Vol.2016-HPC-155 No.27
2016/8/9

PGAS言語XcalableMPによるFiberミニアプリ集の実装と評価

村井 均[1,a)]　中尾 昌広[1,b)]　岩下 英俊[1,c)]　佐藤 三久[1,d)]

概要：XcalableMP (XMP) は，PC クラスタコンソーシアム XMP 規格部会が策定している PGAS 言語である．本報告では，XMP の coarray 機能に基づくローカルビュー並列化により，理研が中心として整備している Fiber ミニアプリ集を実装および評価した結果を示す．多くの場合，XMP による実装は，オリジナル版の MPI 関数を機械的に置き換えることで得られた．Linux クラスタおよび京コンピュータにおける評価の結果，XMP による実装の性能は一部を除いてオリジナル版にほぼ近いものであった．

1. はじめに

分散メモリ並列計算機上のプログラミング手段としては，Message Passing Interface（MPI）が現在広く用いられている．しかし，並列化のあらゆる手順を明示することを強いられる MPI プログラミングは，ユーザにとって負担が大きい．そこで，より容易な並列プログラミングを可能にする手段として，High Performance Fortran (HPF)[1] を始めとするさまざまな並列言語がこれまで提案されており，最近では，Partitioned Global Address Space (PGAS) 言語と呼ばれる並列言語 [2], [3], [4] が提案されている．

そのような並列言語の一つである XcalableMP は，HPF の長所と短所を詳しく分析した上で設計された．その結果，XcalableMP は実用性と利便性を兼ね備えた並列プログラミングモデルとなっている [5], [6], [7]．我々は，筑波大学と共同で，XcalableMP 処理系のリファレンス実装である Omni XcalableMP を開発中である．

XcalableMP の特長の一つは，グローバルビューとローカルビューの 2 つの並列化モデルをサポートしていることである．グローバルビュー並列化では，指示文に基づく抽象度の高い記法によりデータマッピング，ワークマッピングおよび通信を記述することができる．ローカルビュー並列化では，Fortran 2008 規格から導入した coarray 機能を用いて，より細やかで柔軟な並列プログラムの記述が可能である．

一方，理化学研究所は，次世代のスーパーコンピューティング実現のためのミニアプリ集である Fiber の開発および保守を行っている [8]．

本研究は，XMP のグローバルビュー機能およびローカルビュー機能を用いて Fiber を実装・評価することを通じ，XMP の生産性と性能を検証することを目的とする．本報告では，このうちローカルビュー機能による実装と評価について扱う．また，ローカルビュー（coarray）に基づく実装を，必ずしも XcalableMP に限らない「coarray のベンチマーク」として整備していくことも目指す．本研究のさらにもう一つの目的は，coarray を用いた並列プログラミングに関する知見を得ることである．

以下，2 章，3 章，4 章でそれぞれ XcalableMP 言語仕様，Omni XcalableMP，Fiber ミニアプリ集について概観した後，5 章で XMP の coarray 機能による Fiber の実装を説明する．続いて，6 章ではその評価について述べる．最後に 7 章で本報告を総括する．

2. XcalableMP

XcalableMP は，PC クラスタコンソーシアム並列プログラミング言語 XcalableMP 規格部会において検討されている分散メモリ向け並列プログラミング言語である [5]．その主な特徴を以下に挙げる．

- **指示文ベース**
  データマッピング, ワークマッピング, 通信などは, ベース言語である C または Fortran の指示文の形式で記述する（以下，ベース言語 C に対する XcalableMP 仕様を XMP/C，ベース言語 Fortran に対する XcalableMP 仕様を XMP/Fortran と呼ぶ）．

[1] 国立研究開発法人理化学研究所 計算科学研究機構
Advanced Institute for Computational Science, RIKEN
[a)] h-murai@riken.jp
[b)] masahiro.nakao@riken.jp
[c)] hidetoshi.iwashita@riken.jp
[d)] msato@riken.jp

