核融合シミュレーションコードGTC-PのXcalableMPによる実装

津金 佳祐[†1], 奴賀 秀男[†3], 朴 泰祐[†1, 2], 村井 均[†4],
佐藤 三久[†1, 2], William Tang[†5], Bei Wang[†6]

†1 筑波大学大学院システム情報工学研究科
†2 筑波大学計算科学研究センター
†3 日本原子力研究開発機構
†4 独立行政法人理化学研究所 計算科学研究機構
†5 Plasma Physics Laboratory, Princeton University
†6 Princeton Institute for Computational Science and Engineering, Princeton University

目次

- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

目次

- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

研究背景

- 分散メモリ環境上での並列言語はMPIが主流
  - 生産性の低さが問題となっている
    - 高いプログラミングコスト
    - ソースコードが煩雑になりやすい
- 新しい並列プログラミングモデルとして XcalableMP(XMP)が提案・開発
  - 規格：PCクラスタコンソーシアム
  - リファレンス実装：筑波大学, 理化学研究所

⇒2種類のプログラミングモデルにより柔軟に開発可能

研究背景

- 磁場閉じ込め型核融合プラズマの乱流現象の解析
  - 演算量の著しい増加
    - 正確なシミュレーションのための高い空間解像度
    - 国際熱核融合実験炉ITERといった装置規模の巨大化

高い並列性による性能向上や,
プログラム開発コストを下げるための高い生産性が必要

目的

- XMPによる並列プログラムの性能・生産性の評価
  - 核融合シミュレーションコードGTC-PをXMPで実装
  - MPI版と比較
  - 多国間国際共同研究 G8及びCREST
- 目標
  - MPI版と同等の性能
  - より簡易な記述で実装

目次

- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

XcalableMP (XMP)

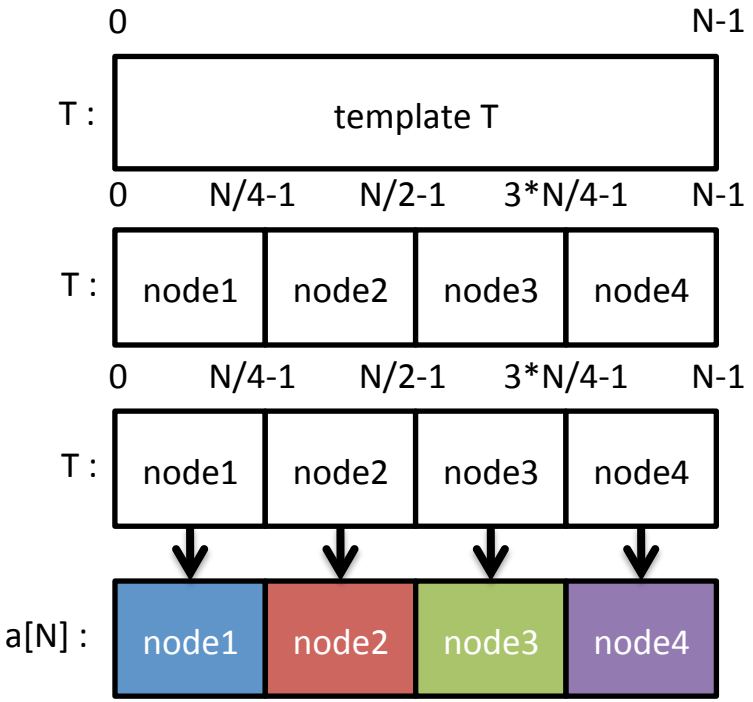
- 分散メモリ環境を対象とした並列言語
  - 既存言語(C, Fortran)の拡張
- OpenMPのような指示文形式
  - 明示的な並列化と通信

  ⇒逐次版と同等のプログラム
- 2種類のプログラミングモデルを実装
  - グローバルビュー
  - ローカルビュー

グローバルビューモデル 1/2

- 指示文によりデータ分散, 並列実行, 通信・同期
  - ステンシル演算のような, 静的な領域分割に適している
- データ分散にテンプレートを使用
  - 仮想的なアドレス空間

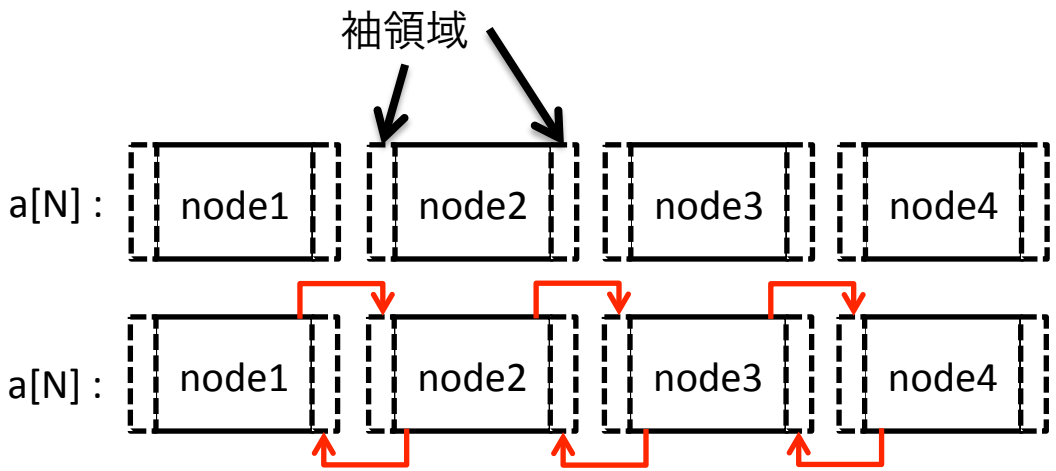
```
int a[N];
#pragma xmp nodes P(4)

#pragma xmp template T(0:N-1)

#pragma xmp distribute T(block) on P

#pragma xmp align a[i] with T(i)
```



