様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　　元 年　　５ 月 ２５ 日現在

機関番号：６３９０３
研究種目：基盤研究(C)（一般）
研究期間：2016～2018
課題番号：１６Ｋ００１７５

研究課題名（和文）大規模並列量子化学計算オープンソースソフトウェアSMASHの高性能化と汎用化

研究課題名（英文）Performance enhancement and generalization of massively parallel quantum chemistry calculation open source software SMASH

研究代表者

石村　和也（Ishimura, Kazuya）

分子科学研究所・理論・計算分子科学研究領域・特任研究員

研究者番号：８０３９０６８１

交付決定額（研究期間全体）：（直接経費）　　3,200,000円

研究成果の概要（和文）：大規模並列量子化学計算オープンソースソフトウェアSMASHのさらなる高性能化及び汎用化を目指して、メニーコアマシン、スーパーコンピュータ向けの新規アルゴリズム開発と実装を行い、並列化効率向上、メモリ使用量削減、通信時間削減を達成した。さらにデータ及びループ構造を改良し、プログラム内のデータの流れを明確にしてPythonによる各計算ステップの制御、組み合わせを容易にした。触媒反応における貴金属削減のための金属ナノ粒子の安定構造の解析、周期境界条件下のあわらに電子-電荷相互作用を取り扱ったQM/MM計算アルゴリズムの開発も行った。

研究成果の学術的意義や社会的意義
本研究で開発した並列化効率の高いプログラムは、演算量及びデータ量ともに膨大となる量子化学計算を高速に実行することができ、ナノサイズ分子系の化学反応の解析や制御、材料設計などをスーパーコンピュータで効率的に行えるようになった。プログラムをオープンソース(Apache 2.0)ライセンスで公開しているため、本研究の成果を他の研究者が自由に使うことができる。また、データを含めたプログラム構造をシンプルにしており、他のプログラムへの移植や融合も容易に行うことができる。

研究成果の概要（英文）：New algorithms for many-core machines and supercomputers were developed and implemented for the performance enhancement and generalization of massively parallel open source software for quantum chemistry calculations SMASH. The improvement of the parallel efficiency, and the reductions of the memory usage and network communication time were achieved. Furthermore, the data and loop structures were modified to combine with other methods by Python and clarify data flow in the program. The stable structures of metal nanoparticles were analyzed to reduce the amount of rare metals in catalytic reactions, and a QM/MM method with the periodic boundary condition treatment of explicit electron-charge interaction was developed.

研究分野：量子化学

キーワード：並列計算アルゴリズム　量子化学　オープンソース　スーパーコンピュータ　大規模並列計算　SMASH

様　式　C－１９、F－１９－１、Z－１９、CK－１９（共通）

１．研究開始当初の背景

化学反応の解析や制御、分子材料設計など幅広く使われている量子化学計算は演算コストが非常に大きいため、本研究代表者はこれまでに大規模並列量子化学計算ソフトウェア SMASH を開発してきた。言語は Fortran90/95、MPI と OpenMP で並列化されており、Apache2.0 オープンソースライセンスで公開されている。最もよく用いられる密度半関数(DFT)法による炭素シート 2 枚(360 原子)のエネルギー計算は、京コンピュータ 10 万 CPU コアで 2 分半、並列加速率約 5 万倍と高い性能を示している。ソースコードの特徴としては、どの計算方法でも用いられる 2 電子反発項などの計算ルーチンがライブラリ化されている。SMASH の様々な計算手法の実装コストを大幅に削減できただけではなく、他の研究者が容易に使えることから、複数の国内のソフトウェアへの一部ルーチンの組み込みも始まっている。

一方、計算機システムの発展状況から、ノード当たりの CPU コアはますます増加し、演算性能と通信性能の比率は広がると予測され、スーパーコンピュータを含めた今後の計算機を効率的に利用するためには、さらなる並列化効率の向上が必要であり、そのための新たなアルゴリズムが求められている。計算内容の観点では、量子化学計算単独ではなく他の計算手法との融合、例えば化学反応の重要な部分を量子化学計算で、それ以外の周囲については分子力場パラメータを用いて簡便に計算する QM/MM 法、電子分布の変化と分子の動きを組み合わせた第一原理分子動力学法などが期待されている。そのような手法の開発のためには、エネルギーや力(エネルギー微分)の計算を容易に呼び出せるようなシンプルなプログラム及びデータ構造が重要になる。

２．研究の目的

(1,2)ポスト京コンピュータを始めこれからのメニーコア計算機でも、今まで以上に大きな分子の量子化学計算を実行するため、ノード内(OpenMP)並列化効率の向上と通信時間を削減する新規アルゴリズムの開発と実装を行う。

