロイダル方向のグリッド数 N_t と等しくなければならない，という制限がある.

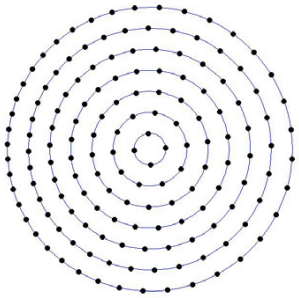

図 4 GTC-P におけるポロイダル断面上での格子点 [16]．ポロイダル方向の格子点間隔は最も外側の円弧の長さを基準に決定されるが，厳密に等しくすることはできないので，全ての円弧は最も外側の円弧以下の長さになるように与えられる．

3.2.1 格子点

GTC-P ではトロイダル方向，径方向の計算格子は等トロイダル角，等径方向距離で与えられるが，ポロイダル方向の格子点は等ポロイダル角ではなく，図 4 に示すように円弧が一定になるように設定されている．これにより，等ポロイダル角で設定する場合に対して，ポロイダル断面上で格子点はできる限り均等に配置されることになる．これは GTC-P の扱う微視的乱流現象がポロイダル断面上ではイオンの旋回半径程度の空間スケールを持つため，ポロイダル断面上では高い解像度を持つ計算格子が必要とされるためである．GTC-P が提供する最小サイズのパラメータではトロイダル方向に 64 点，径方向に 90 点，ポロイダル方向には径方向最大の位置で 640 点が与えられている．ポロイダル断面上の総格子点数はおよそ 30,000 である．

3.2.2 計算領域分割

図 4 に示すように，各径方向位置に対して格子点の数が大きく異なるため，径方向の領域分割を等間隔で行った場合，各プロセス間で計算量に大きな偏りが発生してしまう．このため，GTC-P では径方向の領域分割を等間隔ではなく，径方向に分割された各計算領域におけるポロイダル断面上での格子点数が極力等しくなるように径方向を分割している．このため，径方向の内側は幅が広く，外側になるほど狭くなるように分割される．

トロイダル方向の領域分割は $n_t = N_t$ の制約から，各トロイダル領域はポロイダル面を 1 つだけ持つように分割される．粒子数の分割は各計算領域内の粒子を等分する．

4. XMP による GTC-P の実装

4.1 方針

2.1 節で示したように，領域分割と全体全通信はグローバルビューで，粒子の領域間移動はローカルビューで表現する．ただし 3.2.2 節にある通り，GTC-P の径方向分割は等間隔ではなく，格子点数の計算を経て不均一に分けられるものである．これを正確にプロセスへの分散として記述するためには、グローバルビューによる正方格子での単純な領域分割とは異なり，各プロセスが受け持つ分割領域を、初期計算によって非均質に決定する処理が必要となる。これは一般的に、極座標系や線密度あるいは面密度一定の領域分割を行おうとする場合に必要となる機能である。しかし，現在のグローバルビューの template 指示文は直交座標系相当（多次元配列表現）にのみ対応しており，このような複雑な表現はできない．この機能は gblock という機能によって XMP 上で実現されるが，現時点では仕様上で定められているものの，リファレンス実装としては未実装である．このため領域分割は，各プロセスが担当する配列のサイズを計算し動的にメモリを確保する，という MPI で行っていた方法をそのまま用いる．同様の理由で，隣接格子間の通信もグローバルビューの袖領域通信ではなくローカルビューの Co-array 通信で行う．ただし，全体全通信はグローバルビューで行う．

なお現段階で XMP は OpenMP に非対応であるため，スレッド並列は行わない．

```
1 double *sendr, *sendl;
2 int isr, irl, r_pe, l_pe;
3 MPI_Sendrecv(sendr, isr, MPI_MYREAL, r_pe, istag,
4                recvl, irl, MPI_MYREAL, l_pe, irtag,
5                toroidal_comm, \&istatus1);
```

図 5 一対一通信の MPI 版．通信は MPI_COMM_SPLIT で作成されたサブコミュニケータに対して行われている．

```
1 #define msend 20000
2 double Xsendr[msend], Xrecvl[msend];
3 #pragma xmp coarray Xrecvl:[*]
4 ...
5 int isr, r_pe2;
6 Xrecvl[0:isr]:[r_pe2]=Xsendr[0:isr];
```

図 6 図 5 を XMP の Co-array 通信で置き換えたもの．6 行目左辺にある Xrecvl の 1 つ目のカギ括弧は [先頭の要素番号:送信要素数] である．この場合，受ける側の情報である irl, l_pe は不必要である．また，送信先プロセスである r_pe2 は図 5 の r_pe に相当するプロセスだが，サブコミュニケータ内のランクではなく，全プロセス中の番号が必要である．

