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^{a*}, 塙 敏博^b, 庄司 光男^a, 朴 泰祐^a, 重田 育照^a

^a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)

^b東京大学 情報基盤センター, 〒277-8589 柏市柏の葉5-1-5)

*umeda@ccs.tsukuba.ac.jp

(Received: July 20, 2015; Accepted for publication: August 17, 2015; Online publication: September 25, 2015)

GPU acceleration of four-center (4C) inter-fragment Coulomb interaction term (IFC) for OpenFMO, a fragment molecular orbital calculation program, has been implemented and its performance was examined. FMO calculation has two time-consuming steps: Fock matrix construction and IFC calculation, and in our previous letter, it was reported that the former is successfully accelerated with our GPU-enable code. The 4C-IFC calculation is the core part of the latter and its code is similar to that of Fock matrix construction. In this letter, we briefly describe the GPU-accelerated 4C-IFC calculation routine, and report a performance benchmark for GPU-accelerated FMO calculation. The GPU-accelerated program shows $3.3\times$ speedups from CPU only FMO-HF/6-31G (d) calculation for 642 atomic protein on 8 nodes of HA-PACS base cluster.

Keywords: GPGPU, FMO, Inter-fragment Coulomb interaction, Molecular orbital calculation, Performance benchmark

1 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 [1,2]を行ってきた。さらにフラグメント分子軌道(FMO)計算 [3]プログラムの大規模計算機向け実装であるOpenFMO [4]にGPGPU化Fock行列計算コードを導入する [5]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。前回報告したように、GPUによるFMO計算の高速化では環境静電ポテンシャル(ESP)計算の高速化も重要な要素になっている。本研究ではESP計算のコアとなる4中心クーロン相互作用項(4C-IFC)計算についてGPGPU化を行い、GPGPU化FMO計算の性能を評価した。

2 実装

二つの近接フラグメント間のクーロン相互作用を計算する4C-IFC計算のアルゴリズムは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。大きな違いは計算されたクーロン項の行列への加算手順にある。交換相関項のある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要素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 [1]。今回の4C-IFC計算ではこの行列への加算部分が単純化されるため、積分タイプが増えるものの個々の実装は比較的容易である。

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外側ループ(ij)をGPUのブロックに配分し、フラグメントBについての内側シェルペアループ(kl)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 $(\mathbf{V}_A)_{ij}$ に加算してい

Table 1. Elapsed time [sec.], and speedups or parallelization efficiency for FMO-HF/6-31G (d) calculation of crambin (642 atoms, 20 fragments) and lysozyme (1,961 atoms, 57 fragments) on HA-PACS base cluster.

	Crambin				Lysozyme		
	CPU	CPU	CPU+GPU		CPU+GPU	CPU+GPU	
	Direct	In-core	Direct	Speedups	Direct	Direct	Efficiency
#node	8	8	8		16	32	
SCC	629.8	632.2	207.1	3.0	401.5	244.9	0.82
Dimer SCF	1,266.2	857.3	345.5	3.7	810.1	455.1	0.89
ES Dimer	43.3	42.5	10.5	4.1	52.4	30.0	0.87
Total	1,961.1	1,586.0	590.8	3.3	1,295.1	828.1	0.78

くことになる。これはフラグメントBについての k ルシェルペアループ終了後に同一スレッドブロック内でのリダクション処理(加算)により実装できる。一方それぞれのブロックでは異なる ij について計算するため、行列要素 $(\mathbf{V}_A)_{ij}$ への加算がブロック間で競合することもない。このため行列 $\mathbf{V}_A$ への加算は排他制御なしに実現可能であり、GPU全体で行列 $\mathbf{V}_A$ を共有することが可能である。

3 性能評価と考察

GPGPU化4C-IFC計算コードをOpenFMOプログラムに実装し、GPGPU化FMO計算の性能評価を行った。性能評価には筑波大学のHA-PACSベースクラスター [6] を用いている。HA-PACSベースクラスターの計算ノードには2台の8コアCPU (Intel Xeon E5-2680, 2.6 GHz)と4台のGPU (NVIDIA M2090), 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をそれぞれ利用している。

性能評価としてクランビン(642原子, 20フラグメント)のFMO-HF/6-31G (d)計算をとりあげた。計算には8ノード(128 CPU core + 32 GPU)を利用し、6 MPIプロセスのワーカグループ5グループによる並列計算を行った。Table 1に評価計算の経過時間および性能向上比を示した。表にはFMO計算の主な三つの計算手順(SCC計算, ダイマー SCF計算, 近似ダイマー (ES dimer)計算), およびFMO計算全体に対するマスタプロセスにおける経過時間を示している。in-core計算手法ではノードあたり64GBを積分保存のために利用した。GPGPU化FMO計算は全ての計算手順において3倍以上の性能向上を見せており、FMO計算全体でもCPUによる直接計算手法の3.3

倍の高速化を実現している。Fock行列計算部分のみのGPGPU化では1.5倍の高速化であり [5], 今回の4C-IFC計算のGPGPU化が効果的であったことがわかる。

より大きなFMO計算に向けてリゾチーム(1,961原子, 57フラグメント)のFMO-HF/6-31G (d)計算を評価した。計算は16ノードと32ノードのケースについて15ワーカグループに固定した構成で行った。ワーカグループサイズは4および8 MPIプロセスとなり、ワーカグループ内での強スケール並列性能の評価となる。Table 1に経過時間と並列化効率を示した。ダイマー SCF計算や近似ダイマー計算に比べ、SCC計算はやや低い並列化効率となっている。SCC計算では比較的小さいフラグメントモノマーの計算をするため並列度が不足していると考えられ、より高度な並列化の必要性も示唆される。しかしながらFMO計算全体で0.78の並列化効率は十分高いものであり、今後の大規模FMO計算の実現が期待できる。

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] H. Umeda, T. Hanawa, M. Shoji, T. Boku, Y. Inadomi, *IPSP Trans. Adv. Comput. Syst.*, **6**, 26 (2013).
- [2] H. Umeda, T. Hanawa, M. Shoji, and T. Boku, HPC in Asia poster presentation in ISC'13, 19a (2013).
- [3] K. Kitaura, T. Sawai, T. Asada, T. Nakano, M. Uebayasi, *Chem. Phys. Lett.*, **312**, 319 (1999). [[CrossRef](#)]
- [4] OpenFMO, <http://www.openfmo.org/>
- [5] H. Umeda, T. Hanawa, M. Shoji, T. Boku, Y. Shigeta, *J. Comput. Chem. Jpn.*, **13**, 323 (2014). [[CrossRef](#)]
- [6] HA-PACS Base Cluster, http://www.ccs.tsukuba.ac.jp/research/research_promotion/project/ha-pacs/cluster