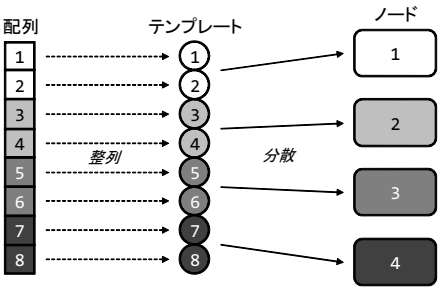

図 1 グローバルビュー並列化におけるデータ分散（一次元の場合）

- **明示的な並列化**
  プログラムの実行は SPMD モデルに従う．処理系がループの並列実行や通信を行うのは，指示文により明示的に指定された場合のみであるため，ユーザはプログラムの挙動や性能を予測しやすい．
- **データ/タスク並列処理の統合**
  データ並列処理とタスク並列処理を統合して記述することができる．
- **グローバルビュー並列化とローカルビュー並列化**
  問題全体を各ノードに分配する方法を記述することにより並列化を行うグローバルビュー並列化（*e.g.* HPF）と，各ノードが実行すべき処理を記述することにより並列化を行うローカルビュー並列化（*e.g.* MPI）の両方が可能である．
- **ハイブリッド並列化**
  MPI および OpenMP との併用が可能である．

さらに，現在，規格部会において，ベース言語 C++のサポート，動的タスク機能等を含む XcalableMP2.0 仕様の策定が進められている．

2.1 グローバルビュー並列化

XcalableMP のグローバルビュー並列化は以下のように記述される．データ（ほとんどの場合，配列）は，`align` 指示文の指定により，**テンプレート**に対して**整列**する．次に，テンプレートは，`distribute` 指示文の指定により，**ノード**の集合へ**分散**される．結果として，配列は，テンプレートを介してノード集合へ分散される（**図 1**）．多くの場合，グローバルビュー並列化に基づく XcalableMP プログラムは，XcalableMP 指示文を無視すれば，通常の C プログラムまたは Fortran プログラムとして解釈することができる．

XcalableMP におけるテンプレートは，データ並列処理の対象である集合（e.g. 差分法における格子点の集合, 粒子法における粒子の集合）を表すと考えられ，並列化の基準としての役割を果たす．また，ノードは，XcalableMP の計算機モデルにおいて，固有のメモリと CPU（複数のコアがあってもよい）を持つ構成単位である．ノード集合は，ノードを要素とする配列（ノード配列）として表現される．

```
1 real a(1024)[*], b(1024)
2
3 a(513:1024)[0] = b(1:512);
4
5 sync all
```

(a) Fortran における coarray

```
1 float a[1024]:[*], b[1024];
2
3 a[512:512]:[0] = b[0:512];
4
5 xmp_sync_all(&status);
```

(b) XMP/C における coarray

図 2 coarray の例

2.2 ローカルビュー並列化

XcalableMP のローカルビュー並列化は，通常の SPMD 実行と同様に，固有のメモリ空間を持つ各ノードが分割済みのデータを保持するというモデルに従う．MPI の通信関数に加え，Fortran 2008 規格で導入された片側通信機能である coarray を利用し，各ノードの挙動を個別に記述することができる [9]．代入文の形式で通信を記述できる coarray 機能による並列化は，MPI による並列化に比べ，生産性の点で優れていると考えられる．

coarray 機能における並列実行の主体は「イメージ」と呼ばれる．coarray を用いると，代入文の形式でイメージ間の片側通信を記述することができる．**図 2**(a) に coarray の例を示す．MPI におけるランク番号が 0 ベースであるのに対し，XMP におけるノード番号と，coarray 機能におけるイメージ番号は 1 ベースであることに注意されたい．

XMP における coarray 機能は，Fortran 2008 のそれに対し，以下の 2 つの点で拡張されている．

- XMP/C における coarray
  ベース言語 C においても coarray を利用できる．図 2(b) に C 言語における coarray の例を示す．通常のデータの宣言または参照の直後にコロンおよびイメージ番号を囲む角括弧を付加することで coarray の宣言または参照を記述する（1 行目，3 行目）．イメージ間の同期のためには `xmp_sync_all` 等の関数が提供される（5 行目）．また，3 つ組による部分配列記法を用いて，複数の配列要素に対する代入処理を一つの代入文で記述することができる（3 行目）[*1]．
- イメージの部分集合
  Fortran 2008 における coarray は常に全イメージで割り付けられ，かつ，イメージ番号は一意である．した

