

SIMD 演算有効利用のための入力データ並列 2 電子フォック行列計算

本田 宏明^{1,4,a)} 稲富 雄一^{2,4,b)} 真木 淳^{3,c)}

概要: 既存のハートリーフォック分子軌道法プログラムでは計算ボトルネック部分である 2 電子フォック (G 行列) 計算において SIMD 演算を有効に利用することは難しい。そのため、分子構造の異なる複数の入力データを同時に計算するアレイジョブ型の計算を SIMD 演算に割り当てた場合の SIMD 演算の有効利用性についての検証を行なった。テスト計算としての複数の同一入力による計算から、(ss,ss), (ps,ss), (ps,ps) の各積分タイプ毎の G 行列計算に対し 2 ウェイの倍精度浮動小数点 SIMD 演算が可能な Intel Xeon X5650 では最大 1.75 倍の速度向上が、4 ウェイの SIMD 演算が可能な Intel Xeon E5-2650 では最大 2.5 倍の速度向上が可能であるとの結果を得た。

1. はじめに

理論化学分野、とりわけ *ab initio* 分子軌道法に基づく量子化学計算の分野は近年の計算機環境の著しい発展とあいまって、現在に至るまで着実に進歩しており、分子の電子状態の計算のみならず、その成果の無機、有機、生化学や創薬などの広汎な分野への応用が進んでいる。しかしながら、生体系など現在望まれている大規模分子系への分子軌道法の適用には依然として種々の解決すべき問題が残されている。その一つが計算に必要なフォック行列要素の効率的な生成法の開拓である。フォック行列の計算アルゴリズムについては、高速な計算機を利用しての大規模分子系の計算に向いている、積分駆動型アルゴリズムが既に確立されており [1]、多くの量子化学計算にて実際に適用されている。しかしながら計算対象の分子系がますます大きくなっている現在では、更なるフォック行列計算の高速化が必要であり、ソフトウェアアルゴリズムのみならず、ハードウェアの特徴を活かした効率の良いアルゴリズムの開拓が急務である。

現在、京コンピュータを越えた、次世代のエクサスケールに至るコンピュータシステムやソフトウェア技術、それに関わる周辺技術を扱った文献は急増しており、多くの研究においてエクサスケールコンピュータ実現を到達目標

とした周辺技術の研究が行われている。有名な技術報告としては米国 DARPA による “Exascale Computing Study” [2] や、国内では「今後の HPCI に関する技術開発に関わる報告書」[3] などがあり、今後のエクサスケールに至るための技術課題やその解決法について様々な議論が展開されている。その多くの研究では、今後、アクセラレーターを含むメニーコアプロセッサを搭載した数十万ノードに達する大規模並列コンピュータが開発されるのでは、と予想されている。

このような大規模並列コンピュータに対して分子軌道法では、フラグメント分子軌道法等の方法の適用により現在のところ MPI や OpenMP によるハイブリッド並列を利用した 2 万並列に達するスケーラブルな計算が可能となっている [4]。

しかしながら、メニーコアプロセッサへの対処については、GPGPU による研究が進んでいるものの、依然として十分に進んでいるとは言えないのが実情である。メニーコアプロセッサについては、高性能低消費電力化の要請から現在のマルチコアプロセッサに比較し、コアについては制御回路を簡素化する方向に、多数のコア間についてはネットワーク型通信方法の方向に向うと考えられており、以下のような主要な特徴を持つと予想されている。1. 100 コア以上のメニーコア構成、2. プロセッサあたりの主記憶量の低減ならびに主記憶に対するコアあたりの提供メモリバンド幅の低下、3. 命令処理におけるインオーダー実行、4. バス方式からパケットスイッチネットワーク通信型へのコア間通信方式の変更、5. 16 ウェイ以上の SIMD 演算器。これらの特徴を考慮した有効なプログラム実装が必要になる

