Vol.2014-HPC-145 No.37
2014/7/30

並列言語XcalableMPによる核融合シミュレーションコードの実装と評価

津金 佳祐[1,a)]　奴賀 秀男[3]　朴 泰祐[1,2]　村井 均[4]　佐藤 三久[1,2]　William Tang[5]

概要：本稿では，磁場閉じ込め核融合プラズマの乱流現象の解析を行うコードである，米国プリンストン大学が開発した GTC-P を，並列言語 XcalableMP が提供するグローバルビューとローカルビューという 2 種類のプログラミングモデルを用いて，実装し評価する．ローカルビューモデルである coarray 記法を用いた XMP（coarray）実装では，MPI_Allreduce のような全体全通信以外を XMP の同機能で記述し，MPI による実装と同等の性能を得る事ができた．また，グローバルビューモデルによる領域分割を用い，指示文による袖領域通信と coarray 記法を用いた XMP（reflect + coarray）実装では，MPI と比較して 5 から 25%の性能劣化となった．生産性については，隣接格子間の通信を指示文一行にすることができ，その他の通信に対しては配列代入文形式で記述可能な coarray 記法を用いた事により，MPI と比較してより簡易に表現する事が出来た．以上より，本コードに代表される PIC 法（Particle-In-Cell）コードを，XcalableMP の様々なデータビューモデルを用いる事により，一定の性能を保ちつつ，簡便かつスケーラブルに記述できる事が示された．

1. はじめに

分散メモリ環境上では並列プログラミングモデルとして，Massage Passing Interface（MPI）が広く普及している．しかし，MPI はプロセス毎のデータの分散配置を考慮するといった，並列化を行う上での様々な手順を明示的に示す必要があるため，プログラミングコストが大きいことやソースコードが煩雑になるといった生産性の低下が問題となっている．そこで，分散メモリ環境上での並列プログラミングをより容易にするために開発されたのが，XcalableMP（以降「XMP」と略す）である [1][2]．XMP は，C や Fortran 言語といった既存言語の拡張であり，OpenMP に似た指示文を用いてループの並列化やプロセス間通信を行うことが可能である．そのため，既存の逐次プログラムを大きく変更することなく容易に並列化を行うことができる．

一方，磁場閉じ込め型核融合装置を対象とした核融合プラズマ中の乱流現象のシミュレーションには，高い次元数と空間解像度が必要とされる．そのため，国際熱核融合実験炉 ITER[3] のような大規模の核融合装置を対象としたシミュレーションを行う場合には，演算量は著しく増大する．従って，高い並列性による性能向上と，プログラムの開発コストを下げるための高い生産性が必要とされている．

そこで本稿では，核融合シミュレーションコードである GTC の並列アルゴリズムの改良版である GTC-P[4] に対して XMP を用いて実装を行い，性能と生産性を評価する．関連研究として，GTC-P に対する最適化 [5] や GPU 化 [6] は行われているが，生産性に関する評価は先行研究 [7] のみである．先行研究では，GTC-P 中の MPI による一対一通信を XMP の coarray 記法によって置き換えた実装の評価は行われているが，reflect 指示文を含む記述についてはまだ検証されていない．

本稿の構成は下記の通りである．2 章は XMP の概要とプログラミングモデルの説明を行い，XMP によるプログラミング例を示す．3 章では対象である核融合シミュレーションコードについて説明を述べ，4 章では XMP による GTC-P の実装の詳細を述べる．5 章で評価を行い，6 章でまとめと今後の課題を述べる．

[1] 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
[2] 筑波大学計算科学研究センター
Center for Computational Sciences, University of Tsukuba
[3] 日本原子力研究開発機構
Japan Atomic Energy Agency
[4] 独立行政法人理化学研究所 計算科学研究機構
Advanced Institute for Computational Science, RIKEN
[5] Plasma Physics Laboratory, Princeton University
[a)] tsugane@hpcs.cs.tsukuba.ac.jp