[*1] Fortran における 3 つ組（下限:上限:刻み幅）とは異なり，XMP/C における 3 つ組は [下限:長さ:刻み幅] の形式である．これは，Cilk Plus [10] や OpenACC [11] における定義に合致している．

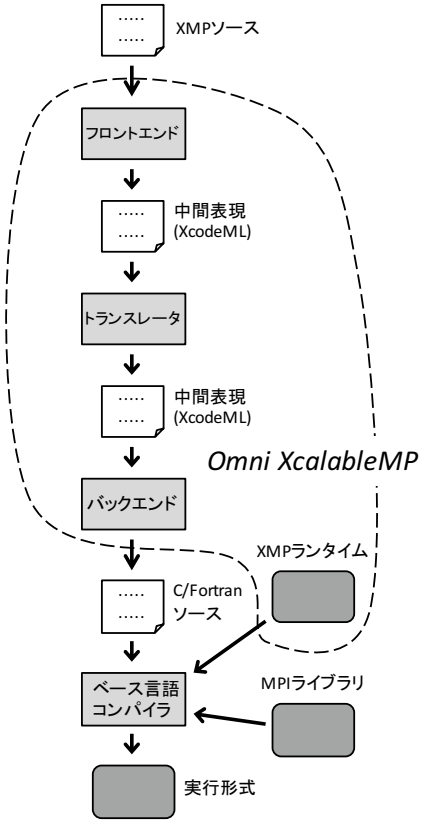

図 3 Omni XcalableMP の構成

がって，特定のイメージのみに coarray を割り付けたり，イメージの部分集合から新たなイメージ集合を定義するといったことはできない [*2]．XMP では，任意のノード集合を指定して coarray を宣言することが可能となっており，Fortran 2008 に比べより自由度の高い記述（e.g. coarray を用いたタスク並列処理や MPMD 実行）ができる．

3. Omni XcalableMP

我々は，筑波大学と共同で，XcalableMP 処理系のリファレンス実装である Omni XcalableMP を開発中である．Omni XcalableMP は，ベース言語として Fortran 95 および C90 をサポートし，京コンピュータや富士通 FX10, NEC SX，日立 SR を始め，MPI が動作する任意の環境をサポートする．

Omni XcalableMP の構成を**図 3** に示す．

XcalableMP ソースは，フロントエンドにおいてパースされ，XcodeML/C 形式 [12] または XcodeML/Fortran 形式 [13] の中間表現へ変換される．中間表現はトランスレータにおいて必要な変換（並列化）を施された後，バックエンド（デコンパイラ）によって，XMP ランタイム呼び出しを含む C または Fortran のソースプログラムへ変換される．この並列化ソースプログラムが，ベース言語コンパイラによりコンパイルされ，XMP ランタイムおよび通信ライブラリとリンクされて，実行形式を得る．原則としてトランスレータは並列化に必要な変換のみを担い，各種の標準的な最適化（e.g. レジスタ割付け，SIMD 化）は後段のベース言語コンパイラに委ねられる．

現在のところ，グローバルビューにおける各通信 (e.g. ステンシル通信）の実装には，MPI が用いられている．また，ローカルビューにおける coarray の実装には，MPI-3 の片側通信機能，GASNet [14] または京の MPI の拡張 RDMA インタフェース [15] が用いられており [16], [17]，Omni XcalableMP のビルド時にいずれかを選択する．

[*2] Fortran の次期規格では，イメージの部分集合を扱うための「team」と呼ばれる機能が導入される予定である．

表 1 Fiber ミニアプリの構成

CCS QCD	格子 QCD
FFVC-MINI	有限体積法に基づく熱流体解析
NICAM-DC	全球大気大循環モデルの力学過程
mVMC-MINI	強相関電子系における物理量の基底状態期待値の計算
NGS Analyzer-MINI	次世代シークエンス解析
MODYLAS-MINI	古典分子動力学シミュレーション
NTChem-MINI	第一原理電子状態計算
FFB-MINI	有限要素法に基づく熱流体解析

4. Fiber

Fiber は，理研が中心として整備を行っている，次世代のスーパーコンピューティング実現のためのミニアプリ集である [8].

