Vol.2018-HPC-166 No.14
2018/9/28

512bit SIMD環境における分子動力学アプリケーションMODYLASの性能評価

大島 聡史[1,a)] 鈴木 惣一朗[2] 坂下 達哉[3] 荻野 正雄[4] 片桐 孝洋[4] 安藤 嘉倫[5]

概要：分子動力学計算は様々な学問分野において実験にならぶ解析ツールとして広く普及しており、分子の特性を活かした機能性材料や高分子材料の設計において必要不可欠になりつつある。我々は分子動力学アプリケーション MODYLAS の高速化と性能評価を実施している。本稿では最新の 512bit SIMD 環境であるマルチコアプロセッサ Intel Xeon スケーラブルプロセッサ (Skylake-SP アーキテクチャ) およびメニーコアプロセッサ Intel Xeon Phi (Knights Landing アーキテクチャ) にて MODYLAS の主要計算部のひとつである粒子対での Lennard-jones 相互作用と静電相互作用計算の SIMD 化について評価を行った結果を報告する。

1. はじめに

分子動力学 (Molecular Dynamics, MD) 計算は、化学・物理・生物・ウィルス学といった様々な学問分野において、実験とならぶ解析ツールとして広く普及している。さらに工業分野においても、分子の特性を活かした機能性材料や高分子材料を設計する際に MD 計算は必要不可欠になりつつある。分子間・分子内の相互作用には様々なものが存在するが、分子集団系の物性をシミュレーションにより正しく再現するには長距離性のある静電相互作用を精度良く計算せねばならない。しかしながら、原子間の長距離静電相互作用を含めた MD 計算は、「京」コンピュータのような大規模なスーパーコンピュータシステムを用いても 1 千万原子系 (空間サイズとして 50 ナノメートル立方程度) に対して数 100 ナノ秒分程度しか行うことができない。より大規模かつ長時間な MD 計算を行うことは、学術的な価値があるのみならず、より高精度な材料設計を可能にするためにも強く求められている。

一方、ムーアの法則に基づく半導体の高性能化が限界を迎えつつある今日、様々な方法による計算機システムの高性能化が進められている。本研究では Single Instruction Multiple Data(SIMD) 計算、特に SIMD 長の長い (より多くのデータに対して一度に計算可能な)SIMD 命令の活用に注目する。SIMD 命令を用いた高性能計算は従来から多くのマルチコア CPU において行われてきたが、近年では倍精度浮動小数点型データを最大 8 個同時に処理することのできる 512bit SIMD 演算命令に対応したプロセッサが普及しはじめている。しかし、それらのプロセッサの持つ性能を十分に発揮するには、対象プログラムを適切に書き直す必要が生じることもある。また、プロセッサによっては SIMD 長の長い命令を使う際に動作周波数が低下するものも存在し、SIMD 命令に対応したからといってプログラムの演算性能が向上する (実行時間が削減される) かどうかは実際に試してみなければわからない状況になっている。

そこで本稿では、分子動力学アプリケーション MODYLAS[1] を対象アプリケーションとして、512bit SIMD 演算に対応したマルチコア CPU およびメニーコアプロセッサにおける性能評価を実施した結果を報告する。以下、2 章では MODYLAS について述べる。3 章では先行研究において我々が実施した SIMD 演算対応の状況について確認する。4 章では 512bit SIMD 演算命令に対応した計算機環境における性能評価結果を示し、5 章はまとめの章とする。

2. MODYLAS

MODYLAS は任意の分子集合体に対する汎用の古典分子動力学計算プログラムであり、その名称は MOlecular DYnamics software for LArge System に由来する。MODYLAS は分子科学研究所や名古屋大学の研究者らが中心となって開発されてきたソフトウェアであり、Web サイト [2] に登録をすれば誰でもダウンロードすることが可能となっている。なお、本研究では公開されている MODYLAS

1 九州大学 情報基盤研究開発センター
2 理化学研究所 計算科学研究センター
3 玉川大学 工学部
4 名古屋大学 情報基盤センター
5 分子科学研究所
a) ohshima@cc.kyushu-u.ac.jp

ではなく、開発中のバージョンを用いている。MODYLAS は Fortran で記述されており、OpenMP と MPI によって並列化されている。MODYLAS の重要な特徴の一つは、静電相互作用を含む大規模計算をスーパーコンピュータのような超並列計算機環境にて効率よく実行できることを目指して開発されている点にある。特に「京」コンピュータを重要な対象環境の一つとして位置づけており、利用実績も多数存在する [3], [4], [5]。一方、「京」コンピュータは 2019 年中の運用停止がアナウンスされており [6]、後継システム（通称ポスト「京」）についての情報も少しずつ公開されはじめている [7]。

「京」コンピュータは 2012 年 9 月の共用開始から約 6 年が経過しているシステムであり、次世代のポスト「京」ではハードウェア仕様に対する様々な変更が予定されている。特に搭載される CPU を比較した場合、SPARC から ARM へとアーキテクチャや命令セットが更新されることに加えて、計算コア数が 8 コアから 48 コア (+2 アシスタントコア) に大きく増加することや、SIMD 長が 2(128bit SIMD、64bit 倍精度浮動小数点型データを同時に 2 つ計算可能) から 8(512bit SIMD、64bit 倍精度浮動小数点型データを同時に 8 つ計算可能) へと増加することなどが明らかになっている。そのため「京」向けに最適化されたプログラムをポスト「京」上でそのまま (再コンパイルして) 実行した場合、プログラムの並列度不足によりコア数や SIMD 演算器を使い切れず十分な性能が得られないという問題が生じる可能性がある。また、現在広く使われている HPC 向けの計算機環境に目を向けると、Intel 社のマルチコアプロセッサ Intel Xeon スケーラブルプロセッサシリーズ (Skylake-SP アーキテクチャ、以下 SKX と記す) は最大 28 コア 56 スレッド同時実行可能であり 512bit SIMD に対応、同社のメニーコアプロセッサ Intel Xeon Phi シリーズ (Knights Landing アーキテクチャ、以下 KNL と記す) は最大 72 コア 288 スレッド同時実行であり 512bit SIMD に対応するなど、多数の計算コアを搭載し長い SIMD 長の SIMD 演算命令に対応したプロセッサが主流となっている。