¹ 九州大学 情報基盤研究開発センター

² 九州大学 システム情報科学研究院

³ 九州先端科学技術研究所

⁴ JST-CREST

a) honda.hiroaki.971@m.kyushu-u.ac.jp

b) inadomi@soc.ait.kyushu-u.ac.jp

c) maki@isit.or.jp

といえる。

これら多くの特徴に対し、本研究では多数のウェイ数を持つ並列度の高い SIMD 演算器の有効利用に着目した。後述するが、本研究が対象とする制限的ハートリーフォック (RHF) 分子軌道法のボトルネックとなる 2 電子フォック行列 (G 行列) 計算では全演算における SIMD 演算割合が 1% 未満と低く、今後例えば 16 ウェイの演算器を持つプロセッサが開発された場合、1/16 程度の実行性能となってしまうことが危惧される。

これに対し本研究では、複数の入力からなるデータを SIMD 演算器に割り当て同時に計算することで分子軌道法計算においても SIMD 演算資源を有効活用出来無いか?との観点にて新規の実装ならびに性能評価を行なった。分子軌道法では、分子の解離過程や反応過程の際に考慮が必要な多数の分子構造に対し、構造以外の全ての計算パラメータについて同じ計算を行なうことが多く、入力データ並列計算が実際に有効利用可能である。

本研究ではまず、複数の同一入力を与えた場合には有効な計算が可能か?について調査し、SIMD 計算利用を行った場合の性能の最大となる結果を求める。また、今回は通常行なわれている MPI や OpenMP 並列化を適用せず、逐次実行でのテスト計算を行なう。

2 章では分子軌道法プログラム全体構成ならびにボトルネックとなる G 行列計算構成、既存アルゴリズムにおける SIMD 演算が非効率となる原因について述べる。3 章では入力データ並列計算について、分子軌道法全体計算での取り扱いならびに G 行列計算における SIMD 並列計算とするための実装の詳細について述べる。4 章ではその性能評価について、5 章ではまとめと今後の課題を述べる。

2. 既存分子軌道法計算と SIMD 演算効率

ここでは RHF 分子軌道法計算手法のアルゴリズムならびに既存手法での SIMD 化効率について説明する。

2.1 分子軌道法計算の全体アルゴリズム

図 1 に RHF 分子軌道法計算全体のアルゴリズムを示す。最初に入力の後、1 電子フォック計算ならびに初期密度行列計算を行なう。次に得られた密度行列を用い G 行列計算を行い、フォック行列対角化を通して新規の密度行列計算を求め、G 行列計算の入力としての密度行列と新規の行列が収束しほぼ同じ結果となるまで繰り返し計算を行う。収束後に得られた密度行列計算から種々の物理量の計算を行なう。

G 行列計算部分が全体計算におけるボトルネックであることが知られており、大きな分子ではその計算実行時間において 99% 以上になることもある。

2.2 G 行列計算

図 2 にボトルネックとなる G 行列計算部分のアルゴリズムを示す。本研究では計算に積分駆動型アルゴリズム [1] を採用した。全体として 8 重ループ構造をとっており、上位 4 重ループのループ長はペプチド分子等の比較的大きな分子にてそれぞれ数百を越える。下位 4 重ループは上位ループ変数の値に応じてループ長が異なり、それぞれ 1~6 の値をとる場合が多い。8 重ループ最下層のループボディ部において原始電子反発積分といわれる量が計算される。本研究では小原積分法 [6] にて計算を行なった。この小原積分法では積分を初期項と漸化項に分け計算する。初期項では 8 重ループの各ループ変数に依存して決定される値 T を計算し、この T 値により条件分岐による不完全ガンマ関数の計算を行う。 T がある閾値より小さい場合にはテーブル参照による Taylor 展開の方法で、 T が閾値より大きい場合には平方逆数計算を含む解析的な方法により計算する。漸化式部分については I,J,K,L の組み合わせにより異なった種類の計算を行う。計算に使用する原子のガウス型関数の最大角運動量により、漸化計算の種類 (積分タイプ) 数が異なり、角運動量 $L = 0, 1, 2, 3 \dots (s, p, d, f \dots \text{関数})$ まで考慮する際、それぞれ 1, 6, 21, 55, $\dots$ 種類の漸化計算が選択され、計算される。

