Vol.2017-HPC-160 No.20
2017/7/27

電子動力学シミュレーション ARTED の KNL システム Oakforest-PACS での全系性能評価

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

概要：これまで，我々は主に「京」コンピュータをメインターゲットに開発してきた電子動力学シミュレータ “ARTED” についてメニーコアプロセッサ，特に Intel Xeon Phi に対し最適化および性能評価を行い，最新アーキテクチャ Knights Landing (KNL) に対しても十分な性能が得られたことを報告している．本研究では，世界最大規模の KNL システムである “Oakforest-PACS” の全系を用いた ARTED による大規模シミュレーションにて行った性能評価について報告する．実問題として，応用上も重要な材料であるシリコンとグラファイトに対して高強度超短パルス光との相互作用で生じる電子動力学のシミュレーションを行った．システム全体の 99.8%に相当する 8192 ノードまでを用い，最も重要なハミルトニアン計算において最高 4 PFLOPS，ピーク性能比約 16%を達成した．スケーラビリティの評価として，全てのケースについて計算全体での Weak Scaling を確認，またシリコンのケースについては計算全体で良好な Strong Scaling が確認された．これにより，我々がこれまで行ってきた最適化が，世界最大の KNL システムにおいても十分な効果を発揮していることを示した．最後に，Strong Scaling で発生した性能低下とその原因，メニーコアプロセッサによって生じる問題について考察した．いくつかの問題は，現在も JCAHPC とともに調査と実験を進めている．

1. はじめに

これまで，我々は主に「京」コンピュータを対象に開発してきた電子動力学シミュレータ “ARTED” を，Intel 社の Xeon Phi を対象として最適化を行ってきた [1]．昨年には Xeon Phi の最新アーキテクチャ Knights Landing (KNL) を搭載したシステムが登場し，我々は既に筑波大学と東京大学が共同設置する Joint Center for Advanced HPC (JCAHPC: 最先端共同 HPC 基盤施設) にて稼働開始したピーク性能 25 PFLOPS の KNL システムである “Oakforest-PACS” を用い KNL 128 ノードまでの性能評価も行っている [2]．

メニーコアプロセッサは，比較的性能の低い演算コアを多数接続することで高い電力当たり性能を達成するため，プロセッサ内において高いスレッド並列性を要求する．加えてベクトル長が長い SIMD 演算器を持つ傾向にあるため，高いベクトル並列性も要求する．さらに，MCDRAM のような高速なメモリを持つとはいえ，理論ピーク性能が極めて高いことから，一般的なマルチコアプロセッサと同様，キャッシュメモリの更なる有効利用が要求される．これらの特性から，実アプリケーションにおいてメニーコアプロセッサの高い演算性能を引き出すのは容易ではない．現在，理化学研究所計算科学研究機構 (AICS) を中心に開発が進められているポスト「京」コンピュータでも汎用メニーコアプロセッサを用いることが公表されており，メニーコアプロセッサである Intel Xeon Phi を対象に行っている性能最適化は，ポスト「京」にも適用できると考えている．このため， Oakforest-PACS は同システムへの準備環境としても注目されている．

昨年度末，KNL への十分な最適化を準備したことを受け， Oakforest-PACS の全系 [*1] を用いた大規模シミュレーションの実行機会を得た．本論文では，全系実行による ARTED のメニーコアシステムでの性能，および問題点について述べる．

2. ARTED: 電子動力学シミュレータ

2.1 概要

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

1 筑波大学大学院 システム情報工学研究科
2 筑波大学 計算科学研究センター
3 Max Planck Institute for the Structure and Dynamics of Matter
a) hirokawa@hpcs.cs.tsukuba.ac.jp

[*1] Oakforest-PACS の総ノード数は 8208 であるが，扱い易い 2 の冪乗のノード数として 8192+16 と分割するのが合理的である．本論文ではこの 8192 ノードを事実上の「全系」と呼ぶ．

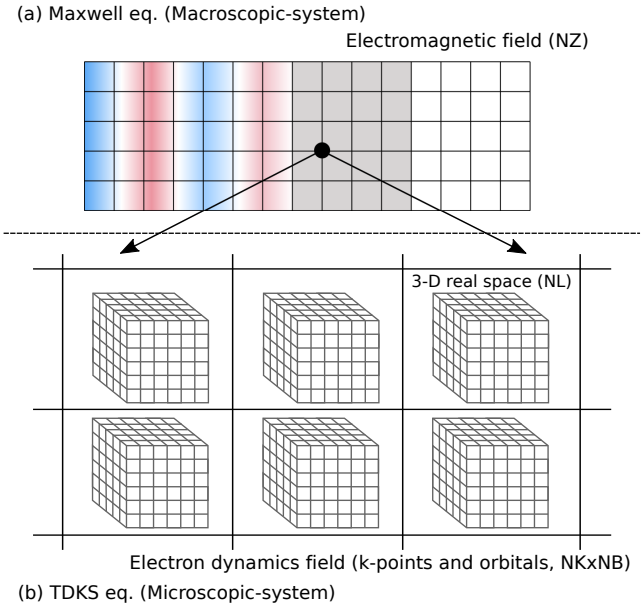

図 1 ARTED の計算領域 (2-D Maxwell + TDKS equation)

