Vol.2012-HPC-135 No.44
2012/8/3

京速コンピュータ「京」における並列プログラミング言語 XcalableMPの評価

村井 均[1]　下坂 健則[1]　大野 善之[1]　宮本 佳明[1]　八代 尚[1]　富田 浩文[1]　佐藤 三久[1,2]

概要：XcalableMP は，MPI や HPF に代わるプログラミング手段を目指して提案された分散メモリ向け並列言語である．並列プログラミング言語 XcalableMP を用いて気象シミュレーションコードを並列化し，京速コンピュータ「京」および標準的なクラスタにおいて評価したところ，生産性と性能の両方について良好な結果を得られた．さらに，XMP の隣接通信を，京の拡張 RDMA インタフェースを用いた通信で置き換えたところ，スケーラビリティおよび絶対性能の改善が見られた．

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) 言語と呼ばれる一連の並列言語 [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，Omni XcalableMP と呼ばれる XcalableMP の処理系を開発中である．

一方，我々は，理化学研究所 計算科学研究機構の「チーム SCALE」において，次世代の大規模高解像度気象モデル SCALE-LES の研究開発を遂行中である．

京速コンピュータ「京」（以下，京と呼ぶ）は，文部科学省が推進する「革新的ハイパフォーマンス・コンピューティング・インフラ（HPCI）の構築」計画のもと，2012 年 6 月の完成を目指して，理研と富士通が共同開発中のスーパーコンピュータである [8]．

本研究は，以下の 3 点を目的とする．

- SCALE-LES の力学コアプロトタイプのコードを XcalableMP を用いて並列化することにより，XcalableMP の生産性を評価する．
- 同コードを Omni XcalableMP により並列化した場合および MPI により並列化した場合の性能を京および標準的なクラスタにおいて評価・比較することにより，Omni XcalableMP の性能を評価する．
- XMP で並列化した同コードの隣接通信を，京の拡張 RDMA インタフェースを用いた通信で置き換え，Omni XcalableMP 処理系の通信ランタイムを，拡張 RDMA インタフェースを用いて実装することの可能性を評価する．

以下，2 章で XcalableMP 言語，SCALE-LES および京について説明する．3 章では Omni XcalableMP の概要を説明する．次に，4 章で XcalableMP による SCALE-LES の力学コアプロトタイプの並列化について述べた後，5 章で評価を行う．最後に，6 章で本研究を総括する．

2. 背景

2.1 XcalableMP

XcalableMP は，次世代並列プログラミング言語検討委員会および 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．XcalableMP プログラムにおいて，データ分散，ループ並列化，通信などは，ベース言語である C または Fortran の指示文の形式でユーザによって明示される．

その主な特徴を以下に挙げる．

[1] 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
[2] 筑波大学 計算科学研究センター
Center for Computational Science, University of Tsukuba

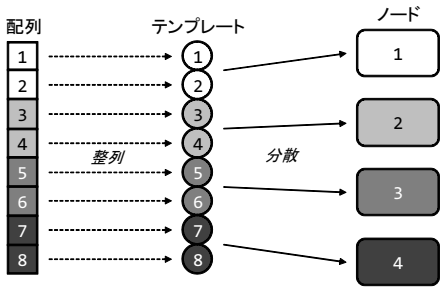

図 1 グローバルビュー並列化におけるデータ分散（一次元の場合）

- **指示文ベース**
  データ分散，通信，ループ並列化などは，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．
- **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 または OpenMP との併用が可能である．

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**）．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．

一方，XcalableMP のローカルビュー並列化においては，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．このとき，Fortran 2008 に基づく coarray 機能を利用し，各ノードの振る舞いを個別に記述することもできる [9]．

2.2 大規模高解像度気象モデル SCALE-LES

理化学研究所 計算科学研究機構に属する計算科学および計算機科学の研究者から成る横断的研究チームである「チーム SCALE (Scalable Computing by Advanced Library and Environment)」は，エクサスケールコンピューティングの時代を見据え，次世代の大規模高解像度気象モデル SCALE-LES の構築を進めている．