また、計 8 重ループにもなる大きな計算のため、不要な計算を削減するため、図中の (A),(B),(C) の位置にてループ内の計算寄与が閾値より小さいと判断される場合にはその計算がスキップされる。

2.3 本研究での対象プログラム

本研究における RHF 法プログラムについては、九大の稲富らが中心になり開発している超並列フラグメント分子軌道法プログラム (OpenFMO) のプロジェクト [8] の一部にて開発された、全て C 言語にて記述されたプログラムをもとにしている。本バージョンでは、GAMESS プログラム [5] と同様に、単一の G 行列プログラムにて複数の積分タイプの計算が可能とする実装を利用した。その際に予め計算で使用するシェルの出現順序を変更することで G 行列計算にて行う電子反発積分の積分タイプが計算中にランダムに出現しないよう工夫している。

2.4 G 行列計算と SIMD 演算効率

上記に説明した通常の計算アルゴリズムは GAMESS 等の多くのプログラムにて使用されているが、その SIMD 演算効率は低い。下位 4 重ループ長を全て即値にて 3 と固定した場合の $(\text{H}_2\text{O})_8$ 分子の (ps, ps) 型積分タイプのみからなる G 行列テスト計算を行った。Intel Xeon X5650 プロセッサ、Intel Compiler 13.0 の O3 最適化にて Perfmon2 ライブラリ [7] によるハードウェアカウンタの取得結果から、SIMD 化された浮動小数点演算数割合は 1% 未満で

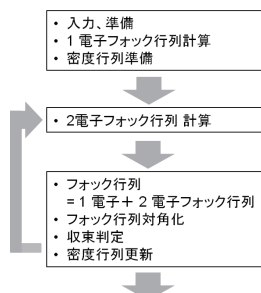


図 1 分子軌道法プログラム主要構成

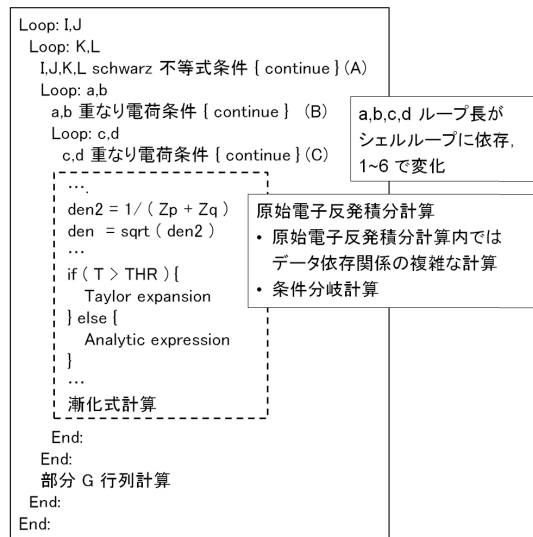


図 2 2 電子フォック行列プログラム

あった。

一般に、コンパイラの自動並列化機能を利用してコード中の多重ループ箇所の SIMD 化オブジェクトコード生成を行うには、最内ループが容易にループアンローリング可能であることならびにループボディ部にデータ依存性が少ない必要がある。しかしながら、上記の一般的な G 行列計算コードでは、1. 下位 4 重ループ長が上位 4 重ループの変数 I,J,K,L に依存し、全て実行時に決定される、2. 原始積分計算からなるループボディ部に複雑なデータ依存性があり、条件分岐も存在する、といった性質があり、効率的な SIMD 計算が妨げられると考えられる。