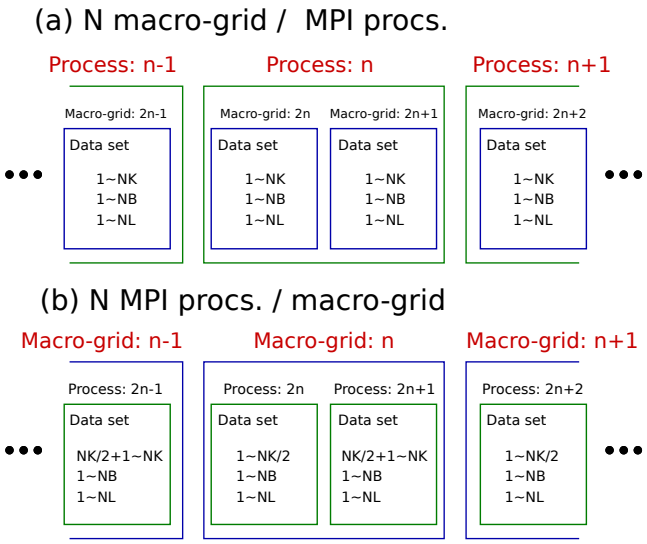

図 2 MPI の並列化方法

ている，光と物質の相互作用の第一原理計算を目的とした，Fortran90 で実装されたマルチスケール電子動力学シミュレータである [3], [4]．このシミュレータは，パルス光のもとでの電子動力学に加え，電磁場と電子の運動を 2 つの方程式を組み合わせた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記述及び求解を行い，光電磁場では，Maxwell 方程式を時間領域差分 (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」,「電磁場空間」と呼ぶ．**図 1** に，Maxwell 方程式と TDKS 方程式を解く際に用いる空間格子について示す．ここでは，2-D Maxwell 方程式と TDKS 方程式で示している．

ARTED は電子動力学の初期条件として，RSDFT (Real-Space Density Functional Theory) [5] と同様の方法を用いて基底状態を求める．ただし，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 では対象問題の特性から，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．ARTED は後で述べるように，時間発展が計算時間の大部分を占め，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．電子波動関数の時間発展は，時間発展演算子の 4 次のテイラー展開によって計算され，1 ステップあたりに波動関数に対する 4 回のハミルトニアンの演算が必要となり，この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．大規模系を対象とする 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，より大きな並列化可能な空間であるブロッホ波数空間を MPI で並列計算し，各実空間での MPI 並列は行わない．1 つの波数の実空間計算は，計算ノード 1 台が持つ計算能力とメモリ容量のみで十分に実施可能である．一方，軌道関数から密度分布を得るために，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．電磁場空間では FDTD 法による隣接格子間の通信のみを行うため，相対的に電子動力学空間の方が電磁場空間より高い通信コストを持つ．しかし，電子動力学空間が持つ 3 次元実空間の 1 個あたりのサイズは RSDFT に対し非常に小さく，1 ステップあたりの計算時間が 10^{-6} [sec] レベルに対し通信は 10^{-9} [sec] レベルのオーダのため，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．

2.2 マルチスケール計算の並列化

図 1 に示すように，ARTED のマルチスケール計算では 2 つの方程式を解くが，Maxwell 方程式はマクロ格子点数 `NZ` をパラメータとし，TDKS 方程式は下記の 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`)

2 つの方程式を解くため，MPI コミュニケータがそれぞれの方程式に対し定義されるが，実行時の MPI の並列化方法は 2 つに分かれる．

- **複数**のマクロ格子点をまとめ，**1 つ**の MPI プロセスで**複数**の TDKS 方程式を解く
- **1 個**のマクロ格子点に対し，**複数**の MPI プロセスで**1 個**の TDKS 方程式を解く

