Vol.2016-HPC-157 No.8
2016/12/21

電子動力学コード ARTED による Knights Landing プロセッサの性能評価

廣川 祐太[1,a)]　朴 泰祐[3,1]　佐藤 駿丞[3]　矢花 一浩[3,2]

概要：2016 年 12 月，東京大学と筑波大学が協働運営する最先端共同 HPC 基盤施設 (JCAHPC) は，Intel Xeon Phi (Knights Landing, KNL) プロセッサを採用した大規模メニーコアシステム「Oakforest-PACS」の運用を開始した．KNL プロセッサは，アクセラレータであった前世代の Knights Corner (KNC) プロセッサと異なり，一般的な Xeon プロセッサと同様にブート可能なソケット版が用意され，KNL 単独による計算機の構築が可能となった．本研究では，我々が KNC に向けて最適化を行ってきた第一原理電子動力学シミュレーションコード ARTED を早期試験運用中の Oakforest-PACS システムに移植し，KNL 向けの変更とその性能について報告する．同コードの支配的な演算は時間発展計算中の 25 点ステンシル計算で，KNC においては Intrinsics を用いた Explicit vectorization を行い，アプリケーション全体を通しメニーコアに向けた最適化を行ってきた．この Explicit vectorization コードを最小変更で KNL に移植し性能評価を行ったところ，Oakforest-PACS の 1 ノード (Xeon Phi 7250) 上で最大 758.4 GFLOPS，理論ピーク性能比 24.8%を達成した．また，時間発展計算では KNC クラスタ COMA (理論ピーク性能 2548 GFLOPS / Node) に対し，Oakforest-PACS (理論ピーク性能 3046 GFLOPS / Node) がより高い性能を示すことを確認した．

1. はじめに

Intel のメニーコア型プロセッサ Xeon Phi は，2016 年 6 月に Knights Landing (KNL) がプロダクトとして正式発表され，これまでの Knights Corner (KNC) と異なりブータブルなホストプロセッサとして利用可能となった．これにより KNL のみをプロセッサとしたシステムの構築が可能となり，主に米国と日本で大規模システムの構築が行われている．国内では，筑波大学と東京大学が協働設置する Joint Center for Advanced HPC (JCAHPC：最先端共同 HPC 基盤施設) では，ピーク性能 25 PFLOPS のシステム「Oakforest-PACS」を 2016 年 12 月に正式に稼働開始し，2016 年 11 月の TOP500 では Linpack 性能 13.55 PFLOPS を達成し世界 6 位，そして国内では「京」コンピュータを凌ぎ最高性能システムとなった．また，近年注目されている HPCG ベンチマークでは，0.3855 PFLOPS を達成し世界 3 位にランクされた [1], [2], [3]．また，文科省が進めるポスト京コンピュータ (フラッグシップ 2020 プロジェクト) ではメニーコアプロセッサを用いたシステムが予定されており，Oakforest-PACS は同システムへ向けたアプリケーション開発準備環境としても期待されている．

KNL は x86 互換の命令セットを持つブータブルなプロセッサであり，Xeon プロセッサで開発されたアプリケーションを，基本的にコード変更なしで動作させることが可能である．しかしながら，その性能特性は Xeon プロセッサとは大きく異なる．ここで重要なのは，単純な移植では実アプリケーションに Xeon Phi を適用し高い性能を得るのは非常に困難で，Xeon Phi つまり大規模スレッド並列かつ長いベクトル長をもつメニーコアプロセッサを意識したコードチューニングが強く要求されることである．我々はこれまで，Oakforest-PACS の導入を視野に入れ，筑波大学計算科学研究センターの KNC クラスタ COMA [4] を用いて，第一原理電子動力学コード ARTED のメニーコアプロセッサ向けコードチューニングを行ってきた [5], [6]．これらの実装は，現在すべて Github で公開している [7]．

本研究では KNC 向けに最適化を行ってきたコードを KNL に移植し，KNC からの改良と KNL の実際の性能について論じ，KNL の利用に関する知見をまとめる．

2. ARTED: 電子動力学シミュレータ

ARTED (Ab-initio Real-Time Electron Dynamics simulator) は，筑波大学計算科学研究センターにて開発されて

1 筑波大学大学院 システム情報工学研究科
2 筑波大学大学院 数理物質科学研究科
3 筑波大学 計算科学研究センター
a) hirokawa@hpcs.cs.tsukuba.ac.jp

いる実時間密度汎関数理論に基づくマルチスケール電子動力学シミュレータである [8], [9]．パルス光の電子動力学を記述するとともに，電磁場と電子の運動をマルチスケール手法を用いて同時に記述する．電子動力学では，電子軌道に対する時間依存 Kohn-Sham (TDKS, Time-Dependent Kohn-Sham) 方程式において，実時間・実空間法を用いて電子の波動関数の導出及び求解を行い，光電磁場では，Maxwell 方程式を一次の有限時間差分 (FDTD, Finite Difference Time-Domain) 法により解く．本研究では，これらをそれぞれ「電子動力学空間 (または TDKS 方程式)」，「マクロ空間」と呼ぶ．