2.5 アクセラレーターによる既存並列計算

G 行列計算をハードウェアサポートによりアクセラレーター加速実行する方法として、これまで ERIC 2 電子積分計算専用プロセッサによる計算 [9] のほか、Cell プロセッサ [10] や GRAPE-DR [11] によるテスト計算、GPGPU による計算が複数報告されている [13]-[16]。特に最近の GPGPU による研究結果では、数倍 ~ 10 倍程度的高速化が達成されている [16]。GPU の計算では上位 4 重ループが互いに計算依存性がないことに着目することでスレッド並列化が行なわれている。

3. 入力データ並列計算コード変更

次に本研究にて提案する分子軌道法計算のための入力データ並列のためのプログラム実装について説明する。

3.1 全体プログラム

前節 2.1 にて説明したが、プログラム全体において、ボトルネック部分は G 行列計算部分のみである。そのため、入力データ並列計算を SIMD 演算に割り当て計算するのは G 行列計算部分のみとした。図 3 に全体プログラムにおける変更点を示す。

1 電子フォック行列、2 電子フォック行列 (G 行列)、密度行列等、プログラム全体を通して利用するデータは全て異なる入力データ毎に離散的な配置とした。このようにすることで SIMD 化計算を行なう G 行列計算内部以外のコードでは既存のプログラムを再利用可能とする事が可能である。G 行列計算内部では、計算の主要な入力である密度行列を、異なる入力データ毎に離散的な配置から、行列要素の異なる入力に対応するデータを連続配置とするよう、並び換えを行う。G 行列の計算後にはこの計算の出力に対応する G 行列について、行列要素毎に異なる入力に対応した連続配置のデータを入力データ毎の離散的な配置へと並び替えを行う。この G 行列により続く計算について既存コードをほぼそのまま利用可能となる。

3.2 G 行列計算プログラム

G 行列計算内部では、入力データ並列の計算に対しても分子構造パラメータ以外の入力とは同一であるため、制御構造はおおまかにはオリジナルプログラムと同様である。しかしながら、図 2 に示した、(A),(B),(C) の 3 箇所の条件において、分子構造の異なる入力に依存して一般に真になる条件が異なる。この場合では、複数の入力データに対応する条件が全て真になる場合のみループ内部の計算を省略可能となる。

G 行列計算内部での 8 重ループ内ループボディ部におけるプログラム変更点を図 4 に示す。前節で説明したように小原積分法を実装しているループボディ部は 8 重ループのそれぞれのループ添字に依存した T ならびに種々のデータの計算を行う箇所とその T の値による条件分岐部分、続く漸化計算部分がある。条件分岐以外の箇所については各文毎に個々の入力に対応する連続データについてループ計算を行う。その際にはループ長に対応する入力データ個数を即値により指定する。この記述を通して、オリジナルの原始電子反発積分の各文が入力データ並列に対するコンパイラの自動 SIMD 化対象コードとなる。条件分岐箇所については各入力毎に通常のループ計算コードを記述する。