SCALE-LES は，LES (Large Eddy Simulation) スケールと呼ばれる，数 m-数百 m の空間解像度での気象現象をターゲットとしたモデルである．力学コア部分は圧縮性流体の支配方程式（連続の式，運動量保存式）を 3 次元カーテシアン座標のもとに完全陽解法で解くスキームを採用しており，空間方向については水平・鉛直共にスタガード格子を用い，4 次の中央差分を用いて有限体積法で解く．時間方向には 3 次の陽的ルンゲ・クッタ法を用いて積分を行う．主要変数は密度，3 次元方向の運動量，温位であり，温位と密度および診断される気圧は気体の状態方程式に従う．スキームの安定性のため，空間スキームには 4 次の人工粘性項を付加している．

チーム SCALE では，気象・気候科学的な要求を満たすシミュレーションモデルの構築と同時に，計算スキームの性能評価，システムソフトウェアの開発およびプログラミング手法・環境の確立を目的として，計算科学分野と計算機科学分野の研究者が共同してモデルデザインと性能評価を行っている．

中でも，我々は，XcalableMP を用いて SCALE-LES を並列化する作業を通じて，XcalableMP の言語仕様および処理系の向上・改善を図り，最終的に高生産性を実現するプログラミング環境を確立することを目的としている．

本研究では，その予備的段階として，SCALE-LES の力学コアプロトタイプ（以下，SCALEp と呼ぶ）をターゲットとして，XcalableMP およびその処理系の評価を行う．

2.3 京速コンピュータ「京」

2.3.1 システム概要

京は，82,944 個の計算ノード[*1]を，「Tofu（Torus fusion）」と呼ばれる 6 次元メッシュ／トーラス構造のネットワークで相互に接続した並列計算機システムである．システム全体のピーク性能は 10.6PFLOPS に達する．

各計算ノードは，1 個の CPU（SPARC64^{TM}VIIIfx），1 個のネットワーク用 LSI （ICC: InterConnect Controller）および 16GB のメモリを持つ．SPARC64^{TM}VIIIfx の緒元を**表 1** に示す．

[*1] さらに，5,184 個の IO ノードが存在する．

表 1　SPARC64^{TM}VIIIfx の諸元

演算性能（ピーク）	128 GFLOPS (16GFLOPS x 8 cores)
コア数	8
クロック周波数	2.0 GHz
浮動小数点演算器	乗加算ユニット x 4 (2 SIMD) 除算器 x 2
レジスタ数	浮動小数点レジスタ (64bit)：256 汎用レジスタ (64bit)：188
キャッシュ	L1I$：32 KB (2way) L1D$：32 KB (2way) L2$：Shared 6 MB (12way)
メモリ帯域	64 GB/s (0.5B/F)

通常，Tofu インターコネクトは，論理的な 3 次元トーラス構造として利用される．その場合の帯域は 3 次元の正負各方向にそれぞれ 5GB/s（双方向）である．

2.3.2　拡張 RDMA インタフェース

京で利用できる MPI は，標準仕様からの拡張として，「拡張 RDMA インタフェース」と呼ばれる機能を持つ [10]．本機能を使用することにより，ICC が備える 4 つの DMA エンジン Tofu Network Interface （TNI）を利用したり，迂回経路を利用することが可能になり，Tofu の特性を最大限に活かした通信を行うことができる．加えて，通信と計算のオーバラップを促進する効果もある．これらの結果，MPI の標準仕様のみを用いる場合に比べ，通信の性能が大きく向上することが期待できる．

3.　Omni XcalableMP

我々は，筑波大学と共同で，XcalableMP 処理系 Omni XcalableMP を開発中である．Omni XcalableMP の構成を**図 2** に示す．

XcalableMP ソースは，フロントエンドにおいてパーズされ，XcodeML/C 形式 [11] または XcodeML/Fortran 形式 [12] 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 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，ネイティブ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．現在の実装では，通信ライブラリとして MPI が用いられている．