ARTED は RSDFT (Real-Space Density Functional Theory) [10] と同様の方法を用いて基底状態を求める．RSDFT が，1,000∼100,000 原子といった大規模な系を対象としているのに対し，ARTED は 10∼100 原子程度の小規模なセルを非常に多くの個数分計算する必要がある．また，ARTED は RSDFT と同様の計算を行うが，時間発展計算が大部分を占め，その初期状態となる基底状態を求める計算時間は非常に短いものとなる．電子の波動関数は 4 次のテイラー展開で計算され，波動関数のハミルトニアンを計算する際にステンシル計算が必要となる．時間発展計算はおおよそ 1 万から 10 万ステップ行われるため，ARTED では時間発展計算が支配的であり，その大部分はステンシル計算に費やされる．大規模原子を対象とする RSDFT は，実空間を domain decomposition により分割し MPI で並列計算を行うため，隣接する MPI プロセス間で袖領域交換が必要となり，通信の隠蔽が大きな課題となっている．一方，ARTED では，電子動力学空間において実空間ではなくより大きな並列空間である波数空間を MPI で分割し並列計算する．1 つの波数の実空間計算は 1 プロセスで十分実行可能な計算量とメモリ量のみを要求する．このため，実空間分割における袖領域の交換が不要な代わりに，全ての波数空間上の実空間データについて，`MPI_Allreduce` 通信を用いて計算結果を束ねる必要がある．マクロ空間では FDTD 法による隣接格子間の通信のみを行うため，相対的にマクロ空間より電子動力学空間は高い通信コストを持つ．しかし，実空間のサイズは RSDFT に対し非常に小さく，計算が 10^{-6} レベルに対し通信は 10^{-9} レベルの時間 [sec] オーダのため ARTED は RSDFT に対して通信コストが低く，大規模並列システム向けのアプリケーションであると言える．

電子動力学空間にて電子軌道に対する波動関数配列を表現するために，計算領域は下記の 3 つのパラメータで構成されている．

- ブロッホ波数空間格子 (`NK`)
- バンド (`NB`)
- 3 次元実空間格子点 `NL`)

マクロ空間 (`NZ`) と，`NK` が MPI 並列され，MPI コミュニケータが 2 つ設定される．電子動力学空間において，各 MPI プロセスは NK/NP×NB 個の 3 次元実空間格子点を OpenMP を用いて並列に計算する．各実空間 (NL) は独立に存在しており，ステンシル計算は各 OpenMP スレッドが独立かつ逐次的に行う．実シミュレーションの際には，マクロ空間格子が複数個設定され，この格子の数の TDKS 方程式を解くことになる．1 個の TDKS 方程式を解く MPI プロセス数は固定されるため，Strong scaling の評価では 1 個の TDKS 方程式の計算性能を評価する．性能評価では，入力データに Si と SiO_2 を用い，パラメータをそれぞれ (NK, NB, NL) = (24^3, 16, $16 \times 16 \times 16$) と (4^3, 48, $20 \times 36 \times 52$) と設定する．Si は Xeon Phi が求める大規模並列性を持ちつつ実空間のサイズが小さいのに対し，SiO_2 は実空間のサイズが大きく，ARTED の並列性を決める波数空間が小さく設定されている．時間発展計算中，TDKS 方程式内では `MPI_Allreduce` が唯一の通信となり，最大でサイズ NL の倍精度浮動小数点ベクトルの総和を行う．通信は，1 個のマクロ空間格子点内の MPI プロセス間と，マクロ空間の全格子点 (全 MPI プロセス) 間の 2 つが必要となる．前者は `MPI_Allreduce` による TDKS 方程式の波数空間を束ねる通信，後者は Maxwell 方程式の袖領域通信だが，現在計算式と実装を簡略化するために後者を `MPI_Allgather` 相当の通信で行っている．これは今後 Peer-to-Peer 通信に書き換える予定である．

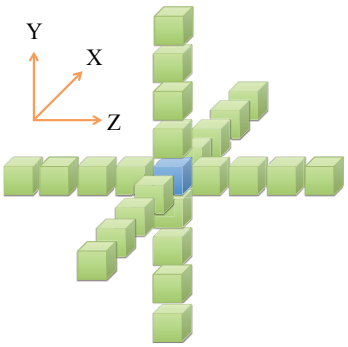

図 1 25 点ステンシル計算のメモリアクセスパターン

計算領域の波動関数配列は，倍精度複素数で表現され周期境界条件による 25 点ステンシル計算が行われる．**図 1** に，25 点ステンシル計算のメモリアクセスパターンを示す．非常にメモリバンド幅律速な問題となるが，次に示す通り一般的なステンシル計算とは異なる．本研究でのステンシル計算は，波動関数のハミルトニアン計算に用いられているため，1 回の時間発展で 1 個の実空間に対し 4 回のステンシル計算が必要となる．ハミルトニアン計算はステンシル計算と擬ポテンシャル計算で構成され，擬ポテンシャル計算も同様に 4 回行われる．前述のとおり，1 個の実空間の計算は OpenMP の 1 スレッドで行われるため，各スレッドは 1 回の時間発展で 4 回のステンシル計算が含まれるハミルトニアン計算を逐次的に行い，複数個の実空間を

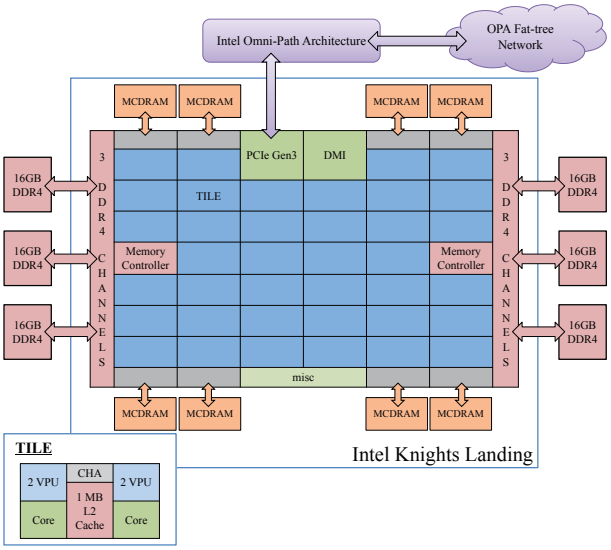

図 2 KNL システムの構成例 (Oakforest-PACS)

計算する．各実空間は独立，閉じた空間のため 1 回の時間発展で行われる 4 回のステンシル計算において OpenMP のスレッド同期または MPI による通信が発生しない．

我々はこれまでに KNC クラスタ COMA を用いて，第一原理電子動力学コード ARTED をメニーコアプロセッサ向けに最適化し，京コンピュータの後継機である FX100 システムや KNL を対象に最適化と性能評価を行ってきた．本研究では，KNL の実際の性能や実行に関する知見をまとめる．

3. Knights Landing アーキテクチャ

KNL は，KNC に続く第 2 世代 Xeon Phi プロセッサである．既に Intel 社より KNC からの変更点などは詳解されているが，ここでは特に重要と考えられる変更について示す [11].