そこで我々は、最新のプロセッサや将来のプロセッサにて MODYLAS をさらに活用することを目指し、コア数や SIMD 長の大きなマルチコア計算機環境及びメニーコア計算機環境における MODYLAS の改良と性能評価を行っている。既に「京」コンピュータの商用版後継機といえる FX100 スーパーコンピュータシステム (以下 FX100 と記す) や、(「京」から見ると新しく、本稿執筆時点から見ると旧世代の) メニーコアプロセッサである Intel Xeon Phi (Knights Corner アーキテクチャ、以下 KNC と記す) における性能評価の結果については文献 [8] にて発表済である。本稿では 512bit SIMD 演算命令に対応したより新しい世代の 2 種類のハードウェア、SKX と KNL を対象として性能評価を行った結果を報告する。

本稿および先行研究で用いた対象問題は以下の通りである。3 次元周期境界条件下の立方体状の基本セル (一辺長さ 58.62 Å) に 6510 個の水分子を含み、初期配置における水分子の配置およびその配向はランダムである。水分子の相互作用モデルとしては、CHARMM 力場とともに利用される mTIP3P モデル [9] を採用した。このモデルでは、酸素および水素原子位置に Lennard-Jones(LJ) 相互作用サイトがあり、さらに酸素原子の位置に-0.834e、水素原子位置に+0.417e の点電荷 (e: 電気素量) が割り振られている。水分子ごと、および系全体では電気的中性条件を満たしている。LJ 相互作用については 12 Å でカットオフする一方、静電相互作用については高速多重極展開法 (FMM)[10] により計算した。FMM においては基本セルを各辺 8 分割、合計 512 個のサブセルに分割した上で、多極子展開および局所展開の次数を 4 として計算した。この展開次数の選択は p2p(後述) でのパフォーマンスには影響しない[*1]。サブセルあたりの平均原子数は 40 である。計算ノード数 (=MPI プロセス数) は 8 とした。

3. 先行研究

MODYLAS は利用者によって選択された計算条件および力場のパラメタに基づき MD 計算を行う。MODYLAS の扱う MD 計算の中でも特に長い実行時間を必要とする処理は主に以下の 2 つであることが判明している。

(1) 粒子対での LJ 相互作用と静電相互作用計算 (p2p)
(2) FMM における多極子展開係数から局所展開係数への変換 (M2L)

これらの処理の関係を**図 1** に示す。我々は文献 [8] にて、p2p 処理の性能に着目し新しく 4 つのスレッド並列化手法を開発して性能評価を行った結果を報告した。本章ではその概要を示し、次章にてより新しい世代のマルチコアプロセッサおよびメニーコアプロセッサとの性能比較結果を示す。なお、文献 [8] 及び本稿にて用いた計算機環境の構成は**表 1** の通りである。多くの場合、FX100 と SKX はマルチコアプロセッサ、KNC と KNL はメニーコアプロセッサと称される。

p2p 処理においては **図 2** に示す 5 つの実装手法が開発されてきた。このうち実装 a(cell 内分割) がオリジナルの実装であり、セル内の原子についてのループを OpenMP ループ並列化の対象としていた。また「京」の 128bit SIMD を前提に IF 文の削除や SIMD 化対象ループの伸張といった最適化も施されている。しかし、**表 2** においてセル内分割のスレッド数の上限 (並列化対象ループ長) が 40 となっているように、実装 a(cell 内分割) は多数のスレッドを同時実行できる最新のプロセッサには適していないことがわかった。そこで、スレッド並列化対象のループ長を伸ばしてス

[*1] 次数は p2p 以外の一連の演算 (p2M, M2M, M2L, L2L, L2p) のパフォーマンスに影響する。

表 1　実験環境

本稿における呼称	FX100	KNC	SKX	KNL
プロセッサ	Fujitsu SPARC64XIfx	Intel Xeon Phi P5110	Intel Xeon Gold 6154	Intel Xeon Phi 7150
搭載システム名 および設置場所	Fujitsu PRIMEHPC FX100 名古屋大学 情報基盤センター	実験用クラスタ 東京大学 情報基盤センター	ITO サブシステム A 九州大学 情報基盤研究開発センター	Oakforest-PACS JCAHPC
ノードあたり搭載数	1	1	2	1
動作周波数	2.2 GHz	1.05 GHz	3.0 GHz - 3.7 GHz	1.4 GHz - 1.6 GHz
1 基あたりコア数	32 + 2 †	60	18	68 ‡
1 基あたり HPL 性能	約 1.0 TF	約 1.0 TF	約 1.1 TF	約 1.6 TF
メモリ種別と 1 基あたり容量	HMC 32 GB	GDDR5 8 GB	DDR4 96 GB	MCDRAM 16 GB
1 基あたり STREAM Triad 性能	105 GB/s	140 GB/s	95 GB/s	495 GB/s
ノード間接続	Fujitsu Tofu2 100 Gbps 6 次元メッシュ/トーラス	*	Mellanox InfiniBand EDR 100 Gbps Full Bisection BW Fat Tree	Intel Omni-Path 100 Gbps Full Bisection BW Fat Tree

† 「＋ 2」コアは通信補助用コアであり計算には未使用
‡ 性能評価時には 64 コアのみを使用
* ホストと PCI-Express Gen2 x16 にて接続。ホスト間は Mellanox InfiniBand FDR 56 Gbps * 2 接続。

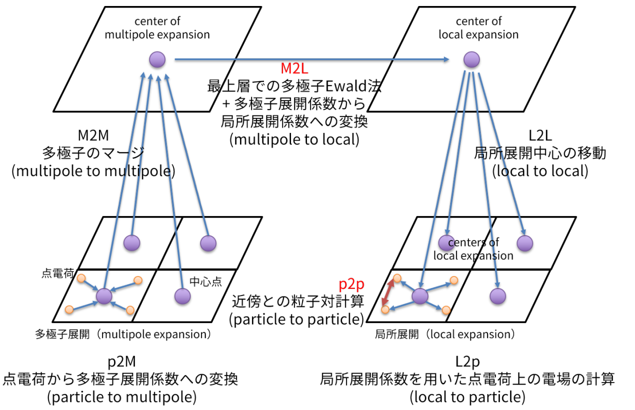

図 1　高速多重極展開法 (FMM) の主要演算のイメージ (赤文字部が並列処理時に高負荷となる処理)