本研究において，Omni XcalableMP を京に移植した．Omni XcalableMP は，京のログインノードにおいてクロスコンパイラとして動作しており，京の計算ノード用の実行形式を生成する．ネイティブコンパイラとして，`fccpx` または `frtpx` が用いられる．

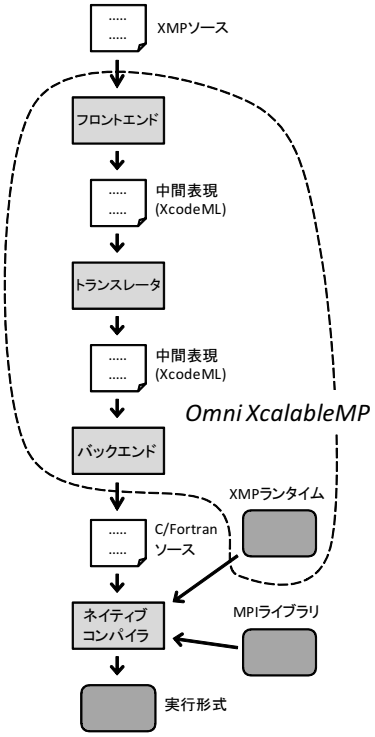

図 2　Omni XcalableMP の構成

4.　SCALEp の並列化

4.1　XcalableMP 通常版

XcalableMP を用いた SCALEp の並列化は，グローバルビュー並列化に基づく．すなわち，水平方向の格子点に相当する 2 次元テンプレートの各次元をブロック分散し，各物理量を保持する配列をこのテンプレートに整列させる．併せて，各配列の次元に，必要に応じて幅 2 のシャドウを付加する．すなわち，逐次版の SCALEp コードに以下の XcalableMP 指示文を挿入することにより並列化を指定する．今回，これらの指示文は，ほぼ機械的に指定可能であった．

- `nodes`
- `template` および `distribute`
- `align` および `shadow`
- `loop`
- `reflect`

XcalableMP では，`shadow` および `reflect` 指示文を用いて，ステンシル計算における「隣接通信」を記述できる．すなわち，`shadow` 指示文の指定により割り付けられたステンシル領域（XcalableMP では「シャドウ領域」と呼ぶ）には，`reflect` 指示文の実行により対応する配列要素の値がコピーされる．

ここで，`shadow` 指示文の指定により，配列の宣言範囲の上下限の外側にもシャドウ領域が暗黙的に割り付けられることに注意されたい．`reflect` 指示文が/periodic/修

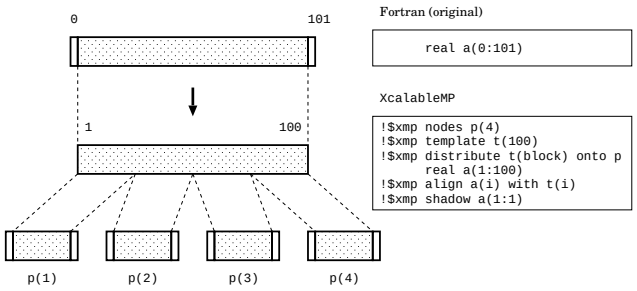

図 3 シャドウ領域の割付け（一次元の場合）

飾子を伴う場合，当該次元は「周期的」であると見なされ，上下限の外側のシャドウ領域も同時に更新される．

したがって，逐次版 SCALEp において，周期的境界条件を実現するために各配列に明示的に付加されていたステンシル領域を除くように，各配列の宣言を修正する必要がある．ただし，この修正により，XcalableMP 指示文を無視して通常の Fortran プログラムとして解釈することはできなくなる（**図 3**）．

XcalableMP で並列化した SCALEp コード（抜粋）を**図 4** に示す．コード中に現れる各 XcalableMP 指示文の意味は次の通りである．