この複数の入力データに対する文毎のループについては、簡単のため、Intel Compiler の拡張仕様による配列表

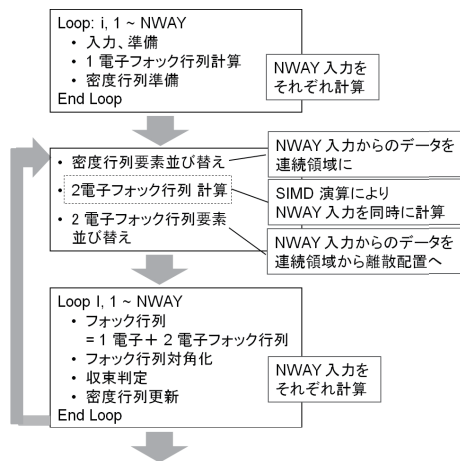


図 3 入力データ並列分子軌道法プログラム

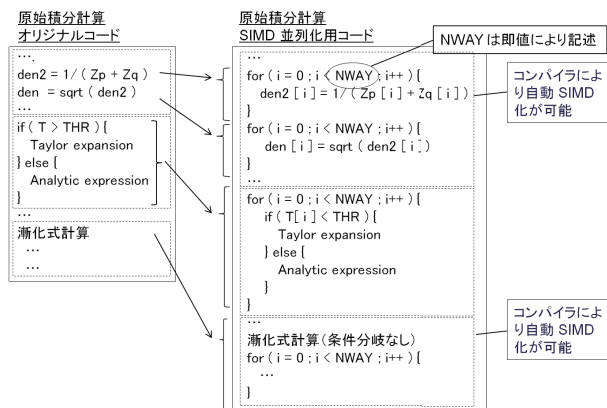


図 4 入力データ並列原始電子反発積分プログラム

記によっても記述可能である。

4. 性能評価実験

4.1 性能評価測定環境

本研究では SSE4.2 命令の 2 ウェイの倍精度浮動小数点計算が可能な Xeon X5650 (2.67GHz, 6 コア) プロセッサならびに AVX 命令の 4 ウェイの倍精度浮動小数点計算が可能な Xeon E5-2650 (2.00GHz, 16 コア) デスクトップパソコンの 2 種類の計算機環境を利用した。コンパイラとして、Intel Compiler 13.0.1 を利用し、両計算機環境における O3 最適化の他、Xeon E5-2650 環境では AVX オプション最適化を追加した。

計算の入力としては、複数の入力データを全て同一とし、 $(\text{H}_2\text{O})_8$ 分子、6-31G 基底関数系を使用した。この入力では図 2 の上位 4 重の各ループ長が最大 104 となり、下位 4 重ループ長は 1~6 にて実行時に変動する。

性能測定方法として、2 電子フォック箇所の実行時間測定ならびに、Perfmon2 ライブラリ [7] によるハードウェアカウンタデータ取得ライブラリを利用した。

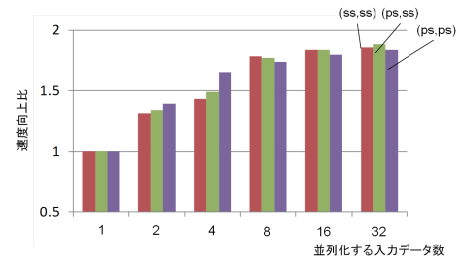


図 5 G 行列計算の入力データ並列度あたりの速度向上比 (2 ウェイ SIMD 演算可)

4.2 入力データ並列計算の速度向上比

逐次実行計算における N 個の同一入力データを処理する計算では、1 個の入力の通常計算に比較して、 N 倍の実行時間が必要であるが、SIMD 演算資源を利用した並列計算が可能である場合には実行時間が減少する。これに対し、本研究では以下の速度向上比を利用し評価を行う。

$$\text{Ratio} = \frac{\text{Exec.time}(1) \times N}{\text{Exec.time}(N)} \quad (1)$$

ここで $\text{Exec.time}(1)$, $\text{Exec.time}(N)$ はそれぞれ 1 入力ならびに N 入力における実行時間を示す。2 ウェイの倍精度 SIMD 演算器が利用可能なプロセッサにて、2 入力の計算を行なった場合、もしも SIMD 演算により 2 倍の実行速度にて計算可能であれば、速度向上比は 2 となる。このように、上式で定義した速度向上比では最大で SIMD 演算器のウェイト数の値を持つ。

4.3 積分タイプ毎の G 行列計算性能評価

G 行列計算の入力データ並列度あたりの速度向上比について、図 5 と図 6 に倍精度 2 ウェイ SIMD 演算器が利用可能な Xeon X5650 と 4 ウェイ SIMD 演算器が利用可能な Xeon E5-2650 による計算結果をそれぞれ示す。