レッド数の多い計算機環境でも十分な性能が得られるようにするとともに、スレッド間の負荷の均衡化や定数計算部分の分離などの改良を行った。文献 [8] では上記の 5 つの実装を FX100 と KNC にて評価した。結果の概要としては、FX100 では実装の改良が効果を発揮し、実装 d(square 単位) や e(column 単位) が良い性能を得た。一方 KNC では実装 b(column 内分割) の性能が最も良く、さらなる改良を加えたはずの実装 c,d,e では性能が低下するという結果となった。KNC の性能とその傾向については、文献 [8] が FX100 を主要な評価対象としていたこと、KNC 上ではプロファイラの機能などに制限があり詳細な性能評価が難しかったこと、KNC から KNL への世代交代によりハードウェアの仕様や性能の特性に大きな変化があったことから、あまり深く調査ができていない。

4. 性能評価

4.1 性能の比較

本章では SKX および KNL にて前章で示した全 5 つの計算カーネルの実行時間を測定し、文献 [8] における FX100 および KNC の実行時間と比較して考察を行う。FX100 および KNC の性能は先行研究において測定された性能値そのものを用いる。各実行環境にて用いたコンパイラおよび MPI と、主な最適化オプション等は以下の通りである。なお、ノード数 (=MPI プロセス数は) 全て 8 であるが、今回は実行時間測定範囲 (p2p) に通信処理が含まれていないため、通信ハードウェアや MPI 処理系の差は実行時間に影響しない。

FX100　Fujitsu TCS 2.0.0, `-Kfast,simd=2,openmp,parallel,ocl`

KNC　Intel コンパイラ 17.0.4, Intel MPI 2017 Update 3, `-qopenmp -O3 -mmic`、native 実行モード

SKX　Intel コンパイラ 18.0.3, Intel MPI 2018 Update 3, `-qopenmp -O3 -xCORE-AVX512 -align array64byte`、1 ノードあたり 1 基のみ使用

KNL　Intel コンパイラ 18.0.1, Intel MPI 2018 Update 1, `-qopenmp -O3 -xMIC-AVX512 -align array64byte`、flat モード（MCDRAM のみ使用）

はじめに全 5 つの計算カーネルの実行時間を SKX および KNL にて測定し、その結果を FX100 および KNC と比較する。対象プログラムのソースコードは先行研究と本稿

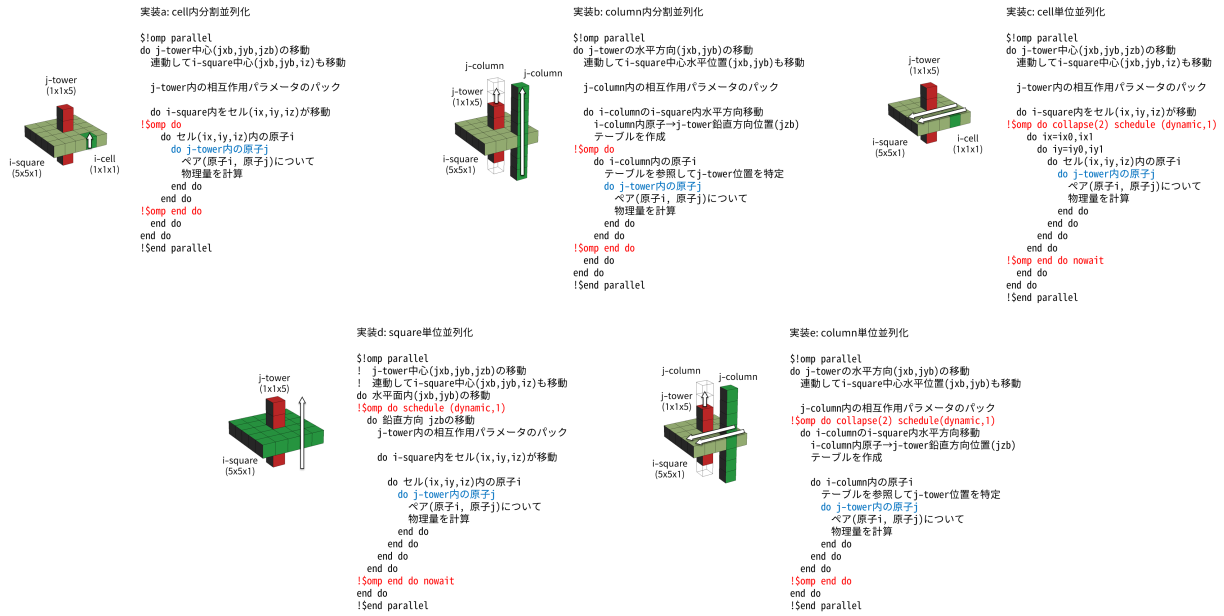

図 2 p2p 処理を行う 5 つの実装 (青文字ループ部が SIMD 化対象)

表 2 p2p 処理における粒度とスレッド数の上限

実装	粒度 (計算原子対数)	スレッド数 N_t の上限 *			
		N×N×N	2×2×2	4×4×4	8×8×8
a: cell 内分割	8000/N_t	40	40	40	40
b: column 内分割	8000N/N_t	40N	80	160	320
c: cell 単位	8000	25N^3	200	1600	12800
d: square 単位	8000 - 200000	$N(N+4)(N+4)$	72	256	1152
e: column 単位	8000N	25N^2	100	400	1600

* 計算領域セル数を N×N×N および具体的な数とした場合に並列化対象ループの長さがいくつになるかの例。いずれもセル内の原子数を 40 とした場合の数を算出。

とで完全に同一ではないが、測定対象となっている部分のコードには性能に大きな影響を与えるような修正は加えられていない。文献 [8] において、32 コア 256bit SIMD の FX100 を対象として、さらにコア数の多い環境や SIMD 長が長い環境での利用も想定して最適化を施してきたため、実装 a(cell 内分割) 以外の実装については SKX や KNL にて良好な性能が得られることが期待される。

