Vol.2013-HPC-139 No.20
2013/5/30

メニーコアプロセッサXeon Phiの性能評価

大島　聡史[1,a)]　金子　勇[1]

概要：本稿では最新のメニーコアプロセッサ Xeon Phi(以下，Phi) の性能について述べる．Phi は高い演算性能およびメモリ転送性能を従来の CPU 同様のプログラミングによって活用できるハードウェアとして高い注目を集めている．その一方でアーキテクチャとしても製品としても新しいものであるため性能を十分に引き出すための知識や技術の共有が十分ではない．そこで本稿では幾つかのベンチマークプログラム等を用いて Phi の性能を評価するとともに，性能に影響を与える実装方法や実行方法について示し情報共有を行う．なお本稿では Phi として先行提供版の Preproduction Xeon Phi を用いている．

1. はじめに

高い演算性能を達成するために様々な並列計算ハードウェアの開発が行われている．特に今日大きな注目を集めているハードウェアの一つとしては，汎用性と高い並列演算性能を兼ね備えたメニーコアプロセッサがあげられる．MIC(Many Integrated Core) の名で注目されていた Intel 社のメニーコアプロセッサは，本年初頭より Xeon Phi（以下，本稿では Phi と表記する）として一般消費者への流通が開始された．最新のスーパーコンピュータのランキング TOP500(2012 年 11 月版) には先行提供された Phi を搭載した計算機もランクインしている [1]．1ExaFlops クラスのスーパーコンピュータを実現するための技術の一つとしても Phi への期待は大きい．著者らの所属する東京大学情報基盤センターにおいても先行提供版の Phi(Preproduction Xeon Phi) を搭載した PC クラスタが設置されており，システムソフトウエアや通信機構に関する研究等 [2], [3] への活用が始まっている．

Phi は既存の汎用 CPU と比べて単純な計算コアを多数搭載した並列計算ハードウェアである．Phi と同様に単純な計算コアを多数用いたハードウェアとしては GPU が広く普及しているが，既存の CPU 向けに作成されたプログラムを GPU 上で高速に動作させるにはアーキテクチャや最適化手法の習得およびプログラム作成に多大な労力が必要であるという意見も少なくない．一方で Phi は GPU と比べて既存の CPU に近い設計・実装となっており，既存の CPU と同様の開発環境やプログラミング手法が利用可能である．そのため，プログラム移植性などの観点から Phi に期待しているという意見もある．

しかしながら，確かに Phi 上で実行可能なプログラムを作成するのは容易である一方で，既存の CPU 向けのプログラムをそのまま Phi 向けにコンパイルしただけで高い性能が得られるとは限らない．Phi の持つ高い性能を発揮するには，Phi の特性に合わせて適切なコーディングやコンパイル，環境変数の設定等を行う必要がある．しかし，Phi はアーキテクチャとしても製品としても新しいものであるため，実際に利用した場合にどの程度の性能が得られるのか，具体的にどのようなプログラムを作成すれば良い性能が得られるのか，プログラムの違いがどの程度の性能差になるのかといった情報の共有が急務である．

そこで本研究では，幾つかのベンチマークプログラムを用いて Phi の性能を評価するとともに，プログラム（ソースコード）記述方法や実行時パラメタ（環境変数）設定が性能に与える影響についても調査を行い結果を報告する．

本稿の構成は以下の通りである．2 章では Phi のアーキテクチャやプログラミング手法の概要を述べる．3 章では幾つかのベンチマークプログラムを用いて性能評価を行った結果を示す．4 章はまとめの章とする．

2. メニーコアプロセッサ Xeon Phi

本章では Phi のアーキテクチャとプログラミング手法の概要について述べる．なお Phi に関する資料は Intel 社の web サイト [4] に多くの資料が掲載されているので参考にされたい．また Phi 向けのプログラム作成には原稿執筆時点で Intel 社製のコンパイラ群と GNU のコンパイラ群が利用可能であるが，ここでは Intel 社製コンパイラの利用を前提として述べる．

はじめに Phi の全体的なハードウェア構成について述べ

[1] 東京大学 情報基盤センター
Information Technology Center, The University of Tokyo
[a)] ohshima@cc.u-tokyo.ac.jp

