Vol.2016-MPS-108 No.17
Vol.2016-BIO-46 No.17
2016/7/5

SHAKE法に着目した 分子動力学ソフトウェアmyPresto/Omegageneの CPU・GPU混在環境における高速化

後藤 公太[1,2,a)]　笠原 浩太[3,b)]　大上 雅史[1,4,c)]　中村 春木[5,d)]　秋山 泰[1,2,4,e)]

概要：近年，タンパク質やDNA，脂質などの解析を行うために，分子動力学シミュレーションが広く用いられており，このようなシミュレーションを行うソフトウェアに，我が国で開発されているmyPresto/Omegageneがある．このソフトウェアは部分的にGPU実装が行われている一方，シミュレーション中の拘束系の運動方程式を解くSHAKE法の並列化は行われていなかった．SHAKE法はCPUによる計算時間のうち1/3を占めており，これを並列化することによる速度向上の寄与は大きいことが期待される．本研究ではSHAKE法について，CPUのみ，GPUのみ，CPU・GPU併用の3通りの並列化実装を行い，最大4.28倍の高速化率を得た．また，SHAKE法の計算でボトルネックになっている部分の解析を行い，高速化率が向上する条件について検討した．

キーワード：分子動力学法, SHAKE, OpenMP, GPU

Accerelating SHAKE Calculation of myPresto/Omegagene Molecular Dynamics Simulation on CPU/GPU heterogeneous environment

Kota Goto[1,2,a)]　Kota Kasahara[3,b)]　Masahito Ohue[1,4,c)]　Haruki Nakamura[5,d)]
Yutaka Akiyama[1,2,4,e)]

Abstract: Recently, Molecular dynamics (MD) simulation is widely used in order to analyze biomolecules (e.g. protein, DNA and lipid). myPresto/Omegagene, MD simulation software, is partially GPU-accelerated but SHAKE calculation in the software is not parallelized. SHAKE calculation spends a one-third of the whole calculation time on CPU, thus parallelization of it is effective to accelerate the calculation of the software. In this study, we parallelized SHAKE calculation in three ways: CPU based parallelization, GPU based parallelization and CPU-GPU co-working parallelization. Our implementation increased the calculation speed 4.28 fold compared to serial execution. We also analyzed the bottleneck of the calculation in order to find rooms to improve.

Keywords: Molecular Dynamics, SHAKE, OpenMP, GPU

[1] 東京工業大学 情報理工学院 情報工学系
Department of Computer Science, School of Computing, Tokyo Institute of Technology
[2] 東京工業大学 情報生命博士教育院
Education Academy of Computational Life Sciences, Tokyo Institute of Technology
[3] 立命館大学 生命科学部
College of Life Sciences, Ritsumeikan University
[4] 東京工業大学 科学技術創成研究院 スマート創薬研究ユニット
Advanced Computational Drug Discovery Unit, Institute of Innovative Research, Tokyo Institute of Technology
[5] 大阪大学 蛋白質研究所,
Institute for Protein Research, Osaka University
a) goto@bi.cs.titech.ac.jp
b) ktkshr@fc.ritsumei.ac.jp
c) ohue@c.titech.ac.jp
d) harukin@protein.osaka-u.ac.jp
e) akiyama@c.titech.ac.jp

Vol.2016-MPS-108 No.17
Vol.2016-BIO-46 No.17
2016/7/5

表 1　myPresto/Omegagene の計算時間.
測定には Trp-Cage (総原子数 13277) を用いた.

計算内容		計算時間 [sec]	割合	全体割合
CPU 処理	SHAKE 法	85.56	36%	18%
	原子が受ける力の計算	87.45	37%	
	その他	64.22	27%	
GPU 処理	LJ ポテンシャル計算	1,006	94%	82%
	その他	63.44	6%	

1. 導入

分子動力学 (MD) シミュレーションは，分子集団の構造や性質を解析するために，コンピュータを用いて原子の軌跡を求める手法の 1 つであり，タンパク質，核酸，脂質やそれらの複合体などの生体高分子の動態を理解する目的で利用されている [1–3]．近年では GROMACS [4–6]，AMBER [7]，NAMD [8]，Desmond [9]，myPresto [10] などの MD シミュレーションソフトウェアが開発されている．

MD シミュレーションは多数の運動方程式を短い時間刻み幅で数値的に解く必要がある．有益な解析結果を得るためには多くのステップ数を計算する必要があるため，この計算には長い時間がかかる．時間刻み幅を長く取ることができれば総計算時間を短縮できるが，系全体のエネルギー保存を正しく計算するためには 1 フェムト秒程度の刻み幅が要求される．これを解決する方法として，振動している原子間距離を固定して扱うことにより，効率的に原子の座標を求める SHAKE 法 [11] が広く用いられている．SHAKE 法を用いることで．系全体のエネルギー保存を 2 フェムト秒程度の刻み幅でも十分な精度で保ちつつ計算を行うことができる．本研究で対象とする MD シミュレーションソフトウェア myPresto/Omegagene においても SHAKE 法が用いられている．myPresto/Omegagene は，我が国で開発されている分子シミュレーションシステムである myPresto のソフトウェア myPresto/psygene [12] の後継として開発されている．myPresto/Omegagene は計算処理の一部は GPU 化されているが，SHAKE 法の計算は並列化されておらず改善の余地がある．そこで，本研究では並列化によって SHAKE 法の高速化を行うことを目的とする．