表 1 に，現在公開中の Fiber ミニアプリの構成を示す．

各アプリは MPI で並列化されており，一部は OpenMP によりスレッド並列化を明示されている．

このうち，FFVC-MINI は，Fortran90 で記述された計算コア部分と，領域分割型のアプリケーションを記述するためのミドルウェアである CPMlib から成る．CPMlib は，C++で記述されており，データ領域確保，並列領域管理，通信などの機能を提供する [18]．すなわち，CPMlib の機能・位置づけは，XMP のそれと同じである．CPMlib を用いて既に並列化されている FFVC-MINI を，同機能の XMP で書き直すことの意義は小さいと考えられること，C++は現在の XMP のベース言語に含まれていないことの 2 つの理由により FFVC-MINI を本研究の対象から除くものとする．

5. 実装

5.1 基本方針

各ミニアプリに含まれる MPI 通信関数を，以下の基本方針に従って，coarray に基づく記述で置き換えることを行う．

- `MPI_Send` / `MPI_Isend` → 左辺に coarray が現れる代

入文（put ベース）

- MPI_Recv / MPI_Irecv → 削除
- 集団通信 → Fortran 2015 組込みサブルーチン co_broadcast，co_sum 等 [*3]
- MPI_Wait → sync all 文

対応する組込みサブルーチンが存在しない MPI_Allgather 等の集団通信も，その通信パターンを分解し，Send/Recv の対を抽出することにより，複数の coarray 代入文の組み合わせとして表現する．

以下に，MPI_Send / MPI_Isend および MPI_Recv / MPI_Irecv に対する置き換えの手順を，さらに詳しく説明する．

(1) 対応する MPI_Send と MPI_Recv の対を見つける．
(2) MPI_Recv のバッファを coarray として宣言する．
(3) MPI_Send を，coarray として宣言した MPI_Recv のバッファを左辺，MPI_Recv のバッファを右辺とする代入文で置き換える．バッファの要素数が 2 以上である場合，この代入文は，部分配列記法による配列値に対する代入を行う．
(4) MPI_Recv を削除する．

図 4 に，MPI 通信の置き換えの例を示す．

5.2 coarray による不規則な通信の記述

一般に，coarray に限らず，片側通信では，通信を起動する側（origin）が，相手側（target）のバッファに関する情報を知っている必要がある．したがって，不規則な通信パターン（e.g. 非構造格子における隣接通信）を片側通信で記述するには，互いのバッファに関する情報を事前に交換しておく必要がある（cf. 定型的な通信パターンでは相手側バッファの情報は自明．両側通信では相手側バッファの情報は不要）．

非構造格子を扱う FFB-MINI では，**図 5**(a) のように，複数の送信元からデータを受信し，かつ，各送信元から受信するデータのサイズは動的に決まるため，オフセット値をインクリメントしながら，一つの連続したバッファにデータを詰め込むという手法が用いられている．この場合，送信側は，自身が送信するデータに対応するオフセット値を事前に知ることができないため，前節で述べた手順をそのまま適用することはできない．そこで，本研究では，以下に示す手法を用いた．

(1) 受信バッファ rbuf を 2 次元にする（2 行目）．
送信元の数の最大値を MAX_SRC，ある送信元から受信するデータのサイズの最大値を MAX_RCOUNT としたとき，$MAX\_RCOUNT \times MAX\_SRC$ の 2 次元配列として rbuf を宣言する．一般に，この書き換え後の rbuf のサイズは，元の MPI プログラムにおけるバッファのサイズ MAX_SUM_COUNT を上回るため，メモリの使用効率は低下する．
(2) 受信バッファの情報（pos）を事前に交換する（6〜8 行目）．
ランク p が送信したデータをランク q が受信するとき，ランク p における pos[q] の値は，q にとって p が何番目の送信元であるかを示す．
(3) put 処理（13〜17 行目）
ランク p は，ランク q の受信バッファ rbuf の第 pos[q] 列にデータを put する．