グローバルビューモデル 2/2

- 通信指示文
  - shadow指示文
    - 隣接ノードが持つデータを保持する袖領域を確保
  - reflect指示文
    - 袖領域の値の更新

```
#pragma xmp shadow a[1]

#pragma xmp reflect (a)
```



ローカルビューモデル

- 各ノードが持つローカルデータに対して通信を行うモデル
  - ローカルデータサイズが動的に変化する場合等に適している
- coarray記法
  - Coarray Fortranの拡張, 片方向通信
  - XMP Cで使用可能

⇒MPIのようなノード毎の振る舞いを記述可能

```
int a[N], b[N];
#pragma xmp coarray a:[*]

if (mype == 2) /* mypeはノード番号 */
  a[0:4]:[1] =b[2:4];
```

ハイブリッドビュー

- グローバルビューは静的な演算領域の分割に最適
  - プロセス毎の演算量が動的に変化する場合に対応不可

- XMPはグローバルビューとローカルビューを組合せて使用可能
  - ローカルビューのcoarray記法で対応

  ⇒ハイブリッドビューと呼ぶ

目次

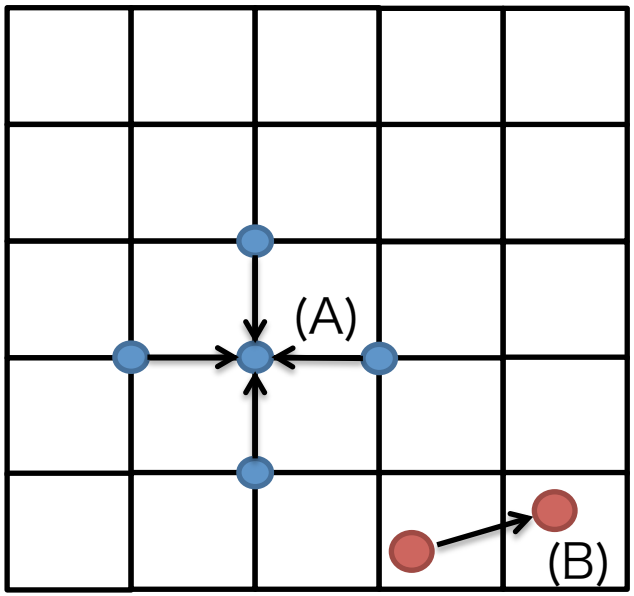
- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

GTC-P

- 磁場閉じ込め型核融合プラズマの乱流現象の解析を行うコード
  - プリンストン大学が開発
  - GTCの並列アルゴリズムの改良版
    - 分割次元数が増えている
  - 3次元PICコード
  - MPI+OpenMPによるハイブリッド並列化

PICコード

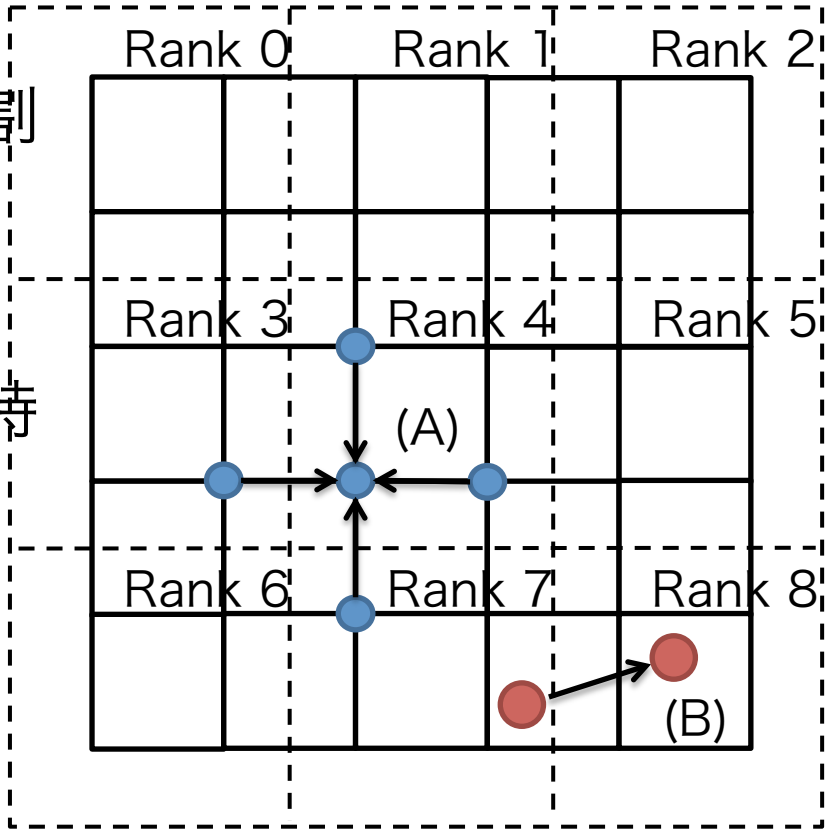
- PIC(Particle-in-Cell)法
  - 核融合プラズマ分野の粒子シミュレーション手法
  - 場の計算を行う計算格子と粒子軌道演算で構成