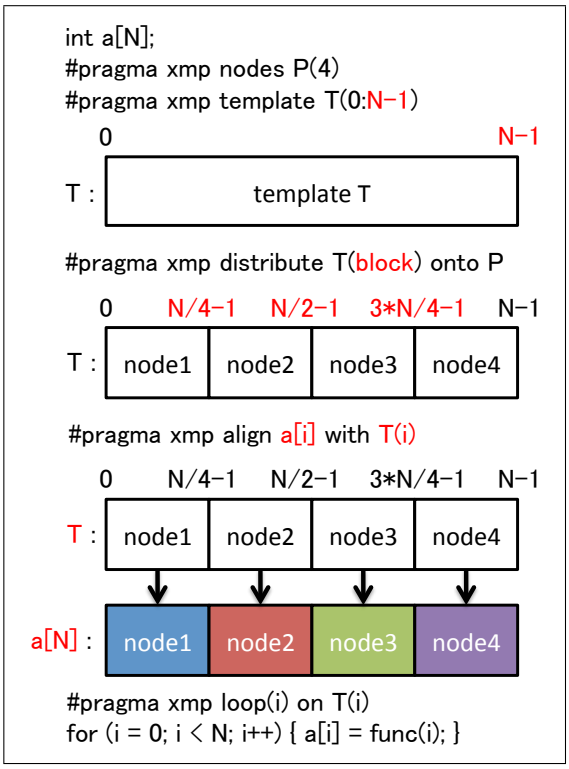


図 1 グローバルビューモデルプログラミング例.

2. XcalableMP

XMP は，次世代並列プログラミング言語検討委員会及び PC クラスタコンソーシアム並列プログラミング言語 XcalableMP 規格部会により，仕様検討及び策定されている分散メモリ型 SPMD（Single Program Multiple Data）を実行モデルとする並列言語である．プログラミングモデルとしては，指示文により各ノードにおいてのデータの分散，ループの並列処理，通信・同期を行う方式をとっており，ユーザが指示文を挿入しない限りそれらの動作が起こることはない．よって，ユーザが意図しない通信が起こることはなく，プログラムの挙動や結果を予測しやすいという利点がある．

XMP はプログラミングモデルとして，グローバルビューモデルとローカルビューモデルの 2 種類のプログラミングモデルが実装されており，これらを用いることで柔軟に並列プログラミングを行うことが可能である．

2.1 グローバルビューモデル

グローバルビューモデルは，問題で扱うグローバルな配列全体を各ノードに分散する指示文を記述することで，並列実行を行うプログラミングモデルである．**図 1** はプログラミング例であり，ノード毎にデータの分散を行うには，テンプレートと呼ばれる仮想的なアドレス空間を用いる．まず，node，template 指示文により実行ノード数（プロセス数）とテンプレート長を指定する．次に，テンプレートに対して distribute 指示文により分割方法（ブロック分割，サイクリック分割及び任意のブロックサイズによる分割）の指定を行い，align 指示文で分散したい配列との対応付けをすることによって，各ノードへとデータの分散を行う．これらの動作は全て指示文によるものであり，ユーザは各ノードへと分散されたデータの配置を意識することなく loop 指示文により，並列実行を行うことが可能となる．また，グローバルビューモデルによる XMP プログラムは，XMP 指示文を無視することで逐次版の C，Fortran 言語によるプログラムとして解釈することが可能である．

2.2 ローカルビューモデル

ローカルビューモデルは，各ノードが持つローカルデータに対して通信を行うプログラミングモデルである．XMP では，Fortran 言語の拡張記法である Coarray Fortran[8] をベースとした coarray 記法 [9] が実装されている．記述方法は配列代入文形式であり，ノード番号を指定することによるデータの片方向通信を行うため，MPI のようなノード毎の振る舞いを個別に記述することが可能である．

3. 核融合シミュレーションコード

核融合プラズマ中の乱流現象のシミュレーションを行う代表的な手法として，PIC（Particle-In-Cell）法とモンテカルロ法が挙げられる．本稿では，PIC 法の実アプリケーションを対象とする事から PIC 法の説明のみを行う．

3.1 PIC シミュレーション

PIC 法によるシミューレションでは，場の計算を行う計算格子と，空間を自由に動き回る，格子によらない粒子軌道計算によって構成される．以下は PIC 法のシミュレーション手順である [10].