この手法により図 5(a) を書き換えたプログラムを同図 (b) に示す．ここで，7 行目および 13 行目の coarray 代入文において，角括弧内の cosubscript として src[i] と dst[i] に 1 が加えられているのは，ランク番号が 0 ベースであるのに対し，イメージ番号は 1 ベースであるためである．

```
1  real a(8), b(8), c(8)
2
3  if (myrank == 0) then
4    call MPI_Isend(a, 4, ..., 1, ...)
5  else if (myrank == 1) then
6    call MPI_Irecv(b, 4, ..., 0, ...)
7  end if
8
9  call MPI_Wait(...)
10
11 call MPI_Bcast(c, 8, ..., 0, ...)
```

(a) MPI プログラム

```
1  real a(8), b(8)[*], c(8)[*]
2
3  if (this_image() == 1) then
4    ! put処理
5    b(1:4)[2] = a(1:4)
6  else if (this_image() == 2) then
7    ! recvを削除
8  end if
9
10 sync all
11
12 call co_broadcast(c(1:8), 1)
```

(b) coarray によるプログラム

図 4 MPI 通信関数の置き換え

5.3 その他の書き換え

XMP 言語仕様または Omni XMP の制限に起因し，以下のようなプログラムの書き換えまたは修正が必要となる場合があった．

- **未サポートのベース言語規格**

[*3] 重要度の高いこれらの組込みサブルーチンを，XcalableMP は先行してサポーしている．

```
real sbuf(MAX_SCOUNT, NDST)
real rbuf(MAX_SUM_RCOUNT)

do i
  call MPI_Send(sbuf(1,i), scount(i), &
                ..., dst(i), ...)
end do

k = 1
do j
  call MPI_Recv(rbuf(k), rcount(j), &
                ..., src(j), ...)
  k = k + rcount(j)
}
```

(a) MPI プログラム

```
real sbuf(MAX_SCOUNT, NDST)
real rbuf(MAX_RCOUNT, MAX_NSRC)[*]
integer pos(NIMAGE)[*]

! posの情報を交換 (all-to-all)
do i
  pos(myrank+1)[src[i]+1] = i
end do

sync all

! put処理
do i
  rbuf(1:scount(i)-1, &
       pos[dst[i]+1])[dst[i]+1] &
    = sbuf(1:scount(i)-1, i)
end do

! recvを削除
```

(a) MPI プログラム

図 5 不規則な通信の場合の MPI 通信関数の置き換え

Fiber の一部のミニアプリ（e.g. MODYLAS-MINI）では，Omni XMP がサポートしない Fortran 2003 の機能である組込みモジュール `ISO_C_BINDING` が参照されている．これに対し，例えば，`ISO_C_BINDING` の中で定義されている名前付き定数 `c_int` （C 言語の `int` 型に対応する `integer` 型の種別型パラメタを表す）が使用されている箇所では，実際の環境に応じて以下のような書き換えを行った．

`integer(c_int)` → `integer(kind=4)`

- **coarray を含むモジュール**
  現在の Omni XMP の制限事項により，coarray を含むモジュールは一つのファイル内に配置されている必要があるため，該当するファイルを結合後にコンパイルを行うように make ファイルを修正した．

表 2 評価の状況

	実装	クラスタ	京
CCS QCD	作業中	-	-
FFVC-MINI	対象外	-	-
NICAM-DC	○	◎	◎
mVMC-MINI	○	N/A	N/A
NGS Analyzer-MINI	○	◎	◎
MODYLAS-MINI	○	○	N/A
NTChem-MINI	作業中	-	-
FFB-MINI	○	○	N/A

表 3 Linux クラスタの評価環境

CPU	Xeon E7-4820 v3 @ 1.90GHz（10 コア）× 4 ソケット
メモリ	256GB
ネットワーク	ノード内（QPI）
OS	Ubuntu Server Release 16.04 LTS x86_64
カーネル	Linux Kernel 4.4.0-22-generic
GCC	5.3.1