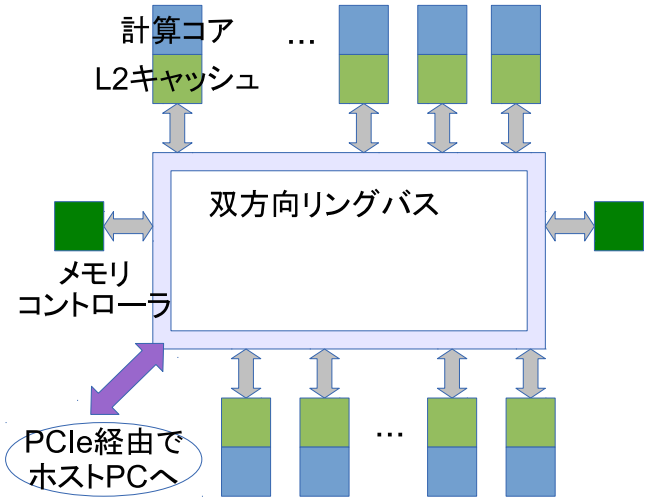

図 1　Xeon Phi の全体構成

る．図 1 に示すように，Phi は計算コアとメモリを双方向の高速なリングバスで繋いだ構成となっている．各計算コアは既存の Intel 社製 CPU をもとにしており，現在の CPU ほど高機能ではなく命令スケジューリングもインオーダー実行であるものの，SIMD 処理 (AVX3) に対応している．命令セットアーキテクチャは k1om であり既存の x86 系とは別なため既存の CPU 向けに作成されたバイナリをそのまま実行することは不可能であるが，既存の CPU と同様の開発環境 (コンパイラやライブラリ) が多く提供されている．また GPU とは異なり Phi 上で Linux OS を動作させることが想定されており，実際に後述の native 実行では Phi 上で動作する Linux OS に ssh ログインしたうえでプログラムを実行している．

現在の Phi は HPC 向けの GPU カードと同様に PCI-Express でホスト PC と接続するアクセラレータカードの形式で提供されている．Phi 1 基 (カード 1 枚) あたりの計算コア数は 60 程度とされており，製品版の 5110P では 60 コアが，本稿で利用している Preproduction Xeon Phi では 57 コアが利用可能である．また HyperThreading 機能に対応しており，1 コアあたり最大で 4 スレッド実行が可能である．メモリとしては GPU と同様に，レイテンシは大きいがスループットの高い GDDR5 メモリを搭載している．

Phi のプログラミングモデルと実行モデルについては，大きく分けて native モデルと offload モデルの 2 種類が利用可能である．

前者の native モデルについては Phi 上で起動し Phi 上で動作するプログラミングモデル・実行モデルを意味する．例えば Phi に対応した Intel コンパイラでは通常の C/C++ プログラムや Fortran プログラムに対して-mmic というオプションを付けることで native 実行向けのプログラムを作成することができる．プログラム実行方法は既存の CPU と同様であり，Phi に ssh ログインして./a.out の形式でプログラムを起動すれば良い．なお CPU 上で動く OS(コンソール，シェル) から明示的な ssh アクセスを行わずに直接 Phi 上のプログラムを実行するサポート機能等も存在するが，本稿では割愛する．

一方で後者の offload モデルはホスト上で起動したプログラムから特定の部分のみを Phi で実行させるプログラミングモデル・実行モデルを意味する．このモデルは GPU プログラムにおいて GPU カーネル関数のみを GPU 上で実行し，その他の部分は CPU 上で実行するのに近い．offload 実行向けのプログラムを作成するには，専用の指示文を用いてソースコードを記述し，対応するコンパイラに与えれば良い．プログラム実行時には，CPU 上でプログラムを実行すれば offload 対象部分のみが Phi 上で実行される．

Phi 向けに提供されている主な並列化プログラミング手法としては，既存の OpenMP と MPI，そして Intel 社の提供する TBB(Thread Building Block) があげられる．特に OpenMP と MPI はすでに HPC を含めた各分野で広く利用されてきたプログラミング手法である．Phi はユーザから見ると多数の計算コアを持つマルチコアプロセッサであるかのように見えるため，OpenMP や MPI を従来と同様の感覚で利用可能であり，既存のプログラムの移植が容易に行えることが期待できる．