- PCIe 版だけではなくソケット版が提供される
- 高バンド幅メモリとして MCDRAM をチップ上に設置している
- KNC では各コアがリングバスで接続されていたのを 2D メッシュネットワークへ変更している
- 動作クロックが向上し Turboboost により逐次性能は KNC に対し最大 3 倍まで向上
- 512-bit ベクトル命令として AVX-512 をサポート

ソケット版の提供により，一般的な Xeon CPU システムと同様に KNL のみを用いたシステムの構築・運用が可能となった．**図 2** に KNL のプロセッサ内部構成と，KNL システムの構成例をまとめたものを示す．ホストプロセッサ (KNL) と通信デバイス，メモリと非常にシンプルな構成となっていることがわかる．

チップ構成について説明する．まず，KNC ではコア間ネットワークがリングバスだったが 2D メッシュへ変更されている．コアの構成も変更され，2 コアを 1 組として “Tile” と呼び，この Tile に 1 MB の L2 キャッシュが接続される形となった．もし L2 キャッシュが 2 つのコア間で均等に分配されるなら，キャッシュメモリの条件は KNC と同様である．各コアは 2 つの Vector Processing Unit (VPU) を持ち，KNC と同様に Hyper-threading により最大 4 スレッドが動作する．チップの両サイドには，DDR4 メモリチャネルが各 3 つ，合計 6 つ用意され，メモリコントローラが 2 つ用意される．そしてチップ上には，高バンド幅メモリである MCDRAM が 2 GB ×8 個接続され，合計 16 GB が利用できる．

KNL は 2D メッシュネットワークの扱い，MCDRAM と DDR4 メモリの扱いについて複数の configuration が存在する．KNL 上のすべてのコアを 1 つのプロセッサとして扱う All-to-All mode と Quadrant mode，コアを複数の NUMA node として仮想的に分割する Sub-NUMA Clustering (SNC) mode がある．All-to-All mode では，チップ上のすべてを 1 つのネットワークとして扱っているためメモリとコア間のパスが長くなってしまう．Quadrant mode では，ネットワークを仮想的に分割しできる限りコアに近いメモリにデータを配置しパスを短くする．SNC mode は 4 分割 (SNC4) と 2 分割 (SNC2) が可能で，両者は分割数が異なるのみで挙動は同じである．以下，SNC4 mode を例に説明する．SNC4 mode では，ネットワークを仮想的に 4 分割して NUMA node を作り，4 ソケットの Xeon CPU ノードのように閉じたメモリアクセスを行う．本研究では Quadrant と SNC4 mode を取り上げ，性能評価を行う．メモリモードは，MCDRAM と DDR4 メモリを NUMA memory node として別々に扱う Flat mode，MCDRAM を L3 cache 的に扱う Cache mode，MCDRAM を NUMA memory と L3 cache に分割する Hybrid mode の 3 つがある．Cache mode では，ノードあたりに扱うデータサイズが MCDRAM より小さければ，キャッシュ利用のオーバーヘッドにより Quadrant mode に対し数%の性能低下が発生するだけと考えられる．これらのメッシュネットワーク構成や，MCDRAM と DDR4 のモードは，BIOS での切り替え，すなわちノードの再起動が必要となる．実アプリケーションでは，これらの configuration により性能特性が変わるため，事前の性能評価が必須である [12].

4. 評価環境

本研究では, KNL システム Oakforest-PACS (JCAHPC), KNC クラスタ COMA (筑波大 CCS), FX100 システム (名古屋大学 ITC) [13] の 3 つのメニーコアシステムの性能評価を記載する．各システムの特徴としては，Oakforest-PACS 及び COMA は Omni-Path Architecture (OPA) か InfiniBand FDR による Fat-tree network であるのに対し，FX100 は 6 次元メッシュトーラスの Tofu2 network である点，及び Oakforest-PACS と FX100 は 1 CPU/Node の一

表 1　本研究の評価環境

	Oakforest-PACS (test operation)	COMA	FX100
# of Node	128 (system total: 8208)	128 (system total : 393)	192 (system total: 2880)
Processor	Intel Xeon Phi 7250 1.4 GHz	Intel E5-2670v2 2.5 GHz ×2 (CPU) + Intel Xeon Phi 7110P ×2 (KNC)	SPARC64 XIfx 2.2 GHz
# of Cores / Node	68	20 (10 × 2, CPU) + 120 (60 × 2, KNC)	32 + 2 assistant-cores
Memory	16GB (MCDRAM) + 96GB (DDR4)	64GB (CPU, DDR3) + 8GB ×2 (Xeon Phi, GDDR5)	32GB (HMC)
Interconnect	Intel Omni-Path Architecture	InfiniBand FDR (Mellanox Connect-X3)	Tohu Interconnect 2
Compiler & MPI	Intel compiler 17.0.1 Intel MPI 2017 update 1	Intel compiler 16.0.2 Intel MPI 5.1.3	Fujitsu Compiler 2.0.0
Theoretical Peak Performance / Node	3046 GFLOPS	400 GFLOPS (200 × 2, CPU) + 2148 GFLOPS (1074 × 2, KNC)	1126 GFLOPS