図 2 に，2 つの並列化方法を図示する．この図では 2 個のマクロ格子点を 1 MPI プロセスで解く場合を (a)，2 MPI プロセスで 1 個の TDKS 方程式を解く場合を (b) に示している．マクロ格子点 1 つあたりのデータサイズが小さい場合，複数のマクロ点を束ねて計算することで，より広い物理空間・より大きい物質でのシミュレーションを実行できる．マクロ格子点を 1 つの MPI プロセスが複数持つため，各 MPI プロセスは複数の閉じた系 (TDKS 方程式) を独立に計算する．TDKS 方程式を計算している間，各 MPI プロセスは通信なしに計算を行えるが，Maxwell 方程式の計算時に TDKS 方程式のデータを全 MPI プロセス間で共有するための通信が発生する．(a) のパターンでは，MPI のコミュニケータはグローバルに 1 つだが，(b) のパターンでは 2 つのコミュニケータを要する．マクロ格子点 1 つあたりのデータサイズが大きい場合，計算時間や計算ノードのメモリサイズなどの関係から，1 個のマクロ格子点を分割し 1 個の TDKS 方程式を複数の MPI プロセスで計算する．TDKS 方程式の MPI 並列化は波数空間 (NK) を分割するため，TDKS 方程式の計算中，1 個のマクロ格子点を計算する MPI プロセス群を MPI のサブコミュニケータとして定義し，サブコミュニケータ間で波数空間を束ねる通信を行う必要がある．ただし，波数空間を分割し，3 次元実空間格子は分割されないため，どちらの並列化方法においても袖領域交換は発生しない．

電子動力学空間において，1 個の TDKS 方程式を計算する MPI プロセス数を NP とすると，各 MPI プロセスでは (NK/NP)×NB 個の電子の波動関数 (3 次元実空間格子点: NL) を，OpenMP を用いてスレッド並列化する．各 3 次元実空間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．時間発展計算中の通信は，1 個の電磁場空間格子点内の MPI プロセス間と，電磁場空間の全格子点 (全 MPI プロセス) 間の 2 つが必要となる．どちらも MPI_Allreduce で，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．前者は TDKS 方程式の波数空間を束ねる通信，後者は Maxwell 方程式の袖領域に相当する通信である．袖領域交換は通常 1 対 1 通信が用いられるが，1 対 1 通信にすることによって起こる計算の煩雑化を防ぐため，MPI_Allgather に該当する通信を MPI_Allreduce で行っている (各プロセスが担当する領域以外をゼロクリアしておくことで結果的に MPI_Allgather と等価になる)．したがって通信サイズはマクロ格子点数に相当する．しかし，マクロ格子点数は Oakforest-PACS の全系を用いても最大 2^{15} 個，通常は MPI プロセス数やノード数に等しく非常に少ないため，ボトルネックとはなっていない．

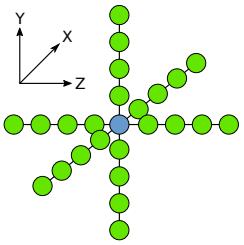

図 3 25 点ステンシル計算のメモリアクセスパターン

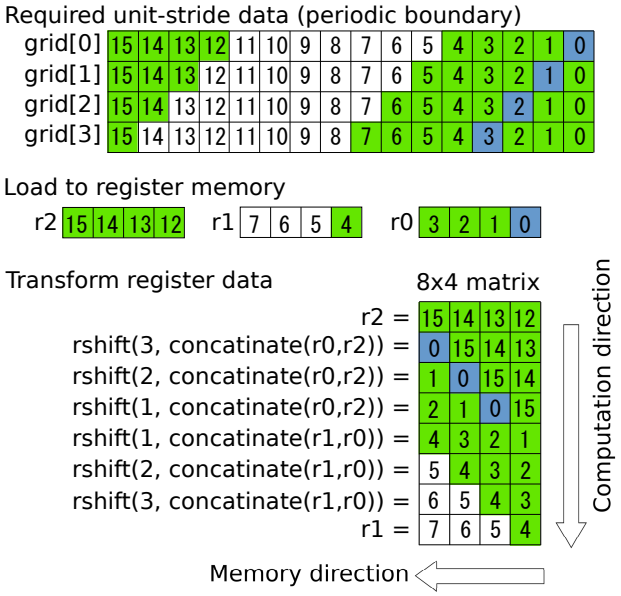

図 4 ステンシル計算の連続領域アクセス最適化

2.3 ステンシル計算

ハミルトニアン計算は，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 3** 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．これは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．

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，1 回の時間発展で 1 個の 3 次元実空間に対し，ステンシル計算と擬ポテンシャル計算が 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，各スレッドは複数個の実空間に対しハミルトニアンを逐次的に計算する．各実空間は独立で閉じた空間のため，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 による通信は発生しない．ハミルトニアン計算は実行時間の 7 割以上を占めるため，我々の最適化は，メモリバンド幅に律速されるステンシル計算を中心としている．

3. これまでの研究

3.1 KNC へのステンシル計算の最適化

本研究の初期段階において，我々は Knights Corner (KNC) に対して最適化を行ってきており，それらの最適化が Knights Landing (KNL) においても有用であることを確認している [2]. 特に，計算時間のうち最大 8 割強程度を占める 25 点ステンシル計算に傾注した最適化を行ってきた.