このように，Phi 向けのプログラム作成においては既存の CPU 向けの知識や技術が大いに活用できると考えられる．特に native モデルについては，もちろんプログラムの内容や用いるライブラリ等の都合はあるが，CPU 向けのプログラムを Phi 向けにコンパイルすればそのまま Phi 上で実行できる可能性があり，容易に Phi 対応を行うことができると考えられる．しかしその一方で，既存の CPU 向けのプログラムをそのまま Phi 上で実行した場合，いくつかの要因によって良い性能が得られない可能性があることが懸念される．

たとえば既存の多くの CPU には Phi と比べてずっと少ない数の計算コアしか搭載されていなかったため，プログラムが低い並列度に最適化されている可能性がある．仮に並列度の低い繰り返し処理 (for ループ，do ループ) が OpenMP によって並列化されていた場合，そのまま Phi で実行すると一部の計算コアしか使われずに低い性能となることが考えられる．また既存の Intel 社製 CPU と Phi では対応する SIMD(AVX) 命令が異なり，最適なメモリアクセス単位にも違いがある．そのため，配列の宣言・確保時にはこれらの差異を考慮することで性能が変化する可能性がある．手動で SIMD 化を行った (組み込み命令を用いて記述してある) プログラムについてはプログラムの書き換えも必要である．その他，キャッシュヒット率向上のためにブロック化を施してあるコードやプリフェッチのための命令を入れて最適化を施したコードなど，既存の CPU 向けに高度な最適化を施してきたコードを Phi 上で用いる場

表 1　実験環境 (CPU は 2 ソケット搭載されているが各実験では 1 ソケットのみ使用している)

	Preproduction Xeon Phi	E5-2670 (SandyBridge)
コア数	57 (228 スレッド)	8 (16 スレッド)
動作周波数	1.10 GHz	2.60 GHz
メモリ	GDDR5 3GB	DDR3(-1666) 64GB
倍精度浮動小数点理論演算性能	1003.2 GFLOPS †	166.4 GFLOPS
メモリバンド幅	320 GB/s †	51.2 GB/s
コンパイラ，etc.	icc 13.1.1 MPPS 2-2.1.5889-16	icc 13.1.1

† 先行提供版のためこれらの値は明確にされていない．1003.2 GFLOPS は次式により算出した：16 DP FLOPS/クロック/コア × 57 コア × 1.10 GHz．320 GB/s は Xeon Phi 5110P の値であり，本システムでは異なる可能性がある．

合はパラメタの変更などの対応を行わねば良い性能が得られない可能性が高い．実行時のスレッドとコアの割り当て方 (アフィニティ設定) についても違いがあり，Phi では既存の compact と scatter 以外に balanced というアフィニティが利用可能となっている．

以上のように，Phi において最大の性能を得るためには Phi 向けの最適化が必要であるため最適化手法や性能へのについての情報共有は重要である．

なお，本稿における性能評価では OpenMP や MPI を用いて記述した native モデルのプログラムを用いている．

3. 性能評価

3.1 評価内容

本章ではベンチマークプログラムを用いて Phi の性能を評価する．実験環境におけるホスト CPU と Phi の仕様を**表 1** に示す．ホスト CPU については同一の CPU が 2 ソケット搭載されているが，各実験ではスレッド数やプロセス数などを調整し，1CPU ソケットと 1Phi カードの比較を行っている．本稿では先行提供版の Phi (Preproduction Xeon Phi) を用いているため，製品版の Phi とは性能や傾向にある程度の差が生じる可能性がある．

今回は以下の 3 つのプログラムについて性能を測定した．

- STREAM ベンチマーク
- HPL ベンチマーク
- GeoFEM ベンチマーク

これらのプログラムは筆者らが過去の研究報告 [6], [7] にてスーパーコンピュータの性能評価にも用いているため，それらの結果との比較も行った．

3.2 STREAM ベンチマーク

STREAM ベンチマーク (Feb.19 2009 版)[5] を用いてメモリ性能を測定した．

STREAM ベンチマークは配列に対して “特定の処理” を繰り返し実行した際の実行時間からメモリ性能 (MB/s) を算出する．“特定の処理” としては，

- 配列のコピーを行う Copy (c[j] = a[j])
- 配列とスカラーとの乗算を行う Scale (b[j] = scalar*c[j])
- 二つの配列を加算する Add (c[j] = a[j]+b[j])
- スカラーとの乗算と配列加算を組み合わせた Triad (a[j] = b[j]+scalar*c[j])

