Vol.2015-HPC-150 No.15
2015/8/5

マルチグリッド法プログラムの京での評価と並列言語 XcalableMP による実装

下坂 健則[1]　村井 均[2]　佐藤 三久[2]

概要：マルチグリッド法は，大規模疎行列連立一次方程式を効率的に解く解法として知られている．近年，代表的な高性能並列計算機向けベンチマークとして，HPL とは異なる性能評価の尺度を提供することを目的としたプログラムが提案されている．その一つに，幾何的マルチグリッド法に基づいて開発された HPGMG がある．本稿では，スーパーコンピュータ京上で，HPGMG の構成プログラムの一つである HPGMG-FV の性能評価を実施し，82944 ノードの実行で，2.83TDOF/s の性能を得た．本結果は，LBNL が公表している世界の主要スパコンにおける性能リストで 1 位にランキングされている．また，京が提供しているプロファイル情報を分析することにより，HPGMG-FV で用意している数種類のスムーザーの違いによる全体性能傾向を把握することができた．また，マルチグリッド法に対する開発生産性向上の試みとして，NPB MG を題材に並列言語 XcalableMP による実装をした結果，MPI 版と比較し 16 ノードで 1.38 倍の実行時間に収まることを確認した．またメモリ管理部分以外は，すべて XMP のグローバルビューで書け，MPI 版と比較し，記述が容易となることを確認した．

1. はじめに

マルチグリッド法は，大規模疎行列連立一次方程式を効率的に解く解法として知られている．

高性能並列計算機向けベンチマークとしては，High Performance Linpack (HPL) が広く知られているが，近年，それとは異なる性能評価の尺度を提供するものとして，HPCG [1] や HPGMG [2] に代表されるマルチグリッド法を用いたベンチマークが提案されている．

HPL は，並列化効率が高く，システムの浮動小数点演算性能の理論最大値に近い値を得られるという特徴がある．しかし，演算量が係数行列のサイズの 3/2 乗に比例することから，問題規模によっては実行時間が数時間から 1 日を超える場合もあり，運用側にとっても，その間システムを占有する必要があること、多大な電力を必要とすることから大きな負担となっている．

HPGMG は，ローレンスバークレー国立研究所 (LBNL) で開発中のマルチグリッド法に基づいて開発されたベンチマークプログラムである．短時間で解け、並列化効率がよいという特徴を持つことから，上記 HPL の課題に対する解として提案されている．2014 年 6 月に v0.1 が公開され，同年 11 月に主要スパコンの測定結果に対する性能リストが公開された．本測定結果は，LBNL の Web サイトで随時更新されている．京では，HPGMG の構成プログラムの一つである HPGMG-FV において，2015 年 3 月現在，2.83TDOF/s を計測し，性能リストの 1 位にランキングされている．本稿では，本結果の詳細について報告する．

また，マルチグリッド法プログラムに対する生産性向上を目的に, PGAS 言語 XcalableMP(XMP)[3] による記述を試みた．題材には，最初に HPGMG よりもプログラム構造が容易な NAS Parallel Benchmark MG(MG) を用い, マルチグリッド法プログラムに対する基本的な実装課題を洗い出すことを目的とした．MG の実装は，同じ並列言語に属する HPF [4] や UPC [5] でも行われている [7][6].

以下では，第 2 章で HPGMG の概要と評価，第 3 章で NAS Parallel ベンチマーク MG の XMP による実装と評価，第 4 章でまとめと今後の課題を述べる．

2. HPGMG の概要と評価

HPGMG は，定数係数及び変数係数の楕円型方程式を幾何的マルチグリッド法を利用して解くベンチマークプログラムである．HPGMG は，離散化に有限要素法を使う HPGMG-FE と有限体積法を使う HPGMG-FV の 2 つのプログラムで構成される．本稿では，HPGMG-FV のみ扱う．

[1] 株式会社 日立製作所
[2] 国立研究開発法人 理化学研究所 計算科学研究機構

表 1　HPGMG-FV で選択できる主な手法

マルチグリッド法	V-cycle
	F-cycle
	U-cycle
スムーザー	Jacobi 法
	chebychev 多項式
	Gauss-Seidel Red-Black(GSRB)
bottom solver	CG
	BiCGStab
	CABiCSstab