(1) 各粒子が持つ電荷を近傍格子点に加算を行う．
(2) ポアソン方程式により，近傍格子点上の電荷密度から格子点上の静電ポテンシャルを求め，それを元にし電場を求める．
(3) 各粒子の近傍格子点から個々の粒子の現在位置での電場を求め，1 ステップ粒子の位置を進める．

通信が発生するステップとして，近傍格子点が別プロセスに分割されている場合や，粒子の移動先が別プロセスが持つ領域の場合が挙げられる．

以上の事から，XMP を用いた実装を行うにあたり，計算格子と言った分割された領域のサイズが変更されない演算の場合には，グローバルビューモデルによる領域分割が適しており，近傍格子点情報は袖領域通信で表す事が可能である．また，粒子運動と言った各プロセスが受け持つデータサイズが動的に変更される場合にはローカルビューモデルによる coarray 通信が適している．

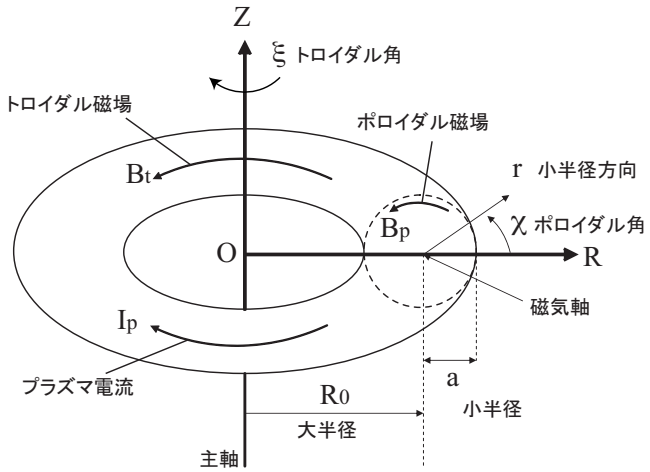


図 2 GTC-P における三次元トーラス空間の簡易図 [11].

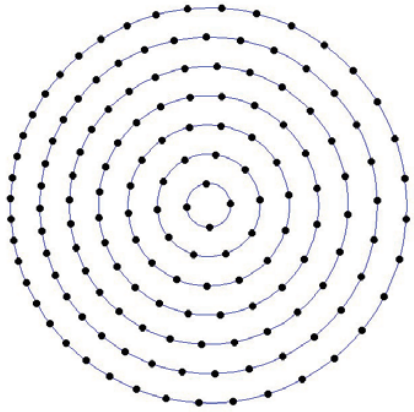


図 3 ポロイダル断面における格子点 [12].

3.2 GTC

GTC（Gyrokinetic Toroidal Code）は，磁場閉じ込め型核融合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 DoE SCiDAC，UC Irvine 等で開発された三次元ジャイロ運動論的 PIC コードである [13]. 本稿で対象とする実アプリケーションである GTC-P は，GTC の並列アルゴリズム改良版であり，GTC では MPI を用いた 1 次元の領域分割と分割された領域内での粒子数の分割に加え，OpenMP によるループレベルでのスレッド並列の三層レベルでの並列化が行われているのに対して，GTC-P では 1 次元の領域分割が 2 次元の領域分割へと改良され，四層レベルでの並列化となっている．また，GTC は Fortran 言語で開発されているのに対して，GTC-P は C，Fortran 言語で開発されている．本稿では C 言語版を用いる．

図 2 は，GTC-P の計算領域を表した三次元トーラス空間の簡略図であり，主軸 Z を回るトロイダル方向，磁気軸を回るポロイダル方向，磁気軸からの距離の方向を径方向と呼ぶ．また，トロイダル方向に分割した際のトーラスの

```
1 double *sendr, *recvl;
2
3 for(i=0;i<nloc_over;i++)
4   sendr[i]=phitmp[i*(mzeta+1)+mzeta];
5
6 MPI_Sendrecv(sendr,nloc_over,double,right_pe,
7     isendtag,recvl,nloc_over,double,left_pe,
8     irecvtag,toroidal_comm,&istatus);
```

図 4 GTC-P 内の MPI による一対一通信

```
1 double Xsendr[nloc_over],Xrecvl[nloc_over];
2 #pragma xmp coarray Xrecvl:[*]
3
4 for(i=0;i<nloc_over;i++)
5   Xsendr[i]=phitmp[i*(mzeta+1)+mzeta];
6
7 Xrecvl[0:nloc_over]:[right_pe]=Xsendr[0:nloc_over];
8 xmp_sync_all(NULL);
```