Fortran90 のコードを最適化し，コンパイラの自動ベクトル化の促進を行ったが，性能が不十分と考え C 言語での手動ベクトル化を実装した [1]. 特に，連続領域である Z 次元のメモリアクセス最適化を行い，メモリ帯域要求を下げ大幅な性能向上に成功した．concatinate shift 命令である `alignr` 命令を用いて，性能が低くボトルネックとなりやすい `gather` 命令を使わずに連続領域のアクセスを最適化した [6]．この最適化の概略を**図 4** に示す．ここで `rshift` はデータ単位の右シフト命令で第一引数個分ベクトルレジスタを右シフトして下位 4 要素を返し，`concatinate` は 2 つのベクトルレジスタを結合したものを返す関数とする．まず `r0`, `r1`, `r2` の 3 つのベクトルレジスタに必要なデータをロードし，これらだけを用いて必要な近傍点を全て格納した 8×4 の行列を `alignr` 命令により生成する．この行列には各格子点が必要とする近傍点が列方向に揃っているため，非常にシンプルな SIMD 演算で，4 つの格子点を同時に計算・更新することが可能となる．この最適化は，Intel が公開したレポート [7] をベースに周期境界条件に対して最適化を行ったもので，[7] とは異なり周期境界条件とそれ以外の場合の両方に適用可能である.

連続領域のメモリロードと同時に L1 キャッシュへのプリフェッチ命令を発行し，メモリアクセスレイテンシの隠蔽を行う．我々のコードは常に 64 Byte アラインされた状態でロード命令が発行されるため，ロードしたアドレスから 64 Byte 先のアドレスをプリフェッチすることで，キャッシュライン単位での効率的なプリフェッチが行える．また，連続領域のメモリロードに合わせて発行するため，次の反復で用いられるデータが L1 キャッシュにロードされる.