が用意されている．

STREAM ベンチマークは OpenMP を用いた並列実行に対応した単一プロセスプログラムであり，コンパイラオプション以外に修正することなく Phi 上で実行可能であることが確認できた．そこで，プログラムの書き換えを行わずにコンパイラオプションや環境変数のみを調整して性能を測定した．測定結果を**図 2** に示す．問題サイズ N は 80,000,000 を指定しており，繰り返し回数 TIMES は 10(初期値) である．図中には以下の組み合わせからなる結果が含まれている．

- コンパイラオプション (-O3 -mmic -openmp に加えて)：なし, -no-vec, -no-opt-prefetch, -no-vec -no-opt-prefetch
- スレッド数：57, 114, 171, 228
- アフィニティ：scatter, balanced

図中のグラフは 2 つのコンパイラオプションによって 4 つの部位 (A 部，B 部，C 部，D 部) に別れている．特に-no-vec の有無が性能に大きく影響を与えており，-no-vec を指定した「B 部と D 部」では，指定しなかった「A 部と C 部」と比べて Copy 以外の性能が大きく劣っていることが確認できる．-no-vec はコンパイラによるベクトル化 (AVX3 の活用) を抑制するオプションであり，この性能差から Phi の性能を引き出すうえでベクトル化が重要であることが確認できる．一方でコンパイラによるプリフェッチ命令の生成を抑制する-no-opt-prefetch の有無については性能に有意な差は見られなかった．スレッド数については，全体的にコア数の 2 倍である 114 スレッドの際に最も良い性能が得られている．スレッド数をさらに多くした場合に性能が低下している件については，STREAM ベンチマークのようにメモリアクセスに負荷のかかる処理ではコアあたり 3 以上のスレッドを生成してもメモリ処理が追いつかずに性能向上が得られなかったことを意味している．アフィニティ設定については，全体的に scatter よりも balanced の方がわずかながら良い性能が得られた．なお Copy のみ-no-vec の有無で性能に差が生じなかった理由については，コンパイラの-S オプションにて出力されたアセンブラコードを見るといずれの場合も Copy に該当する処理は_intel_fast_memcpy という外部関数の呼び出しに置き換わっており，STREAM ベンチマークのコンパイルオプションが影響しないためであることが判明している．

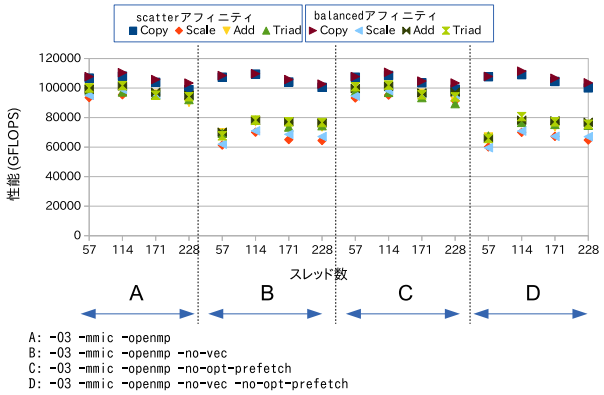

図 2 STREAM の性能

さらにプログラムの書き換えを行い，以下に示す最適化手法の効果を測定した．

メモリアラインメント 配列の宣言時に 64byte アラインメント調整を行う指示子を挿入 (__declspec(align(64))，AVX 組み込み関数を用いる場合は必須)

メモリの確保方法 アラインメントを意識してメモリの確保を行うことができる_mm_malloc や，ラージページを活用可能な MAP_HUGETLB オプションを有効にした mmap 関数を用いた動的なメモリ確保

AVX 組み込み関数を用いた AVX プログラムの記述

non-temporal オプション AVX を用いたロード・ストア命令に対する non-temporal オプションの設定

プリフェッチ命令 プリフェッチ命令の挿入および挿入の仕方 (レベルと幅)

メモリアラインメント，メモリの確保方法，non-temporal オプションについては選択肢が多くはないことから網羅的に実験を行った．AVX については STREAM ベンチマークの内容にあわせてシンプルな AVX コードを記述した．プリフェッチについてはプリフェッチのレベル (NTA,T0,T1,T2) とプリフェッチ幅の選択肢が多いため，幾つかのパターンを試してみた．以上のように様々な最適化を施して性能を測定してみたが，プログラムの内容が単純でコンパイラによる最適化が十分効果を発揮していたためか，明確な性能向上は観測できなかった．