般的な CPU クラスタと同様の構成であるのに対し，COMA では KNC と Ivy-Bridge CPU のヘテロジニアス構成となっている点である．Oakforest-PACS では 8208 ノードが利用できる予定だが，現在試験運用期間のため 1 ジョブあたり 128 ノードまでの利用に制限されており，本研究では 128 ノードまでの Strong scaling の評価を行う．

KNL では高バンド幅メモリである MCDRAM と，大容量の DDR4 メモリが利用できるが，本研究では MCDRAM のみを利用する．この場合，`numactl` コマンドを利用する方法がもっとも簡易な方法である．Quadrant mode の場合，NUMA memory 0 が DDR4，NUMA memory 1 が MCDRAM となっているため，`numactl -m 1 ${PROGRAM}` とする．SNC4 mode の場合，NUMA memory 0,1,2,3 が DDR4，NUMA memory 4,5,6,7 が MCDRAM となっている．このとき，各コアは自身の Sub-NUMA ノードが持つ MCDRAM にアクセスするため，`numactl -m 4,5,6,7 ${PROGRAM}` として実行すれば目的が達成できる．このような構成を Flat-Quadrant 及び Flat-SNC4 mode などと呼ぶが，本研究ではメモリモードは Flat で固定のため，単に Quadrant 及び SNC4 mode と呼ぶことにする．

5. ステンシル計算の性能評価

5.1 Knights Landing への移植

文献 [5], [6] などにて，我々はコンパイラ自動ベクトル化 (Compiler vectorization，以後 Compiler vec. とする) を施した後，KNC (512-bit SIMD) に向けて ARTED の 25 点ステンシル計算を Intrinsics を用いて実装し (Explicit vectorization，以後 Explicit vec. とする)，最適化を行ってきた．ここでは，KNC の IMCI (Initial Many-Core Institution) から KNL などが提供する AVX-512 命令への移植について論ずる．

IMCI と AVX-512 は，四則演算などの基本的な命令のフォーマットは同じだが，シャッフルや入れ替えといった命令のフォーマットが異なる．また，非アラインロード命令についても IMCI がキャッシュラインを跨がないように

```
#ifdef __AVX512F__
/* Intrinsics for KNL and AVX-512 processors */
#define _mm512_loadu_epi32   _mm512_loadu_si512
#define _mm512_storenrngo_pd _mm512_stream_pd
#elif __MIC__
/* Intrinsics for KNC */
inline
__m512i _mm512_loadu_epi32(int const* v)
{
  __m512i w;
  w = _mm512_loadunpacklo_epi32(w, v + 0);
  w = _mm512_loadunpackhi_epi32(w, v + 16);
  return w;
}
#endif
```

図 3　プリプロセッサを用いた IMCI から AVX-512 へのコード変換イメージ

2 命令を発行する必要があるのに対し，AVX-512 では AVX 命令などと同様に 1 命令で実行できる．一部では，IMCI では実装されているが AVX-512 では実装されていない命令も存在する．

本研究で用いた IMCI Explicit vec. のうち，AVX-512 において書き換えが必要な命令，または計算は以下の 4 つである．

- 128-bit 単位での並べ替え (shuffle 命令)
- non-temporal store 命令
- 倍精度浮動小数点数から倍精度複素数への変換
- 非アラインロード命令

128-bit 単位での並べ替えと non-temporal store 命令は，フォーマットや名前が異なるのみで，パラメータはほぼ同じである．倍精度浮動小数点数から倍精度複素数への変換，非アラインロード命令は必要命令数や必要な命令そのものが異なるが，5~10 行程度のインライン関数の置換のみが必要となる．名前の変換とインライン関数実装の入れ替えのみを必要とするため，これらはプリプロセッサを用いることで，KNC から KNL へ容易に移行できる．プリプロセッサによる置換イメージを**図 3** に示す．`__AVX512F__`マ

表 2 ベクトル化性能 [GFLOPS]

Si case	Compiler vec.	Explicit vec.
KNC ×2	251.4	467.9
KNL	547.0	729.1
SiO_2 case	Compiler vec.	Explicit vec.
KNC ×2	185.0	230.6
KNL	442.0	542.9

クロは，AVX-512 をサポートするプロセッサで定義され，__MIC__は KNC でのコンパイル時に定義される．本研究の KNL 向け Explicit vec. コードでは，KNL 向け最適化を目的としたコードの追記は行わず，全て KNC 向けのコードを利用し，プリプロセッサによる置換のみで実装した．

AVX-512 命令は複数のサブセットに分かれており，同じ AVX-512 をサポートするプロセッサであっても，利用できる命令が異なる点に注意が必要となる．本研究で実装した Explicit vec. コードでは，AVX-512 対応の全プロセッサがサポートする基本命令セットである AVX-512F (Foundation) のみを利用している．すなわち，AVX-512 をサポートする全てのプロセッサで，本研究のコードの実行が可能である．COMA には 1 ノードあたり KNC 2 台が接続されていること，KNL の理論ピーク性能は KNC ほぼ 3 台分に相当することから，ステンシル計算では KNC 2 台と KNL 1 台での性能比較，つまり COMA と Oakforest-PACS のそれぞれ 1 ノードでの比較を行う．