対象プログラムは SIMD 化をコンパイラによる判断に任せており、言い換えれば、SIMD 化により高性能が期待できる部分はコンパイラが適切に SIMD 化可能と判断できるようなコーディングを行ってきた。今回、FX100, SKX, KNL におけるコンパイラによる最適化レポートを比較したところ、いずれの環境においても図 2 に示した各 OpenMP ループ並列化の内側の適切なループが SIMD 化対象と判定されていることがほとんどであった。ただし、多くの場合は実行時にプロセス数や入力データによって決まるループの長さ等によって SIMD 実行するかどうかが判断されるため、コンパイルの時点では、実行時に各環境において同一のループが同様に SIMD 実行されるかどうかは判断できない。

性能測定結果を図 **3** から図 **6** に示す。いずれもプロセス番号 0 の実行時間であり、p2p 部を 1000 回程度実行した際の平均値である。結果を比べると、SKX の実行時間は全 18 コア使用まで徐々に減少しており、最短平均実行時間は実装 d(square 単位) の 18 スレッドにおける 1.86 ミリ秒であった。一方 KNL の実行時間は 32 スレッドまたは 64 スレッドの実装 b(column 内分割) が高速であり、HyperThreading を用いた 128 スレッド以上の実行では性能が向上せず、最短平均実行時間は実装 b(column 内分割) の 64 スレッドにおける 3.97 ミリ秒であった。最短実行時間を得たスレッド数と実装は表 **3** となっており、HPL 性能や STREAM 性能では最速の KNL に比べて FX100 や SKX が 2 倍ほど高性能、FX100 と SKX はほぼ同等性能という結果が得られた。それぞれの傾向としては、メニーコアプロセッサである KNC と KNL、マルチコアプロセッサである FX100 と SKX の傾向がそれぞれ似た傾向と言

表 3　最短実行時間と対応する実装

	最短実行時間(ミリ秒)	実装	スレッド数
FX100	1.99	d:square 単位	32
KNC	17.97	b:column 内分割	180
SKX	1.86	d:square 単位	18
KNL	3.99	b:column 内分割	64

える。FX100 と SKX は実装 c(cell 単位)・実装 d(square 単位)・実装 e(column 単位) の 3 実装の実行時間が近く、KNC と KNL は実装 b(column 内分割) 以外において全物理コア以上使用時の性能低下が顕著である。

つづいて、OpenMP 並列化対象ループ部における各スレッドごとの実行時間を比較する。測定方法としては、前述のプログラムにおけるスレッド並列化 (OpenMP ループ並列化) 対象部にてスレッドごとに `omp_get_wtime` 関数によって実行時間を測定する処理を追加し、1MD ステップ内におけるスレッド単位の実行時間を確認できるようにした。今回は第 10MD ステップにおける各スレッドの実行時間を用いて、実行時間の長短とスレッド間の実行時間のばらつきを比較する。

図 7 には SKX のスレッドごとの実行時間を、**図 8** には KNL のスレッドごとの実行時間を示す。SKX の結果に注目すると、実装 a(cell 内分割) はスレッドごとの実行時間のばらつきがやや大きくスレッド番号が小さいほど実行時間が長い傾向にあるのに対して、他の実装はスレッド間の実行時間の差が小さい。いずれのスレッドサイズについても最も実行時間が短い実装 (最終的には全スレッドの計算が終わらねばならないため、最も実行時間が長いスレッドの時間が短い実装ほど高速な実装と言える) は実装 b(column 内分割) であるように見えるが、実装 a(cell 内分割) 以外の実装には大きな差はなく、スレッド処理以外の部分の時間も含めたうえで総合的に速かったのが実装 d(square 実装) であったようである。

一方で KNL の結果に注目すると、まず実装 a(cell 内分割) については OpenMP 並列化ループのループ長が 40 しかないため、64 スレッド以上の実行においてはスレッド番号 40 以降のスレッドが活用できていない。また、128 スレッド以上の実行においては特に実装 b(column 内分割)・実装 e(column 単位) においてスレッドごとの実行時間のばらつきが大きい。ループイタレーションごとの処理に大きな差は生じないはずでありループ長も十分長いため不可解な結果にも見えるが、HyperThreading 実行による資源競合の影響が大きいと考えている。

なお、実装 a(cell 内分割) についても 256 スレッド実行時にばらつきが大きいように見えるが、処理が割り当たっていないスレッド番号における時間のばらつきであり、処理が割り当たっているスレッド番号の実行時間に影響はないためここでは無視する。

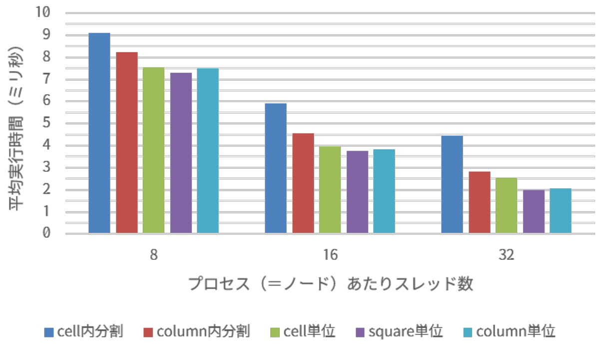

図 3　FX100 における p2p 部の実行時間 (先行研究)

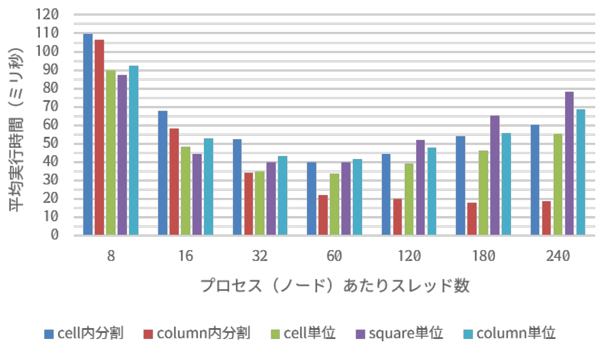

図 4　KNC における p2p 部の実行時間 (先行研究)