	Omni	MPI
MODYLAS-MINI	0.9.3-release	MPICH3.2
NGSA-MINI		
NICAM-DC	0.9.3-20160224	
FFB-MINI	1.0-20160526	Intel MPI 5.1.3

6. 評価

5 章で示した方法により実装した各プログラムの性能を，Linux クラスタおよび京コンピュータ [19] 上で評価した．

表 2 に，実装および評価の状況を示す．表中の◎はスケーラビリティの評価を行ったことを，○は特定の並列度で並列実行を行ったことを意味する．

6.1 Linux クラスタにおける評価

Linux クラスタにおける評価で用いた環境を**表 3** に示す．coarray 機能のための通信ライブラリとしては，MPI-3 を選択した．また，ベース言語コンパイラである GCC には，Fiber のパッケージに付属する makefile で指定されているものと同じコンパイラ・オプションを指定した．

表 4 および**表 5** は，それぞれ FFB-MINI および MODYLAS-MINI の 8 並列実行の結果である．これらの表より，FFB および MODYLAS では，XMP 版の性能はオリジナルの MPI 版にほぼ近いことがわかる．特に FFB で XMP 版の性能が MPI 版をわずかに上回るのは，coarray による片側通信と MPI の一対一通信の性能差に依ると考えられる．

図 6 および**図 7** は，それぞれ NGS Analyzer-MINI および NICAM-DC のスケーラビリティ評価の結果である．グラフより，XMP 版の性能はオリジナルの MPI 版とほぼ同

表 4　FFB-MINI の評価結果（8 並列）

	経過時間 (秒)	
	オリジナル	XMP
time コマンド	183.60	174.57
LES3X	182.96	174.00
PREP	13.16	12.71
MAIN LOOP	169.63	161.13
POST	0.17	0.17

表 5　MODYLAS-MINI の評価結果（8 並列，データ wat222）

	経過時間 (秒)	
	オリジナル	XMP
time コマンド	336.97	350.74
Total	336.66	350.64
Setup	5.16	5.33
MD main loop	331.36	345.17
Output	0.07	0.06
Closing	0.14	0.14