Xeon X5650 における計算では、並列入力データ数が増加するに従い、 $(ss, ss) \sim (ps, ps)$ のそれぞれについて、速度向上比が増加してゆき、8 並列にて 1.75 倍程度に達する。一方 Xeon E5-2650 では、X5650 と同様に、並列入力データ数が増加するに従い増加し、8 並列以降では (ss, ss) 2 倍程度、 (ps, ps) では 2.5 倍程度となっており、両プロセッサにおいて高速化が達成された。しかしながら、2 ウェイの演算器の利用に対しては十分に性能向上が果されたといえるが、4 ウェイでの結果については 2 ウェイの向上比に比較すると高速化の度合いが低い結果となった。

これらの結果に対し、全浮動小数点計算数あたりの SIMD 演算数割合についてのハードウェアカウンタ測定結果を図 7 ならびに図 8 に示す。図 7 から、Xeon X5650 では、入力データ並列度が 1 にて SIMD 演算数比がほぼ 0 であったのに対し、データ並列 2 ならびに 4 では 0.6~0.7 に達している。さらに 8 以降ではほぼ 1.0 と、100% 近くの浮動小数点演算が SIMD 化されている。一方、図 8 から、2 ウェ

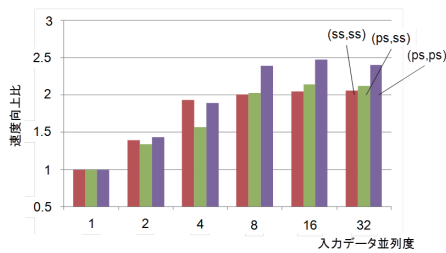


図 6 G 行列計算の入力データ並列度あたりの速度向上比 (4 ウェイ SIMD 演算可)

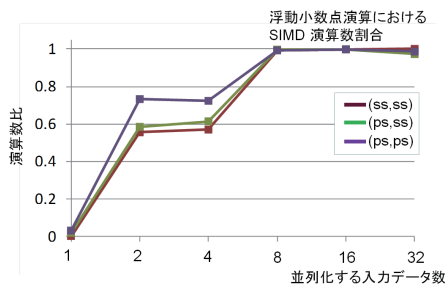


図 7 全浮動小数点演算における SIMD 演算数比 (Xeon X5650, 2 ウェイ SIMD 演算可)

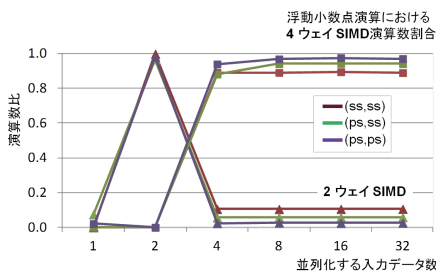


図 8 全浮動小数点演算における SIMD 演算数比 (Xeon E5-2650, 4 ウェイ SIMD 演算可)

い、4 ウェイの 2 種類の倍精度浮動小数点 SIMD 演算が可能な Xeon E5-2650 の結果では、入力データ並列度が 1 にて SIMD 演算数比が 0.1 以下であったのに対し、データ並列度が 2 の場合では 2 ウェイ SIMD 演算比が 0.98 となった。つぎにデータ並列度が 4 の場合では 2 ウェイ SIMD 演算が最大 0.1, 4 ウェイ SIMD 演算が最大 0.95 と、ほぼ 4 ウェイ SIMD 演算での計算がなされている。入力データ並列度 8 以上では、(ss,ss), (ps,ss), (ps,ps) のそれぞれの計算に対し 2 ウェイと 4 ウェイについて 0.1 と 0.9, 0.08 と 0.92, 0.03 と 0.97 の結果であり、ほぼ一定である。X5658 と E5-2650 との比較から、データ並列度 8 以上では X5658 の 2 ウェイ演算数比がほぼ 1.0 であったのに対し、E5-2650 における 4 ウェイ演算数比では最大 0.97 程度となっていた。