(A)場の計算
(B)粒子軌道演算

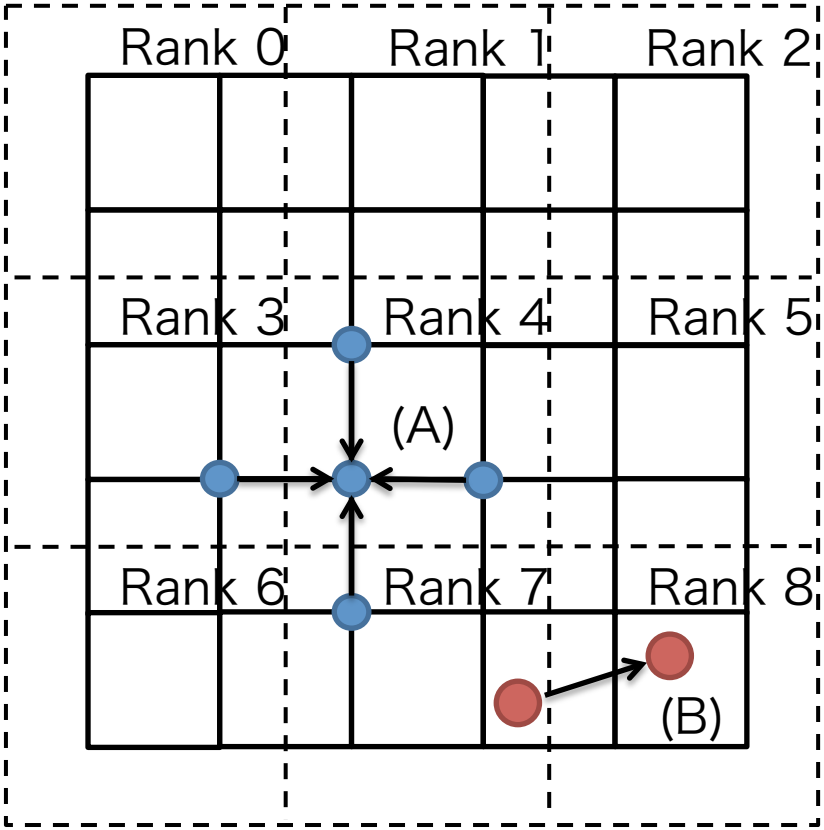
PICコードの並列化

- 領域分割
  - 計算格子を複数プロセスに分割
- 通信
  - 他プロセスが近傍格子点を所持
  - 粒子の移動先が他プロセスが持つ領域



PICコードのXMP実装例

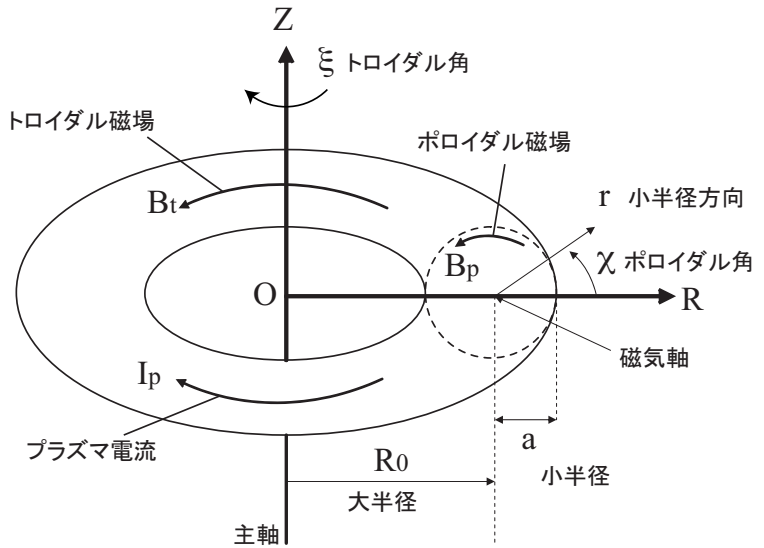
```
double f[X][Y]; /* 格子データ */
double myp[N]; /* 粒子データ */
double send[N], recv[N];

#pragma xmp nodes P(3,3)
#pragma xmp template T(0:X-1, 0:Y-1)
#pragma xmp distribute T(block, block) on P
#pragma xmp align f[i][j] with T(i, j)
#pragma xmp shadow f[1][1]
#pragma xmp coarray recv:[*]

for (t = 0; t < TIME; t++) {                    (A)
 /* 格子演算 */
#pragma xmp reflect (f) /* 隣接格子点の更新 */
 /* 粒子演算 */                                    (B)
 /* 別プロセスのもつ領域へ
    移動する粒子情報を”myp”から”send”へ */
 recv[0:icount]:[pe] = send[0:icount]; /* 粒子の移動 */
 xmp_sync_all(NULL); /* 同期 */
}
```