3.2 KNL への移行

KNC では IMCI が提供されているのに対し，KNL では AVX-512 命令が提供されているため，この 2 つの命令フォーマットの違いを吸収する必要がある [8]. 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 2 命令を発行

```
#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 5 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

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 の手動ベクトル化のうち，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 [9]
- 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 5** に示す．`__AVX512F__`マクロは，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 向け手動ベクトル化コードでは，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．本研究で実装した手動ベクトル化コードでは，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，Skylake-EP など 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.

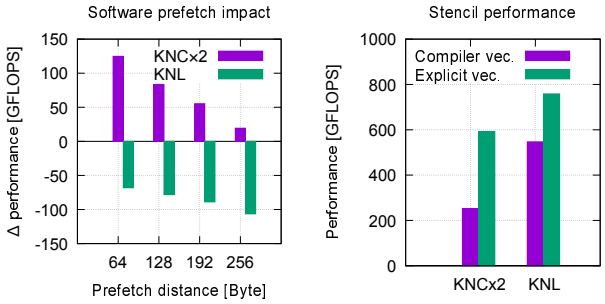

図 6 コンパイラベクトル化と手動ベクトル化コードでのソフトウェアプリフェッチの効果とステンシル計算性能 [GFLOPS]

表 1 本研究の評価環境 (Oakforest-PACS system)

CPU	Intel Xeon Phi 7250 with 68 cores, 1.4GHz base clock
# of nodes	8208 (use up to 8192)
Memory	16GB of MCDRAM and 96GB of DDR4-2400
Interconnect	Intel Omni Path Architecture with 100Gbps link
Network topology	Full bisection bandwidth of Fat-Tree
File system	26PB Lustre by DDN SFA7700X, 500GB/s bandwidth
File cache	940TB Burst Buffer by DDN IME14K, 1560GB/s bandwidth
Cooling	Water colling for CPU and Air cooling for others
Power consumption	Max.: 4.2MW (including cooling)
Operating system	CentOS 7 and McKernel (developed by RIKEN)
Compiler and MPI	Intel compiler 17.0.1 and Intel MPI 2017 update 1

3.3 ステンシル計算の性能比較

図 6 に，KNC 2 台と KNL 1 台でのソフトウェアプリフェッチの効果とステンシル計算に絞った性能比較を示す．マクロ格子点は 1 個として，電子動力学空間には実空間格子点を 16^3，波数空間とバンドの数は $8^3 \times 16$ とした．KNL のハードウェアプリフェッチの性能向上に伴いソフトウェアプリフェッチの手動挿入が不要となりつつあり [10]，我々のステンシル計算コードでも KNC では 1 台あたり 60 GFLOPS 以上の性能が得られたのに対し，KNL では逆に大幅な性能低下が見られた．

我々のステンシル計算コードは，KNL が KNC からの移植であるのにも関わらずどちらのプロセッサにおいても Intel コンパイラによるベクトル化よりも高い性能を達成した．KNL では 758.4 GFLOPS，ピーク性能比約 24.8%を達成し，KNC に対する十分な最適化が KNL にも非常に有用であることを示すことができた．

4. Oakforest-PACS 全系を用いた性能評価

Oakforest-PACS は 2017 年 6 月現在，ピーク理論性能は 25 PFLOPS，Linpack の性能は 13.55 PFLOPS を達成し，TOP500 リストの中で世界 7 位にランクされる日本最高性能のスーパーコンピュータである [11]．各計算ノードは CPU に KNL を搭載し，通信デバイスとして Intel Omni-Path Architecture (OPA) を持つ [12]．2016 年 12 月に筑波大学と東京大学が共同設置した Joint Center for Advanced HPC (JCAHPC) にてフルシステムの稼働を開始し，2017 年 4 月より正式運用が開始された [13]．

KNL を採用した一般的な CPU クラスタとして見ると，同じく KNL を採用した米国の国立エネルギー研究科学計算センター (NERSC) の Cori が Cray の MPP である XC40 システムのため，Oakforest-PACS は世界最大の KNL クラスタである．各計算ノードは約 3 PFLOPS のピーク演算性能を持つ Xeon Phi 7250 を唯一のプロセッサとして持ち，8208 台の計算ノードで構成される．また，2017 年 6 月現在において，同システムは OPA ネットワークを持つ世界最大のシステムでもある．**表 1** に，システムの基本性能について示す．

