2P128

GAMESSに実装された分割統治(DC)量子化学計算法のハイブリッド並列化

（早大高等研[1],早大先進理工[2],早大理工研[3], JST-CREST[4]）○小林正人[1]，中井浩巳[2,3,4]

【緒言】

並列計算は現在の計算機アーキテクチャにとって欠かすことのできない高速化技術となっている。計算の並列化は、OpenMP などを用いたスレッド並列化と MPI などを用いたプロセス並列化に大別することができる。我々は分割統治(DC)量子化学計算法を GAMESS パッケージに実装してきた[1–3]。GAMESS には Distributed Data Interface (DDI)という独自のプロセス並列インターフェースが用いられており、DC プログラムもこれを用いて並列化されている。さらに DC-MP2 計算に対しては、Generalized DDI (GDDI)を用いた二段階並列計算法を提案しており、その高い並列計算パフォーマンスが実証されている[4]。しかし、配布されている GAMESS はスレッド並列化は施されていない。『京』のような巨大なスーパーコンピュータで有意な計算を実行するためには、2 つの並列化技法を組み合わせたハイブリッド並列による効率的な実装が必要である。これまでにも GAMESS もしくはこれに付随するフラグメント分子軌道(FMO)法のハイブリッド並列化に関する報告がある[5,6]が、本発表では、特に DC-MP2 計算に注目し、そのハイブリッド並列化の試みについて報告する。

【DC-MP2 エネルギー計算のハイブリッド並列化】

DC-MP2 法では、部分系の MO を用いて以下の式で部分系 α の相関エネルギーを求める。

$$\Delta E^{\alpha}_{\mathrm{MP2}} = \sum_{i,j}^{\mathrm{occ}} \sum_{a,b}^{\mathrm{vir}} \sum_{\mu \in \boldsymbol{S}(\alpha)} C^{\alpha *}_{\mu i} \left\langle \mu j^{\alpha} \middle| a^{\alpha} b^{\alpha} \right\rangle \left[2\tilde{t}^{\alpha}_{ij,ab} - \tilde{t}^{\alpha}_{ij,ba} \right] \tag{1}$$

ここで $\boldsymbol{S}(\alpha)$は部分系 α の中央領域に属する AO の集合である。有効二電子励起係数$\tilde{t}^{\alpha}_{ij,ab}$ も部分系の MO を使って下式より求める。

$$\tilde{t}^{\alpha}_{ij,ab} = -\frac{\left\langle a^{\alpha} b^{\alpha} \middle| i^{\alpha} j^{\alpha} \right\rangle}{\varepsilon^{\alpha}_{a} + \varepsilon^{\alpha}_{b} - \varepsilon^{\alpha}_{i} - \varepsilon^{\alpha}_{j}} \tag{2}$$

(1)式で μ に関する部分和を取ることを除けば、すでに GAMESS に実装されている MP2 計算のプロセス並列アルゴリズム[7,8]をそのまま流用して部分系の相関エネルギーを算定することが可能である。さらに、DC-MP2 法では全系の相関エネルギーは下式で与えられる。

$$\Delta E_{\text{DC-MP2}} = \sum_{\alpha} \Delta E^{\alpha}_{\mathrm{MP2}} \tag{3}$$

(3)式で部分系ごとの計算は互いに独立に実行することが可能である。この性質を利用した手法が、二段階並列アルゴリズム[4]である。

OpenMP によるプログラムのスレッド並列化は、梅田ら[6]による実装に基づいて行った。具体的には、二電子積分の計算は shell quartet と呼ばれる四重ループを演算コアごとに分担する形で行った。また積分変換部分は、分散メモリを利用する DDI アルゴリズム[7]とメモリ利用量を減らしてディスクを使用する IMS アルゴリズムに対して OpenMP 化を行った。DDI アルゴリズムでは、演算コアが担当する二電子積分インデックスの排他的制御が困難であったため、ATOMIC ディレクティブを利用したメモリの競合回避を行った。行列演算は、可能な限りスレッド並列化された BLAS ライブラリを利用するものに書き換えた。