⇒ ハイブリッドビューにより領域分割とcoarrayを同時に記述

GTC-Pの座標系

- 3次元トーラス空間(トロイダル方向, ポロイダル方向, 径方向)
  - トロイダル方向分割時の断面がポロイダル断面
- GTC-Pの分割方法
  - MPIによる領域分割
    - トロイダル方向, 径方向
    - 粒子数
  - OpenMPによるスレッド並列化

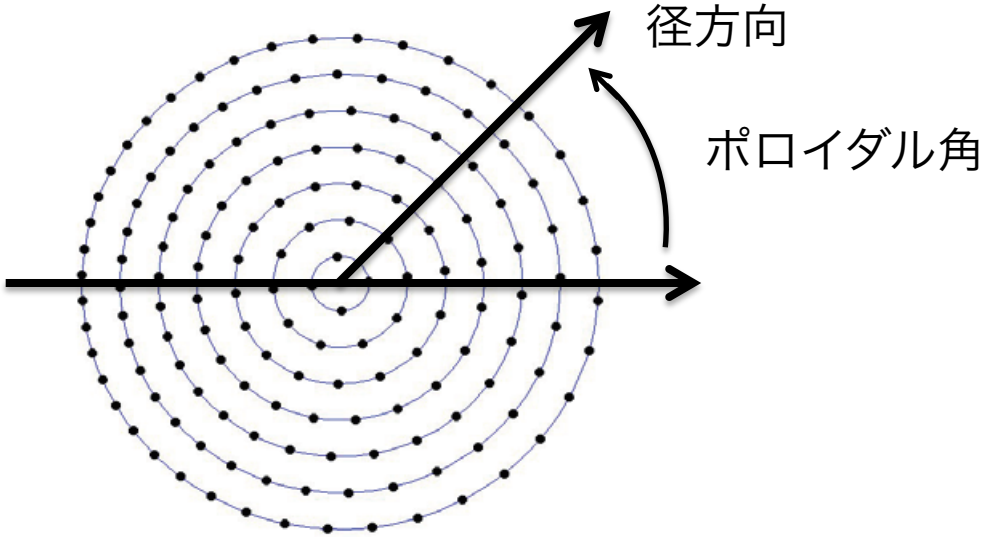


座標系[1]

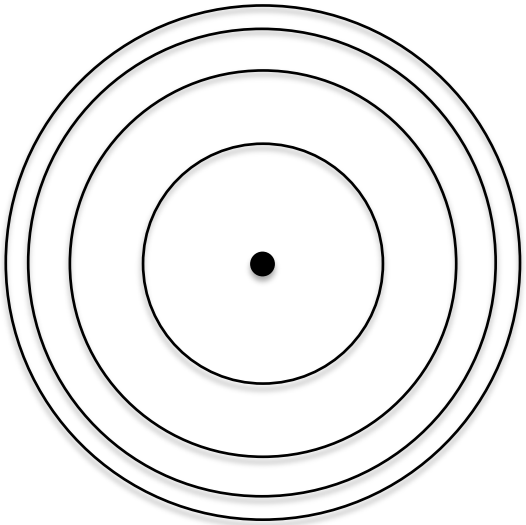
1 : Hideo, N. and Atsushi, F.: Kinetic modeling of the heating processes in tokamak plasmas, PhD Thesis, Kyoto Univ (2011).

GTC-Pの格子点

- ポロイダル断面上の格子点
  - 等ポロイダル角ではなく，等径方向距離



ポロイダル断面図[2]



径方向の領域分割

⇒ 演算量の均等化のために径方向の領域分割は非均等に行われる

2 : 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 (2005).