4.2 ローカルビュー

GTC-P において粒子の計算領域間の移動は MPI_Sendrecv もしくは MPI_Isend/Irecv によって記述されている．通信は常に 1 次元的に行われ，通信相手は左右の隣り合う相手だけであり，一部の例外を除いて両隣のプロセスに対し右方向送信，左方向送信と順次行う．図 5, 6 は右方向送信の部分をそれぞれ MPI，XMP で行った例である．XMP の Co-array 通信は代入形式で記述される．また，Co-array 通信を行う変数はポインタではなく配列である必要がある．

図 5, 6 の例は常に一つのプロセスが別の一つだけのプ

```
1 double efield;
2 ... // calculate efield on each rank
3 double tmp = 0.0;
4 MPI_Allreduce(\&efield, \&tmp1, 1, MPI_MYREAL,
      MPI_SUM, partd_comm)
5 efield = tmp1;
```

図 7 MPI を用いて集計計算を行っている例．サブコミュニケータ partd_comm に対して MPI_SUM を行っている．

```
1 #pragma xmp nodes p(32)
2 #pragma xmp nodes p2(8,4)
3 ...
4 double efield;
5 ... // calculate efield on each rank
6 #pragma xmp reduction(+:efield) on p2(:,*)
```

図 8 XMP で集計計算を行った例．一時変数が不要で，各ランクで efield が求まった後に指示文で集計計算を実行している．指示文の p2(:,*) は MPI 版のコミュニケータ partd_comm に相当する部分である．この場合，1 ∼ 8, 9 ∼ 16, 17 ∼ 24, 25 ∼ 32 のように 8 プロセスから構成されるグループの中で通信が行われる．

ロセスからデータを受ける場合の例であるが，一つのプロセスが複数のプロセスからデータを受信する場合 MPI と異なる仕組みが必要となる．これは，MPI は recvbuf の先頭アドレスを知っているのは受信側のプロセスであるが，XMP は送信側が知っていなければならない，という相違によるものである．この場合，送信側のプロセスで recvbuf の先頭アドレスを計算する，もしくは，受信側から recvbuf の先頭アドレスを送信側に送信するなどの方法で対応する必要がある．今回の場合は後者の方法で対応した．

4.3 グローバルビュー

MPI_Bcast や MPI_Allreduce などの全体全通信をグローバルビュー機能を用いて記述した．XMP において，この通信は指示文の形で記述される．図 7, 8 はそれぞれ MPI, XMP で同様に集計計算を行った例である．サブコミュニケータ単位で通信が行われる場合は図 8 の 2，5 行目のように通信に参加するプロセスを指定する必要がある．

5. 評価

5.1 環境

以下の計算は全て筑波大学計算科学研究センターの HA-PACS[18] 上で行った．HA-PACS の主な環境を表 1 に示す．コンパイラには Intel C Compiler を，MPI には IntelMPI をそれぞれ用いた．

なお、HA-PACS は高密度・大規模 GPU クラスタであり、我々の最終目的は本コードをさらに GPU 対応させることであることから同システムを用いている。ただし、現時点では GPU は用いず、CPU のみによる実装と評価を行っている。

表 1 計算機環境：HA-PACS

CPU	Intel Xeon E5-2670 × 2 (2.6GHz)
	CPU (8 cores/CPU) × 2 = 16 cores
CPU Memory	128GB, DDR3 1600MHz
OS	CentOS 6.1
Intel Compiler	13.0
Compiler options	-O3 -mavx -Wall -std=gnu99 -limf -lirc
XcalableMP Compiler	R1336
MPI	intelmpi/4.1
Interconnect	Infiniband QDR 4 Lines, 2 Rails

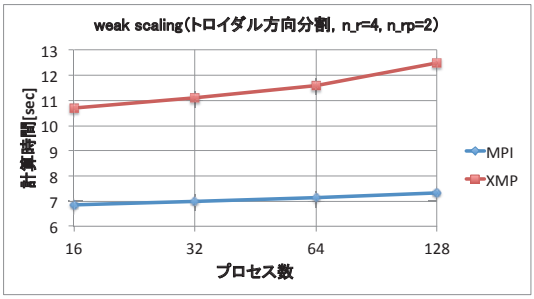

図 9 弱スケーリングの図．横軸は総プロセス数で，$n_r = 4, n_{rp} = 2$ は固定し，トロイダル分割数を $2 \leq n_t \leq 16$ で変化させたもの．