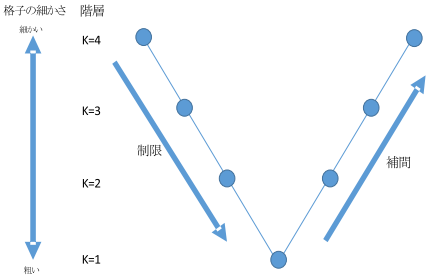

図 1　V-cycle

2.1　HPGMG-FV の概要

HPGMG-FV は C で実装され，MPI+OpenMP で並列化されている．実行時には，引数に 1 ノードあたりの問題サイズを指定し，実行ノード数に比例した問題規模で実行される．ベンチマーク性能は，Degree of Freedom per second (DOF/s) で示される．また，プログラムは以下に示す順序で実行される．

- 初期処理 (V-cycle の深さの算出, V-cycle の各階層のコミュニケータ作成やパラメータ初期化など).
- マルチグリッド法ソルバーの予備実行.
- マルチグリッド法ソルバーの本番実行.

HPGMG-FV では，マルチグリッド法の種類，および，そこで使用するスムーザーや、最も粗い格子で使う共役勾配法系ソルバー (bottom solver) を選択することができる．表 1 では，HPGMG-FV のコンパイル時に選択できる主な項目を示す．

マルチグリッド法は，多重の格子を用いて，粗い格子に対応した方程式を再帰的に解き近似解を改良していく手法の総称である．マルチグリッド法には様々な手法があり，図 1 に代表的な再帰的実行例として V-cycle の例を示す．

図 2 では，HPGMG-FV がデフォルトで設定している F-cycle の例を示す．

F-cycle は，最も粗い格子から処理が始まり，階層を一つ上る度に，一つ下の階層で求めた解の補間解を初期値として使い，該当する階層で V-cycle や W-cycle を実施する．HPGMG-FV で使用する F-cycle は，階層をひとつ上る度に V-cycle を実施する方式を採用している．

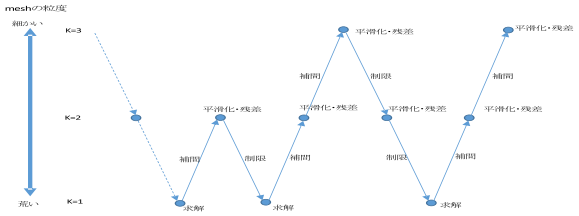

図 2　F-cycle の例

表 2　京コンピュータでの測定条件

ノード性能	128GFLOPS
メモリ帯域	64GB/Sec
最大使用可能ノード数	82944
コンパイラ	mpifccpx -Xg -std=gnu99 -Kfast,openmp

2.2　実験環境

実験には、理化学研究所で運用されているスーパーコンピュータ京を使用した．表 2 に，京の主な諸元，および本稿での測定条件を示す．

HPGMG-FV は，OpenMP の入れ子並列処理が実行可能である．京は，デフォルトでは，OpenMP 向けに独自実装された高速ライブラリが実行されるが，この高速ライブラリは，OpenMP の入れ子並列処理では使用できない．OpenMP の入れ子並列処理を使いたい場合には，高速ライブラリの実行を OFF にしなければならない．実際に，本研究でも，OpenMP 入れ子並列処理の ON と OFF の違いを計測したが，HPGMG-FV では，高速ライブラリを使用した方が性能的に優位なことを確認している．

本稿では，さらに京のシステム標準のプロファイラを利用する．プロファイラには，アプリケーション全体の性能の概要を把握するための基本プロファイラと，アプリケーションの性能向上に重要なホットスポットを詳細に解析するための詳細プロファイラがある．本稿では，主に詳細プロファイラを使用して性能傾向を分析した．

2.3　HPGMG-FV の評価

本稿では，HPGMG-FV について，京を使い，実行時間が最も長いスムーザーの処理に焦点を絞り評価した．スムーザーは，主に chebychev 多項式と Gauss-Seidel Red-Black(GSRB) を比較対象とした．マルチグリッド法の手法と bottom solver はそれぞれ F-cycle と BiCGStab に固定した．また，実行時に単位ノードあたりの問題サイズを変更することができるため、その性能最適化も実施した．