図 9 と図 10 にそれぞれの計算の全演算あたりの浮動小数点演算割合を示す。X5650, E5-2650 の両計算ともにデータ並列度が増加するに従い、徐々に減少しているが、入力データ並列度 8 以降では 0.25 程度であり、若干の増加傾

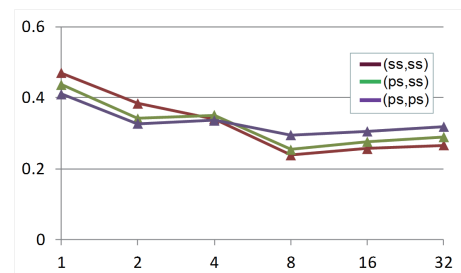


図 9 全演算における浮動小数点演算数比 (Xeon X5650, 2 ウェイ SIMD 演算可)

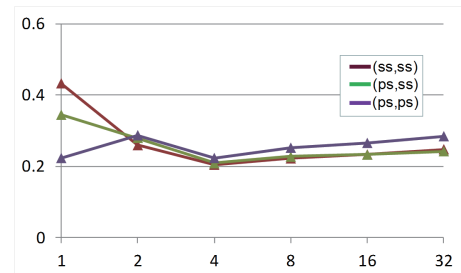


図 10 全演算における浮動小数点演算数比 (Xeon E5-2650, 4 ウェイ SIMD 演算可)

向となっている。この 0.25 程度の浮動小数点演算数比により SIMD 演算による入力データ並列計算の性能向上が抑えられていると考えている。

入力データ並列 8 以上にて、両者の全演算における浮動小数点演算比がおおよそ同じであるが、その一方で全浮動小数点演算数における SIMD 演算数比については、E5-2650 の 4 ウェイ演算比の方が X5650 における 2 ウェイ演算比より若干小さく、2 ウェイ演算も混在している。これが X5650 に比較し E5-2650 において使用可能な SIMD ウェイ数に対し低い性能となる原因の一つではないかと考えられる。

5. まとめと今後の課題

本論文では、分子軌道法の制限ハートリーフォック計算において、効率的な SIMD 計算を行なうことを目的とし、複数の入力データ並列計算を SIMD 演算に割り当て計算する試みを行なった。その際に、同一入力を用い、SIMD 演算が最も効率良く実行可能な条件にて結果を調べた。その結果、2 ウェイ SIMD 演算が可能な Intel Xeon X5650 では最大 1.75 倍、4 ウェイ演算が可能な Intel Xeon E5-2650 では最大 2.5 倍の速度向上との結果が得られた。実際の計算では入力毎に異なる計算により計算が省略されるが、これを SIMD 演算では考慮出来ず余分な計算が必要となるため性能が低下するが、効率的な演算が可能であることが期待される。

現在 Intel の Xeon Phi や Haswell プロセッサにて、より多数のウェー数を持つ SIMD 演算も利用可能であり、本研究の有効性についてのさらなる検証が必要である。ま

た、実際の分子軌道法計算では MPI や OpenMP によるノード内外のハイブリッド並列計算が行われる。今回の入力データ並列計算は、この通常のハイブリッド並列計算とは独立であるため両立が可能である。しかしながら、同一プロセッサ内にてハイブリッド並列計算を行なう場合には逐次計算とは異なる共有キャッシュ利用状況となるため、通常の分子軌道法ではそれほど顕在化しないバンド幅の問題が発生する可能性がある。今後はハイブリッド並列との計算とも組み合わせ、入力データ並列を SIMD 演算計算に割り当てる方法の検証をすすめる必要がある。また、入力データ並列を押し進めることで、プロセッサオンチップメモリでの使用メモリ量がデータ並列数に比例して増加してしまい、将来のコア当りのメモリ量減少の問題に直面する問題についても危惧される。これに対しては、分子軌道法独自の省メモリアルゴリズムである RT 並列化法 [17] の同時利用についての検討も必要であると考えている。