図 5 GTC-P 内の通信を coarray 記法で置き換えた例

断面をポロイダル断面と呼ぶ．GTC-P では，MPI を用いてトロイダル方向に N_t，径方向に N_r と二次元の領域分割を行い，さらにそれぞれ分割された領域が持つ格子上の粒子数を N_{rp} と分割する．従って，総 MPI プロセス数は $N = N_t * N_r * N_{rp}$ となる．

図 3 に示すように，GTC-P ではポロイダル断面上の格子点数は各径方向位置ごとに異なる．そのため，トロイダル方向に対して等トロイダル角での分割は可能だが，径方向に対して等間隔に分割を行った場合には格子点数の偏りにより，プロセス毎の演算量に大きな差が生じる．そこで，GTC-P では，径方向の分割においての格子点数の偏りを減らすために，径方向の内側の領域を広く，外側の領域を狭く分割する事で，格子点数を極力揃えている．

4. 実装

GTC-P の隣接格子点間と粒子の移動による通信を全て coarray 記法で実装した XMP（coarray）と，領域分割をグローバルビューモデルを用いて行い，隣接格子点間の通信を reflect 指示文 [14] による袖領域通信とし，粒子の移動による通信を coarray 記法で実装した XMP（reflect + coarray）の二種類の実装を行う．また，両実装共に MPI_Bcast，MPI_Allreduce による全体通信は，グローバルビューモデルが提供する bcast，reduction 指示文を用いる．

4.1 XMP（coarray）実装

GTC-P における粒子の計算領域間の移動時の通信は MPI_Sendrecv または，MPI_Isend/Irecv によって記述されている．通信先は常に同一であり，一部を除き左右の隣接格子間で通信を行う．また，通信サイズは動的に変更される事から coarray 記法による通信を用いる．**図 4** は MPI

```
1 #pragma xmp reflect (phitmp) width (0,/periodic/1:0)
```

図 6 GTC-P 内の通信を reflect 指示文に置き換えた例

```
1  #define n_t 2
2  /* トロイダル方向の分割数*/
3  #define n_r 4
4  /* 径方向の分割数*/
5  #define n_rp 2
6  /* 粒子数の分割数*/
7
8  #define nloc_over_all 107722
9
10 real phitmp_g [nloc_over_all][2*n_t];
11 int b [n_r*n_rp]
12   = {10967,10967,14086,14086,16164,16164,12644,12644};
13 /* {gblock 分散時の各プロセスのブロックサイズ } */
14
15 #pragma xmp nodes P2(n_r * n_rp, n_t)
16 /* ノードの分割数の指定*/
17 #pragma xmp template T(0:nloc_over_all−1, 0:2*n_t−1)
18 /* テンプレート長の指定*/
19 #pragma xmp distribute T(gblock(b), block) onto P2
20 /* テンプレートの分割方法の指定*/
21 #pragma xmp align phitmp_g [i][j] with T(i, j)
22 /* 配列とテンプレートの対応*/
23 #pragma xmp shadow phitmp_g [0][1:0]
24 /* 袖領域の確保*/
```

図 7 XMP による GTC-P の実装

による実装，**図 5** は coarray 記法による実装の例である．例では，通信データを 1 次元配列へと格納し，隣接する右辺のプロセスへの通信を表している．coarray 通信は，ノンブロッキング通信である事から XMP が提供する全ノードの通信完了を保証する xmp_sync_all を用いている．XMP の仕様上，coarray 記法の両辺共にポインタを指定する事が出来ず，通信相手のプロセスを指定する配列は静的に記述されている必要がある．

