4P075　　大規模 Fock 行列計算プログラムの開発

(科技振 CREST[1], 産総研 計算科学[2], 九大 情基[3])
○梅田宏明[1,2], 稲富雄一[3], 渡邊寿雄[1,2], 石元孝佳[1,2], 長嶋雲兵[1,2]

序

生体分子のような大規模系の分子軌道計算ではフォック行列の生成に全計算時間の 9 割以上の時間を費やすことが知られている。これまで我々は Ninf-G を利用することで、Grid 上の多数の計算機を効果的に利用する並列 Fock 行列計算プログラムを開発してきた[1]。しかし我々の開発してきたアルゴリズムは Fock 行列及び密度行列の全体を全ての計算機が保持していることを仮定しており、数万次元を超えるような計算を実行するにはメモリ量が問題となってしまっていた。例えば 2 万次元の実対称行列のサイズは 1.5GB であり、このサイズの Fock 行列・密度行列を保持するためには最低でも 3GB 以上のメモリが必要となる。これまで広く利用されてきた 32bit Linux 計算機ではプロセスあたりのメモリ上限が(通常は)2GB であり、2 万基底を超えるような計算は不可能であった。またこれまで行なわれてきた HF 計算を SCF 法により解く手法では、このサイズの計算は計算量的にも非常に困難であり、実際にメモリ量が問題となるような計算を実行されることはあまりなかった。しかし我々の開発した FMO-MO 法[2]では、計算量の大きい Fock 行列を一度だけ計算すればよいため、SCF 法では困難なサイズの大規模 HF 計算が比較的容易に実行可能となっている。本発表では、FMO-MO 法による大規模分子軌道計算のための大規模 Fock 行列計算プログラムを開発したので、これを紹介する。

大規模 Fock 行列計算プログラム

メモリの使用量を考慮した並列分散 Fock 行列計算プログラムは既にいくつか発表されている。代表的なアルゴリズムとして Global Arrays Toolkit [3]を用いて Fock 行列をブロック分散した R. J. Harrison らのアルゴリズム[4]や、H. Takashima らの RT 並列アルゴリズム[5]がある。Global Arrays Toolkit は ARMCI ライブラリ等を用いて非同期にアクセスできる分散共有メモリ機構(リモートメモリアクセス機構)を実現する開発環境であり、Fock 行列作成のような大規模行列の分散共有には非常に適している。Harrison らのアルゴリズムでは、Global Array 上の密度行列および Fock 行列をブロック化して扱うことで計算におけるデータの局所性を確保し、通信自体の頻度を減らしている。この時の通信も非同期アクセスであるため、複雑なアルゴリズムなしに効率の良い並列化が実現可能となっている。一方、Takashima らの RT 並列は、縮約シェル R,T 間でのスクリーニングを有効に活用することでデータ転送量を削減する方法である。各計算ノードで必要となるメモリ量が非常に少ないため、MPP 計算機のような個々のプロセッサのメモリ搭載量が少ない計算機には向いている。しかし RT 並列ではホスト計算機の上に何らかの形で密度行列および Fock 行列を保持する必要がある。我々の開発した大規模 Fock 行列計算法は、これまで我々が開発してきた計算法を拡張し、Global Array を用いて Fock 行列や密度行列を保持する方法である。このため Ninf-G などによる GridRPC/MPI ハイブリッドモデルへの拡張も同様に可能となっている。ここでは、MPI レベルでの我々の大規模並列 Fock 行列計算プログラムを用いた FMO-MO 計算をいくつかのテスト分子に対して実行する。

テスト計算および結果

開発したプログラムのテストとしてリゾチーム分子(129 残基, 1,961 原子)の FMO-MO/HF/STO-3G 計算(6,002 基底)を行なった。この分子は Fock 行列のサイズが 200MB 以下であり、以前の Fock 行列計算プログラムでも比較実行可能である。テストは AIST スーパークラスタ(ASC)[6]の F32 クラスタ(32bit Xeon x2, 3.06GHz, 4GB memory)を用い、127 ノード(254 プロセッサ)を利用して行なった。F32 クラスタの内部ネットワークは Gigabit ethernet であり、MPI ライブラリとして GridMPI 2.1 を用いている。Global Arrays Toolkit 4.0.6 は GridMPI に加え ARMCI 用に TCP/IP socket を選択している。

Global Array を利用することによる通信の影響を確認するため、我々の以前のプログラムと速度を比較した。アルゴリズムとしては静的負荷分散アルゴリズムを用い、両者で同じ負荷分散をあたえるようにした。以前のプログラムの経過時間が 854 秒であったのに対し、Global Array を利用した今回のプログラムの経過時間は 871 秒であった。今回のプログラムでは通信の隠蔽など十分なチューニングを施していないが、2%(17 秒)程度の遅れとなっている。我々のアルゴリズムでは計算サイズが大きくなった場合に通信の隠蔽がより効果を発揮すると予想され、大規模な Fock 行列でも効率良く生成することが可能である。

さらに大きなテスト系として、溶媒(2,096H_2O + 9Cl^-)中のリゾチーム分子(図 1)についての FMO-MO計算を行なった。この系はSTO-3G基底関数で 20,758 基底になり、Fock行列・密度行列のサイズがそれぞれ 1.6GB、合計で 3.2GBになってしまうため、これまでF32 クラスタでは計算が不可能であった。しかし今回開発したプログラムを使えば、このサイズのFock行列が 4 時間弱で生成できる。またFMO-MO計算全体で見れば 5 時間程度でこの分子軌道(図 1)が得られており、高速な大規模分子軌道計算を実現している。

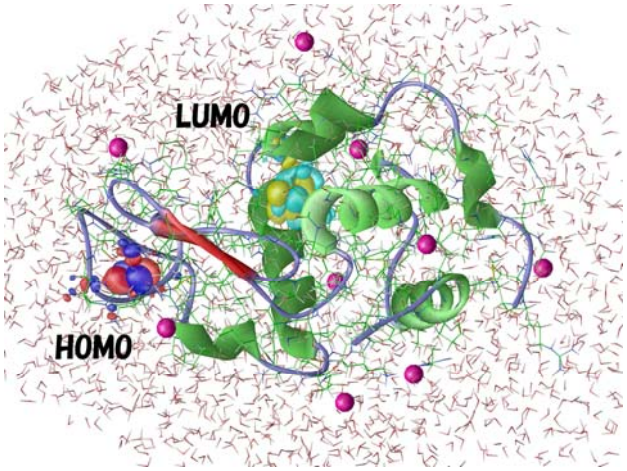

図 1 リゾチーム分子とそのフロンティア軌道

[1] H. Umeda et al., *J. Comput. Chem. Jpn.*, **4**, 179(2005). [2] Y. Inadomi et al., *Chem. Phys. Lett.*, **378**, 589(2002). [3] Global Arrays Toolkit; http://www.emsl.pnl.gov/docs/global/. [4] R. J. Harrison et al., *J. Comp. Chem.*, **17**, 124(1996). [5] H. Takashima et al., *J. Comp. Chem.*, **23**, 1337(2002). [6] http://unit.aist.go.jp/tacc/ci/supercluster.html.