目次

- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

XMP実装の方針

- XMPにより2種類の実装を行う
  - ローカルビュー実装：XMP-localview
    - GTC-Pの通信をcoarray記法で実装
  - ハイブリッドビュー実装：XMP-hybridview
    - 計算格子の分割, 格子点間の通信
      - グローバルビュー
    - 粒子の移動
      - ローカルビュー
  - 共通
    - MPI_AllreduceなどのcollectiveはreductionへAllreduceなどのcollective通信はreduction指示文へ

ローカルビュー実装

- MPI_Sendrecvをcoarrayで記述
  - 隣接プロセス間の通信例

```
double *sendr, *recvl;

for(i=0;i<nloc_over;i++)
 sendr[i]=phitmp[i*(mzeta+1)+mzeta];

MPI_Sendrecv(sendr,nloc_over,double,right_pe,
        isendtag,recvl,nloc_over,double,left_pe,
        irecvtag,toroidal_comm,&istatus);
```

MPI

```
double Xsendr[nloc_over],Xrecvl[nloc_over];
#pragma xmp coarray Xrecvl:[*]

for(i=0;i<nloc_over;i++)
 Xsendr[i]=phitmp[i*(mzeta+1)+mzeta];

Xrecvl[0:nloc_over]:[right_pe]=Xsendr[0:nloc_over];
xmp_sync_all(NULL);
```

XMP-localview

  - xmp_sync_all
    - 全プロセス間でのcoarray通信の完了を保証

⇒coarray記法による実装はMPIと比較してより簡易な記述

ハイブリッドビュー実装 1/2

- XMP宣言部の実装例
  - 径方向の分割はプロセス毎に分割サイズが異なる
  - 非均等なブロックサイズによる分割が必要

  ⇒ gblockを使用
    - ユーザ定義のブロック分割

- MPI実装では, 動的にブロックサイズを決定
  - XMPは, 動的な2次元分散をする事が現在不可能

  ⇒例同様, 静的に記述

```
#define n_t        2 /* トロイダル方向の分割数 */
#define n_r        4 /* 径方向の分割数 */
#define n_rp       2 /* 粒子数の分割数 */
#define nloc_over_all          107722

real phitmp_g  [nloc_over_all][2 * n_t];
int  b [n_r * n_rp]
  ={10967,10967,14086,14086,16164,16164,12644,12644};
    /* gblock分散時の各プロセスのブロックサイズ */

#pragma xmp nodes P2(n_r * n_rp, n_t)
    /* ノードの分割数の指定 */
#pragma xmp template T(0:nloc_over_all-1, 0:2*n_t-1)
    /* テンプレート長の指定 */
#pragma xmp distribute T(gblock(b), block) onto P2
    /* テンプレートの分割方法の指定 */
#pragma xmp align phitmp_g      [i][j]     with T(i, j)
    /* 配列とテンプレートの対応 */
#pragma xmp shadow phitmp_g     [0][1:0]
    /* 袖領域の確保 */
```

ハイブリッドビュー実装 2/2

- 分散配列により近傍格子間の通信を指示文へ

```
double *sendr, *recvl;

for(i=0;i<nloc_over;i++)
 sendr[i]=phitmp[i*(mzeta+1)+mzeta];

MPI_Sendrecv(sendr,nloc_over,double,right_pe,
        isendtag,recvl,nloc_over,double,left_pe,
        irecvtag,toroidal_comm,&istatus);
```

MPI

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

XMP-hybridview

⇒reflect指示文により，近傍格子間の通信を1行で記述

目次

- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

実験環境

- HA-PACS (筑波大学計算科学研究センター)
  - 1ノード16MPIプロセス
  - 最大32ノード512MPIプロセスで計測
  - OpenMPによるスレッド並列化は行わない
    - MPIとXMPの比較のため

CPU	Intel Xeon E5-2670 x2 (2.6GHz)
	CPU (8 cores/CPU) x2 = 16 cores
Memory	128 GB, DDR3 1600 MHz
Interconnect	Infiniband QDR 4 Lanes x2 Rails
OS	CentOS 6.1
C Compiler	GCC 4.4.7
MPI	MVAPICH2 2.0

分割方法

- トロイダル方向, 径方向, 粒子数の分割数
  - 各分割数でStrong scaling, Weak scalingを計測
- 1次元の分割数を変動させ, 他2次元は2x2に固定
  - トロイダル方向のみGTC-Pの制約上, 演算量を揃えるため2x4

プロセス数	トロイダル方向	径方向	粒子数
16	2x2x4	2x4x2	2x2x4
32	4x2x4	2x8x2	2x2x8
64	8x2x4	2x16x2	2x2x16
128	16x2x4	2x32x2	2x2x32
256	32x2x4	2x64x2	2x2x64
512	64x2x4	2x128x2	2x2x128

(トロイダル方向分割数 x 径方向分割数 x 粒子数分割数)