HPGMG-FV では，スムーザーの実行時間の改善を目的としたいくつかのチューニングオプションを用意している．表 3 に主なオプションを示す．

-DGSRB_FP は，GSRB に対して、SIMD 化を容易にするよう実装されたアルゴリズムを選択するオプションである．-DGSRB_STRIDE2 も GSRB に対する最適化オプ

表 3　チューニングオプション

タイリング	-DBLOCKCOPY_TILE_I -DBLOCKCOPY_TILE_J -DBLOCKCOPY_TILE_K
Gauss-Seidel Red-Black 向け最適化オプション	-DGSRB_FP -DGSRB_STRIDE2

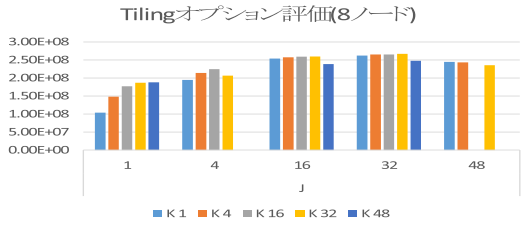

図 3　Tiling オプション性能評価

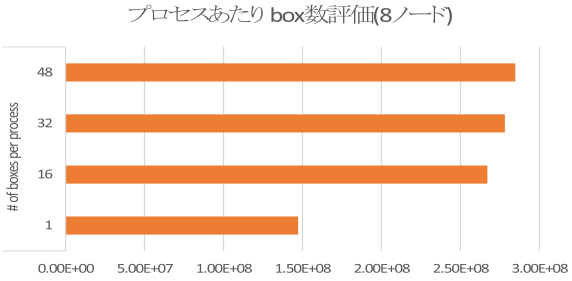

図 4　1 ノードあたりの問題規模に対する性能評価

ションであるが，ストライド幅 2 でメモリアクセスする実装を含んでいるため，-DGSRB_FP に比べ実行効率や SIMD 化率は低くなるものの，実行時間が速くなる効果が期待できる．

図 3 にキャッシュ向けのチューニングオプションである-DBLOCKCOPY_TILE_I,J,K を変化させたときの結果を示す．図 3 では，特に効果が期待できる-DBLOCKCOPY_TILE_J, K を変化させたときの結果を示す．京では，-DBLOCKCOPY_TILE_J による変化の度合いが大きく，-DBLOCKCOPY_TILE_J=32, -DBLOCKCOPY_TILE_K=32 のとき，最適なことがわかる．

図 4 に，HPGMG-FV で現れる box サイズを 128 の 3 乗に固定し，1 ノードあたりの box 数を変化させたときの性能を示す．図 4 からは，1 プロセスあたりの box 数が最も大きい 48 のときの性能が最もよい．したがって，1 プロセスあたりのデータ量が大きい方が，絶対性能は良い傾向であることがわかる．

表 4 に，チューニングオプション有無の GSRB と chebychev 多項式の場合のスムーザーの性能比較結果を示す．

表 4 から絶対性能は，チューニングオプションなしの GSRB，chebychev 多項式，-DGSRB_FP, -DGSRB_STRIDE2 の順に高くなっていくことがわかる．また，SIMD の効果を確認するため，括弧内に SIMD を OFF にしたときの性能を示した．SIMD 有無による性能差異は，数%とほとんど差異がないことがわかる．

表 4　スムーザー性能比較 (DOF/s)

ノード数	1	8	64
chebychev	3.99E+07 (3.88E+07)	3.06E+8 (2.96E+08)	2.40E+09 (2.33E+09)
GSRB	3.39E+07 (3.33E+07)	2.57E+08 (2.49E+07)	2.02E+09 (1.96E+09)
GSRB FP	4.30E+07 (4.11E+07)	3.28E+08 (3.15E+08)	2.57E+09 (2.47E+09)
GSRB STRIDE2	4.32E+07 (4.22E+07)	3.29E+08 (3.21E+08)	2.58E+09 (2.52E+09)

表 5　実行効率比較 (%)

ノード数	1	8	64
chebychev	10.31	9.86	9.67
GSRB	4.74	4.50	4.41
GSRB FP	10.06	9.59	9.39
GSRB STRIDE2	6.26	5.95	5.82

表 6　SIMD 化率比較 (%)

ノード数	1	8	64
chebychev	61.59	61.34	61.99
GSRB	7.43	7.43	7.43
GSRB FP	61.61	61.35	61.25
GSRB STRIDE2	27.22	27.12	28.08

