3P100　OpenFMO における Fock 行列計算ルーチンの GPGPU 化

(筑波大 CCS) ○梅田宏明、塙敏博、庄司光男、朴泰祐

Fock matrix calculation with GPGPU for OpenFMO

(CCS, Univ. Tsukuba) ○H. Umeda, T. Hanawa, M. Shoji, T. Boku

序

近年、GPU を高性能計算向けの演算加速装置として利用する動きがある。量子化学計算アプリケーションについてもこのような GPGPU に対応した実装が求められている。そこで我々は HF 計算において最も計算量の多い Fock 行列計算について GPGPU による高速化を試みた。ターゲットプログラムとして超並列計算機向けの FMO 計算プログラムである OpenFMO[1]を取り上げ、そのフラグメント計算の HF 計算部分だけを切り出したコードについて CUDA による GPGPU 化を行なった。

OpenFMO は九州大学の稲富らが開発している超並列計算機向けフラグメント分子軌道（fragment molecular orbital, FMO）計算プログラムである。このプログラムは C 言語で記述されており、CUDA 実装でも扱いやすい。また OpenMP/MPI ハイブリッド並列による超並列計算機向けの並列化だけでなく、RPC 化などによる耐故障性の検討にも使われるなど次世代の量子化学計算アプリのプラットフォームとしても利用が期待されている。

GPGPU 化実装

GPGPU を用いて高性能計算を実現するためには、大量の演算コアを有効に使うことが必要である。このために重要なのがメモリアクセスの効率化とプログラムの SIMD 並列化となる。また特に Fock 行列計算においては各演算コアが計算した二電子積分を Fock 行列に加算する操作が存在するため、Fock 行列の取り扱い手法についても工夫が必要である。本研究では 1)行列加算手法、2)Schwarz の不等式によるスクリーニング、3)動的負荷分散、4)基底関数のソート、5)複数 GPU 計算、6)CPU との混合計算の 6 つの改良を検討した。

1)　行列加算手法

GPGPU 1 台あたり数百もの演算コアが利用可能であるのに対し、利用できるメモリは僅かに数 G バイト程度しかない。このため多くの並列実装で行われているような Fock 行列を演算コアが独立に保持する実装は現実的には不可能である。一方で Fock 行列を複数演算コアで共有する場合には Fock 行列への加算にコストの高い排他制御が必要となり、性能が出ないことも知られている。そこで我々は FMO-MO 計算のために開発した並列分散共有 Fock 行列計算アルゴリズム[2]を GPGPU に適用することでアトミック加算を回避する実装を行った。

2)　Schwarz の不等式によるスクリーニング

ホスト CPU と違い GPU は SIMD 動作をするため、Schwarz スクリーニングによる積分計算の省略のための分岐が効果的に動作しない可能性がある。そこで我々はスクリーニング処理だけを積分計算の前に実行するよう改良を行い、効果的な Fock 行列計算を試みた。

3)　動的負荷分散

GPGPU では演算コアが多いため静的な負荷分散では十分な負荷バランスが得られない可能性がある。そこで GPU ボード内についての動的負荷分散コードを実装し効率の良い並列化を目指した。

4)　基底関数のソート

積分計算そのものの実装についても SIMD 動作を考える必要がある。OpenFMO で利用している小原積分では積分計算において原始シェルペアについての二重ループが存在しており、効果的な SIMD 動作のためにはこのループ長を揃えることが望まれる。我々は縮約シェルペアを原始シェルペア長でソートすることにより、これを実現した。

5)　並列化による複数 GPU による計算

OpenFMO は OpenMP/MPI のハイブリッド並列化が実装されており、それを利用した複数 GPU 実行も高速化には有効である。我々は 1GPU/1MPI ランクの構成で並列実行を可能にした。

6)　CPU-GPU 混合計算

GPU による計算中に CPU でも計算を行なう混合計算によりさらなる高速化を実現した。また、GPU では性能が出ない積分タイプを CPU で計算させるなどの最適化も可能になる。

ベンチマーク

性能評価としてモデル DNA 分子$(CG)_2$の HF/6-31G(d)計算を筑波大の GPGPU クラスタ HAPACS を用いて実行した。図は 4CPU コア(Intel E5 2.6GHz)と 1GPU(M2090)を用いて計算した場合の経過時間である。CPU だけで実行していた場合に比べ、部分的に GPGPU 化を行った場合は半分程度の時間で、CPU と GPU の混合計算を利用した場合には 1/3 程度の時間で計算が実行できていることがわかる。

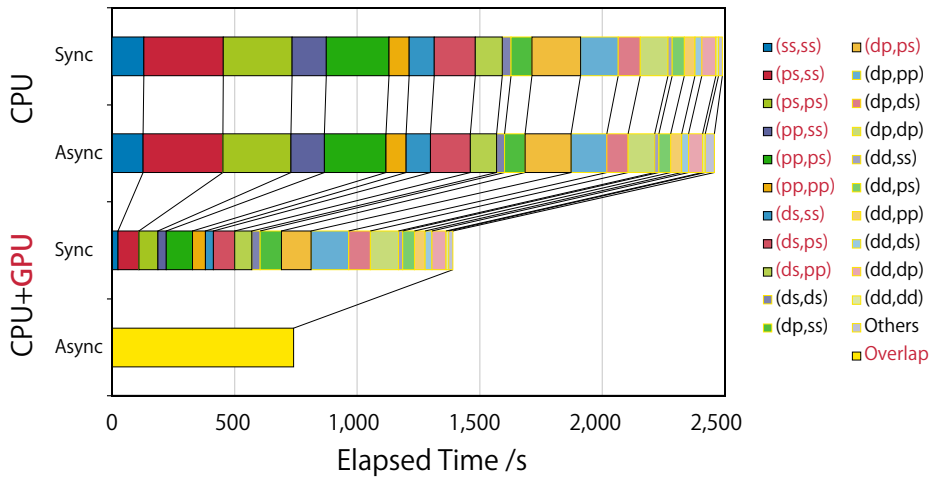

図　CPU-GPU 混合実行による高速化

謝辞: 本研究で使用した HF 計算コードは九州大学の稲富らによる OpenFMO プログラムの一部を抜粋して提供していただいている．また本研究の一部は文部科学省特別経費「エクサスケール計算技術開拓による先端学際計算科学教育研究拠点の充実」事業，および JST-CREST 研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」，研究課題「ポストペタスケール時代に向けた演算加速機構・通信機構統合環境の研究開発」による．

[1] OpenFMO; http://www.openfmo.org/OpenFMO/index.html; 本学会講演 1E01.
[2] Umeda, H., Inadomi, Y., Watanabe, T., Yagi, T., Ishimoto, T., Ikegami, T., Tadano, H., Sakurai, T. and Nagashima, U., *J Comput. Chem.*, **31**, 2381-2388(2010).