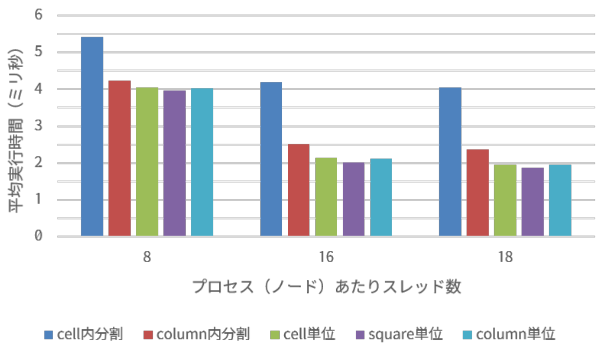

図 5　SKX における p2p 部の実行時間

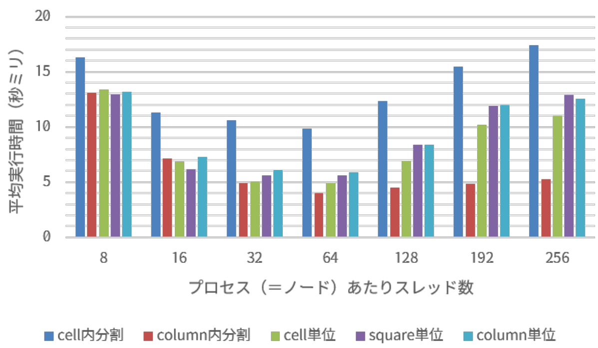

図 6　KNL における p2p 部の実行時間

4.2　SIMD 化状況の詳細確認

本節では SKX と KNL の SIMD 化状況についてより詳細に確認する。

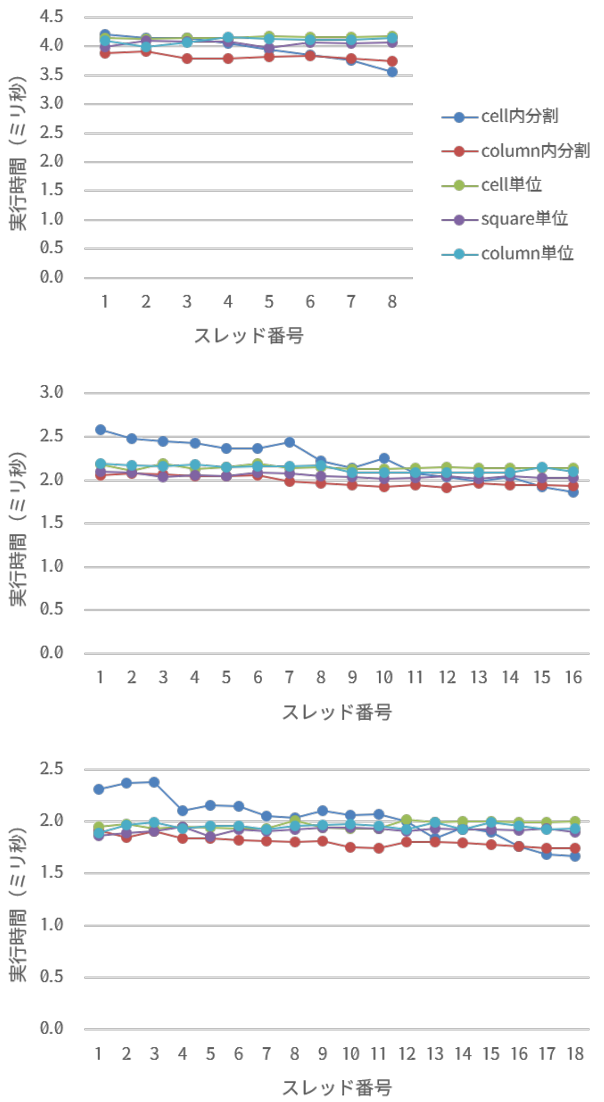

図 7　スレッドごとの実行時間の比較 (SKX)