ところで，筆者らは参考文献 [6], [7] においても同様に STREAM ベンチマークを用いた性能評価を行っている．そこで，Phi とホスト CPU の STREAM 性能を参考文献 [7] における結果とあわせて**表 2** に示す．ホスト CPU の STREAM 性能についてはコンパイラオプションや SIMD 実装をいくつか試した中で最大の性能が得られたものを示しており，いずれも 128bitSSE 組み込み命令を用いて記述したものが採用されている．なお AVX と SSE との性能差はわずか (1%程度) であった．

表 2 によると，Phi の STREAM 性能は非常に高いメモリバンド幅を持つ Power7 と比べると半分弱であるが，ホスト CPU や SPARC64IXfx と比べると 1.5 倍から 2 倍程度の高い性能を得られていることが確認できる．一方で Phi の理論性能比は他のプロセッサの結果と比べると低い値となっている．しかし前述のように Preproduction Xeon Phi の理論メモリバンド幅は不正確である可能性がある (実際はもっと低い値である可能性がある) ため，他のプロセッサ並みの値が得られている可能性があることを記しておく．

表 2 STREAM 測定結果 (単位は MB/sec，括弧内は理論性能比)

	Xeon Phi Preproduction	ホスト CPU E5-2670	Oakleaf-FX SPARC64IXfx	Yayoi Power7
Copy	111430.2433 (34.82%)	76022.5024 (45.7%)	59987.3012 (68.9%)	224825.3361 (42.9%)
Scale	99048.1914 (30.95%)	76131.3847 (45.7%)	59768.9227 (68.7%)	226349.5329 (43.2%)
Add	101421.4211 (31.69%)	75599.2948 (45.4%)	64640.5627 (74.3%)	256364.6680 (48.9%)
Triad	102030.5048 (31.88%)	75700.2066 (45.5%)	64712.2441 (74.3%)	255192.6583 (48.7%)

※ Oakleaf-FX と Yayoi は参考文献 [7] から引用 (ともに 1 ノードの性能)

3.3 HPL ベンチマーク

HPL ベンチマーク (HPL-2.1)[8] を用いて演算性能を測定した．

HPL ベンチマークは LU 分解による連立一次方程式の求解を行うものであり，特に行列-行列積計算 (BLAS3 DGEMM) の性能がベンチマークスコアに大きな影響を与えるベンチマークである．

HPL ベンチマークについても STREAM ベンチマークと同様に，ソースコードの修正を行うことなく HPL の用意している通常の手順によって native モデルで動作する実行ファイルを作成することができた．プログラム作成時の主なコンパイルオプションとして-O3 -openmp -mmic -mkl を指定した．-mkl を指定しているため，性能に大きな影響を与える DGEMM は MKL ライブラリによって行われる．様々な問題設定 (HPL.txt に設定する値および実行時パラメタ) においてベンチマークを実行した結果，以下の設定において 151.0 Gflops の性能を得た．

- N = 16,000
- NBs = 32
- P×Q = 1×14
- MPI プロセス数 14 × プロセスあたりスレッド数 16
- KMP_AFFINITY=compact,granularity=fine (MKL 推奨値)

得られた性能は約 1Tflops の理論性能と比べると非常に低い値である．この主な理由については，HPL 全体を Phi で実行したこと，Phi 上の単純な計算コアが DGEMM 以外の様々な処理についても実行したことにあると考えられる．

そこで，MKL ライブラリを用いて正方行列同士の

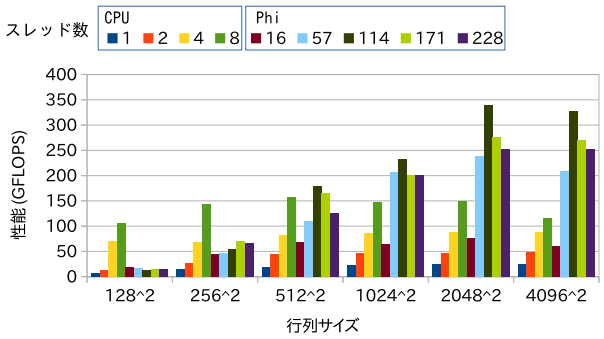

図 3　mkl を用いた DGEMM の性能

DGEMM を実行し性能を確認した．結果を**図 3** に示す．図中のグラフが示すように，Phi は行列サイズと使用スレッド数が十分大きいときに高い性能を発揮することができる．そのため HPL ベンチマークの全てを Phi 上で実行した場合は，DGEMM 以外の Phi にとって得意ではない処理も行うことになるうえに，DGEMM についても十分な性能の得られる問題設定ではないものまで解くことになるため，十分に性能を発揮できない状況となることがわかる．