(3)今後ますます期待される量子化学計算と他の計算手法の融合を容易にするため、SMASH ソフトウェアでのサブルーチン及び関数のデータの受け渡しを全て引数で行い、module 変数を削除して、データの流れを明確化する。開発したソフトウェアをオープンソースライセンスで公開し、他の研究者が自由に使えるようにする。

```
!$OMP parallel do schedule(dynamic,1)
do μν = nbasis*(nbasis+1)/2, 1, -1
  μνからμ*とνを逆算
  λstart=mod(μν+mpi_rank, nproc)+1
  do λ = λstart, nbasis, nproc
    do σ = 1, λ
      2 電子反発項(μν|λσ)計算+
      Fock 行列 F_μνへの足し込み
    enddo
  enddo
enddo
call MPI_AllReduce(Fock)
```

図１　2 電子積分項並列計算アルゴリズム

３．研究の方法

(1)Hartree-Fock 及び DFT 法における 2 電子積分項計算の 4 重ループは、SMASH ソフトウェアでは最外(第 1)ループのみ OpenMP でスレッド並列化されていた。振り分ける作業の粒度を小さくして分散数を増やして OpenMP 並列化効率を向上させるため、第 1,2 ループをまとめて OpenMP 並列化を行うが、第 2 ループの回転数は第 1 ループのインデックスに依存しており collapse 節を導入できない。そこで、2 つのループを融合し 3 重ループにして、元のインデックスを後で逆算する(図 1)。金属原子などを含む複雑な電子状態の収束を向上させるため、同様のアプローチで 2 次収束法の MPI/OpenMP 並列化アルゴリズムも開発する。

(2)2 次の摂動(MP2)エネルギーは式(1)で表され最もコストがかかるのが原子軌道 2 電子反発項($\mu\nu|\lambda\sigma$)から分子軌道 2 電子反発項($ia|jb$)への 4 つのインデックス変換(式(2))である。ここで i,j,a,b は分子軌道、μ,ν,λ,σ は原子軌道、$C_{\mu i}$ は Hartree-Fock 計算で得られる分子軌道係数である。これまでに開発したアルゴリズムでは、変換の前半(第 1,2)は原子軌道を、後半(第 3,4)は分子軌道を各プロセスに分散させ、各変換においては BLAS(行列-行列積)を用いたスレッド並列化を行っている。前半と後半で各プロセスが担当するインデックスが変わるため、後半では

```
do i-block
  do μλ= 1, nbasis*(nbasis+1)/2 (MPI 分散)
    do σν= 1, nbasis*nbasis
      AO 積分計算 (μν|λσ)
      第 1 変換 (μi|λσ) (partial i)
    enddo σν
    第 2 変換 (μi|λj)
  end do μλ

  MPI_sendrecv (ij=1 で必要になる(μi|λj))
do ij (MPI 分散)
    MPI_isend,irecv (次の ij で必要になる(μi|λj))
    第 3 変換 (μi|bj)
    第 4 変換 (ai|bj)
    MP2 エネルギー計算
    MPI_wait
  end do ij
end do i-block
MPI_reduce(MP2 エネルギー)
```

図 2 MP2 エネルギー並列計算アルゴリズム

$$E_{MP2} = \sum_{ij,ab} \frac{(ia|jb)\{2(ia|jb) - (ib|ja)\}}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b} \quad (1)$$

$$(ia|jb) = \sum_{\mu\nu\lambda\sigma} C_{\mu i} C_{\nu a} C_{\lambda j} C_{\sigma b} (\mu\nu|\lambda\sigma) \quad (2)$$

データの送受信が必要となる。
　従来のアルゴリズムでは、前半(第 2 変換)終了後の中間データ全てを複数ノードのメモリ上に分散保存しているが、その総量は原子数の 4 乗に比例して増加するため巨大分子の計算が不可能であった。そこで、$(\mu\nu|\lambda\sigma)$項を複数回計算することになるが、計算全体の外にループを 1 つ追加し、変換する分子軌道のうち第 1 変換の i を分割することで、中間データの配列サイズを小さくする(図 2)。通信はこれまで MPI_sendrecv を用いて行っており、後半の変換において数多く呼ばれる。その全通信量は原子数の 4 乗に比例して増加する。そこで演算と通信をオーバーラップさせ通信時間削減を図る。後半の変換時に受信データ用配列を 2 つ用意し、受信用と受信後の演算用としてこの 2 つを交互に利用する。MPI_Isend,Irecv,Wait に MPI 通信を変更して、演算を行っている最中に通信を行う(図 2)。MP2 エネルギー微分計算についても、同様の手順で使用メモリ量及び通信時間削減アルゴリズムの開発と実装を行った。エネルギー1 次微分計算についても同様のアルゴリズムを開発した。