Oakforest-PACS では通常利用時，Xeon Phi 7250 が持つ 68 コアのうち 64 コアの利用を推奨している．ステンシル計算では，64-cores/256 OpenMP threads での性能評価を行う．Affinity 設定には Intel OpenMP 環境変数 KMP_PLACE_THREADS と KMP_AFFINITY を併用する．KMP_PLACE_THREADS は，OpenMP を実行するプロセッサの中で利用するコア数，コアあたりの OpenMP スレッド数，利用するコアのオフセットを指定できる．例えば，KMP_PLACE_THREADS=64c,4t,20 とすると，64-cores/256 OpenMP threads を利用し，2 つ目のコアから利用する．KMP_AFFINITY を用いて，CPU ID と OpenMP スレッド番号の対応付けを設定する．KNL では，2-cores/8 threads が 1 つの Tile を構成し 1MB の L2 キャッシュを共有するため，スレッド番号は可能な限りコア内で連続とすることが望ましいと考えられる．本研究では KMP_AFFINITY=balanced,granularity=fine とした．

5.2 性能評価

表 2 に，Si と SiO_2 での Compiler vec. と Explicit vec. による演算性能を示す．KNC では，Compiler vec. の性能に対して最大 1.86 倍の性能が Explicit vec. で得られるのに対し，KNL では性能差が縮まり最大 1.42 倍となった．KNC 向けに実装したコードが KNL に最適でない可能性も考えられるが，少なくとも依然として Explicit vec. の方が Compiler vec. より高性能である．Explicit vec. の更なる最適化も考えるが，KNL (AVX-512) に対する Intel コンパイラの Compiler vec. の性能が KNC (IMCI) に比べ向上していると推察される．

```
#define L1PREFETCH_DISTANCE 64
inline
__m512d _mm512_load_prefetch_pd(void const* c) {
  __m512d a = _mm512_load_pd(c);
  _mm_prefetch(
    ((char const*)c) + L1PREFETCH_DISTANCE,
    _MM_HINT_T0
  );
  return a;
}
```

図 4 Load 命令へのソフトウェアプリフェッチの挿入

表 3 ソフトウェアプリフェッチの挿入による性能変化 [GFLOPS]

Si case	64 B	128 B	192 B	256 B
KNC ×2	+124.76	+83.24	+55.25	+19.28
KNL	-68.06	-78.23	-88.82	-106.36
SiO_2 case	64 B	128 B	192 B	256 B
KNC ×2	+105.94	+109.48	+95.27	+73.89
KNL	+25.43	+2.11	-16.35	-47.45

表 4 各プロセッサのステンシル計算性能 [GFLOPS]

Si case	Compiler vec. (F90)	Explicit vec. (C)
KNL	547.0	758.4
KNC ×2	251.4	591.4
SPARC64 XIfx	176.0	192.3
IVB ×2	225.5	233.1
SiO_2 case	Compiler vec. (F90)	Explicit vec. (C)
KNL	442.0	593.8
KNC ×2	185.0	336.4
SPARC64 XIfx	219.9	179.1
IVB ×2	229.8	239.0

次にソフトウェアプリフェッチの効果について示す．本研究ではデータの空間的局所性を意識し，レジスタに読み込んだデータを全て利用して計算するようにしている [5]．そこでロード命令の次にソフトウェアプリフェッチを差し込み，次のループで利用するデータを先読みすることで性能向上を図った．Aligned Load 命令を利用しているため，ロードした先頭アドレスから 64 Byte 先をプリフェッチすることで，次のキャッシュラインを読み込むことになる．実装したインライン関数を**図 4** に示す．この関数は，Load 命令をラップするもので AVX-512 intrinsic の_mm512_load_pd 関数の代わりに呼び出す．

表 3 に prefetch distance を変え，ソフトウェアプリフェッチを挿入しなかった場合に対する性能差分を示す．SiO_2 では，どちらのプロセッサでもソフトウェアプリフェッチ

により性能が向上し，KNC では distance を 64 Byte としたときに 1 台あたり 50 GFLOPS 程度の性能向上が得られている．Si では，KNC で 1 台あたり 60 GFLOPS 以上の性能向上が得られているのに対し，KNL ではどのケースでもソフトウェアプリフェッチの挿入によりかえって性能が低下している．KNL ではハードウェアプリフェッチの性能が KNC に比べ向上しており，手動でソフトウェアプリフェッチを挿入する必要性が低下していると考えられる [14]．Si と SiO_2 では各スレッドで計算する小領域のサイズが異なり，それぞれ 64 KB と 585 KB で，SiO_2 の場合は L2 キャッシュからデータが追い出されていると考えられる．KNL では，L2 キャッシュサイズより各スレッドが計算するデータサイズが大きい場合に，ソフトウェアプリフェッチの効果があるのではないかと考えられ，今後詳細を調査する予定である．

最後に，KNL/KNC/SPARC64 XIfx/Intel Ivy-Bridge (IVB) の性能比較を**表 4** に示す．同表では，KNC/IVB はプロセッサ 2 台での性能，KNL/SPARC64 XIfx はプロセッサ 1 台での性能である．つまり，いずれも計算機の 1 ノード相当の性能となる．IVB/SPARC64 XIfx はそれぞれ 256-bit SIMD の AVX と HPC-ACE2 で実装しているが，その実装詳細については省略する．KNC では 1 スレッドで計算する実空間サイズが L2 キャッシュを超えているため SiO_2 の性能が Si の半分程度の性能しか得られていないが，KNL では 593.8 GFLOPS の性能が得られ KNC 2 台に対し約 1.77 倍，KNC 3 台以上の性能が得られている．KNL は Turboboost により逐次性能が向上しているため，SiO_2 のようにスレッド並列性が低く逐次性能を要求するような計算でも高い計算性能が期待できる．SPARC64 XIfx ではソフトウェアパイプラインにより性能向上を図っているため，ベクトル長が短い Si では性能を得るのが難しい．並列性が高く実空間のサイズが小さくなっている Si では，KNL は 758.4 GFLOPS，ピーク性能比 24.8%を達成した．このコードは KNC に最適化した実装を，プリプロセッサを用いてコンパイル時に KNL 対応のインライン関数や Intrinsics に置き換えているだけのものなので，KNC へ最適化されたコードが KNL でも有効であることを示している．