【OpenMP によるスレッド並列化の効率】

まず、今回 OpenMP 並列化した 2 つの MP2 計算アルゴリズムの並列パフォーマンスを検証した。図 1 は PC 1 ノード[CPU: Xeon X5650 (2.66 GHz)×2, メモリ: 24 GB]を用いてポリエン $C_{30}H_{32}$ の DC-MP2 計算を行った時の実計算時間を、用いたスレッドの数に対してプロットしたものである。プログラムは Intel Fortran Compiler 12.0.2 を用い、O2 最適化オプションでコンパイルした。1 スレッドでは DDI、IMS の両アルゴリズムの実行時間はほとんど同じであるが、スレッド数を増やすと IMS アルゴリズムの方が高速になることがわかる。これは DDI アルゴリズムにおいて ATOMIC ディレクティブによるメモリの競合回避が性能を大きく低下させているためであると考えられる。IMS アルゴリズムを用いることで、6 スレッド並列程度で計算時間を半分程度まで減少させることができている。しかしスレッド数をさらに大きくしても改善はほとんどみられないことがわかった。

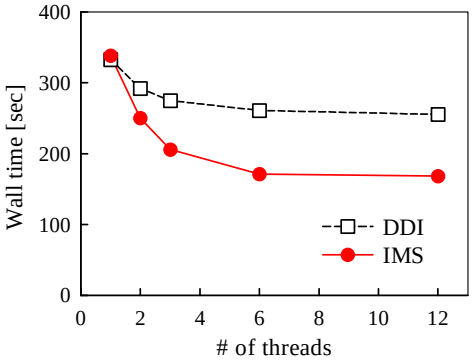

Fig. 1. Wall-clock time for DC-MP2 calculations of $C_{30}H_{32}$ with respect to the number of OpenMP threads (6-31G).

【ハイブリッド並列計算のパフォーマンス】

次に、DC-MP2 法の二段階並列アルゴリズムと組み合わせたハイブリッド並列による大規模計算の結果を示す。表 1 にポリエン $C_{300}H_{302}$ の DC-MP2 計算を 144、288、及び 576 プロセスを使用して実行した時の実計算時間を示す。積分変換部分は IMS アルゴリズムを用い、全プロセスを 18 プロセスごとにグループ化して二段階並列計算を行った。ノード間並列に用いる DDI ライブラリは、MPI と ARMCI を用いて実装した[6]。数千スレッド程度の並列計算でも、ストロングスケーリングで 80%程度の実行効率を出しており、効果的な実装が行えていることが確認された。

Table 1. Parallel execution performance of the DC-MP2 calculations of $C_{300}H_{302}$ with the 6-31G* basis set.

# of processes	# of threads	Wall time [s]	Efficiency
144	1008	2040.4	-
288	2016	986.4	1.03
576	4032	623.6	0.82

謝辞 GAMESS/FMOハイブリッド並列コードをご提供いただいた梅田宏明氏に感謝いたします。

[1] M. Kobayashi, T. Akama, and H. Nakai, *J. Comput. Chem. Jpn.* **8**, 1 (2009).

[2] M. Kobayashi and H. Nakai, in *Linear-Scaling Techniques in Computational Chemistry and Physics* (Springer, 2011), pp. 97-127.

[3] M. Kobayashi and H. Nakai, *Phys. Chem. Chem. Phys.* **14**, 7629 (2012).

[4] M. Katouda, M. Kobayashi, H. Nakai, and S. Nagase, *J. Comput. Chem.* **32**, 2756 (2011).

[5] K. Ishimura, K. Kuramoto, Y. Ikuta, and S.-a. Hyodo, *J. Chem. Theory Comput.* **6**, 1075 (2010).

[6] 梅田宏明, 佐藤三久, 分子科学討論会 2010 大阪, 1P094.

[7] G.D. Fletcher, M.W. Schmidt, and M.S. Gordon, *Adv. Chem. Phys.* **110**, 267 (1999).

[8] K. Ishimura, P. Pulay, and S. Nagase, *J. Comput. Chem.* **27**, 407 (2006).