- 1 行目: `N1xN2` の 2 次元のノード配列 `p` を宣言する．
- 2 行目: `IAxJA` の 2 次元のテンプレート `t` を宣言する．
- 3 行目: テンプレート `t` の各次元をブロック形式で分散する．
- 7, 8 行目: 配列 `dens` 他の 2 次元目と 3 次元目を，テンプレート `t` の 1 次元目と 2 次元目にそれぞれ整列する．
- 9 行目: 配列 `dens` 他の 2 次元目と 3 次元目にそれぞれ幅 2 のシャドウ領域を付加する．
- 11 行目: 配列 `dens` 他のシャドウ領域を（周期的に）更新する．
- 13 行目: 直後のループ `jy` および `ix` を，テンプレート `t` の分散に基づいて並列化する．

4.2 XcalableMP+RDMA 版

3 章で述べたように，現在の Omni XMP の実装では，通信ライブラリとして MPI が用いられる．一方，京の MPI が提供する拡張 RDMA インタフェースを用いれば，より効率の良い通信を実現できる可能性がある．

そこで，拡張 RDMA インタフェースを用いて `reflect` 指示文に対応する XcalableMP ランタイムを実装した場合を想定し，前節で開発した SCALEp コード中の `reflect` 指示文を，拡張 RDMA インタフェースを用いた通信で置き換える．それ以外の箇所は，通常版と同じである．したがって，各配列のシャドウ領域は，通常版と同様に，XcalableMP の `shadow` 指示文の指定により割り付けられる．

RDMA 版の SCALEp コードを**図 5** に示す．コード中の

```
1  !$xmp nodes p(N1,N2)
2  !$xmp template t(IA,JA)
3  !$xmp distribute t(block,block) onto p
4  ...
5    real(8) :: dens(0:KA,IA,JA)
6    ...
7  !$xmp align (*,i,j) &
8  !$xmp     with t(i,j) :: dens, ...
9  !$xmp shadow (0,2,2) :: dens, ...
10 ...
11 !$xmp reflect (dens(0,/periodic/2,&
12 !$xmp                  /periodic/2), ...)
13 ...
14 !$xmp loop (ix,jy) on t(ix,jy)
15         do jy = JS, JE
16          do ix = IS, IE
17           ...
18           do kz = KS+2, KE-2
19            ... dens(kz,ix+1,jy) + ...
20            ...
21           end do
22           ...
23          end do
24         end do
```

図 4 SCALEp コード（通常版）

各手続きの機能は次の通りである．

- `rdma_setup`
  拡張 RDMA インタフェースを初期化（`FJMPI_Rdma_init`）する．
- `set_rdma_variable`
  対象データのメモリ領域を，拡張 RDMA インタフェースに登録する（`FJMPI_Rdma_reg_mem`）とともに，リモート DMA アドレスを獲得する（`FJMPI_Rdma_get_remote_addr`）．
- `rdma_put`
  対象データをリモートライトする（`FJMPI_Rdma_put`）．

今回用いた実装では，`rdma_put` 内でリモートライトの完了確認を行っているため，通信と計算のオーバラップは起きていない．

5. 評価

5.1 生産性

逐次版の SCALEp コード 1088 行（空行およびコメント行を除く）[*2]に対し，挿入した XMP 指示文は 77 行（7%）であった．一方，MPI 版の SCALEp コードは 1099 行である．一般に，SCALEp のような規則的なコードでは，逐次版，XcalableMP 版，MPI 版のコードのサイズは大きくは変わらない．しかし，XcalableMP 版では，少数の（抽象

[*2] 行数のカウントは CLOC[13] による．

```
1  ! 初期化
2  call rdma_setup(nvars,                &
3                  isize, jsize, ksize, &
4                  lshadow, ushadow,    &
5                  ilb, iub,            &
6                  jlb, jub,            &
7                  rank_W, rank_N,      &
8                  rank_E, rank_S)
9
10 call set_rdma_variable(dens, 0)
11 ...
12 ! 通信
13 !!$xmp reflect (dens, ...)
14 call rdma_put(0,6)
```