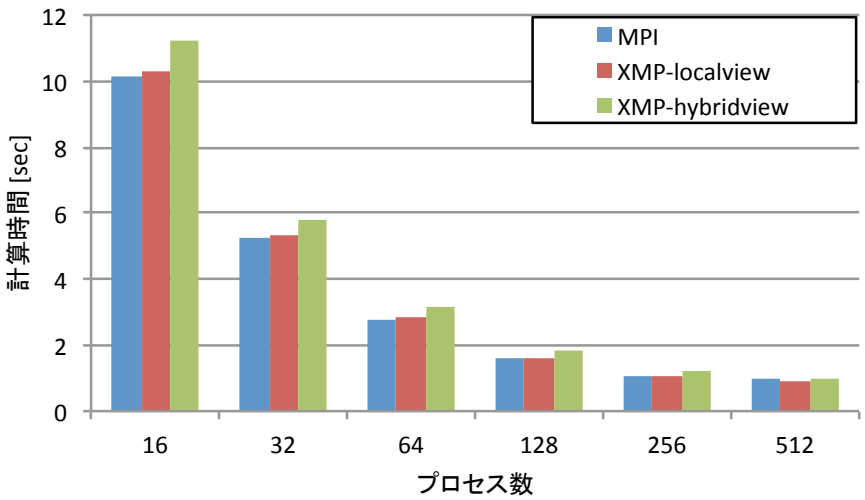
問題サイズ

- GTC-Pが提供する問題サイズAを用いる
  - Strong scalingは表中の最小値を使用
  - Weak scalingは表のようにパラメータを変更

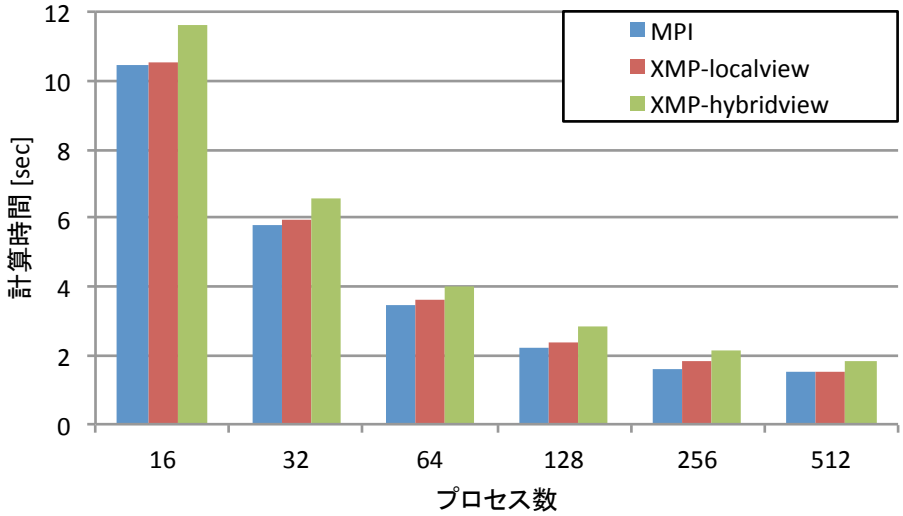
問題サイズA	Default	トロイダル方向	径方向	粒子数
mstep	100	20	20	20
mpsi	90	90	90 ~ 2880	90
mzetamax	64	2 ~ 64	2	2
格子点当たりの粒子数	100	100	100	100 ~ 3200

- mstep : 演算ステップ数
- mpsi : 径方向の格子点数
- mzetamax : ポロイダル格子数

評価 (Strong scaling)



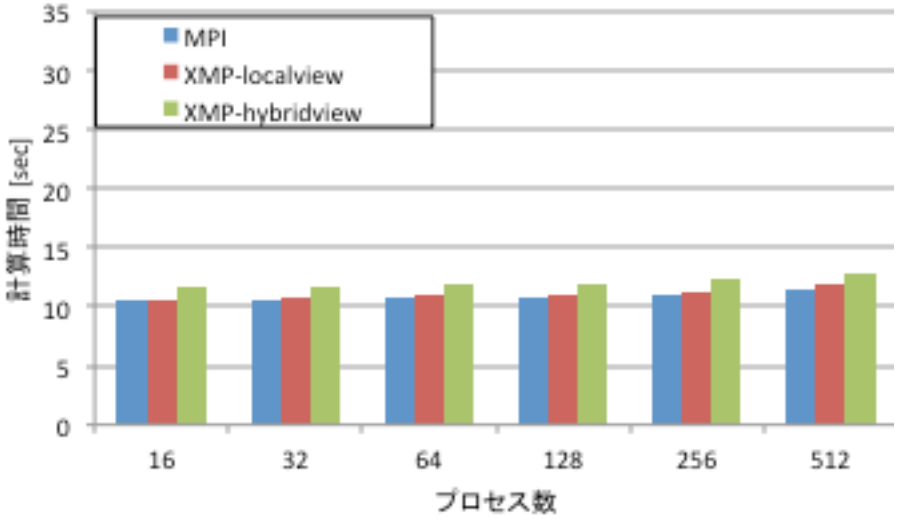
径方向分割 (2xNx2)