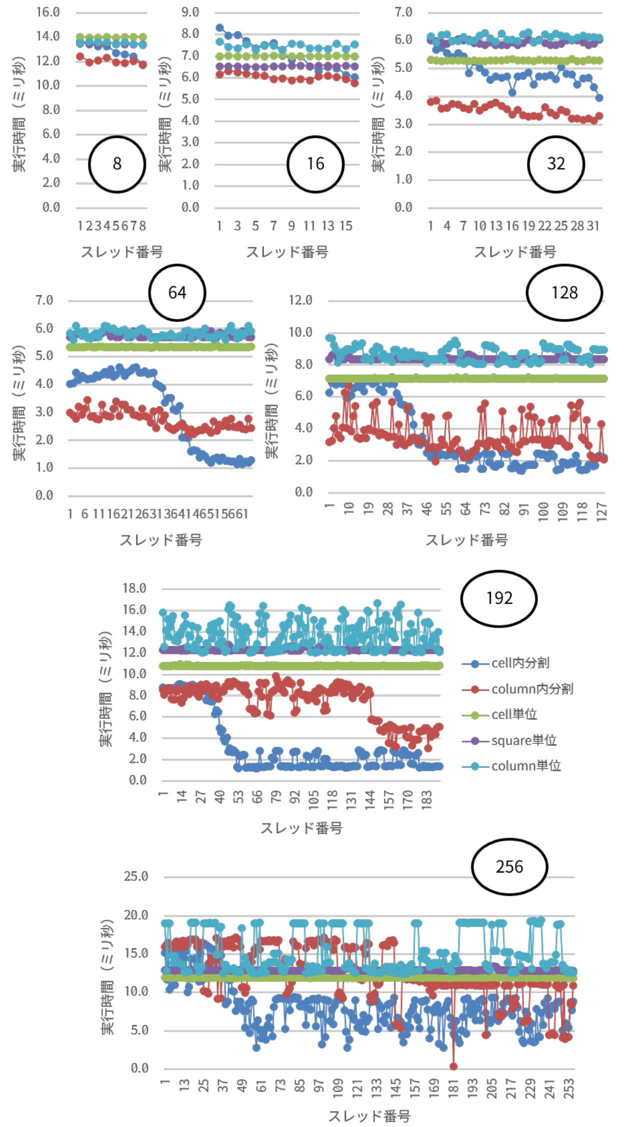

図 8　スレッドごとの実行時間の比較 (KNL):丸付きの数字 (8 から 256) はそれぞれのグラフにおける使用スレッド数

はじめに、各環境において Intel Advisor を用いて `advixe-cl -collect survey` にて実行履歴を取得しベクトル命令の実行状況を確認した。本機能を用いることで、どの SIMD 化ループ部分がどの程度 SIMD 化されて実行されたかを確認することができる。図 9 に SIMD 化率を示す。実装 a(cell 内分割) および実装 b(column 内分割) についてのみ対象部が 2 つずつ (A と B)、他は 1 つずつ (A のみ) であるのは、全 5 実装ともに物理量を計算する際に「(A) 自身のセルを含む場合」と「(B) 自身のセルを含まない場合」であらかじめ分岐した後に SIMD 化ループ部分が存在しており、実装 a(cell 内分割) および実装 b(column 内分割) では A 部と B 部の両方が、他の実装では A 部のみが実際に SIMD 実行されたことによる。これらの結果からは、A 部についてはどの実装も SKX では 75%程度・KNL では 70%弱とある程度高い SIMD 化実行率が得られている一方、B 部については 30%台の低い値が得られていることがわかる。ただし、B 部は A 部と比べて実行される回数が非常に少ないため、B 部が SIMD 実行されていないもしくは SIMD 化率が低いことによる性能への影響は小さい。

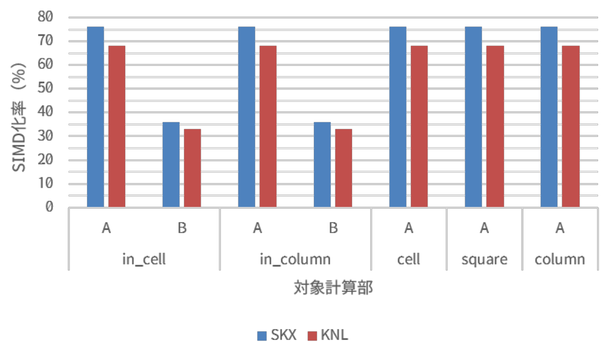

図 9　SIMD 化率の比較

ところで、Advisor の結果からは SKX も KNL も全ての実装について AVX512 による SIMD 化が行われている

ように見えるものの、コンパイル時の最適化レポートによると SKX では ZMM レジスタ (512bit 長の SIMD 計算用レジスタ) を使わない SIMD 化が行われた旨が出力されている。そのため、実際には 512bit SIMD(AVX512) ではなく 256bit SIMD(AVX2) が使用されていると考えられる。Intel コンパイラのマニュアルによると、Intel コンパイラは`-xCORE-AVX512` オプションにて 512bit SIMD 命令が有効化された場合、デフォルト設定では積極的に ZMM レジスタを使用しないという設定になっている。これを変更するには`-xCORE-AVX512` オプションの代わりに`-xCOMMON-AVX512` オプションを使用する、または`-qopt-zmm-usage=high` オプションを付加すれば良い。そこで、`-qopt-zmm-usage=high` を付加した場合の SKX の性能を測定した。

コンパイラオプションにより AVX512 命令の利用を促進した場合の SKX の実行時間を**図 10** に示す。**図 5** と比較すると、いずれも実行時間が 10%前後増加する結果となった。これは ZMM レジスタ (512bit SIMD 演算) を使ってもメリットが得られないであろうというコンパイラによる総合的な判断が正しかったことを意味する。コンパイル時の最適化レポートによると、上記のコンパイラオプションの変更はループ長が短く SIMD 化に適さないことが多いピールループや剰余ループも SIMD 化してしまうため、これが性能に悪影響を及ぼしている可能性も考えられる。(`-CORE-AVX512` の場合には、ピールループや剰余ループに対しては、ベクトル化可能ではあるが非効率であるため行わない旨のレポートが出力される。) そこで、指示文の挿入による部分的な 512bit SIMD 化の促進を試みた。対象は図 9 にて高い SIMD 率を得ていた各実装の (A) のループ部とし、`!$omp simd simdlen(8)` 指示文を加えることで 512bit SIMD 演算命令が生成されるよう促した。しかしながら、この場合も**図 11** に示すように元の結果 (図 5) と比べて実行時間は 10%から 20%程度増加する結果となった。**図 12** は図 5 の実行時間を 1 とした場合の図 10 および図 11 の実行時間の比である。なお、この指示文の追加による AVX512 命令の利用の促進は、今回の性能評価対象部である p2p 部以外の幾つかの処理においては僅かながら性能向上に寄与したことを付け加えておく。ちなみに、`-xCORE-AVX2` オプションにより AVX1/AVX2 環境向けにコンパイルして実行した場合は、`-xCORE-AVX512` オプション指定時 (図 5) よりも全体的に 3%程度実行時間が増加する結果となった。

さらに、PMlib[11] を用いて SIMD 命令実行数の確認を行った。PMlib は理化学研究所にて開発されている性能モニタリングのためのソフトウェアライブラリであり、指定した区間の性能統計情報を簡単に測定し出力する機能を備えている。今回は特に、指定区間内におけるベクトル命令実行回数をレポートする機能を用いて SIMD 化状況の詳細確認を試みた。PMlib は SPARC64 系 (「京」/FX10/FX100) および x86 系 (Xeon) のいくつかの CPU に対応しており、SKX の命令セットについても AVX512 命令を含めて対応しているものの、KNL の命令セットには対応していない。そのため、SKX についてのみ評価を行う。

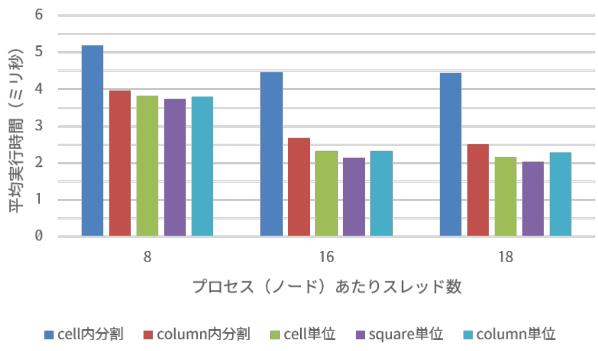

図 10 SKX における p2p 部の実行時間 (-qopt-zmm-usage=high 版)