5.2 性能

計算量に関係するパラメータにはトロイダル格子数 N_t, 径方向格子数 N_r, 格子点あたりの粒子数 N_{rp}, 最外殻でのポロイダル格子数 N_{pol}，計算ステップ数 N_{step} がある．以下の計算では $N_r = 90$, $N_{rp} = 100$，$N_{pol} = 640$, $N_{step} = 20$ とする．N_r, N_{rp}, N_{pol} は GTC-P が提供する最小スケール用のパラメータであり，時間ステップ数はそれよりもさらに小さい値としている．[*3]．弱スケーリングについては $n_r = 4, n_{rp} = 2$ を固定し，$2 \leq n_t = N_t \leq 16$ として計測を行った．最小スケール用のトロイダル格子数は $N_t = 64$ なので実際の計算に用いられるものより小さい範囲での計測である．また，強スケーリングについては $n_t = 2, n_{rp} = 2$ を固定し，径方向分割数を $4 \leq n_r \leq 32$ と変化させたものと，$n_t = 2, n_r = 2$ を固定し，$4 \leq n_{rp} \leq 32$ としたものを計測した．

HA-PACS は 1 ノードあたり 16 コアなので，1 ノードあたりに 16 プロセスを分配した．

5.2.1 弱スケーリング

図 9 は径方向分割数，粒子数分割数を固定し，トロイダル格子数と分割数を変化させたものである．XMP 版は MPI 版に比べて速度が遅く，あまりスケールも良くないという結果となった．XMP は全体全通信に関しては MPI を呼び出すだけなので MPI より速くなることは原理的にな

[*3] 物理計算には 10000step 以上が必要とされる．性能評価用には 100step を用いるとあるが，今回はより小さい値を用いた．

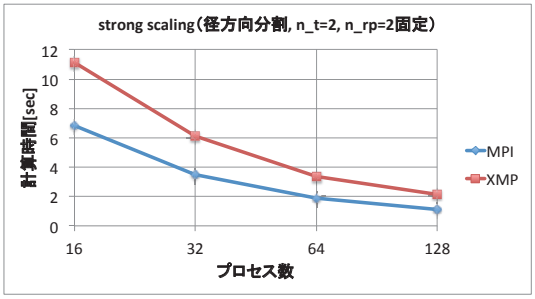

図 10 強スケーリングの図．横軸は総プロセス数で，$n_t = 2, n_{rp} = 2$ を固定し，径方向分割数を $4 \leq n_r \leq 32$ で変化させたもの．

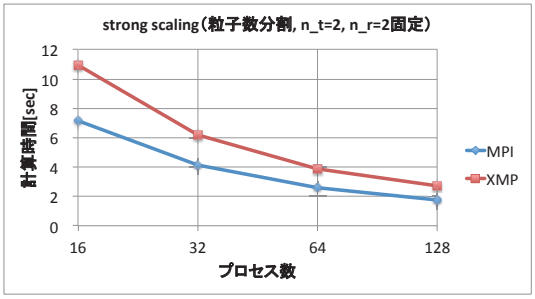

図 11 $n_t = 2, n_p = 2$ を固定し，粒子数分割数を $4 \leq n_{rp} \leq 32$ で変化させたもの．

表 2 関数別で見た MPI 版と XMP 版の性能比．このときのプロセス数は $n = 128$，分割数はそれぞれ $n_t = 16, n_r = 4, n_p = 2$ である．通信は，なし，Co-array のみ，reduction のみ，両方の 4 通りで分けている．

関数名	経過時間（MPI）	性能比	通信内容
total	7.34 [sec]	0.59	–
charge	3.13	0.56	Co + reduction
push	2.72	0.64	reduction
shift_t	0.34	1.24	Co + reduction
shift_r	0.14	0.76	Co + reduction
sorting	0.59	0.80	なし
poisson	0.22	0.48	Co
field	0.070	0.29	Co
smooth	0.078	0.29	Co + reduction
setup	0.33	0.49	reduction
poisson_init	0.036	0.51	なし

い．また，一対一通信に関しては最適化の余地が残っており，これに関しては改善の見込みがある．

5.2.2 強スケーリング

図 10 はトロイダル分割数，粒子数分割数を固定して，径方向分割数を変化させたもの，図 11 はトロイダル分割数，径方向分割数を固定して粒子数分割数を変化させたものである．弱スケーリング同様，XMP 版は MPI 版に対し性能で劣るものの，プロセス数の増加に伴う性能の相対的な低下はどちらの場合もほぼ見られなかった．

5.2.3 関数別

表 2 に関数別の XMP/MPI 性能比を示す．関数の重みとしては，全体の計算時間に対し，charge と push がそれぞれ全体の 5 割，および 4 割程度を占めている．

通信が発生しない関数の性能比がコンパイラの性能比を示すものと考えられるが，これに相当する関数 sorting と poisson_init の性能比はそれぞれ 0.80, 0.51 と開きがある結果になった．poissin_init の処理時間は MPI 版で 0.5%程度であり，sorting の 8%と比べて十分小さいので，コンパイラの性能差として 0.80 程度であるとして考える．