本研究では，扱いやすい 2 の冪乗のノード数での評価が合理的であるとして 8192 ノード，システム全体の 99.8%を用いて性能評価を行う．同システムは Burst Buffer が利用できるが，ARTED はファイル IO が一般的なシミュレーションに比べて非常に少ないため，本研究では用いていない．上述の理由より，我々はファイル IO について時間発展計算の終了後に一括で行うこととしたため，本研究の評価にはファイル IO の時間が含まれていないことに注意されたい．

KNL プロセッサでは，コア間ネットワークモードとメモリモードが複数用意されているが，Oakforest-PACS ではコア間ネットワークは Quadrant モードで固定され，メモリは MCDRAM を DDR4 メモリと同等に扱う Flat モードと，MCDRAM を DDR4 メモリのキャッシュとして扱う Cache モードの 2 種類のみが提供されている．この組合せを通常 Flat-Quadrant と Cache-Quadrant と呼ぶ．ARTED は，メモリバンド幅律速であるステンシル計算が支配的な計算となるため，本研究では DDR4 メモリを使用せず，Flat-Quadrant モードで MCDRAM のみを使用した．MCDRAM のサイズを超えるデータセットの場合，Cache モードを用いること性能と計算サイズのバランスを取れると考えられる．しかしながら，Cache モードはダイレクトマップ方式のためキャッシュの有効利用には非常に注意が必要で，Flat モードで利用する計算ノードを増加させた方がより高い性能を達成可能と考えられる [14]．

Oakforest-PACS は，OS ジッタの影響回避のためタイ

表 2　各シミュレーションのデータセット

	シリコン	グラファイト
MPI procs. / Macro-grid	–	8
Macro-grid / MPI procs.	1–4	–
Total # of macro-grid (NZ)	32768	1024
# of wave count (NK×NB)	$8^3 \times 16$	7928×16
Size of 3-D real space (NL)	16^3	$26 \times 16 \times 16$

マー割込をコア 0 のみ受け付ける，tickless 設定が行われている [15]．このため，Xeon Phi 7250 が持つ 68 コアのうち 64 コアの利用を推奨しており，本研究でも 64 コア・最大 256 スレッドで性能評価を実施する．また，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，コア 0 で発生する割り込みがコア 1 にも影響を与えると考えられる．したがって，本研究ではコア 0 および 1 を避けてスレッドアフィニティを設定し，割り込みの影響を抑える．

4.1　問題設定

本研究では，シリコン及びグラファイトの 2 次元薄膜と高強度超短パルス光の相互作用を対象とする計算を行う．シリコン及びグラファイトは応用上も重要な材料であるが，本計算はそれらのレーザー加工初期過程の解明に資するものである．高強度超短パルス光を用いた加工技術は，質の高い微細加工や難削材料への適用可能性により，応用上重要な加工技術として精力的に研究が進められている [16]．しかしながら，高強度な光電磁場が物質中に誘起する極めて非線形の強い電子動力学の複雑さのため，その微視的機構は未だ明らかになっていない．本研究は，これまで経験的なパラメータを用いた研究にとどまっていたレーザー加工の初期過程に対して，量子論の第一原理計算に基づく電子動力学計算と Maxwell 方程式を組み合わせることにより，精緻で予言力のあるシミュレーションを実現するものである．本計算により加工初期過程を理解することは，学術的な興味にとどまらず，応用上も加工の精度や効率を向上させるために重要な意味をもつ．

本研究では，これら 2 つの物質のシミュレーションから，計算性能を評価する．**表 2** に各シミュレーションのデータセットを示す．シリコンでは，TDKS 方程式のサイズが $8^3 \times 16 \times 16^3$ と小さいため，1–4 個のマクロ格子点をまとめて 1 個の MPI プロセスで計算する．グラファイトは，TDKS 方程式のサイズがシリコンに対し約 25 倍あり非常に大きいため，各マクロ格子点を 8 個の MPI プロセスで計算する．8192 ノードで計算できるマクロ格子点の数は，シリコンで 32768 点，グラファイトが 1024 点である．Weak Scaling では 128 ノードから計測し，シリコンは 512–32768 点，16–1024 点のマクロ格子点を計算し評価した．Strong Scaling では，それぞれ MPI の並列化方法が異なるため，MPI プロセスあたりの問題サイズの設定が異

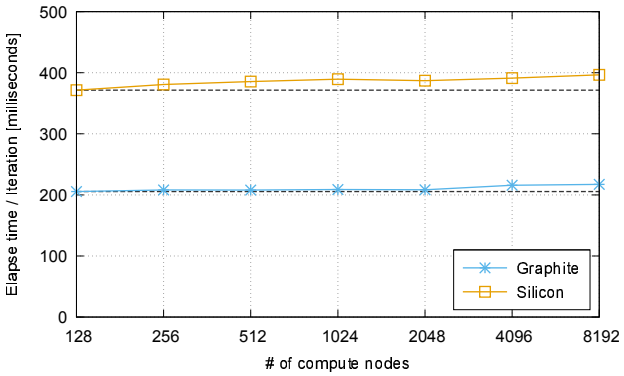

図 7　Weak Scaling での時間発展計算の評価

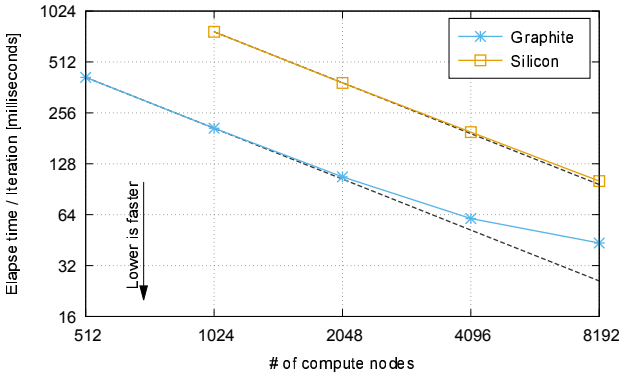

図 8　Strong Scaling での時間発展計算の評価

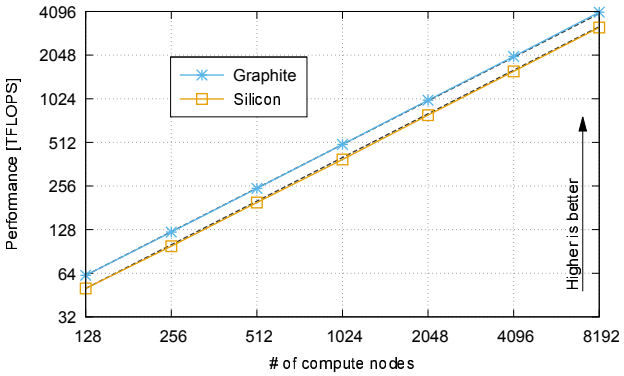

図 9　Weak scaling でのハミルトニアン計算性能 [GFLOPS]

なっている．シリコンは 1024 ノードから計測し，マクロ格子点を 8192 点に固定して MPI プロセスあたりのマクロ格子点を減らして評価した．グラファイトは 512 ノードから計測し，マクロ格子点を 128 点に固定して，マクロ格子点あたりの MPI プロセス数を増やして評価を行った．