4.2 XMP（reflect + coarray）実装

GTC-P におけるトロイダル方向，径方向の領域分割は，3.2 節よりトロイダル方向は等間隔に分割されるが，径方向に対しては初期格子点演算時に不均一に分割される．そこで，グローバルビューモデルが提供する distribute 指示文の gblock 分割を用いる．gblock は各プロセスが受け持つ分割領域のサイズを非均等に配置することが可能な分割方法である．**図 7** は，計算領域に対して gblock 分割を用いた場合のグローバルモデルによる GTC-P の実装例である．19 行目の distribute 指示文で gblock 指定の際に，それぞれのプロセスが持つ分割領域のサイズが格納された配列（例では，11 行目の配列 b を指す）を指定することで不均一な分割を行う．現在，XMP は一次元分散の配列は動的確保可能だが，二次元分散の配列は動的確保をすることができない．そのため，XMP の初期化（ノード数，テンプレート長，分割方法，配列とテンプレートの対応付け）は図 7 と同様に全て静的に記述している．

隣接格子点間の通信にはグローバルビューモデルが提供する reflect 指示文を用いる．reflect 指示文は shadow 指示文により，分散配列の分散境界を挟んで隣接する上端・下端の要素の値を保持する領域である，袖領域の値を更新する指示文である．**図 6** は，reflect 指示文による実装例であり，図 4，図 5 と同様の通信を表している．width 節により片袖のみの通信とし，periodic 指定により極座標系といった周回データに対して，末端プロセス同士の袖領域の更新を行う事を表している．袖領域として隣接格子点間の通信を行うことで，MPI，XMP（coarray）実装時に行っていた通信のために一次元配列へのデータの格納を行う必要がなくなる．

表 1 実験環境

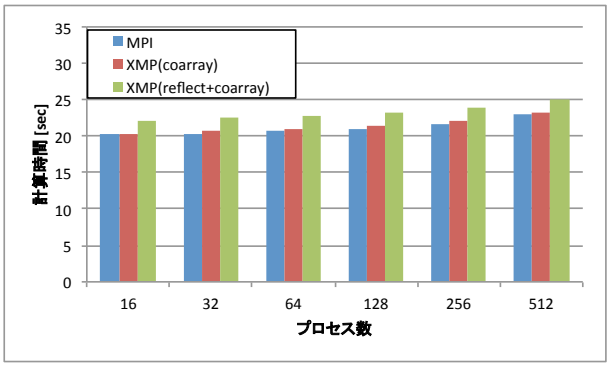
CPU	Intel Xeon E5-2670 × 2 (2.6GHz)
	CPU (8 cores/CPU) × 2 = 16 cores
Memory	128GB, DDR3 1600MHz
Interconnect	Infiniband QDR 4 Lines, 2 Rails
OS	CentOS 6.1
C Compiler	gcc 4.4.7
MPI	MVAPICH2 1.8.1

5. 評価

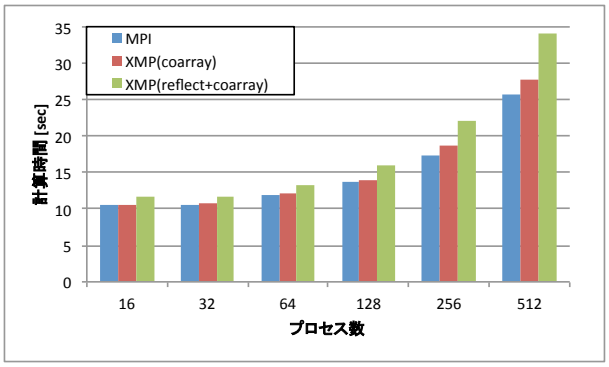
5.1 実験環境

4 章で述べた MPI による通信を coarray 記法へと変更した XMP（coarray）実装と，グローバルビューモデルが提供する gblock 分割を用いて領域分割を行い，隣接格子間の通信を袖領域通信の reflect 指示文とし，それ以外の通信を coarray 記法へと変更した XMP（reflect + coarray）実装の評価を行う．実験環境としては，筑波大学計算科学研究センターの超並列 GPU クラスタである HA-PACS[15] を用いた．HA-PACS の 1 ノードの計算機環境を**表 1** に示す．現時点では本研究は，CPU のみを用いた実装となっているが，将来的に GPU 対応を行うことから同システムを用いている．