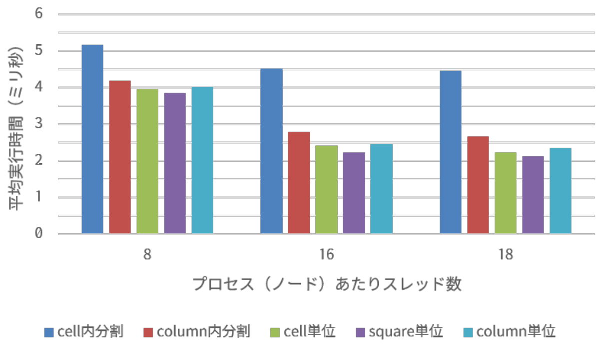

図 11 SKX における p2p 部の実行時間 (SIMD 指示文追加版)

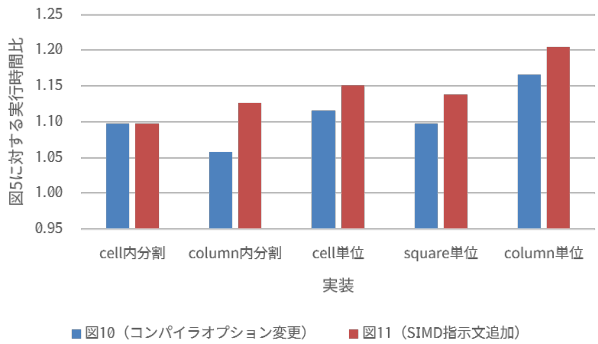

図 12 SKX における p2p 部の実行時間の比較

図 13 には`-xCORE-AVX512` オプションでコンパイルした場合、**図 14** には`-xCORE-AVX512 -qopt-zmm-usage=high` オプションでコンパイルした場合の、AVX 命令種別ごとの実行命令数を積み上げ棒グラフで示す。測定対象は全 5 実装それぞれ 1MD ステップのみとした。`-xCORE-AVX512` オプションのみでコンパイルした場合にはコンパイル時の情報として ZMM レジスタを使わない SIMD 化が行われた旨が出力されたことを述べたが、図 13 を見ると確かに AVX512 命令 (PMlib 上では DP_AVXW と出力される、倍精度浮

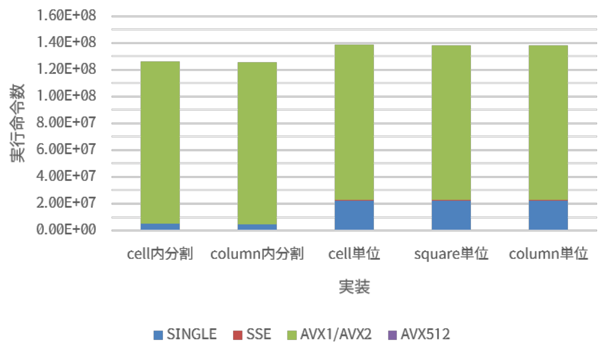

図 13 SKX における p2p 部の SIMD 演算実行命令数 (-xCORE-AVX512)

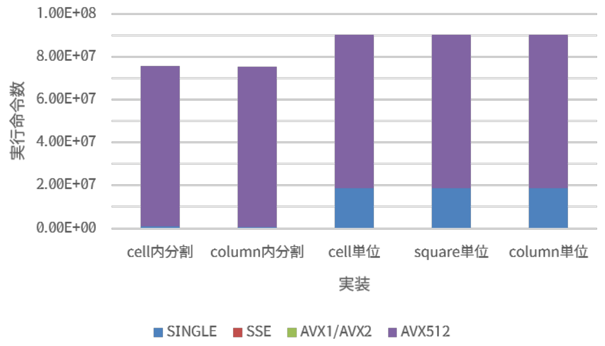

図 14 SKX における p2p 部の SIMD 演算実行命令数 (-xCORE-AVX512 -qopt-zmm-usage=high)

動小数点型データ向けの AVX512 命令) は実行されておらず、AVX1/AVX2 命令 (PMlib 上では DP_AVX と出力される、倍精度浮動小数点型データ向けの AVX1/AVX2 命令) のみが実行されている。一方、ZMM レジスタの利用を促進させるオプションを付加した図 14 では AVX1/AVX2 命令の代わりに AVX512 命令が利用されるようになったことがわかる。図 13 における AVX1/AVX2 と図 14 における AVX512 の実行命令数を比較すると、いずれも約 62%に減少している。理想的にはベクトル長が 2 倍になることで実行命令数は半減することが期待されるが、SINGLE(SIMD 演算以外の倍精度浮動小数点演算) や SSE(64bit SIMD 演算) の命令数も減少していることからこれらの命令も全て AVX512 にて処理されており、非効率な AVX512 演算が多数行われた可能性が高いと考えられる。また SKX では、SIMD 演算を行わないときよりも AVX1/AVX2 命令を実行する際、AVX1/AVX2 命令を実行する際よりも AVX512 命令を実行する際の方が、動作周波数が低下する [12]。今回の性能評価において AVX512 演算命令の利用を促進しても実行時間が短くなっていない理由は、これらの複数の要因が影響したためと考えられる。

5. まとめ

本稿では分子動力学アプリケーション MODYLAS、特にその中でもホットスポットである p2p 処理の 5 種類の実装を対象として、512bit SIMD 環境における SIMD 化の効果を中心とした性能評価を行った。4 章における性能評価結果から、今回の性能評価対象部である MODYLAS の p2p 部については、少なくとも Intel コンパイラが AVX512 の利用について検討できる程度には 512bit SIMD 環境に対応したコードとなっていることが確認できた。p2p 処理の実行時間を比較したところ、HPL ベンチマークや STREAM ベンチマークによる性能スコアが近い FX100(先行研究) と SKX は、ほぼ同じ最短実行時間を示した。この最短実行時間が実装 d(square 単位) にて得られたという点も共通であった。一方 KNL は、ベンチマークスコアは FX100 や SKX を大きく上回り、前世代の KNC と比べて p2p 処理の実行時間を約 78%削減するという大きな性能向上を得た一方、FX100 や SKX と比べると約 2 倍の時間を要した。また最短実行時間を得られた実装は実装 b(column 内分割) であり、KNC(先行研究) と同様であった。