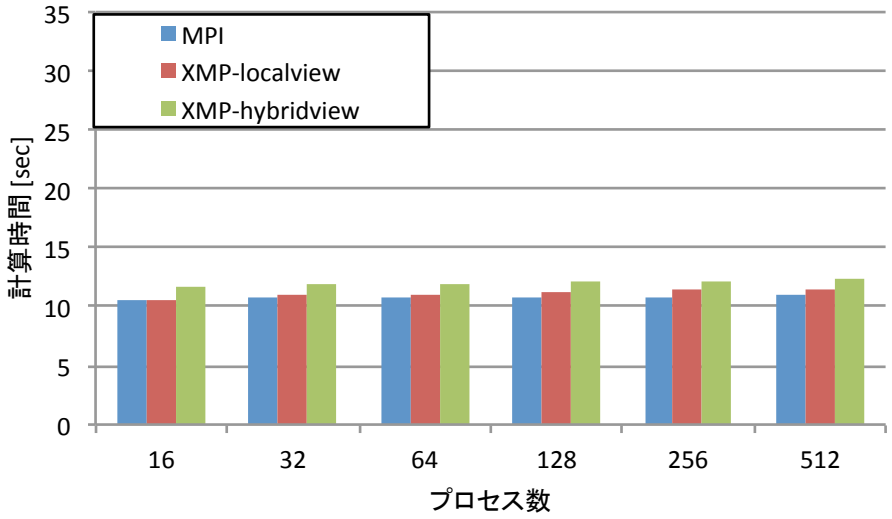
粒子数分割 (2x2xN)

- XMP-localviewとMPI実装は, ほぼ同等の性能
- XMP-hybridviewは5 ~ 25%の性能低下

評価 (Weak scaling) 1/2



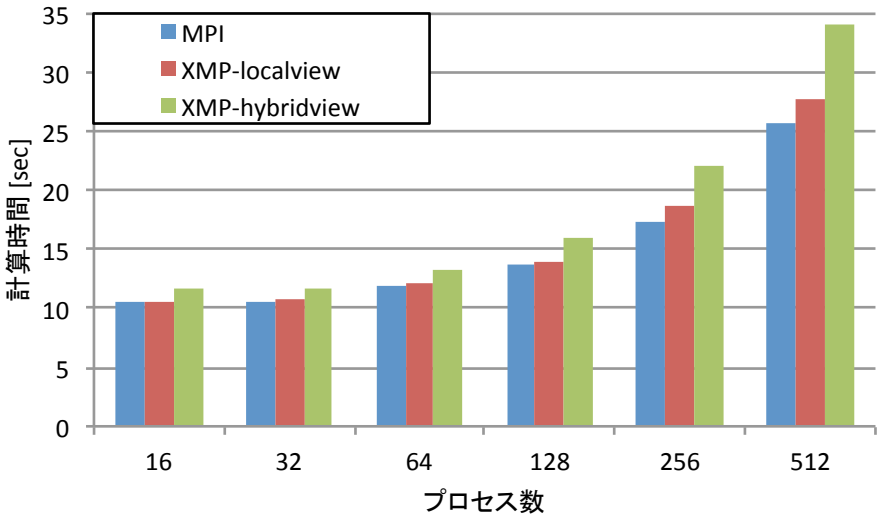
トロイダル方向分割 (Nx2x４)



粒子数分割 (2x2xN)

- XMP-localviewとMPI実装は，ほぼ同等の性能
- XMP-hybridviewは8 ~ 12%の性能低下

評価 (Weak scaling) 2/2



径方向分割 (2xNx2)

- XMP-localviewは8%, XMP-hybridviewは25%の性能低下
- 3つの実装とも同様の傾向
  - 他の分割方法と比較して, 全体的に性能が低い

考察 (Weak scaling)

- 各プロセスの実行時間の調査 (演算する格子点数)
  - 通信時間を除く

径方向分割

プロセス数	Min	Max
16	9.902 (10967)	10.144 (16164)
32	9.905 (12104)	10.405 (24200)
64	9.916 (14130)	11.354 (33462)
128	9.974 (17422)	12.798 (74745)
256	10.197 (23198)	14.737 (138127)
512	10.670 (34522)	18.165 (276110)