6. 時間発展計算の性能評価

6.1 予備評価: 通信性能

Oakforest-PACS は，Omni-Path Architecture (OPA) を搭載した 2016 年 11 月にて最大システムである．OPA は InfiniBand EDR と同様に 100 Gbps の通信バンド幅を提供しているが，OPA ネットワークの大規模システムは Oakforest-PACS が最初のため，実際の通信性能についてはまだ共有されていないことが多い．本研究では，基本的な通信性能についてのみ述べる．

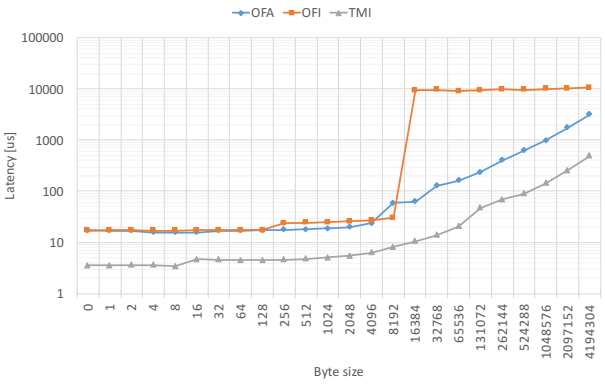

図 5 KNL+OPA の通信レイテンシ [us]

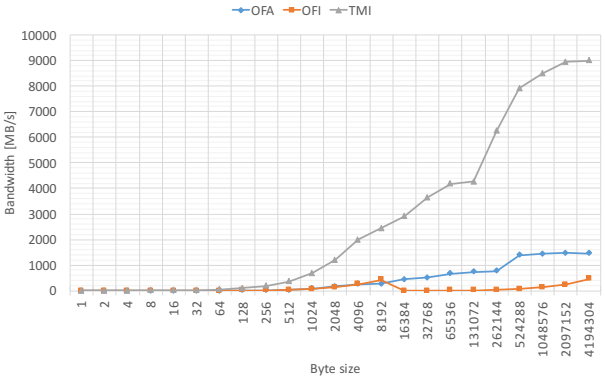

図 6 KNL+OPA の通信バンド幅 [MB/s]

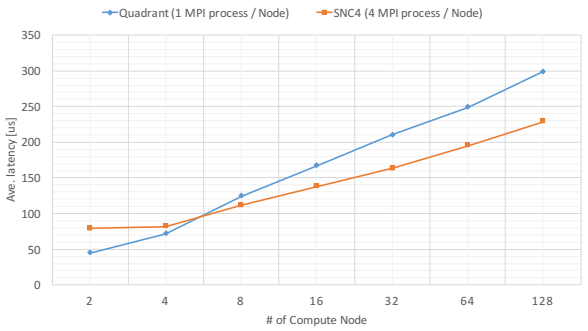

図 7 KNL+OPA での `MPI_Allreduce` 通信レイテンシ [us] (Si case)

Oakforest-PACS の Intel MPI では OPA を利用するための通信スタックとして，下記 3 つが利用できる [15]．

- Tag Matching Interface (TMI)
- OpenFabrics Alliance (OFA)
- OpenFabrics Interfaces (OFI)

これらのうち，Oakforest-PACS では TMI をデフォルト通信スタックとして利用している．**図 5** に各通信スタックの通信レイテンシを，**図 6** に各通信スタックの通信バンド幅を示す．TMI では，最大バンド幅が 8.9 GB/s とほぼバンド幅上限の性能が得られているのに対し，OFA や OFI では OPA の性能を活かせていない．また，Haswell

表 5　Quadrant 及び SNC4 mode の実行構成

	Process/Node	Max thread/Process	Fabric
Quadrant	1	256 (64-cores)	TMI
		266 (66-cores)	TMI
		272 (68-cores)	OFA
SNC4	4	64 (16-cores)	TMI

Xeon CPU+OPA では MVAPICH2 の最小通信レイテンシが InfiniBand と同様に 1 [us] 程度と報告されているのに対し [16]，KNL+OPA では 3.55 [us] と高い．これは，KNL の単体コア性能に起因するものと推察される．**図 7** に，ARTED が必要とする `MPI_Allreduce` 通信のレイテンシを示す．ここでは，Si を対象とした場合に必要となる最大通信サイズ (32 KB) での結果を示しているが，Quadrant mode より SNC4 mode の通信レイテンシが小さい．SNC4 mode は Quadrant mode の 4 倍の MPI プロセスを持っており，通信量がより多くなっているにも関わらず SNC4 mode の性能が高いことから，OPA は複数プロセスで利用することでスループットが向上できるのではないかと考えられる．

TMI にはトレードオフもあり，1 つ以上のスレッドを専有し通信を行うため，フルコア・フルスレッド利用時に性能が低下する．Oakforest-PACS で利用されている Xeon Phi 7250 は本来 36 Tile ある中で 34 Tile が有効化され 68 コアが利用できるが，2D メッシュネットワーク上のどの 2 Tile が無効化されているかは個体差がある．そのため，Oakforest-PACS では通常利用時には 64 コアでのアプリケーション実行を推奨している．図 6 の通り，TMI を利用しないと通信バンド幅が大幅に低下してしまうため，少なくとも 1 コアを TMI 用に残す必要がある．また，L2 キャッシュは 2 コアで共有されているため，キャッシュ汚染の影響を考えると 2 コアを利用せず 66 コアで利用したほうがよいとも考えられる．

Quadrant mode では，64/66-cores with TMI と，68-cores with OFA でのアプリケーション実行性能を評価する．SNC4 mode では，各仮想 NUMA ノードに割り当てられるコア数が 16 または 17 と異なるため，SNC4 mode においては 16-cores with TMI で動作させる．また，68-cores with TMI では大幅に実行性能が低下したため，評価には記載していない．