図 5 SCALEp コード（RDMA 版）

表 2 XcalableMP 指示文の内訳

nodes	1
template	1
distribute	1
align	11
shadow	4
reflect	9
loop	50

度の高い）指示文をほぼ機械的に挿入するだけで並列化を実現できていることから，MPI 版に比べ生産性は高いと言える．

挿入した XMP 指示文の内訳を**表 2** に示す．XcalableMP 仕様によると，array 指示文により Fortran の配列代入文の並列化を指定することができる．しかし，現在の Omni XcalableMP は array 指示文をサポートしていないため，コード中の配列代入文をループに展開した上で，loop 指示文を指定した．

5.2 京における評価

評価に用いたソフトウェア環境は，次の通りである．

- Omni XcalableMP Compiler 1（リビジョン 783）
- 言語環境 K-1.2.0-05

また，指定したネイティブコンパイラ・オプション（性能に関係するもののみ）とその意味は次の通りである．

- -Kfast： ターゲットマシン上で高速に実行させることを指示する．
- -Kparallel：自動スレッド並列化を行うことを指示する．

計算ノード内のスレッド並列処理では 8 スレッドを用いる．すなわち，XcalableMP の 1 ノードが 1 台の計算ノードに割り当てられ，自動スレッド並列化によって計算ノード内の 8 コアでスレッド並列処理が行われる．

水平方向の格子点数は 512x512，鉛直方向の格子点数は 128 であり，時間発展ループ 500 回転に要する時間を評価した．

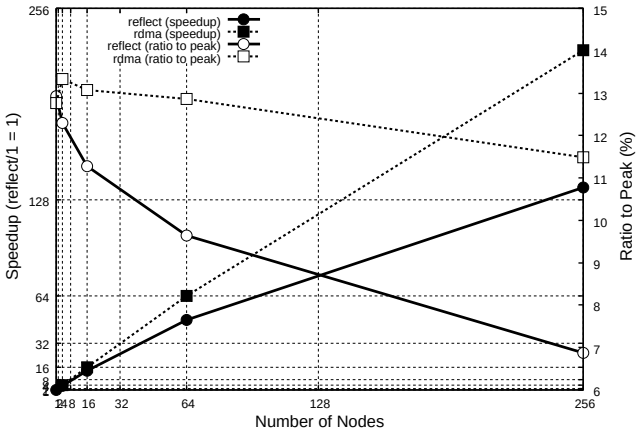

図 6 京における評価結果

表 3 評価環境（クラスタ）

製品名	HP Proliant SL390s G7 2U
CPU	Xeon X5670 (2.93 Ghz 6 core) * 2
memory	24 GB
network	1 Gbps Ethernet * 2 + Infiniband 4x QDR
OS	CentOS 5.5

京における通常版および RDMA 版の評価結果を**図 6** に示す[*3]．グラフの左縦軸は通常版の 1 ノードの性能を 1 とする相対性能，右縦軸はピーク性能比[*4]である．

通常版では，16x16 ノード実行時に，136 倍（並列化効率 53%）程度の性能しか得られていない．なお，本来であれば，この結果の妥当性を検証するには，逐次版および MPI 版との比較が必要であるが，時間の制約によりそれらの性能を測定することができなかった．逐次版および MPI 版との比較については，次節においてとりあげる．

一方，RDMA 版の性能は通常版を大きく上回っている．この点から，通常版における性能低下の原因が隣接通信であることが推測できるとともに，拡張 RDMA インタフェースを用いることによりスケーラビリティおよび実行効率を改善できることが示される．

5.3 クラスタにおける評価

評価に用いたハードウェア環境を**表 3** に挙げる．また，ソフトウェア環境は次の通りである．

- Omni XcalableMP Compiler 1（リビジョン 783）
- インテル Fortran コンパイラー 12.1.2
- インテル MPI ライブラリ 4.0