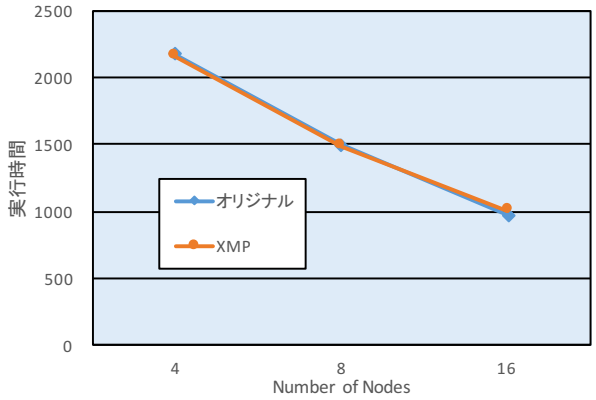

図 6　NGS Analyzer-MINI の評価結果（データ: 日本人男性ゲノムデータ）

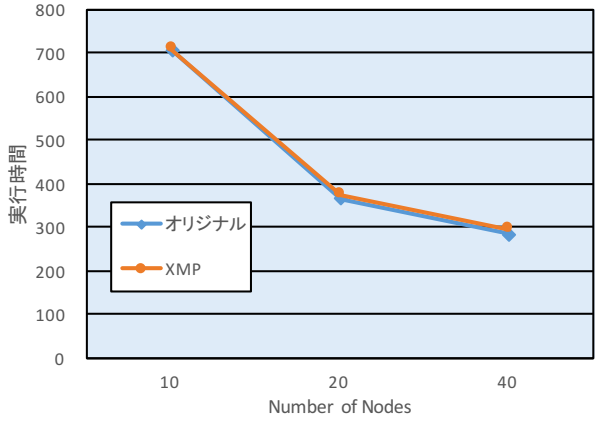

図 7　NICAM-DC の評価結果（データ: gl06rl01z80pe{10,20,40}, LSTEP_MAX=132）

じであることがわかる．

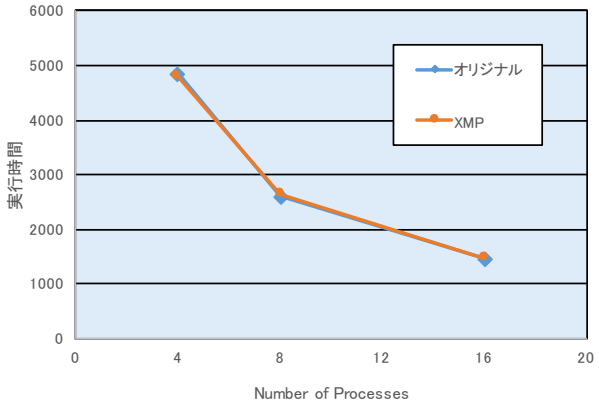

図 8　NGS Analyzer-MINI の評価結果（データ: 日本人男性ゲノムデータ）

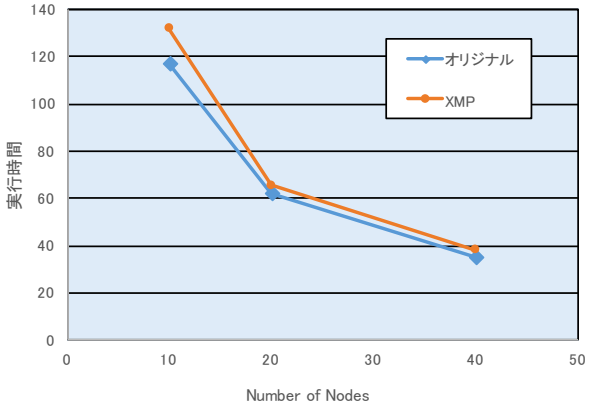

図 9　NICAM-DC の strong scaling 評価結果（データ: gl06rl01z80pe{10,20,40}, LSTEP_MAX=132）

6.2　京コンピュータにおける評価

京コンピュータにおける評価では，言語環境 K-1.2.0-20-1 および Omni コンパイラ Version 1.0.1 を用いた．ベース言語コンパイラである fccpx および frtpx には，Fiber のパッケージに付属する makefile で指定されているものと同じコンパイラ・オプションを指定した．

NGS Analyzer-MINI の評価結果を**図 8** に示す．計算ノード内の各コアに XMP ノード（MPI プロセス）を割り当てるフラット並列を用いた．グラフより，XMP 版の性能はオリジナルの MPI 版とほぼ同じであることがわかる．

NICAM-DC の strong scaling の評価結果を**図 9** に，同じく weak scaling の評価結果を**表 6** に示す．ベース言語コンパイラの自動並列化機能を適用し，計算ノード内の 8 コアでスレッド並列処理を行うハイブリッド並列を用いた．グラフおよび表より，XMP 版の性能は，オリジナルの MPI 版に比べやや低くなっていることがわかる．特に 10 並列と 640 並列の場合に差が大きくなっている．詳しい原因については，現在調査中である．

表 6 NICAM-DC の weak scaling 評価結果（データ: gl06rl01z80pe40, gl07rl02z80pe160, gl08rl03z80pe640, LSTEP_MAX=132）

	40	160	640
オリジナル	35.15	35.91	36.67
XMP	37.92	39.16	42.71

7. まとめ

XcalableMP の coarray 機能に基づくローカルビュー並列化により Fiber ミニアプリ集の実装と評価を行った．

多くの場合，coarray による実装は，オリジナル版における MPI 関数の呼び出しを，ほぼ機械的に置き換えることで得られる．ここから，生産性の点で，coarray と MPI はほぼ同等であるか，coarray がやや優れているものと考えられる．しかし，FFB-MINI のような非定型的なコードでは，通信（coarray 代入文）に先立って，ノード（イメージ）の間で互いのバッファに関する情報を交換する処理が必要になる場合もある．