6.2　性能評価

Si case での Quadrant mode の各構成の性能を**図 8** に示す．図 8 中の “no affinity” は，利用するコア等を手動設定していないため，コア 0 にスレッドが割り当てられる可能性がある．その影響で，8，16 ノードでは affinity を設定した 64-cores 版に比べて性能低下が見られるが，Strong scalability が増加するに従って性能差が見られなくなっている．OFA スタックの場合は，Strong scalability とともに通信にかかる割合が増えているため，32 ノード以降で TMI スタックよりも性能差が広がっている．

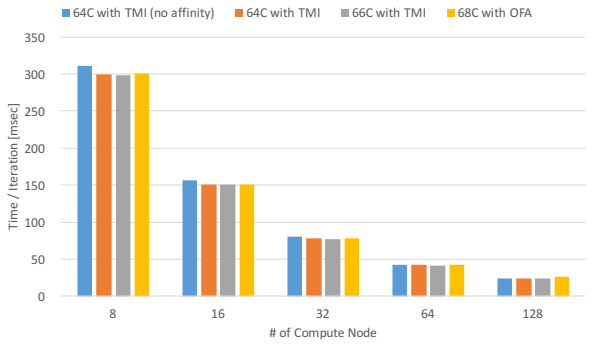

図 8　Si での Quadrant mode の性能，no affinity は OpenMP thread affinity を手動設定していないバージョンである

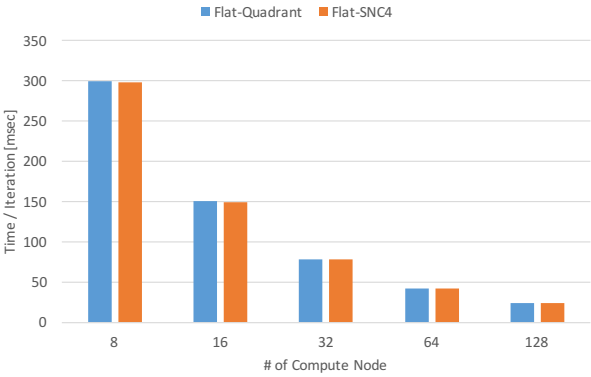

図 9　Si での Quadrant，SNC4 mode の性能比較

Si case での Quadrant (64-cores with TMI) と SNC4 mode の比較を**図 9** に示すが，大きな性能差は見られない．本研究で利用した ARTED は，波数空間並列であることから process/thread affinity を柔軟に設定できるため，メッシュネットワークを自由に選択できる．しかし，アプリケーションによってはプロセス内並列性の不足等により，64-cores/256-threads で動作させることが難しい場合がある．Xeon CPU のように 20 コア程度までを想定した計算の場合，SNC4 mode を用いて各プロセスが 16-cores/64-threads で計算することで高い性能を得られる可能性がある．前述の通信性能の通り，複数プロセスで OFA を用いることでスループットを上げられるため，より通信過密なアプリケーションでは SNC4 mode でのノードあたり 4 MPI プロセス実行が有利になる可能性もある．ARTED では，通信性能がボトルネックとなりづらいため，大きな性能差が得られていないとも考えられる．今後，より多くの計算リソースを必要とするシミュレーションを実行し，通信性能の影響について検証していく．

全システムの Si での性能を**図 10**，SiO_2 での性能を**図 11** に示す．FX100 では Tohu2 のネットワーク単位である 12 ノード単位で評価を行い，COMA では IVB only，KNC

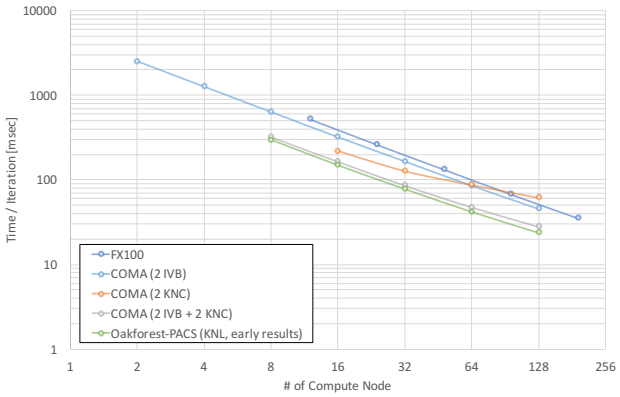

図 10 Si case の全システム性能比較

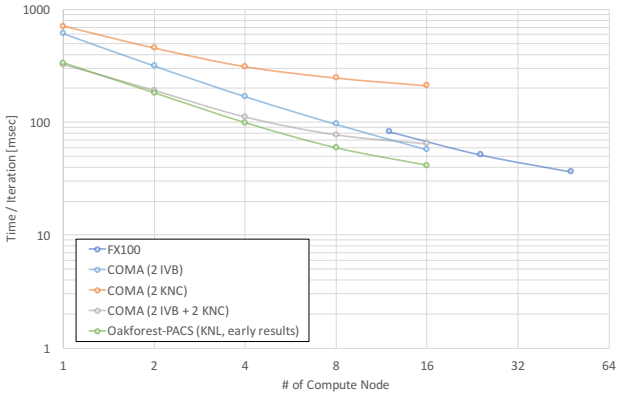

図 11 SiO_2 case の全システム性能比較

only，KNC と IVB を組合せた Symmetric 実行の 3 つをそれぞれ評価した．Si は KNC only を除きほぼ線形に性能向上し，Symmetric 実行と同等以上の性能が得られた．1.5 倍の 192 ノードを用いた FX100 に対し，128 ノードの KNL にて約 1.5 倍の性能が得られている．SiO_2 では，KNC only と Symmetric 実行の性能飽和が発生しているが，KNL や FX100 でも性能飽和が見てとれる．しかしグラフの通り，16 ノードの KNL と 48 ノードの FX100 がほぼ同性能で，プロセッサの理論ピーク性能と同じ約 3 倍の性能差が得られることを確認できた．