指定したネイティブコンパイラ・オプションは-O3 である．

水平方向の格子点数は 256x256，鉛直方向の格子点数は 64 であり，時間発展ループ 500 回転に要する時間を評価した．

*3 本結果は，整備中のシステムにおける暫定的なものである．
*4 ピーク性能比の値は，基本プロファイラによる．

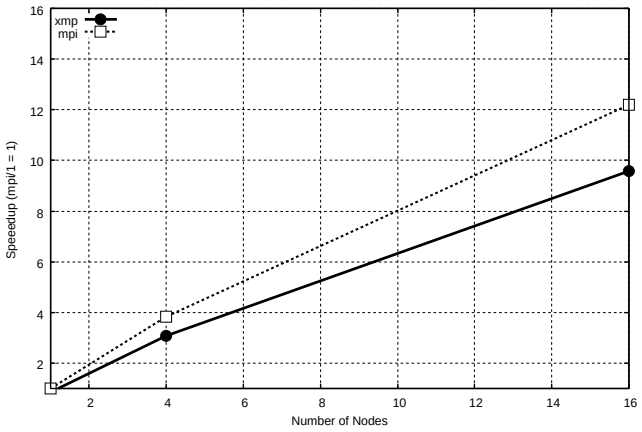

図 7 クラスタにおける評価結果

クラスタにおける XcalableMP 版（前節における通常版に相当）および MPI 版の評価結果を**図 7** に示す．グラフの縦軸は MPI 版の 1 ノードの性能を 1 とする相対性能である．

グラフより，XcalableMP 版の性能は MPI 版の性能を下回っていることがわかる．これは，次の 2 つの理由による．

- 隣接通信の効率が悪い．
  MPI を用いて直接に記述した隣接通信に比べ，各 `reflect` 指示文に対応して Omni XcalableMP のランタイムが行う隣接通信の性能が低い．ランタイムは任意の条件（配列サイズ，分散形式，シャドウの幅など）に対応する必要があるため，人手で直接に記述した通信の性能を上回ることは一般にないとはいえ，チューニングの余地はまだ残ると考えられる．
- ベクトル化されないループがある．
  ループを並列化するために Omni XcalableMP が挿入するランタイム呼び出しが，ネイティブコンパイラ（インテル Fortran コンパイラー）の最適化 (ベクトル化) を阻害している．しかし，問題のランタイムは対象のループの外側に挿入されているため，本来であれば，当該ループの最適化には影響しないはずである．問題が生じる厳密な条件を含め，詳細については調査中である．

6. おわりに

本研究では，並列プログラミング言語 XcalableMP およびその処理系 Omni XcalableMP の生産性と性能を評価した．

XcalableMP のグローバルビュー並列化の機能により，少数の指示文をほぼ機械的に挿入するだけで気象シミュレーションコード SCALEp を並列化できた．このコードを Omni XcalableMP を用いてコンパイルし，性能を京および Linux クラスタで評価した結果，一定の性能を達成できているものの，特に隣接通信の性能については課題が残ることがわかった．

一方，XMP で並列化した SCALEp の隣接通信を，京の拡張 RDMA インタフェースを用いた通信で置き換えたところ，スケーラビリティおよび絶対性能の点で良好な結果が得られた．従来の MPI 関数ではなく，拡張 RDMA インタフェースを用いて Omni XcalableMP の通信ランタイムを実装することにより，さらなる性能向上を図ることも今後の課題として挙げられる．

なお，時間の制約により今回示すことができなかった京における MPI 版の評価結果と，調査中とした問題の原因については，研究会の場において報告できる見込みである．

謝辞 本研究の成果（の一部）は，理化学研究所が実施している京速コンピュータ「京」の試験利用によるものです．また，SCALEp 本体および拡張 RDMA インタフェースを用いた通信部の開発にご協力頂いたチーム SCALE の各メンバーに感謝します．

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 1.0, `http://www.xcalablemp.org/xmp-spec-1.0.pdf` (2011).
[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] 高瀬 亮，横川三津夫（編）：情報処理，Vol. 53, No. 8, 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).
[9] 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).
[10] 富士通株式会社：Parallelnavi for MP10 V1.0 MPI 使用手引書 (2012).
[11] 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).
[12] 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).
[13] Northrop Grumman Corporation: CLOC, `http://cloc.sourceforge.net/`.