粒子数分割

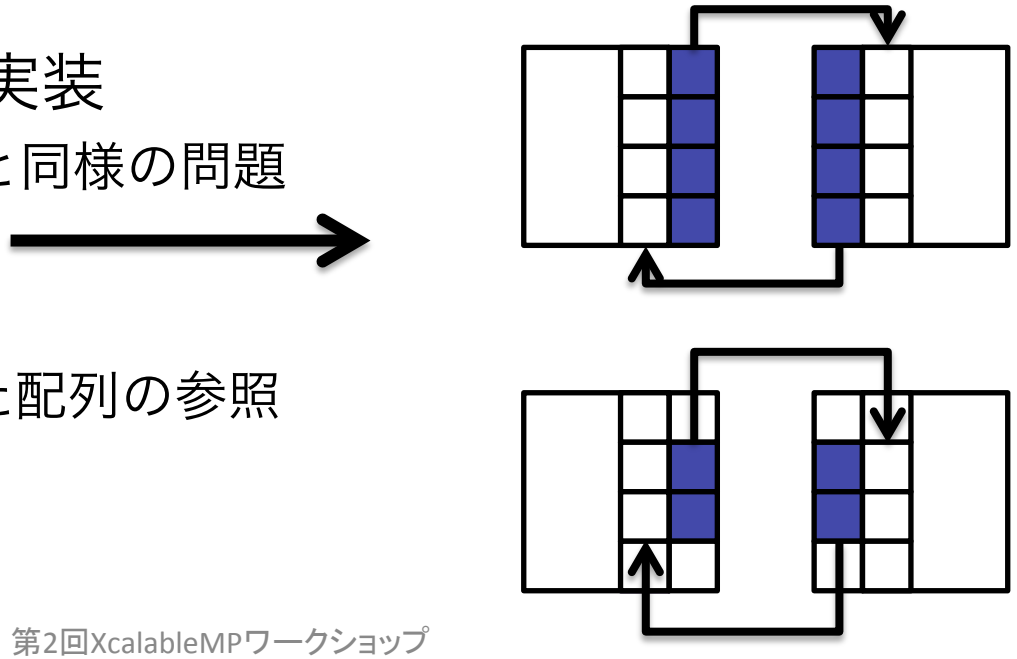
プロセス数	Min	Max
16	10.236 (19805)	10.399 (19916)
32	10.241 (19805)	10.425 (19916)
64	10.238 (19805)	10.431 (19916)
128	10.220 (19805)	10.444 (19916)
256	10.225 (19805)	10.447 (19916)
512	10.223 (19805)	10.462 (19916)

  - 径方向分割時のロードバランスが悪い
    - プロセス当たりの演算する格子点数に大きな差があるため

⇒ GTC-P自体の問題

考察

- ローカルビュー実装
  - 負荷が非均等のため通信時の同期処理がネック
    - MPI_Sendrecvは2プロセス間の同期
    - xmp_sync_allは全プロセスでの同期
  - post/wait指示文, xmp_sync_memoryで実装可能
- ハイブリッドビュー実装
  - ローカルビュー実装と同様の問題
  - 袖の部分的な通信
  - XMPのオーバヘッド
    - gblock分割された配列の参照



生産性

- ローカルビュー実装
  - coarray記法による配列代入文形式
- ハイブリッドビュー実装
  - reflect指示文による1行での袖領域通信
  - MPI版と比較してシリアル版からの変更点が少ない

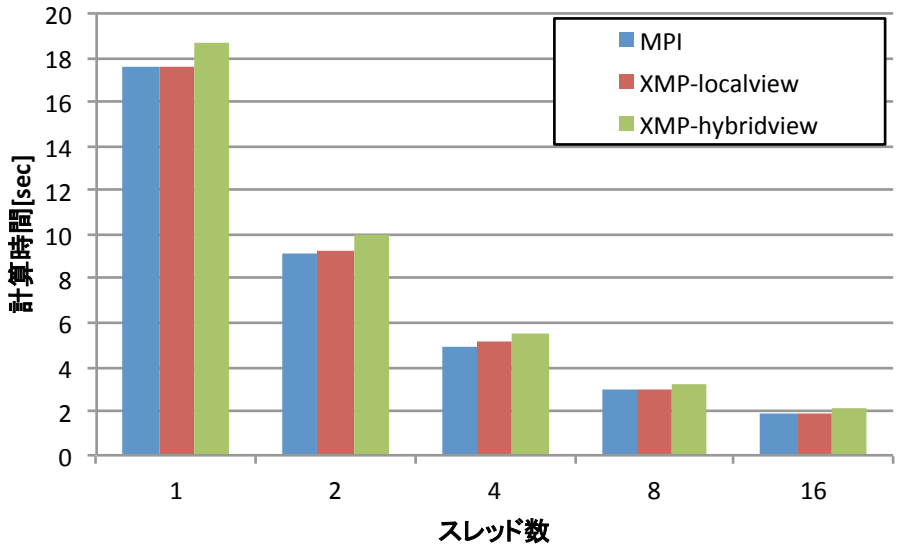
Serial	MPI	XMP-hybridview
4386	5319	5052

ソースコード(行数)

⇒ いずれの実装も見通しが良く生産性が高い

評価 (XMP+OpenMP)

- 1ノード1MPIプロセス, 8ノードを使用
- (トロイダル方向, 径方向, 粒子数) = (2, 2, 2)



- 3つの実装とも同様の傾向
  - XMP+OpenMPで性能が下がるということはない

目次

- 研究背景・目的
- XcalableMP
- 核融合シミュレーションコードGTC-P
- 実装
- 性能評価
- まとめ

まとめ・今後の課題

- GTC-PのXMP実装を行った
  - ハイブリッドビュー実装
    - 指示文による領域分割, 隣接格子間の通信を袖領域通信へ
    - 動的に演算量が変動する場合もcoarrayで対応可能
    - MPIと比較して5~25%の性能差
  - ローカルビュー実装
    - MPIの通信とcoarrayがほぼ同等の性能であることを示した
  - 各XMP実装の生産性の高さを示した
- 今後の課題
  - GPU対応
    - XcalableACC
  - 実問題サイズを解いた場合の問題点の調査
    - ITER規模の問題サイズ