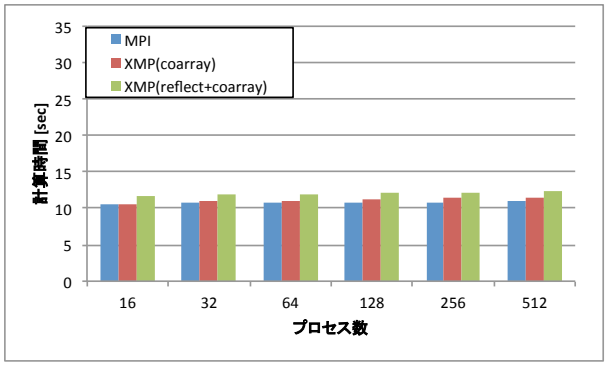
GTC-P には演算量を決定するパラメータとして，演算ステップ数である mstep，径方向の格子数である mpsi，最外殻でのポロイダル格子数である mthetamax，トロイダル格子数である mzetamax，格子点あたりの粒子数がある．本稿で評価に用いる問題サイズとして，GTC-P が提供する問題サイズ A を用いており，これは mstep = 100，mpsi = 90，mthetamax = 640，mzetamax = 64，格子点当たりの粒子数は 100 であるが，演算ステップ数は mstep = 20 としている．また，GTC-P ではトロイダル方向の分割数とトロイダル格子数 mzetamax が同値である必要がある事から，このパラメータについても問題サイズ A から変更を



(a) トロイダル方向の分割数を変動させた場合の演算時間.



(b) 径方向の分割数を変動させた場合の演算時間.



(c) 粒子数の分割数を変動させた場合の演算時間.

図 8 弱スケーリングの評価.

加えている.

それぞれの次元での分割数は，2 つの次元の分割数をそれぞれ 2 に固定し，一次元の分割数を 4 から 128 と増加させる．例えば，トロイダル方向の分割数を 4 から 128 へと増加させる場合は，径方向と粒子数の分割数はそれぞれ 2 となる．また，GTC-P の OpenMP によるスレッド並列実行は行わず，1 ノード 16MPI プロセスとして最大 32 ノードを用いて計測を行った.

5.2 性能

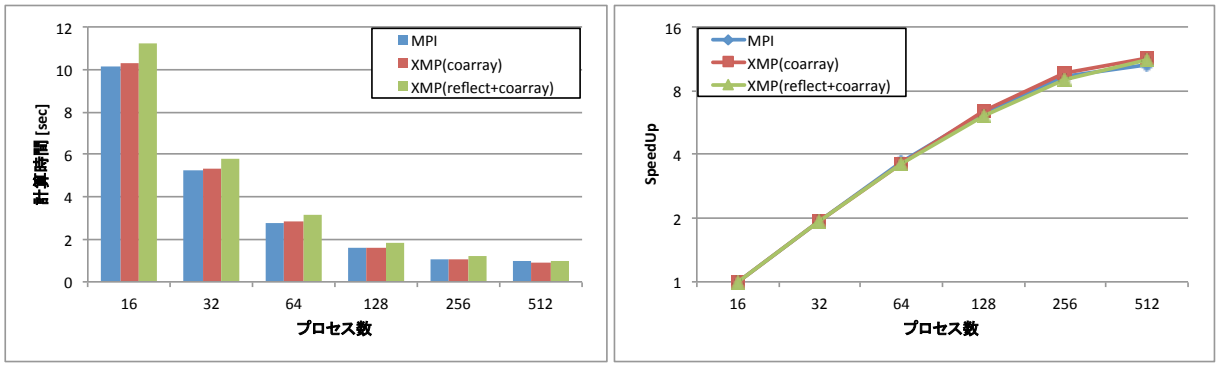
図 8 に弱スケーリングによる評価を示す．XMP（coarray）による実装は，MPI の通信を XMP の reduction 指示文，または，coarray 記法による通信に置き換えただけなので MPI 実装とほぼ同等の性能となっている．XMP（reflect + coarray）による実装では，MPI 実装と比較して性能は約 8 から 25%の差がある．径方向による分割数を変動させた評価 (b) の弱スケーリングが，他の二つの分割数を変動させた場合と比較して MPI と XMP 実装に差が生じた．これは，径方向分割時のロードバランスが特に悪く，MPI_Sendrecv による 2 プロセス間の同期が，coarray 通信時に用いる xmp_sync_all による全体同期となり，同期待ち時間が増加したためである．しかし，全体として径方向分割時の弱スケーリングが悪くなるのは不自然であり，現在調査中である.