表 7　メモリスループット比較 (GB/peak(%))

ノード数	1	8	64
chebychev	64.98 (63.13)	62.98 (61.10)	61.99 (60.20)
GSRB	50.28 (49.36)	48.54 (47.00)	47.76 (46.27)
GSRB FP	64.00 (61.18)	62.09 (59.59)	61.03 (58.66)
GSRB STRIDE2	64.25 (62.68)	62.19 (60.75)	61.06 (59.73)

2.3.1　プロファイル採取結果と考察

表 5 に表 4 の各項目に対する実行効率を示す．

実行効率は，キャッシュ効果の低い順に GSRB, -DGSRB_STRIDE2, -DGSRB_FP, chebychev 多項式となることがわかる．

次に，表 6 に表 4 の各項目に対する SIMD 化率を示す．

GSRB の最適化の影響により，-DGSRB_FP オプションは，chebychev 多項式と同程度の高い数値を示している．

次に，表 7 に各項目に対するメモリースループットを示す．

表 7 では，全項目で 40〜60%程度の高い数値を示している．括弧内は SIMD を OFF にしたとき数値を示している．

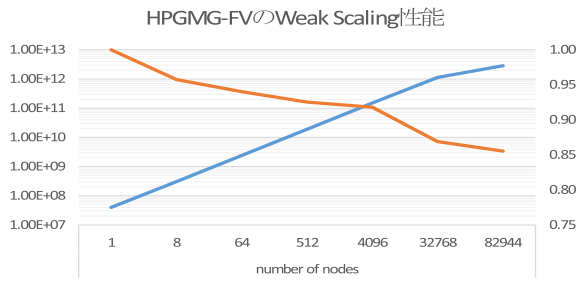

図 5 Weak scaling 性能 (DOF/s), スムーザー：chebychev 多項式

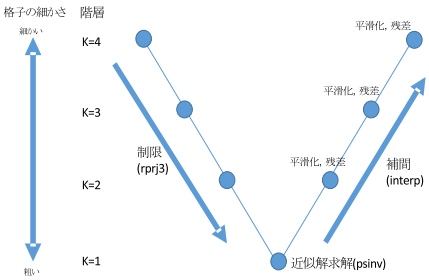

図 6 MG の V-cycle

SIMD の効果が低いのは，メモリバンド幅に対する依存度が高いためと考えられる．

最後に高並列環境での性能を確認する．図 5 では，スムーザーを chebychev 多項式にしたときの Weak scaling の結果を示す．図 5 では，京の最大ノード数である 82944 ノードまでの性能を測定した．スムーザーは，デフォルトである chebychev 多項式で計測した．このとき，並列化効率は，82944 ノードでも 86%と高い効率を保つことができていることがわかる．表 5 から-DGSRB_FP の方が高い性能が出ると推測できるが，計測を実施できていない．

3. NAS Parallel Benchmark MG の XMP による実装と評価

マルチグリッド法プログラムの生産性向上の可能性を確認するため，まず，HPGMG よりもプログラム構造が容易な NAS Parallel Benchmark MG で XMP による実装を試し，従来の実装との性能や生産性の比較を実施する．

3.1 NPB MG

NAS Parallel Benchmark は，NASA Ames Research Center で，並列計算機性能を評価するために開発されたプログラムセットであり，5 つのカーネルコード (CG, EP, FT, IS, MG) と 3 つの疑似アプリケーションコード (BT, LU, SP) からなる．NAS からは，逐次版，MPI 版，HPF 版が公開されている．MG は，3 次元ポアソン方程式を，簡略化されたマルチグリッド法で解くベンチマークである．図 6 に示す V-cycle にしたがって処理が進む．

