1P15

OpenFMO における 4 中心クーロン相互作用項計算の GPGPU 化の試み

○梅田宏明[1]、塙敏博[2]、庄司光男[1]、朴泰祐[1]、重田育照[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)

序

GPU 等の演算加速装置を用いた高性能科学技術計算システムへの対応は、量子化学計算においても重要なトピックとなっている。これまでに我々は Hartree-Fock(HF)計算のホットスポットである Fock 行列計算の GPGPU 化[1]を行ってきた。フラグメント分子軌道(FMO)計算[2]プログラムの大規模計算機向け実装である OpenFMO [3]に GPGPU 化 Fock 行列計算コードを導入する[4]など、GPU を使った大規模分子軌道計算に向けた開発を進めている。本発表では OpenFMO による FMO 計算でホットスポットになる 4 中心静電相互作用項(ESP)計算についての GPGPU 化の取り組みについての現状を報告する。

実装

二つの近接フラグメント間の静電相互作用を計算する 4 中心 ESP 計算のアルゴリズム構成は Fock 行列計算と類似しており、Fock 行列計算の GPGPU 化で行なった手法の多くを活用できる。大きな違いとしては、計算された二電子積分を用いて行列に加算するのが一方のフラグメントに対するクーロン項に限られていることが挙げられる。Fock 行列計算では行列全体に広がる可能性のある 6 つの行列要素の加算が必要であったため、当時の GPU が苦手としていた排他的な行列加算を多数回行う必要があり、これを回避するための特別なアルゴリズムが要求されていた。今回の 4 中心 ESP 計算ではこの行列への加算部分がクーロン項の 1 要素のみであるため、比較的容易に行列加算が可能である。

具体的な実装方針としては、フラグメント A のシェルペアについてのループを GPU のブロックに配分し、フラグメント B についてのシェルペアループをブロック内のスレッドに配分する並列化を行った。この際 Fock 行列計算の GPGPU 化と同様に、シェルペアの並べ替えや Schwarz 不等式によるスクリーニングプロセスの分離、さらにはブロック間での動的負荷分散などを適用した。

このような並列化のもとでは、同一ブロック内の全てのスレッドが同じ行列要素に加算していくことになる。これはフラグメント B についてのシェルペアループ終了後に同一ブロック内スレッドで加算要素のリダクション処理を行うことで置き換えられるが、このリダクション処理については共有メモリの利用やシャッフル演算などのアルゴリズムが知られており、それを活用することで高速なリダクションが可能となる。一方、異なるブロックでは必ず異なる行列要素に加算することも容易にわかる。このため結果の行列は GPU 全体でただ一つだけ持てば良く、各ブロックのマスタスレッドがこの行列に単純加算する形で実装できる。

性能評価

実装した GPGPU 化 4 中心 ESP 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[5] 2 ノードを用いて行った。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 QDR2 ポートにより接続されている。また複数 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た。コンパイルや実行には Intel コンパイラ 15.0.2, CUDA 6.5.14, IntelMPI5.0 をそれぞれ利用した。OpenFMO は耐故障性ミドルウェアである falanx により実装されたものを利用した。

性能評価としてアラニンの 10 量体(112 原子、5 フラグメント)の FMO-HF/6-31G(d)計算を取り上げ、この計算の 4 中心 ESP 計算[(ss,ss)タイプ]についての実行時間を測定した。フラグメント間での並列化の影響を除外するため、ワーカ数を 1 とした場合の経過時間を測定し、GPGPU 化による高速化率を CPU 1 コア比として求めた(Table 1)。Falanx 版 OpenFMO を 2 ノード(8MPI プロセス)で起動させた場合にはワーカ内の MPI プロセス数は 4 となるが、この時の GPGPU 化による高速化率は 19.6-26.5 であった。Fock 行列の GPGPU 化に場合と比較した場合に、SCC 計算については高速化率が低い値となっている。これは FMO 計算で計算したフラグメントのサイズが小さいことが理由だと考えられる。GPU は非常に多くの並列度を持つため、これを効率良く利用するにはそれ以上の並列度が計算自体に要求される。このケースではたかだか 20 原子程度のフラグメント計算を 4 台の GPU を利用して並列計算しているため、計算の並列度が不足していると考えられる。そこでワーカのサイズを 1 MPI プロセスとした場合にも同様の性能評価を行なった(Table 1 右)。この場合にはフラグメントモノマーを計算する SCC 部分でも大きな高速化を得られていることがわかる。

今後の超並列計算機の利用を考える上ではワーカサイズは大きくせざるをえないため、今回見られた性能の劣化を防ぐ手法の開発が求められる。このため近接フラグメントが複数存在することを考慮して、複数のフラグメント-フラグメント間 ESP 計算をワーカ内で並列に計算することも検討中である。

Table 1 Elapsed time[s] and speedups [from CPU 1core] of (ss,ss)-type 4-center ESP calculation for FMO calculation of (Ala)10 (112 atom, 5 fragments)

	4 MPI / worker			1 MPI / worker		
	Elapsed Time /s		SpeedUps	Elapsed Time /s		SpeedUps
	CPU (4 core)	GPU	/CPU (1core)	CPU (4 core)	GPU	/CPU (1core)
SCC	12.0	2.5	19.6	42.3	7.3	23.2
Dimer SCF	3.4	0.5	26.5	11.9	1.8	26.8
ES Dimer	0.3	0.0	25.8	0.9	0.1	28.4

謝辞: 本研究の一部は 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．

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] 梅田宏明，塙敏博，庄司光男，朴泰祐，重田育照, J. Comp. Chem. Japan, **13**, 323(2015).
[5] HA-PACS ベースクラスタ;
http://www.ccs.tsukuba.ac.jp/research/research_promotion/project/ha-pacs/cluster