図 9 に径方向の分割数を増加させた場合，**図 10** に粒子数の分割数を増加させた場合の強スケーリングによる評価を示す．トロイダル方向の分割数を増加させた場合は，GTC-P の制約により実行する事ができないため二種類の評価となっている．図 9 (a)，図 10 (a) より，XMP (coarray) による実装は弱スケーリング同様，ほぼ MPI 実装と同等の性能となっており，XMP (reflect + coarray) による実装では，およそ 5 から 25%の性能差が生じた．また，図 9 (b)，図 10 (b) より XMP を用いた両実装についても MPI 同様のスケーリングをしている.

XMP（reflect + coarray）実装と MPI による実装との差としては，XMP（reflect + coarray）実装での reflect 指示文による袖通信の通信量が大きくなる事が挙げられる．これは，一部の通信において，袖領域の中の限られた部分のみを通信するといった動作があるためである．また，gblock により分散された配列に対する参照は，現在の XMP の実装上，関数を呼び出す実装となっていることから，分散配列参照時にかかるオーバヘッドが MPI による実装との差であると考える.

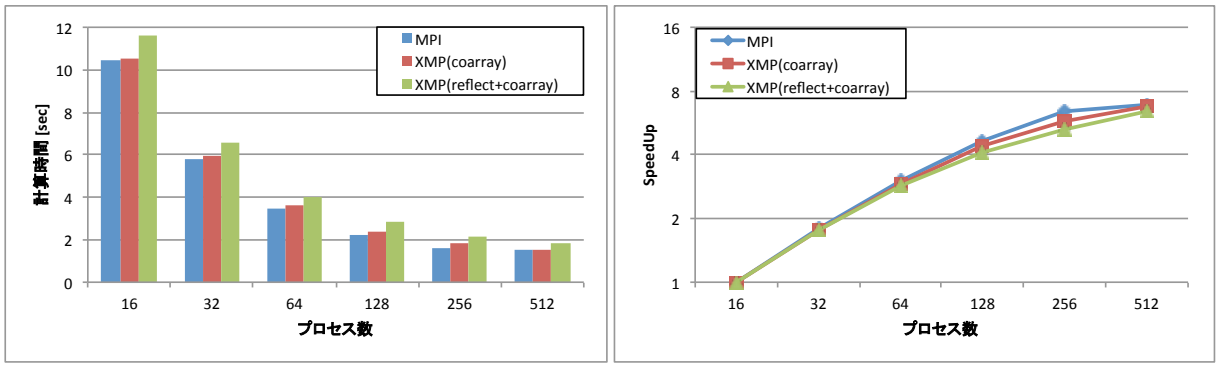
5.3 生産性

XMP（coarray）による実装は，MPI 実装と比較して一対一通信の記述が coarray 記法による配列代入文形式で表現していることから，簡易な実装であると言える．また，



(a) 演算時間　　(b) SpeedUp

図 9　強スケーリングの評価．径方向分割数を変動させた場合．



(a) 演算時間　　(b) SpeedUp

図 10　強スケーリングの評価．粒子数の分割数を変動させた場合．

MPI，XMP（coarray）実装では，通信時に 1 次元配列へと値の格納を行い，その配列を隣接プロセス間で通信といった実装をとるのに対して，XMP（reflect + coarray）実装では，reflect 指示文により袖領域として値の更新する事で，通信自体を 1 行の指示文で記述する事ができる．よって，より見通しがよく生産性が高いと言える．しかし，GTC-P では一次元で確保したポインタに対して配列参照形式で二または，三次元として扱っている．XMP の仕様上 loop 指示文による分散は全ての次元数分の for ループが必要となるため，MPI 実装と比較して行数が増加している．現在，XMP には array 指示文が実装されており，この指示文では初期化と言った処理を分散配列に対して行う場合に，指示文を用いて for ループを使用する事なく値の代入を記述する事が出来る事から，この指示文を用いる事でより生産性の向上が見込まれる．