4.2　全系性能

図 7 および**図 8** に，Weak Scaling および Strong Scaling での ARTED の性能を示す．Strong Scaling では物質により結果が異なっているが，128 ノードから全系に相当する 8192 ノードまで Weak Scaling をほぼ完璧に達成している．**図 9** には，Weak Scaling でのハミルトニアン計算の

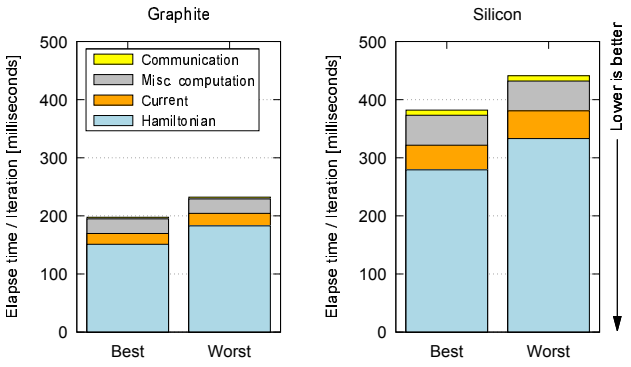

図 10 時間発展計算の計算時間内訳

FLOPS を示している．時間発展計算で最も重要なハミルトニアン計算において，我々は最大 4 PFLOPS，全系のうちピーク比で約 16%相当の演算性能を達成した．

シリコンは Strong Scaling をほぼ完璧に達成していると言えるが，グラファイトの計算で大幅な性能低下が見られる．この問題について，我々は各計算ノード間でおよそ 20%程度の計算時間の差が生じていることを発見しており，この差が原因であると考えている．全ての計算ノードには全く同じ計算量が与えられており，図 7 に示す Weak Scaling の結果からも見て取れるように計算負荷はバランスが取れている．しかしながら，**図 10** に示す最速・最遅の計算時間内訳で見ると，通信を全く含まず計算のみで構成されるハミルトニアン計算の時間に大きな差が生じている．

4.3 考察

我々は，通信を一切含まないハミルトニアン計算に発生しているノード間の性能差について，Turbo Boost が原因ではないかと考えている．Turbo Boost は，動作クロックをプロセッサの TDP を守りつつベースクロックから最大で数百 MHz ほど動的に引き上げ，瞬間的に高い性能を得られる．しかしながら，Turbo Boost は当然プロセッサ温度も条件として含めているため AVX-512 を使い演算器をフルに活用すれば，温度上昇により Turbo Boost によるクロック向上の効果時間が短くなることは容易に予想できる．また Oakforest-PACS では 2 ノードペアでプロセッサを冷却する水冷却システムを導入しているため，1 番目に冷却されるノードと 2 番目に冷却されるノードでは，プロセッサ温度が異なったとしても不思議ではない．

シリコンとグラファイトでは Strong Scaling 時の性能が異なっており，上述の Turbo Boost によるノード間性能差だけでは説明できていない．これは，図 2 に示すように MPI の並列化方法の違いによるものと考えられる．シリコンでは計算コストの大部分を占める電子動力学空間の計算において MPI での分割を行っておらず，各 MPI プロセスが独立して複数の閉じた系を計算するため，同期 (MPI_Allreduce) が一切発生しない．したがって，シリコンは全 MPI プロセスでの全体同期だけが唯一必要となり，上述の性能ギャップはハミルトニアン計算を行ったあと即座に是正されると考えられる．

電子動力学空間を MPI で分割しているグラファイトでは，1 個の系を計算する MPI プロセス群での同期，電磁場空間での全 MPI プロセスの同期，2 ステージの同期が必要となる．各系で閉じた同期は小数の MPI プロセス群，今回は 8 プロセス単位で発生するため，性能ギャップによる遅延はシリコンの場合に比べてランダム性が増加し，より広範囲に影響するものと考えられる．以上より，性能ギャップは最終的に通信レイテンシとして表面化し，通信時間の比が増加する Strong Scaling の評価で問題が顕在化した．もしこのアルゴリズムと一切関係ない性能ギャップが改善した場合，我々のアプリケーションは Strong Scaling においても高い性能を実現できると見込んでいる．

Turbo Boost のような動的なクロックアップを行う機能は，計算性能と供給可能電力のバランスという観点から非常に重要である．特にメニーコアプロセッサでは，通常の CPU の数倍のコアを搭載するため，TDP 要求はより厳しいものとなる．しかしながら，我々が行った大規模全系実行では，この機能がアプリケーションのアルゴリズムとは一切関係無い計算時間のずれを引き起こす可能性を示唆している．我々はこの問題について，JCAHPC と協力し現在も実験を行っている．

5. まとめ

本研究では，我々がこれまでに Intel Xeon Phi に対し最適化を進めてきた電子動力学シミュレータ “ARTED” について，世界最大の KNL クラスタである “Oakforest-PACS” を用いた全系での性能評価を行った．シリコンとグラファイトの 2 つの実問題を取り上げ，システム全体の 99.8%に相当する 8192 ノードまでの性能評価を行い，最も重要な TDKS 方程式のハミルトニアン計算について，全体で最高約 4 PFLOPS，ピーク演算性能比 16%を達成した．Weak Scaling をほぼ完璧に達成し，ARTED のスケーラビリティが非常に高いことを確認したが，Strong Scaling ではグラファイトで大きな性能低下が発生した．これについて，我々は計算ノード間で発生しているハミルトニアン計算の約 20%の性能差を発見し，アルゴリズムとは一切関係ない，メニーコアプロセッサによる問題ではないかと考察した．

計算負荷のバランスが取れているのにも関わらず，計算ノード間で性能差が発生するこの問題は，メニーコアプロセッサが世界的に広まった動機の 1 つである消費電力と熱問題に起因するため，Intel Xeon Phi のみならずメニーコアプロセッサ全てに共通して発生しうる問題と考えられる．我々はこの問題について，JCAHPC と引き続き協議・実験を行っていく予定である．

