1P094　大規模並列環境に向けた GAMESS-FMO の並列化

(1 筑波大 計算科学研究センター, 2 自然科学研究機構 分子研)
○梅田宏明 1,2, 佐藤三久 1

序

メニーコアプロセッサなどに象徴される計算機技術の発展に伴い、数万コアを越えるような超並列計算機が現実的なものとなってきている。このような大規模並列計算機を有効に利用するには、アルゴリズムの選択も含めたアプリケーションの改良が必要になってくる。例えば従来の MPI のみを用いた並列化手法ではコア数の増加が MPI ランク数の増加に直結しているが、これがメモリ使用量の増加など MPI 自体の動作に影響を及ぼすケースも見られるようになってきている。また大量の並列プロセスに対する同期的な集合通信などのコストも大きい。この問題を回避するには、階層的な並列化手法が重要となってくる。例えば、ノード内の複数の計算コアについては OpenMP 等のスレッド並列の手法を用いた細粒度の並列化し、ノード間については MPI による粗粒度の並列化とするなどの並列実装が要求される。

計算化学分野に目を向けると、大規模並列計算に的した計算手法としてフラグメント分子軌道法(FMO 法)[1]があげられる。FMO 法では分子全体を比較的小さなフラグメントに分割して取り扱う。これによりフラグメント内・フラグメント間といったような階層的な並列化が自然に実現可能となっている。我々がターゲットとする分子軌道計算プログラムである GAMESS[2]における FMO の実装では、GDDI と呼ばれるプロセス間並列 API により、フラグメント内・フラグメント外の二階層の並列化が実装されている。一方上述のように、MPI ランク数を減らす必要性があるため、スレッド並列化も不可欠である。我々は超並列計算機上で GAMESS の FMO 計算を高速に実行させるために、この計算に関わるルーチンの OpenMP によるスレッド並列化を試みている。本発表では、FMO-HF 計算で重要となる Fock 行列の生成ルーチンを OpenMP 化したので、その結果を中心に発表する。

GAMESS の並列化

GAMESS は並列化については、独自の通信レイヤーである Distributed Data Interface (DDI)を用いたプロセスレベルの並列化が行なわれている。DDI では、broadcast や global sum 等の集合通信や send/recv のような一対一の通信の他に GET/PUT/ACC といったリモートメモリアクセス(RMA)機構が実装されている(version 1)。またサブグループの概念が導入されフラグメント分子軌道法計算で利用されている(GDDI, version 2)。最近では、マルチプロセッサノードの普及に対応してノード内共有メモリを利用する機構が実装され、Coupled Cluster 計算においてより大きな計算をオンメモリで高速に計算できるようになってきている(version 3)。現在 GAMESS と共に配布されている DDI には TCP/IP socket, MPI-1/socket mixed, MPI-1, ARMCI 等による実装が用意されているが、前回報告したように MPI+ARMCI の実装がテストに用いた T2K 筑波システムでは高速であったため、ARMCI を用いた実装を利用した。

OpenMP 並列化に際し GAMESS プログラム自体への変更を最小限にするため、COMMON 文で定義される共通ブロックは必要に応じ threadprivate 化し、copyin で初期化した。GAMESS では独自のメ

モリ管理機構を利用しており、巨大な配列についてはそれを使うことが望まれる。一方で局所的な配列を利用した方が速度的には有利であり、これを使い分ける必要がある。今回の実装では Fock 行列自体については GAMESS のメモリ管理機構を利用し、二電子積分の結果が格納されるワーク領域については OpenMP による private 配列とした。FMO 計算では Fock 行列がフラグメントのサイズで抑えられるため、通常の大規模 HF 計算で行なわれる direct SCF 計算だけでなく、in-core 計算の OpenMP 並列化も必要である。今回 OpenMP 並列部分については単純化のため静的負荷分散を採用し、DDI の動的負荷分散に関わる通信についてはマスタースレッドのみが実行するようにした。

なお、GAMESS の HF 計算等の OpenMP 化については石村らによる発表があり非常に良い成果をあげている[3]。しかしながら彼等のコードは公開を前提としたものではないため、今回は独自にコードを実装した。

テスト計算および結果

開発したコードを評価するため、CG 対 2 対からなる DNA 分子の HF/6-31G 計算(126 原子, 814 基底)を direct SCF 法で行なった。テストは筑波大の T2K システム(4 コア/CPU, 4CPU/ノード, Infiniband, [4])を利用し 40 ノード(640 コア)までの並列性能を測定した。図 1 には計算コア数ごとの速度向上比を全体および SCF 部分に対して示している。Hybrid(n)は MPI ランクあたり n スレッドを使った OpenMP 並列を意味しており、Original は元となる GAMESS(12 JAN 2009 (R1))のプロセス並列による結果である。

利用した MPI ライブラリの問題で、64 ランクを越えたところで並列化による性能の向上が劣化していることがわかる。このためプロセス並列のみでは 128 コア程度で性能が頭打ちになってしまっている。一方でスレッド並列も併用した場合には MPI 64 ランクに対応するコア数が多いため、より高い性能を示している。

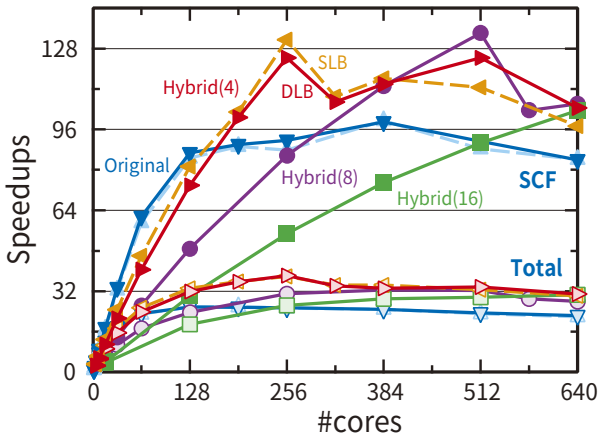

図 1 HF 計算経過時間の速度向上比(direct SCF)

[1] Kitaura, K.; Sawai, T.; Asada, T.; Nakano, T.; Uebayasi, M. *Chem Phys Lett* 1999, **312**, 319.

[2] GAMESS (General Atomic and Molecular Electronic Structure System), http://www.msg.chem.iastate.edu/gamess/

[3] Ishimura, K.; Kuramoto, K.; Ikuta, Y.; Hyodo, S.-a. *J Chem Theory Comput* 2010, **6**, 1075.

[4] T2K Open Supercomputer Alliance, http://www.open-supercomputer.org/