7. まとめ

本研究では，我々が以前よりメニーコアプロセッサ，特に Intel Xeon Phi (Knights Corner, KNC) に向け最適化してきた電子動力学コード ARTED を用いて，Knights Landing (KNL) プロセッサへの移植とその性能について論じた．支配的計算である 25 点ステンシル計算では，KNC 向けに実装した 512-bit SIMD Explicit vectorization コードを最小変更で KNL に移植し最大 758.4 GFLOPS，ピーク性能比 24.8%を達成した．KNC での最適化は，KNL にも有効であることを示したといえる．時間発展計算の性能評価では，ノードあたり 2548 GFLOPS を持つ KNC クラスタ COMA に対しノードあたり 3046 GFLOPS である KNL システム Oakforest-PACS はより高い性能が得られている．

本研究から KNL システムでは，下記に関する性能評価が必要であると考えられる．

- メッシュネットワーク (Quadrant や SNC4) の違いによる process/thread affinity
- 通信スタック (TMI，OFA，OFI) に違いによる通信性能と計算スレッドへの影響
- ソフトウェアプリフェッチの有無

今後，KNL へ向け更なる最適化を試行し，Oakforest-PACS フルシステム稼働開始後により多くの計算リソースを必要とするシミュレーションや Weak scaling の評価を行う予定である．

謝辞 本研究の一部は，筑波大学計算科学研究センター平成 28 年度学際共同利用プログラム課題「時間依存密度汎関数理論によるパルス光と物質の相互作用」，文部科学省ポスト「京」重点課題 (7)「次世代の産業を支える新機能デバイス・高性能材料の創成」，JST-CREST 研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」研究課題「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」による．本研究の Knights Landing プロセッサの評価環境は最先端共同 HPC 基盤施設 (JCAHPC) による．

参考文献

[1] 最先端共同 HPC 基盤施設：入手先 ⟨http://jcahpc.jp/⟩.
[2] TOP500: 入手先 ⟨http://www.top500.org/⟩.
[3] HPCG Benchmark: 入手先 ⟨http://www.hpcg-benchmark.org/⟩.
[4] 筑波大学計算科学研究センター：スーパーコンピュータ COMA (PACS-IX) について，入手先 ⟨http://www.ccs.tsukuba.ac.jp/files/coma-general/coma_outline.pdf⟩.
[5] 廣川 祐太，朴 泰祐，佐藤 駿丞，矢花一浩：電子動力学シミュレーションのステンシル計算最適化とメニーコアプロセッサへの実装，情報処理学会論文誌コンピューティングシステム (ACS)， Vol. 9, No. 4, pp. 1–14 (2016).
[6] Y. Hirokawa: Electron Dynamics Simulation with Time-Dependent Density Functional Theory on Large Scale Many-Core Systems, *SC16 Student Research Competition Poster* (2016).
[7] ARTED Github: 入手先 ⟨https://github.com/ARTED/ARTED⟩.
[8] S. A. Sato and K. Yabana: Maxwell + TDDFT multiscale simulation for laser-matter interactions, *J. Adv. Simulat. Sci. Eng.*, Vol. 1, No. 1, pp. 98–110 (2014).
[9] M. Schultze, K. Ramasesha, C. D. Pemmaraju, S. A. Sato, D. Whitmore, A. Gandman, J. S. Prell, L. J. Borja, D. Prendergast, K. Yabana, D. M. Neumark and S. R. Leone: Attosecond band-gap dynamics in silicon, *Science*, Vol. 346, No. 6215, pp. 1348–1352 (online), DOI: 10.1126/science.1260311 (2014).
[10] Y. Hasegawa, J. Iwata, M. Tsuji and *et al.*: First-principles Calculations of Electron States of a Silicon Nanowire with 100,000 Atoms on the K Computer, *Proceedings of 2011 International Conference for High Performance Computing, Networking, Stor-*

age and Analysis, SC '11, ACM, (online), DOI: 10.1145/2063384.2063386 (2011).

[11] A. Sodani: Knights landing (KNL): 2nd Generation Intel Xeon Phi processor, *2015 IEEE Hot Chips 27 Symposium (HCS)*, pp. 1–24 (online), DOI: 10.1109/HOTCHIPS.2015.7477467 (2015).

[12] C. Rosales, J. Cazes, K. Milfeld, A. Gómez-Iglesias, L. Koesterke, L. Huang and J. Vienne: A Comparative Study of Application Performance and Scalability on the Intel Knights Landing Processor, pp. 307–318 (online), DOI: 10.1007/978-3-319-46079-6_22 (2016).

[13] 名古屋大学情報基盤センター：スーパーコンピュータシステム，入手先 ⟨http://www.icts.nagoya-u.ac.jp/ja/sc/⟩.

[14] B. Joó, D. D. Kalamkar, T. Kurth, K. Vaidyanathan and A. Walden: Optimizing Wilson-Dirac Operator and Linear Solvers for Intel KNL, *High Performance Computing: ISC High Performance 2016 International Workshops, Revised Selected Papers*, pp. 415–427 (online), DOI: 10.1007/978-3-319-46079-6_30 (2016).

[15] Intel MPI: 入手先 ⟨https://software.intel.com/en-us/intel-mpi-library⟩.

[16] MVAPICH2 Pt-to-Pt Performance on Intel Haswell Architecture with Intel Omni-Path: 入手先 ⟨http://mvapich.cse.ohio-state.edu/performance/pt_to_pt/#haswell-omnipath⟩.