ほとんどの通信がある関数で性能比は 0.80 以下となったが，shifti_toroidal だけは MPI 版よりも高速化するという結果となった．これについては原因を調査中であるが，この関数では他の関数に比べて大きい Co-array 通信を含むため，MPI_Sendrecv に比べて Co-array 通信の速度が出た可能性が考えられる．

6. まとめ

本研究では核融合プラズマ中の乱流現象解析コードである GTC-P の XMP による並列化を行った．5 節に示したように計算性能自体は MPI に劣るが，計算はスケールするということが示された．また，XMP 版は通信の最適化の余地が残っており，改善の可能性がある．また、今回の評価は利用可能ノード数の制約等から、小さいモデルに対するものであり、特に強スケーリングの結果については、参考にはなるが、今後の大規模問題・大規模ノードでの検証が必要である。

今後は XMP の機能として実装される gblock 機能を用いて，空間の領域分割を XMP のグローバルビュー機能によって置き換え，PIC シミュレーション全体をグローバルビューとローカルビューの組み合わせで記述できる，ということを検証していく予定である．

謝辞 本研究の一部は日本学術振興会・多国間国際研究協力事業 (G8 Research Councils Initiative) プログラムの研究課題「エクサスケール規模の核融合シミュレーション」ならびに，筑波大学計算科学研究センター平成 25 年度学際共同利用プログラム課題「核融合シミュレーションの GPU 化と性能評価」によるものである．

参考文献

[1] Shimomura, Y., Aymar, R., Chuyanov, V., Huguet, M., Parker, R. and ITER Joint Central Team: ITER overview, *Nuclear Fusion*, Vol. 39, p. 1295 (1999).
[2] 日本学術振興会：多国間国際研究協力事業，`http://www.jsps.go.jp/j-bottom/01_b_gaiyo.html`.
[3] The Nu-FuSE Project: `http://www.nu-fuse.com/`.
[4] Ethier, S., Adams, M., Carter, J. and Oliker, L.: Petascale Parallelization of the Gyrokinetic Toroidal Code, *In proceedings of VECPAR'10 9th International Meeting on High Performance Computing for Computational Science*, Berkeley, CA (2010).
[5] XcalableMP project: `http://www.xcalablemp.org`.
[6] Snir, M., Otto, S., Huss-Lederman, S., Walker, D. and Dongarra, J.: MPI-The Complete Reference, Volume 1,

The MPI Core, 2nd ed. Cambridge, MA, USA: MIT Press (1998).
[7] OpenMP.org: http://openmp.org/wp/.
[8] Nakao, M., Murai, H., Shimosaka, T. and Sato, M.: Productivity and Performance of the HPC Challenge Benchmarks with the XcalableMP PGAS Language, 7th International Conference on PGAS Programming Models, Edinburgh, Scotland, UK (2013).
[9] PGAS: Partitioned Global Adress Space Languages, http://www.pgas.org/.
[10] Numrich, R. W. and Reid, J.: Co-array fortran for parallel programming, *SIGPLAN Fortran Forum*, Vol. 17, No. 2, pp. 1–31 (1998).
[11] 内藤裕志，佐竹真介：粒子シミュレーションのコーディング技法，*J. Plasma Fusion Res.*, Vol. 89, No. 4, pp. 245–260 (2013).
[12] Brizard, A. J. and Hahm, T. S.: Foundations of nonlinear gyrokinetic theory, *Reviews of modern physics*, Vol. 79, No. 2, p. 421 (2007).
[13] Idomura, Y., Ida, M., Kano, T., Aiba, N. and Tokuda, S.: Conservative global gyrokinetic toroidal full-f five-dimensional Vlasov simulation, *Computer Physics Communications*, Vol. 179, No. 6, pp. 391–403 (2008).
[14] Hosogane, N., Ninomiya, H., Matsukawa, M., Ando, T., Neyatani, Y., Horiike, H., Sakurai, S., Masaki, K., Yamamoto, M., Kodama, K. et al.: Development and operational experiences of the JT-60U Tokamak and power supplies, *Fusion science and technology*, Vol. 42, No. 2-3, pp. 368–385 (2002).
[15] 加藤千幸：戦略分野 4:次世代ものづくり (<特集>次世代スーパーコンピュータ「京」:動き出した大型プロジェクトの全体像，日本物理學會誌， Vol. 66, No. 7, pp. 542–547 (2011).
[16] Ethier, S., Tang, W. M. and Lin, Z.: Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms, *Journal of Physics: Conference Series*, Vol. 16, No. 1, p. 1 (2005).
[17] Nuga, H. and Fukuyama, A.: Kinetic modeling of the heating processes in tokamak plasmas, PhD Thesis, Kyoto Univ. (2011).
[18] Center for Computational Sciences, University of Tsukuba: HA-PACS プロジェクト，http://www.ccs.tsukuba.ac.jp/CCS/research/project/ha-pacs.