図 7 は，逐次版の V-cycle の処理を示している．粗い格子の方へシフトしていく (k が減少方向) ときには制限 (rprj3) を行い，最も格子の粗い最下層で近似解を求め (psinv)，その後，細かい格子の方へシフトしていく (k が増加方向) ときには補間 (interp) を行い，さらに補間を実施する各階層で，残差計算 (resid) と平滑化 (psinv) を実施する．

```
1         do  k= lt, lb+1 , -1
2            j = k-1
3            call rprj3(r(ir(k)),m1(k),m2(k),m3(k),
4           >        r(ir(j)),m1(j),m2(j),m3(j),k)
5         enddo
6
7         k = lb
8         call zero3(u(ir(k)),m1(k),m2(k),m3(k))
9         call psinv(r(ir(k)),u(ir(k)),
10        >m1(k),m2(k),m3(k),c,k)
11
12        do  k = lb+1, lt-1
13            j = k-1
14           call zero3(u(ir(k)),m1(k),m2(k),m3(k))
15           call interp(u(ir(j)),m1(j),m2(j),m3(j),
16          >              u(ir(k)),m1(k),m2(k),m3(k),k)
17           call resid(u(ir(k)),r(ir(k)),r(ir(k)),
18          >             m1(k),m2(k),m3(k),a,k)
19           call psinv(r(ir(k)),u(ir(k)),m1(k),
20          >             m2(k),m3(k),c,k)
21        enddo
```

図 7 逐次版 V-cycle のコード

3.2 XcalableMP

Partitioned Global Address Space (PGAS) 言語 XcalableMP(XMP)[3] は，C，Fortran 言語で書かれた逐次プログラムを基に，最小限の指示文を用いて効率よく並列プログラムを開発するための言語である．仕様は PC クラスタコンソーシアムの XcalableMP 規格部会で検討されている．XMP のリファレンスコード Omni XcalableMP は，理化学研究所と筑波大学が共同で開発に取り組んでいる．XMP は，グローバルビューとローカルビューという 2 つのプログラミングモデルを備えていることを特徴とする．本稿では，XMP のグローバルビューだけを用いて，MG の逐次 Fortran プログラムを並列化することを目標とした．MG 以外の NAS Parallel ベンチマークの XMP による実装，評価は SP を除き既に行われている．

3.3 従来手法

MG の並列化は，MPI, HPF, UPC で実装が行われている．MPI 版では，V-cycle の階層間で関係があるデータは同一プロセスで実行し，階層間で通信しないように，あらかじめ 1 次元のデータとして格納している．これによりメ

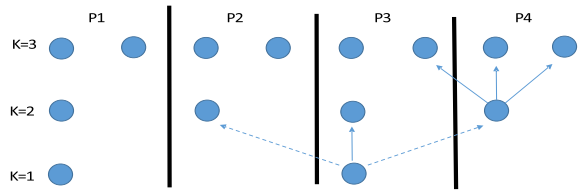

図 8 例外処理

モリを無駄なく使い，通信を効率よく行うことができる．

MPI 版 MG は， 図 7 で示すように，一次元でメモリ管理されているデータ r，u の各階層 k での起点を実引数 ir(k) として渡すことで，手続き側では，各階層で 3 次元のデータとして処理している．

HPF 版 [7] でも，階層間で通信を行わないようにするため，データを分散配置するときに，ストライドによる整列を行い，実質的に MPI 版と同様のデータ配置を行っている．しかし，XMP の v1.2 仕様では，ストライドによる整列を許していないため，この HPF での方法を踏襲することはできない．

MPI 版では，プロセス数 2 のときは 1x1x2，プロセス数 4 では 1x2x2，プロセス数 8 では 2x2x2, プロセス 16 では 2x2x4 の分散というように周期的な 3 次元分散方法で実行されている．

図 8 では，補間のときに，該当次元の要素数が分散数よりも少なくなる場合の処理 (以下，例外処理) を示している．k=1,2 の補間処理が該当する．P1 から P4 は各ノードを，○は k=1 から 3 までの各分散配列の要素を示している．このとき，制限，補間のときの通信は原則，隣り合うノード間での通信となるが，k=1 のときには隣のノードに配列がない状況が発生する．HPF でも隣接ノードに配列がない場合には，ダミーの配列を用意し，そこに該当する配列要素を置くことで通信を可能にしている．MPI 版では直接通信させることで実現している．

3.4 XMP による実装方針

本稿では, 従来手法に対する検討に基づき, NPB-3.3-SER をベースに，XMP による実装を，以下の方針で実施した．

- メモリ管理に，XMP を使用しない．
- mg3P サブルーチンで制御している V-cycle の各手続きのみ XMP 化する．
- 分散は, 簡単のため, 3 次元配列の 3 次元目のみで行う．
- 例外処理は，簡単のため，該当要素を全ノードにコピーした上で，逐次ルーチンで実行する．

