2P17

GPGPU 化した Fock 行列計算ルーチンの OpenFMO への組み込み

○梅田 宏明[1]、塙 敏博[2]、庄司 光男[1]、朴 泰祐[1]
[1]筑波大学 計算科学研究センター(〒305-8577 つくば市天王台 1-1-1)
[2]東京大学 情報基盤センター(〒277-8589 柏市柏の葉 5-1-5)

【緒言】

近年の大規模高性能並列計算機の発展において、アクセラレータを用いた高性能科学技術計算は重要なトピックとなっている。量子化学計算においても大規模高性能計算のためのアクセラレータ対応は喫緊の課題であり、これを利用したアプリケーションの開発が求められている。これまで我々は Hartree-Fock(HF)計算のホットスポットである Fock 行列計算の GPGPU 化を試みてきた[1]。この成果を大規模分子軌道計算に応用するために、フラグメント分子軌道(FMO)法[2]プログラムである OpenFMO [3]に我々の GPGPU 化 Fock 行列計算コードを導入した。本発表では、GPGPU 化手法の説明とこれを用いた FMO 計算についての性能評価を報告する。

【実装】

GPGPU 化を行った Fock 行列計算コードは OpenFMO においてフラグメント(及びフラグメントペア)の計算で利用される HF 計算コードをスケルトンコードとして切り出して提供されているものである。コードは簡潔であり機能追加や改良を容易にすることができる。またこのスケルトンコードへの改良は OpenFMO プログラムへの導入も容易であり、FMO 計算に適用することが可能となる。我々はこのスケルトンコードに対して CUDA 言語による GPGPU 実装を行った。

多くの計算コアを持つ GPU では行列への加算にコストの高い排他制御が必要となるため、これを避けるようなアルゴリズムを開発した。また実装においては GPU スレッドの SIMD 動作が効果的に実行可能となるような最適化を行っている。さらなる高速化として CPU と GPU を同時に実行させるコードも開発した。Fock 行列計算の GPU 化は使用する二電子積分の積分タイプごとに行っているため、GPU 化で十分な速度の出ない積分タイプについての Fock 行列計算を CPU で実行させることにより、効果的な CPU と GPU の同時実行を可能としている。また複数 GPU を利用した並列化も実現した。

組み込みに利用したバージョンの OpenFMO プログラムは動的プロセス生成を利用する MIMD プログラムとなっており、マスタ MPI プロセスとメモリサーバ MPI プログラム、そして多数のワーカ MPI プログラムからなっている。我々のコードを実装するのは、このうちでワーカプログラムだけである。GPGPU 化コードの組み込みは 1)GPU の初期化、2)分子データの転送、3)Fock 行列計算ルーチンの差し替え、4)分子データの消去、5)GPU の終了処理の各コードを OpenFMO ワーカコードに挿入することで実装できる。同じプログラムで複数のフラグメントに対する Fock 行列計算を実行するため、一つの SCF 計算終了後に不要となった分子データを正しく開放しなければメモリリークの原因となる。誤差関数テーブルなど使い回すことが可能なデータもあり、利用期間の異なるデータの取り扱いには十分な注意が必要である。

【性能評価】

OpenFMO に実装した GPGPU 化 Fock 行列計算コードの性能評価は筑波大学の HA-PACS ベースクラスタ[4]を用いて行った。HA-PACS ベースクラスタの計算ノードには 2 台の 8 コア Intel E5 CPU(Sandy Bridge-EP, 2.6GHz)と 4 台の Fermi 世代の GPGPU(NVIDIA M2090 GPU)、および 128GB のメモリが搭載されており、それらが InfiniBand により接続されている。複数 GPU を活用するためノードごとに 4MPI プロセスを起動し、それぞれのプロセスが OpenMP 並列で 4 CPU コアと 1 台の GPU を利用することとしている。コンパイルや実行には Intel コンパイラ 14.0, CUDA 5.0.35,

mvapich2 1.8.1 をそれぞれ利用した。OpenFMO では動的プロセス生成や片側通信を利用するため、MPI 処理系である mvapich2 については適切に環境変数を設定している。

ベンチマークとしてグリシンの 10 量体(112 原子, 5 フラグメント)及びクランビンタンパク(642 原子, 20 フラグメント)の FMO-HF/6-31G(d)計算を行った。それぞれ計算には HA-PACS ベースクラスタの 2 ノード(8MPI プロセス)および 8 ノード(32MPI プロセス)を用いている。HA-PACS クラスタではノードあたり 128GB と大きなメモリ空間を利用可能なため、FMO 計算のフラグメントサイズ程度であれば二電子積分をメモリに保存しておいて何度も利用する in-core 手法が可能になる。そこで性能評価では CPU や GPU による direct 計算だけでなく、この in-core 計算手法も比較の対象とした。表 1 にはそれぞれ計算手法による実行時間を示した。我々が実装した GPGPU 化 Fock 行列計算コードは CPU による direct 計算より速いだけでなく、in-core 計算手法を用いた場合よりも高速に動作していることがわかる。

表 1 GPGPU 化 Fock 行列計算コードを用いた FMO 計算の実行時間(秒)

	$(Gly)_{10}$			Crambin		
Algorithm	SCC	DimerSCF	Total	SCC	DimerSCF	Total
Direct	240	166	414			
In-core	214	113	335	1,513	1,382	3,007
Direct(GPU+CPU)	196	88	302	1,355	1,185	2,661

参考文献

[1] 梅田宏明, 塙敏博, 庄司光男, 朴泰祐, 稲富雄一, 情報処理学会論文誌コンピューティングシステム(ACS), **6**, 26(2013).

[2] K. Kitaura et al., Chem. Phys. Lett., **312**, 319(1999).

[3] OpenFMO; http://www.openfmo.org/

[4] HA-PACS ベースクラスタ;
http://www.ccs.tsukuba.ac.jp/research/research_promotion/project/ha-pacs/cluster