謝辞 本研究における Oakforest-PACS の利用は最先端共同 HPC 基盤施設 (JCAHPC) のご協力による．本研究の一部は文部科学省ポスト「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」(CDMSI)，筑波大学計算科学研究センター平成 29 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」の一環として実施された．本研究の一部は JST-CREST 研究課題「光・電子融合第一原理計算ソフトウェアの開発と応用 (課題番号: JPMJCR16N5)」により支援された．

参考文献

[1] 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).

[2] 廣川 祐太，朴 泰祐，佐藤 駿丞，矢花 一浩：電子動力学コード ARTED による Knights Landing プロセッサの性能評価，情報処理学会研究報告，Vol. 2016-HPC-157, No. 8 (2016).

[3] 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).

[4] 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).

[5] 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, Storage and Analysis*, SC '11, ACM, (online), DOI: 10.1145/2063384.2063386 (2011).

[6] J. Hofmann, J. Treibig, G. Hager and G. Wellein: Comparing the Performance of Different x86 SIMD Instruction Sets for a Medical Imaging Application on Modern Multi- and Manycore Chips, *Proceedings of WPMVP'14*, pp. 57–64 (online), DOI: 10.1145/2568058.2568068 (2014).

[7] C. Andreolli: Eight Optimizations for 3-Dimensional Finite Difference (3DFD) Code with an Isotropic (ISO), available from ⟨https://software.intel.com/en-us/articles/eight-optimizations-for-3-dimensional-finite-difference-3dfd-code-with-an-isotropic-iso⟩.

[8] Intel Intrinsics Guide: available from ⟨software.intel.com/sites/landingpage/IntrinsicsGuide/⟩.

[9] R. Krishnaiyer, E. Kultursay, P. Chawla, S. Preis, A. Zvezdin and H. Saito: Compiler-Based Data Prefetching and Streaming Non-temporal Store Generation for the Intel(R) Xeon Phi(TM) Coprocessor, *Proceedings of the 2013 IEEE 27th International Symposium on IPDPSW*, pp. 1575–1586 (2013).

[10] 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).

[11] TOP500: available from ⟨http://www.top500.org/⟩.

[12] M. S. Birrittella, M. Debbage, R. Huggahalli and *et al.*: Enabling Scalable High-Performance Systems with the Intel Omni-Path Architecture, *in IEEE Micro*, Vol. 36, No. 4, pp. 38–47 (2016).

[13] 最先端共同 HPC 基盤施設：入手先 ⟨http://jcahpc.jp/⟩.

[14] C. Yount and A. Duran: Effective use of large high-bandwidth memory caches in HPC stencil computation via temporal wave-front tiling, *Proceedings of the 7th International Workshop on PMBS'16*, pp. 65–75 (online), DOI: 10.1109/PMBS.2016.12 (2016).

[15] 塙 敏博，中島 研吾，大島 聡史，星野 哲也，伊田明弘：パイプライン型共役勾配法の性能評価，情報処理学会研究報告，Vol. 2016-HPC-157, No. 6 (2016).

[16] M. Malinauskas et.al: Ultrafast laser processing of materials: from science to industry, *Light: Science and Applications*, Vol. 5, p. e16133 (2016).