Linux クラスタおよび京コンピュータにおける評価で，XMP 版の性能は，一部を除いてオリジナル版にほぼ近いものであったが，京コンピュータにおける NICAM-DC ではオリジナル版よりやや低くなっている．この原因については現在調査中である．

今後の課題は以下の通りである．

- 残りのミニアプリについて，ローカルビュー（coarray）による実装と評価を行う．
- グローバルビューによる実装と評価を行う．

謝辞 本論文の結果（の一部）は，理化学研究所のスーパーコンピュータ「京」を利用して得られたものです．

参考文献

[1] High Performance Fortran Forum: High Performance Fortran Language Specification Version 2.0, http://hpff.rice.edu/versions/hpf2/hpf-v20.pdf (1997).
[2] Robert W. Numrich and John Reid: Co-array Fortran for parallel programming, *ACM SIGPLAN Fortran Forum*, Vol. 17, No. 2 (1998).
[3] UPC Consortium: UPC Specifications, v1.2, Technical report, Lawrence Berkeley National Lab (LBNL-59208) (2005).
[4] Cray Inc: Chapel Language Specification 0.91, http://chapel.cray.com/spec-0.91.pdf (2012).
[5] XcalableMP Specification Working Group: XcalableMP Specification Version 1.2.1, http://www.xcalablemp.org/download/spec/xmp-spec-1.2.1.pdf (2014).
[6] Nakao, M., Lee, J., Boku, T. and Sato, M.: XcalableMP Implementation and Performance of NAS Parallel Benchmarks, *Fourth Conference on Partitioned Global Address Space Programming Model (PGAS10)*, New York (2010).
[7] 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).
[8] 小村幸浩，鈴木惣一朗，三上和徳，滝澤真一朗，松田元彦，丸山直也：Fiber ミニアプリの性能評価，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2014, No. 28, pp. 1–12（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009808123/⟩ (2014).
[9] 中尾昌広，佐藤三久：京速コンピュータ「京」における PGAS モデルによる気象コード NICAM の実装，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2013, No. 6, pp. 1–5（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110009588126/⟩ (2013).
[10] Intel Corporation: Intel®Cilk™Plus Language Extension Specification Version 1.2, https://www.cilkplus.org/sites/default/files/open_specifications/Intel_Cilk_plus_lang_spec_1.2.htm (2013).
[11] OpenACC-Standard.org: The OpenACC Application Programming Interface Version 2.5, http://www.openacc.org/sites/default/files/OpenACC_2pt5.pdf (2015).
[12] XcalableMP/Omni Compiler Project: XcodeML/C 仕様書 Version 0.9J, http://www.hpcs.cs.tsukuba.ac.jp/omni-openmp/xcodeml/XcodeML-C-0.9J.pdf (2009).
[13] XcalableMP/Omni Compiler Project: XcodeML/Fortran 仕様書 Version 0.9J, http://www.hpcs.cs.tsukuba.ac.jp/omni-openmp/xcodeml/XcodeML-Fortran-0.9J.pdf (2009).
[14] Bonachea, D.: GASNet specification Version 1.8, https://gasnet.lbl.gov/dist/docs/gasnet.pdf (2008).
[15] 富士通株式会社：Parallelnavi for MP10 V1.0 MPI 使用手引書 (2016).
[16] 中尾昌広，Tuan, T. M.，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，*Proc. SACSIS2012* (2012).
[17] Iwashita, H., Nakao, M. and Sato, M.: Preliminary Implementation of Coarray Fortran Translator Based on Omni XcalableMP, *Proc. 9th Int'l Conf. Partitioned Global Address Space Programming Models (PGAS2015)* (2015).
[18] Institute of Industrial Science, T. U. o. T.: CPMlib, https://github.com/fiber-miniapp/ffvc-mini/tree/master/src/CPMlib (2012).
[19] 高瀬 亮，横川三津夫（編）：情報処理，Vol. 53, No. 8, chapter 特集：京速コンピュータ「京（けい）」，一般社団法人 情報処理学会 (2012).