データの分散方法は，MPI 版で採用されている分散方法を踏襲するものとする．ただし，3 次元目のみ分散させる方針としたため，MPI 版よりも簡略的なメモリ管理方法となる．

図 9 では，2 つの異なるグローバル配列 r, s に対して，

```
1         subroutine rprj3( r,m1k,m2k,m3k,s,m1j,m2j,m3j,k )
2         integer m1k, m2k, m3k, m1j, m2j, m3j,k
3         real(8) r(m1k,m2k,m3k), s(m1j,m2j,m3j)
4
5   !$xmp nodes p(*)
6   !$xmp template t1(m3k)
7   !$xmp template t2(m3j)
8   !$xmp distribute t1(block) onto p
9   !$xmp distribute t2(block) onto p
10  !$xmp align r(*,*,i3) with t1(i3)
11  !$xmp align s(*,*,i3) with t2(i3)
12  !$xmp shadow r(0,0,1)
13  !$xmp shadow s(0,0,1)
14  ....
15  !$xmp reflect (r)
16  ....
```

図 9 制限ルーチンの分散宣言

```
1         if(k .eq. npower) then
2              call gtol(r(ir(k)),rs(irs(k)),
3         >           m1(k),m2(k),m3(k))
4              call rprj3_s(rs(irs(k)),m1(k),m2(k),m3s(k),
5         >           rs(irs(j)),m1(j),m2(j),m3s(j),k)
6   ....
7         subroutine gtol(u,us,m1,m2,m3)
8         real(8) u(m1,m2,m3),us(m1,m2,m3+2)
9   ....
10  !$xmp align u(*,*,i3) with t(i3)
11  !$xmp shadow u(0,0,1)
12
13  !$xmp gmove
14        us(:,:,2:m3+1)=u(:,:,:)
15  ....
16
```

図 10 XMP 版 V-cycle の例外処理のコード例

3 次元目のみを同じノード配列 p を使ってブロック分散している．このことにより，MPI 版と同様に階層間の通信がなくなり，通信は周期的な隣接通信のみとなる．隣接通信は，shadow 指示文により，r，s 双方に幅 1 の袖領域を定義し，reflect 指示文で袖領域を更新する．

本稿では, V-cycle の分散処理を容易にするため, V-cycle の制限の過程で, z 軸の要素数がプロセス数より小さくなったときには，分散された該当配列要素を，1 ノードに集約し，全ノードで同じデータを持つこととした．. 図 10 に，本処理によるコード例を示す．2 行目の gtol ルーチンが，本処理に該当する．集約処理は、XMP の gmove 指示文で実施している．

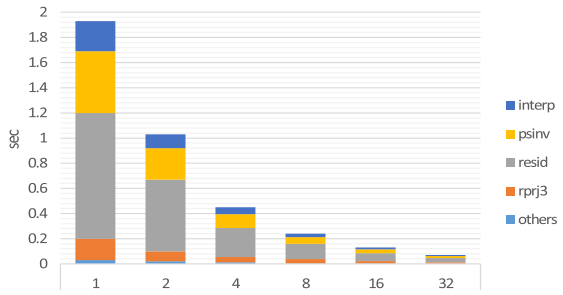

図 11 MPI 版性能

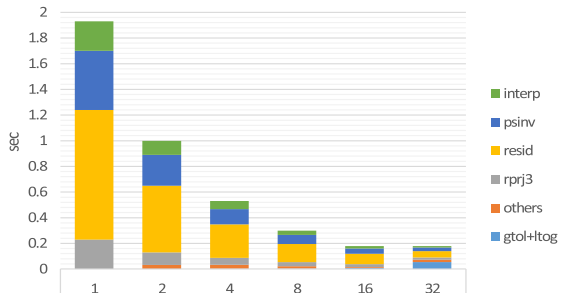

図 12 XMP 版性能

3.5 実験結果

図 11，図 12 はそれぞれ CLASS A における MPI 版，XMP 版でのストロングスケーリングの結果を示している．XMP 版では 3 次元目だけを分散させているのに対して，MPI 版ではプロセス数 2 のときは単位プロセスあたりの形状を 256x256x128，プロセス数 4 のときには 256x128x128，プロセス数 8 のときには 128x128x128 としているため，隣接通信のときの通信方法や表面積の違いによる通信量に差異が発生すると考えられる．