謝辞

本研究の一部は JST-CREST の研究課題「省メモリ技術と動的最適化技術によるスケーラブル通信ライブラリの開発」による。

参考文献

- [1] J. Almlöf, K. Faegri, Jr., and K. Korsell, Principles for a Direct SCF Approach to LCAO-MO Ab Initio Calculations, *J. Comput. Phys.*, Vol.3, pp.385-399 (1982).
- [2] P.Kogge *et al.*, ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, September 28, [Online].
- [3] SDHPC: 今後の HPCI 技術開発に関する報告書, [Online], Available: <http://www.open-supercomputer.org/workshop/sdhpc/>.
- [4] 稲富雄一 他, 京コンピュータでの効率的な動作を目指した並列 FMO プログラム OpenFMO の高性能化, *J.Comp.Chem.Japan*, Vol.12, pp.145-155. 2013.
- [5] Gordon Research Group: The General Atomic and Molecular Electronic Structure System (GAMESS), Iowa State University [online], Available <http://www.msg.chem.iastate.edu/gamess/>.
- [6] S. Obara *et al.*, Efficient recursive computation of molecular integrals over Cartesian Gaussian functions, *J.Chem.Phys.*, Vol.84, pp.3963-3974, 1986.
- [7] Perfmon2, [online] Available <http://perfmon2.sourceforge.net/>.
- [8] OpenFMO, 九州大学, [online], Available <http://www.openfmo.org/OpenFMO/>.
- [9] K. Nakamura *et al.*, Eric: A Special-Purpose Processor for ERI Calculations in Quantum Chemistry Applications, HPC-Asia,2002, 2002.12.
K. Nakamura *et al.*, A HighPerformance, LowPower Chip Multiprocessor for Large Scale Molecular Orbital Calculation, Workshop on unique chips and systems, 2005.03.
- [10] 林徹生 他, Cell プロセッサへの分子軌道法プログラムの実装と評価, 情報処理学会研究報告, HPC, 2006(87), 103-108, 2006-07-31.
- [11] J. Makino *et al.*, GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing, Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, 2007. SC '07. pp.1-11, 2007.
- [12] K. Yasuda, Two-electron integral evaluation on the graphics processor unit, *Journal of Computational Chemistry*, Vol. 29, No. 3, pp. 334-342 (2008).
K. Yasuda, Accelerating Density Functional Calculations with Graphics Processing Unit, *J.Chem.Theor.Comp.*, Vol.4, pp.1230-1236 (2008).
- [13] I.S. Ufimtsev *et al.*, Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics, *J.Chem.Theor.Comp.*, Vol. 5, No. 10, pp. 2619-2628 (2009).
- [14] A. Asadchev *et al.*, Uncontracted Rys Quadrature Implementation of up to G Functions on Graphical Processing Units, *J.Chem.Theo.Comp.*, Vol. 6, No. 3, pp. 696-704 (2010).
- [15] 成瀬彰, 分子軌道法プログラムの GPGPU 化, サイエンスフィックスシステム研究会アクセラレータ技術 WG 成果報告 (2012).
- [16] 梅田宏明 他, 分子軌道法の GPGPU 化に向けた行列加算手法の提案, 情報処理学会研究報告, Vol.2013-HPC-138 No.19, 2013.
梅田宏明 他, フラグメント分子軌道法に現われる Fock 行列計算の GPGPU 化, 情報処理学会論文誌 (採録済み).
- [17] H.Takashima *et al.*, A Novel Parallel Algorithm for Large-Scale Fock Matrix Construction with Small Locally Distributed Memory Architectures: RT Parallel Algorithm, *J.Comput.Chem.*, Vol.23, pp.1337-1346, 2002.