その一方で、SKX においてはコンパイラの判断により 512bit SIMD 命令は用いられず、256bit 以下の SIMD 命令のみが用いられた。指示文を追加することで容易に 512bit SIMD 命令の利用を強制できることが確認できたが、これにより実行時間は増加してしまった。その原因としては、典型的には SIMD 化対象ループのループ長が短いことが理由にあげられやすいが、本稿の場合には、512bit SIMD 命令に適していないと思われる命令までもが 512bit SIMD 命令によって処理されてしまったり、512bit SIMD 命令時に動作周波数が低下するというハードウェアの都合による部分が大きいと考えられる。

以上のように、512bit SIMD 環境における MODYLAS の利用について様々な性能情報を得ることができた一方で、SKX については 256bit SIMD から 512bit SIMD にすることで性能向上が得られておらず、KNL については FX100 や SKX より性能が大きく劣っている。これらの最新の計算機環境においてさらに性能を向上させることができるかについては、512bit SIMD 演算の活用以外の視点も含めてさらに検討したい。その他、MODYLAS の高速化については p2p 以外の部分の高速化、例えば大規模な問題に向けた通信の最適化などにも取り組む余地があると考えている。さらに我々は FDM アプリケーションに対して最新の計算機環境向けの最適化・自動チューニングに取り組んだ実績があり [13], [14]、MODYLAS についても同様に自動チューニングの観点からの最適化を検討したい。

謝辞 MODYLAS は文部科学省ポスト「京」重点課題 5「エネルギーの高効率な創出，変換・貯蔵，利用の新規基盤技術の開発」の成果物であり、提供いただいた名古屋大学岡崎進教授に感謝します。PMlib について助言をいただきました理化学研究所 計算科学研究センター 三上和徳氏と九州大学 情報基盤研究開発センター 小野謙二教授に感謝します。本研究は大規模学際情報基盤共同利用・共

同研究拠点 (JHPCN、課題番号 jh180050-NAJ)、JSPS 科研費 16K21094(若手研究 (B) 非対称な脂質組成をもった脂質二重層膜物性の解明)、JSPS 科研費 16H02823(基盤研究 (B) 通信回避・削減アルゴリズムのための自動チューニング技術の新展開) の助成を受けています。本研究を実施するにあたり、九州大学情報基盤研究開発センターに設置されている ITO および JCAHPC に設置されている Oakforest-PACS を使用させていただきました。

参考文献

[1] Yoshimichi Andoh, Noriyuki Yoshii, Kazushi Fujimoto, Keisuke Mizutani, Hidekazu Kojima, Atsushi Yamada, Susumu Okazaki, Kazutomo Kawaguchi, Hidemi Nagao, Kensuke Iwahashi, Fumiyasu Mizutani, Kazuo Minami, Shin-ichi Ichikawa, Hidemi Komatsu, Shigeru Ishizuki, Yasuhiro Takeda, Masao Fukushima: "MODYLAS: A Highly Parallelized General-Purpose Molecular Dynamics Simulation Program for Large-Scale Systems with Long-Range Forces Calculated by Fast Multipole Method (FMM) and Highly Scalable Fine-Grained New Parallel Processing Algorithms", Journal of Chemical Theory and Computation, Vol.9, Num.7, pp.3201–3209 (2013).

[2] MODYLAS — MOlecular DYnamics software for LArge System `http://www.modylas.org/` (Accessed: 2018.08.23)

[3] Yoshimichi Andoh, Noriyuki Yoshii, Atsushi Yamada, Kazushi Fujimoto, Hiroki Kojima, Keisuke Mizutani, Atsushi Nakagawa, Akio Nomoto, Susumu Okazaki, "All-atom molecular dynamics calculation study of entire poliovirus empty capsids in solution", The Journal of Physical Chemistry, Vol.141, Num.16, p.165101 (2014).

[4] Takuma Yagasaki, Masakazu Matsumoto, Yoshimichi Andoh, Susumu Okazaki, Hideki Tanaka: "Effect of bubble formation on the dissociation of methane hydrate in water: A molecular dynamics study", The Journal of Physical Chemistry B, Vol.118, Num.7, pp.1900–1906 (2014).

[5] Takuma Yagasaki, Masakazu Matsumoto, Yoshimichi Andoh, Susumu Okazaki, Hideki Tanaka, "Dissociation of methane hydrate in aqueous NaCl solutions", The Journal of Physical Chemistry B, Vol.118, Num.40, pp.11797–11804 (2014).

[6] 「京」の運用停止とポスト「京」への移行に関する説明会の開催のお知らせ｜ HPCI `http://www.hpci-office.jp/pages/seminar_180124` (Accessed: 2018.08.23)

[7] ポスト「京」の CPU の仕様を公表 : 富士通 `http://pr.fujitsu.com/jp/news/2018/08/22-1.html` (Accessed: 2018.08.23)

[8] Yoshimichi Andoh, Soichiro Suzuki, Satoshi Ohshima, Tatsuya Sakashita, Masao Ogino, Takahiro Katagiri, Noriyuki Yoshii, Susumu Okazaki: "A thread-level parallelization of pairwise additive potential and force calculations suitable for current many-core architectures", The Journal of Supercomputing, Vol.74, pp.2449–2469 (2018).

[9] Stewart R. Durell, Bernard R. Brooks, Arieh Ben-Naim: "Solvent-Induced Forces between Two Hydrophilic Groups", The Journal of Physical Chemistry, Vol.98, Num.8, pp.2198–2202 (1994).

[10] Leslie Frederick Greengard: "The rapid evaluation of potential fields in particle systems", Doctoral Dissertation, Yale University (1987).

[11] 三上 和徳, 小野 謙二: "PMlib を用いた計算性能測定と性能可視化手法", 情報処理学会 研究報告 2018-HPC-163, pp.1–5 (2018).

[12] Intel®Xeon®Processor Scalable Family Specification Update, Feburuary 2018 `https://www.intel.com/content/www/us/en/processors/xeon/scalable/xeon-scalable-spec-update.html` (Accessed: 2018.08.23)

[13] Takahiro Katagiri, Satoshi Ohshima, Masaharu Matsumoto: "Auto-tuning on NUMA and Many-core Environments with an FDM code", Proc. of IEEE IPDPSW2017, pp.1399–1407 (2017).

[14] Takahiro Katagiri: "Auto-tuning for The Era of Relatively High Bandwidth Memory Architectures: A Discussion Based on an FDM Application", Proc. of IEEE IPDPSW2018, pp.1084–1092 (2018).