MPI 版は，16 プロセスのときに，1 プロセスのときの 14.8 倍の加速度であることに対し，XMP 版は 10.7 倍程度と 1.38 倍の開きがある．32 プロセスでは，MPI 版は 27.5 倍，XMP 版は加速せず 10.7 倍のままのため，双方の性能差は 2.57 倍に拡大している．最も大きな原因は，図 9 の gtol ルーチンの処理が 32 ノードに拡大したことで全体実行時間の中の割合が 5.5%から 30%に急増したためである．また，rprj3, resid, psinv, interp の各ルーチンについても，16 プロセスから 32 プロセスでは，MPI 版はどれもほぼ 2 倍に加速しているのに対して，XMP 版は 2〜4 割の加速に留まっている．

図 13 は，逐次コード，MPI 版，XMP 版のステップ数を比較している．

XMP 版の総ステップ数は，MPI 版と大差なくなっているが，350 ステップは逐次コードの流用であるため．実質的な開発コストは MPI 版ほど大きくはない．その上，通信はすべて XMP 指示文で書けていることから，逐次流用コード以外の部分についても MPI 版と比べて容易に記述できていると考えられる．

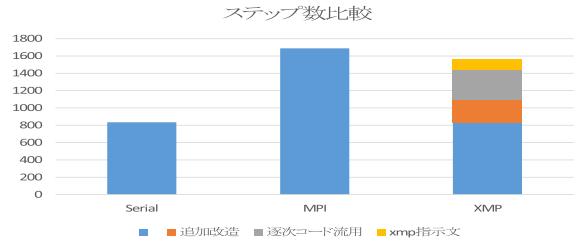

図 13 ステップ数比較

3.6 考察

XMP 版が MPI 版の性能に見劣りする点は，V-cycle の通信が発生する各処理の実行時間が MPI 版よりも遅いことから，データの分散方法の違いが影響していると考えられる．また，例外処理については，16 ノードで性能の伸びが頭打ちになる直接の原因になっていることから別の対策を打つ必要がある．また，今回はメモリ管理部分に XMP 指示文を使わなかったが，XMP のメモリレイアウトを考慮しながら書くことになるため，記述コストは大きい．この部分の記述の容易性は，今後検討する余地がある．

4. まとめと今後の課題

HPGMG-FV の京への移植および性能評価を実施した．その結果，LBNL で公表している世界の主要スパコンにおける性能ランキングで 1 位を得た．また，京が提供しているプロファイル情報を分析することにより，HPGMG-FV で用意している数種類のスムーザーの違いによる全体性能傾向を把握することができた．この傾向からは，HPGMG-FV で使う 2 階の楕円型方程式の場合，メモリスループットが高い傾向があるため，京の SIMD 有無はあまり効果的ではないことが判明した．

NPB MG を XMP で簡易的に実装した結果，MPI 版と比較して 16 ノードで 1.38 倍の実行時間に収めることができた．またメモリ管理部分以外は，すべて XMP のグローバルビューで書け，MPI 版と比較し，記述が容易となることを確認した．

今後の課題には，XMP 版 NPB MG の性能および記述容易性のブラッシュアップと，HPGMG-FV を XMP で記述することが挙げられる．

謝辞 本研究は，著者が理化学研究所出向中に得た結果を纏めたものである．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ある．

参考文献

[1] HPCG: http://www.hpcg-benchmark.org/index.html.
[2] HPGMG: https://hpgmg.org/.
[3] XcalableMP Specification Working Group:

Specification of XcalableMP, Version 1.2, http://www.xcalablemp.org/spec/xmp-spec-1.2.pdf.
[4] High Performance Fortran Forum : High Performance Fortran Language Specification Version 2.0, hpff.rice.edu/versions/hpf2/hpf-v20.pdf.
[5] UPC Consortium: UPC language specifications, v1.2, Tech Report LBNL-59208, Lawrence Berkeley National Lab (2005).
[6] UPC NAS Parallel Benchmarks: https://threads.hpcl.gwu.edu/sites/npb-upc.
[7] 村井均, 岡部寿男：NAS Parallel Benchmarks による HPF の評価, 情報処理学会論文誌, Vol. 47, No. 7 (2006).