(3)Python で各計算ステップを容易に制御し、アクセスするデータを明確化するため、物理定数などパラメータ変数以外のすべての変数を引数で関数及びサブルーチンに渡すように変更した。分子の座標など計算で必要になる基礎データは従来 Fortran の module 変数で扱っているが、構造体を用いることでシンプルな受け渡しを実現しながら、SMASH ソフトウェア内の module 変数を削除した。

４．研究成果

(1)図 1 のアルゴリズムを実装し、Xeon Phi(Knights Corner, 60 コア)で Taxol($C_{47}H_{51}NO_{14}$)分子、cc-pVDZ 基底の Hartree-Fock 計算を行った(表 1)。少ないスレッド数ではオリジナルと差はほとんどなかったが、60 スレッド以上でも改良版の並列加速率は伸び続け、メニーコアマシンでも効率的に計算実行できるアルゴリズムであることを実証した。

表 1 Taxol 分子の Hartree-Fock 計算時間(秒)と並列加速率

	オリジナル		改良版	
スレッド数	計算時間	並列加速率	計算時間	並列加速率
15	9034.6	15.0	9057.9	15.0
30	4559.0	29.7	4565.8	29.8
60	2391.9	56.7	2302.6	59.0
120	2652.9	51.1	1592.9	85.3
240	2888.6	46.9	1333.5	101.9

(2)まず、MP2 エネルギー計算についてメモリ使用量削減のためのループ追加と分子軌道分割を実装した。Intel Xeon マシン(28 コア、128GB メモリ)1 ノードで Taxol 分子の計算を行ったところ、表 2 のように分子軌道を 2 分割、3 分割した場合、メモリ使用量はそれぞれ 1/2、1/3 になり、計算時間は約 20%、40%の増加となった。分割した場合、複数回計算するのは初めのステップの原子軌道 2 電子積分項のみで、その後の 4 回の積分変換の演算量は分割数に依存しないため、分割数に比べて計算時間増加の割合は少ない。少ない計算時間の増加で大幅にメモリ使用量を削減することができ、メモリ不足による計算可能サイズ制限の問題は解消した。京コンピュータで $C_{150}H_{30}$、cc-pVDZ 基底の MP2 エネルギー微分並列計算を行ったところ、ノード数を増やすにしたがって全体のメモリ量が増え、軌道分割数が減るため、使ったノード数以上の並列加速率を達成した(表 3)。

表 2 MP2 エネルギー計算における軌道分割数、計算時間(秒)、メモリ使用量(GB)

軌道分割数	1	2	3
MP2 計算時間	765.7	943.4	1121.5
メモリ使用量	101	51	34

表 3 MP2 エネルギー微分並列計算の実行時間(秒)と並列加速率

ノード数	192	384	768	1536
軌道分割数	3	2	1	1
MP2 微分実行時間	4253.6	2002.0	743.6	382.0
並列加速率	192.0	407.9	1098.2	2137.9

表 4 MP2 エネルギー計算の実行時間と通信時間(秒)

	ノード数	96	192
改良前	MP2 実行時間	1089.9	421.9
	(うち通信時間)	9.7	8.7
改良後	MP2 実行時間	1082.2	416.7
	(うち通信時間)	5.2	3.4

さらに、MPI_Isend、Irecv、Wait を用いた演算と通信のオーバーラップも実装し、京コンピュータで $C_{150}H_{30}$、cc-pVDZ 基底の MP2 エネルギー計算を行った。表 4 に示す通り、通信時間は半減し、今後演算と通信性能の比率がますます広がると予測される計算機システムでも高い並列性能を維持したまま実行できるソフトウェアになったと考えられる。

(3)構造体を使って module 変数をサブルーチン・関数の引数に変更したコードを作成し、Luciferin($C_{11}H_8N_2O_3S_2$)分子の B3LYP/cc-pVDZ エネルギー計算を行った。さらにそのコードを Python から呼び出して実行した。どのプロセス数、スレッド数でも、変数を引数に変えたコードでは実行時間にほとんど変わりはなく、コード書き換えによる影響は無かったが、Python から SMASH を呼び出した場合、約 3%のオーバーヘッドがあった。この改良により、Python を使って他の計算手法との組み合わせが、多少実行時間は増えるものの容易に行えるようになった。

表 5 B3LYP エネルギー計算実行時間(秒)