6. まとめと今後の課題

本稿では，各融合シミュレーションコードである GTC-P の XcalableMP 化を行った．その際に隣接格子点間と粒子の移動による通信を全て coarray 記法で実装した XMP（coarray）と，領域分割をグローバルビューモデルの gblock 分散を用いて行い，隣接格子点間の通信を reflect 指示文による袖領域通信とし，粒子の移動による通信を coarray 記法で実装した XMP（reflect + coarray）の二種類の実装を行った．XMP（coarray）実装は，MPI 実装とほぼ同等の性能となり，XMP（reflect + coarray）実装は，約 5 から 25%の性能劣化となった．生産性の観点からは，隣接格子間の通信を指示文一行にすることができ，その他の通信に対しては配列代入文形式で記述可能な coarray 記法を用いた事により，MPI と比較してより簡易に表現する事が出来た．

今後の課題として，ITER といった大規模な核融合装置に対応した問題サイズにした場合の評価や，XMP と OpenMP によるハイブリッド並列化による評価等が挙げられる．また，現在 Omni Compiler Project では，XMP と OpenACC の統合が検討されており，それらを用いた GPU 化などが挙げられる．

謝辞　本研究の一部は JST-CREST 研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」，研究課題「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」による．

参考文献

[1] XcalableMP Specification Working Group: *XcalableMP Specification Version 1.2, (2013)*, http://www.xcalablemp.org/download/spec/xmp-spec-1.2.pdf.
[2] 李　珍泌，朴　泰祐，佐藤三久：分散メモリ向け並列言語 XcalableMP コンパイラの実装と性能評価，情報処理

学会論文誌. コンピューティングシステム， Vol. 3, No. 3, pp. 153–165 (2010).

[3] Shimomura, Y., Aymar, R., Chuyanov, V., Huguet, M., Parker, R. and Team, I. J. C.: ITER overview, *Nuclear Fusion*, Vol. 39, No. 9Y, p. 1295 (1999).

[4] Ethier S, Adams M, C. J. and L, O.: Petascale Parallelization of the Gyrokinetic Toroidal Code, *In proceedings of VECPAR'10 9th International Meeting on High Performance Computing for Computational Science* (2010).

[5] Wang, B., Ethier, S., Tang, W., Williams, T., Ibrahim, K. Z., Madduri, K., Williams, S. and Oliker, L.: Kinetic Turbulence Simulations at Extreme Scale on Leadership-class Systems, *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, New York, NY, USA, ACM, pp. 82:1–82:12 (2013).

[6] Madduri, K., Ibrahim, K. Z., Williams, S., Im, E.-J., Ethier, S., Shalf, J. and Oliker, L.: Gyrokinetic Toroidal Simulations on Leading Multi- and Manycore HPC Systems, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, New York, NY, USA, ACM, pp. 23:1–23:12 (2011).

[7] 奴賀秀男，朴　泰祐，藤田典久，中尾昌広，佐藤三久，WilliamTang：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，研究報告計算機アーキテクチャ（ARC）2013-ARC-207, 5 (2013).

[8] Numrich, R. W. and Reid, J.: Co-array Fortran for Parallel Programming, *SIGPLAN Fortran Forum*, Vol. 17, No. 2, pp. 1–31 (1998).

[9] 中尾昌広，TranMinhTuan，李　珍泌，朴　泰祐，佐藤三久：PGAS 言語 XcalableMP における coarray 機能の実装と評価，先進的計算基盤システムシンポジウム論文集，Vol. 2012, pp. 289–297 (2012).

[10] 内藤裕志，佐竹真介：5. 粒子シミュレーションのコーディング技法 (核融合プラズマシミュレーションの技法-大規模並列計算環境の活用-)，プラズマ・核融合学会誌，Vol. 89, No. 4, pp. 245–260 (2013).

[11] Hideo, N. and Atsushi, F.: Kinetic modeling of the heating processes in tokamak plasmas, *PhD Thesis, Kyoto Univ*, p. 7 (2011).

[12] 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. 8 (2005).

[13] DoE SCiDAC, UC Irvine: *Gyrokinetic Toroidal Code*, http://phoenix.ps.uci.edu/GTC/index.php.

[14] Murai, H. and Sato, M.: An Efficient Implementation of Stencil Communication for the XcalableMP PGAS Parallel Programming Language, *7th International Conference on PGAS Programming Models, Edinburgh, Scotland, UK, Oct.* (2013).

[15] 筑波大学 計算科学研究センター：*HA-PACS*, http://www.ccs.tsukuba.ac.jp/research/computer.