3.4　GeoFEM ベンチマーク

GeoFEM プロジェクト [9] で開発された並列有限要素法アプリケーションを元に整備した性能評価のためのベンチマークプログラム GeoFEM-Cube[10] による評価を実施した．オリジナルの GeoFEM ベンチマーク [11] は，
(1) 三次元弾性静解析問題（Cube 型モデル，PGA モデル）
(2) 三次元接触問題
(3) 二重球殻間領域三次元ポアソン方程式
に関する並列前処理付き反復法ソルバーの実行時性能（GFLOPS 値）を様々な条件下で計測するものである．プログラムは全て OpenMP ディレクティヴを含む FORTRAN90 および MPI で記述されている．各ベンチマークプログラムでは，GeoFEM で採用されている局所分散データ構造 [9] を使用しており，マルチカラー法等に基づくリオーダリング手法によりベクトルプロセッサ，SMP，マルチコアプロセッサにおいて高い性能が発揮できるように最適化されている．また，MPI，OpenMP，Hybrid（OpenMP ＋ MPI）の全ての環境で稼動する．

中島らは参考文献 [11] において，3 種類の GeoFEM ベンチマークのうち 一様な物性を有する単純形状（Cube 型）を対象とした三次元弾性静解析問題について cc-NUMA アーキテクチャを有する HA8000 に対して様々な最適化を試みた．この成果を性能評価用のベンチマークプログラムとして整備したものが GeoFEM-Cube である．

GeoFEM-Cube の主要な計算部においては共役勾配法 (Conjugate Gradient，CG) 法が用いられている．また係数行列の格納法としては CRS(Compressed Row Storage) 法が用いられており，並列性を抽出するためにマルチカラー法などのカラーリング処理が実装されている．

GeoFEM ベンチマークについても Phi 向けにソースコードの修正は行わず，コンパイルオプションには-O3 -mmic を指定した．**表 3** に性能評価結果を示す．筆者らは参考文献 [6], [7] においても同様に GeoFEM ベンチマークを用いた性能評価を行っているため，参考文献 [7] に掲載した結果の一部も同時に示している．並列プログラミングモデルとしてはいずれも Flat MPI を使用している．また Phi はコア数 (スレッド数) に対して搭載メモリ量が非常に小さいため，他の環境と比べて小さな問題サイズで実験を行っている．

実験の結果によると，Phi はホスト CPU とほぼ同じ 16GFLOPS 程度の性能を得た．この性能は Oakleaf-FX の 1 ノードとも同程度である．しかしピーク性能比で比べると他の環境に比べて非常に低い値となった．Phi のピーク性能比が低い理由としては，HPL ベンチマークと同様に Phi が不得意とするような計算も行っていることがあげられるが，性能値の差を考慮するとコアあたりのキャッシュサイズの小ささや Byte/Flop の低さの影響も考えられる．

4.　おわりに

本稿では Preproduction Xeon Phi を用いてベンチマークプログラムを実行し，性能評価を行った．

STREAM ベンチマークについては幾つかのコンパイラオプションや実行時環境変数を調整するのみで 100MB/s 程度以上の高い性能を得ることができた．一方でより高い性能を得るために試みた幾つかの最適化手法は効果的ではなかった．ただし，STREAM ベンチマークは処理の内容が簡単であることに加えてコンパイラやシステムソフトウェアの性能が良いために最適化の努力をせずとも十分な性能が得られている，という可能性もあると考えられる．

HPL ベンチマークについては 150GFLOPS 程度という理論演算性能に比べて低いスコアとなってしまったが，これはベンチマークの全てを Phi 上で実行したことが影響している可能性が高い．Phi によって高い性能を得やすい計算，すなわち大きなサイズの DGEMM のみを Phi で実行し，その他の部分はホスト CPU で計算するのが妥当である．

GeoFEM ベンチマークについてはホスト CPU と同程度の性能を得たが，ピーク性能に対する性能比は低い値となった．これについても HPL ベンチマーク同様に全て Phi 上で実行したことが影響していると考えられる．native モデルではなく offload モデルを使用するなど，性能向上の期待できる部分のみを Phi に実行させることで良い性能が得られるだろう．