プロセス数	スレッド数	オリジナル	構造体導入	Python 制御
1	1	381.2	385.6	393.4
1	28	18.7	18.7	19.6
2	28	10.7	10.6	11.0

周期境界条件下であらわに電子-電荷相互作用を取り扱った QM/MM 法を開発、実装した。この相互作用を正確に記述するためには、Ewald 法における surface-dipole 項が必要不可欠であることを示した。さらに、触媒反応における貴金属削減のための金属ナノ粒子の安定構造の解析も行った。

５．主な発表論文等

〔雑誌論文〕（計　　６件）

① Kawashima Y.、Ishimura K.、Shiga M., Ab initio quantum mechanics/molecular mechanics method with periodic boundaries employing Ewald summation technique to electron-charge interaction: Treatment of the surface-dipole term, The Journal of Chemical Physics, 査読有, 150, 2019, 124103-124103
DOI:10.1063/1.5048451

② Furukawa Shunsuke、Fujita Masahiro、Kanatomi Yoshihiko、Minoura Mao、Hatanaka Miho、Morokuma Keiji、Ishimura Kazuya、Saito Masaichi, Double aromaticity arising from σ- and π-rings, Communications Chemistry, 査読有, 1, 2018, 60
DOI: 10.1038/s42004-018-0057-4

③ Yanai Kazuma、Ishimura Kazuya、Nakayama Akira、Hasegawa Jun-ya, First-Order Interacting Space Approach to Excited-State Molecular Interaction: Solvatochromic Shift of p-Coumaric Acid and Retinal Schiff Base, Journal of Chemical Theory and Computation, 査読有, 14, 2018, 3643-3655
DOI: 10.1021/acs.jctc.7b01089

④ Takagi Nozomi、Ishimura Kazuya、Matsui Masafuyu、Fukuda Ryoichi、Ehara Masahiro、Sakaki Shigeyoshi, Core-Shell versus Other Structures in Binary $Cu_{38-n}Mn$ Nanoclusters (M = Ru, Rh, Pd, Ag, Os, Ir, Pt, and Au; n = 1, 2, and 6): Theoretical Insight into Determining Factors, The Journal of Physical Chemistry C, 査読有, 121, 2017, 10514-10528
DOI:10.1021/acs.jpcc.6b13086

⑤ Yanai Kazuma、Ishimura Kazuya、Nakayama Akira、Schmidt Michael W.、Gordon Mark S.、Hasegawa Jun-ya, Electronic Polarization Effect of the Water Environment in Charge-Separated Donor-Acceptor Systems: An Effective Fragment Potential Model Study, The Journal of Physical Chemistry A, 査読有, 120, 2016, 10273-10280
DOI:10.1021/acs.jpca.6b10552

⑥ SAITOU Sona、MOCHIZUKI Yuji、YAMAZAKI Yutaka、ISHIMURA Kazuya, Performance Evaluations of Parallelized DFT Calculations with SMASH on Intel Xeon Phi Processor, Journal of Computer Chemistry, Japan, 査読有, 15, 2016, 92-96
DOI:10.2477/jccj.2016-0047

〔学会発表〕（計　１４件）

① Kazuya Ishimura, Development of Massively Parallel Quantum Chemistry Calculation Program SMASH, International Workshop on Massively Parallel Programming for Quantum Chemistry and Physics 2019, 2019

② Kazuya Ishimura, SMASH: Massively Parallel Software for Quantum Chemistry Calculations, 16th International Congress of Quantum Chemistry, 2018

③ 石村和也, Python を用いた大規模並列量子化学計算プログラム SMASH の制御, 第 21 回理

論化学討論会，2018
④ Kazuya Ishimura, SMASH: SMASH: Massively Parallel Quantum Chemistry Program, 11th Triennial Congress of the World Association of Theoretical and Computational Chemists, 2017
⑤ 石村和也，大規模並列 MP2 エネルギー微分計算アルゴリズムの開発と実装，第 19 回理論化学討論会，2016

〔図書〕（計　1件）
下司雅章編、片桐孝洋,中田真秀,渡辺宙志,山本有作,吉井範行,Jaewoon Jung,杉田有治,石村和也,大石進一,関根晃太,森倉悠介,黒田久泰著、大阪大学出版会、計算科学のための HPC 技術 1、2017、300（219-248）

〔産業財産権〕
○出願状況（計　0件）

○取得状況（計　0件）

〔その他〕
ホームページ
http://smash-qc.sourceforge.net/

６．研究組織
(1)研究分担者

※科研費による研究は、研究者の自覚と責任において実施するものです。そのため、研究の実施や研究成果の公表等については、国の要請等に基づくものではなく、その研究成果に関する見解や責任は、研究者個人に帰属されます。