今回の各性能評価においては基本的に既存の CPU 向けプログラムをそのまま Phi 上で実行するのみとしており，

表 3　各計算機環境における GeoFEM-Cube 性能評価結果 (1 ノード，Flat MPI 実行)

			Hitachi SR16K/M1	T2K 東大	Fujitsu FX10 Oakleaf-FX	「京」
Processor	Preproduction Xeon Phi 1.10 GHz	Intel E5-2670 2.60 GHz	IBM Power7 3.83 GHz	AMD Opteron8356 2.3 GHz	SPARC64 IXfx 1.848 GHz	SPARC64 VIIIfx 2.0 GHz
Core #/Node	57 (228)	8 (16)	32	16	16	8
Peak Performance (GFLOPS)	1003.2	166.4	980.5	147.2	236.5	128.0
STREAM Triad (GB/s)	102.0	75.7	264.2	20.0	64.7	43.3
Byte/Flop	0.102	0.455	0.269	0.136	0.274	0.338
GeoFEM-Cube (GFLOPS)	16.6	16.8	72.7	4.69	16.0	11.0
% to Peak	1.65	10.10	7.41	3.18	6.77	8.59
Last Level Cache/core (MB)	0.512	2.50	4.00	2.00	0.75	0.75

※ Hitachi SR16000/M1 (Hitachi SR16K/M1)，Hitachi HA8000 クラスタシステム (T2K 東大)，Fujitsu PRIMEHPC FX10 (Oakleaf-FX)，「京」は参考文献 [7] から引用
コアあたり問題サイズ：Phi 以外は 40^3 節点=3×64,000=192,000 自由度，Phi のみコア数に対してメモリ容量が小さいため 10^3 節点 =3×1,000=3,000 自由度

プログラムの修正は STREAM ベンチマークについてのみ行った．Phi 向けにプログラムを修正せずに実行できることは，特に GPU に対する Phi の大きな利点である．ただし，今後 Phi に特化した最適化を施していくうえではプログラムの修正も必要になると考えられる．

今後は Phi の最適化についてさらなる調査や研究を行う予定である．また今回実行した以外のプログラムについての性能評価，native 実行だけではなく offload 実行を用いた最適化についても取り組む予定である．これらから得られた知見が Phi を用いたアプリケーションの高速化につながると考えている．

謝辞　日頃より最適化プログラミングについて議論をさせていただいている東京大学情報基盤センタースーパーコンピューティング研究部門の皆様に感謝します．

参考文献

[1] TOP500 Supercomputer Sites: Stampede `http://www.top500.org/system/177931`
[2] Balazs Gerofi, Akio Shimada, Atsushi Hori, Yutaka Ishikawa: Towards Operating System Assisted Hierarchical Memory Management for Heterogeneous Architectures, ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC) (2012).
[3] Min Si, Yutaka Ishikawa: Design of Communication Facility on Heterogeneous Cluster, 情報処理学会研究報告 (HPC-133-16) (2012).
[4] Intel: Xeon Phi Coprocessor, Intel Developer Zone `http://software.intel.com/en-us/mic-developer`
[5] STREAM BENCHMARK `http://www.cs.virginia.edu/stream/`.
[6] 大島聡史, 實本英之, 鴨志田良和, 片桐孝洋, 田浦健次朗, 中島 研吾: 大規模 SMP 並列スーパーコンピューター (HITACHI SR16000 モデル M1) の性能評価, 情報処理学会研究報告 (HPC-133-5) (2012).
[7] 大島聡史, 實本英之, 鴨志田良和, 片桐孝洋, 田浦健次朗, 中島 研吾: 大規模超並列スーパーコンピューターシステム Oakleaf-FX(Fujitsu PRIMEHPC FX10) の性能評価, 情報処理学会研究報告 (HPC-135-43) (2012).
[8] HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers `http://www.netlib.org/benchmark/hpl/`
[9] GeoFEM `http://geofem.tokyo.rist.or.jp/`.
[10] UT-HPC benchmark `http://www.cspp.cc.u-tokyo.ac.jp/ut-hpc-benchmark/`.
[11] 中島研吾，片桐孝洋：マルチコアプロセッサにおけるリオーダリング付き非構造格子向け前処理付反復法の性能，情報処理学会研究報告（HPC-120-6）（2009）．