2. 手法

2.1 対象とする計算

図 1 に myPresto/Omegagene の計算処理の概要を示す．また，**表 1** に myPresto/Omegagene の主な処理の計算時間を示す．表 1 の計算に用いた計算機環境とパラメータを**表 2** に示す．

原子間の相互作用を表すレナード-ジョーンズ（LJ）ポテンシャルの計算，相互作用する原子のペア生成などの処理を GPU で行っている． 表 1 に示す計算時間の内訳を

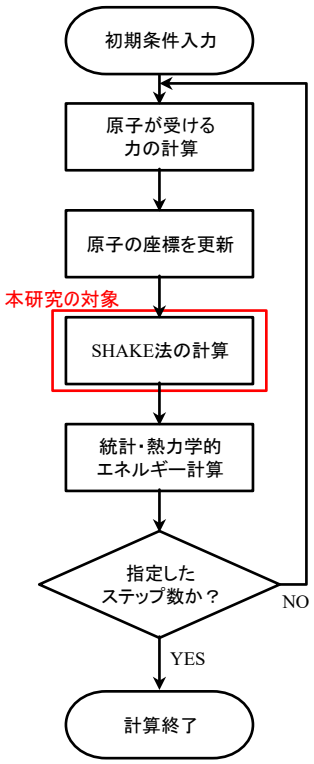

図 1　myPresto/Omegagene の計算処理の概要

表 2　実験に使用した計算機環境とパラメータ

CPU	Intel(R) Xeon(R) CPU E5-1630 v3 3.7 GHz 4 コア 8 スレッド
GPU	NVIDIA Tesla K40
メモリ	32 GB
OS	CentOS 6.5
コンパイラ	gcc 4.4.7 (オプション -O3) CUDA 6.5
統計アンサンブル	NVE
静電相互作用	ZD 法 [18–20]
カットオフ距離	12 Å
タイムステップ	2 フェムト秒

見ると全体の 82%を GPU が占めているが，この値は今後の GPU アーキテクチャの発展に伴って減ることが予想され，相対的に CPU の割合が増加すると考えられる．特に myPresto/Omegagene で用いられている実装は計算密度が高く，将来の GPU に対してもスケールすることが期待できる [13]．したがって，さらなる高速化のためには CPU 計算部分の並列化が求められる．現在の myPresto/Omegagene の実装では CPU の計算処理は全て CPU 1 コアのみで行われているため，この並列化は将来的なボトルネックの回避に繋がると考えられる．CPU での計算時間は SHAKE 法の計算とエネルギー計算がほぼ同じであるため，本研究では SHAKE 法の計算に着目して，myPresto/Omegagene の高速化を行った．

表 3　本研究で実装した提案手法

提案手法	使用する計算資源	内容
1	CPU	原子組ごとの CPU による並列化
2-1	GPU	原子組ごとの GPU による並列化
2-2	GPU	原子組内での GPU による並列化
3	CPU + GPU	CPU と GPU の両方を用いる並列化

表 4　提案手法のメリットとデメリット

提案手法	メリット	デメリット
1	実装コストが最も低い	高速化率の上限が CPU スレッド数依存
2-1，2-2	GPU による大規模並列計算	1 スレッドの能力は低いため一部の計算により低速化する可能性
3	多くの計算資源を使える	CPU 側がボトルネックになる可能性

2.2　改良前の SHAKE 法の実装

myPresto/Omegagene の元の実装では，SHAKE 法の全ての計算を CPU 1 コアで行っている．**図 2** に処理の概念図を示す．SHAKE 法は 2 個の原子（2 原子組），3 個の原子（3 原子組），4 個の原子（4 原子組）について適用される．2 原子組に SHAKE 法を適用する場合，1 個の拘束条件を満足する次の時刻の座標を求めるため，運動方程式を 1 回解いて座標の変位を得ることで終了する．一方で 3 原子組は 3 個，4 原子組は 6 個の拘束条件を満足する座標を求めなければならない．複数個の運動方程式を解いて得られた変位を順番に適用させ，全ての拘束条件を十分な精度で満足するまでこれを繰り返す．この繰り返しはほとんどの場合 3 回で終了するため，2 原子組の計算時間と比べると平均して 3 原子組は約 9 倍，4 原子組は約 18 倍の計算時間になる．

2.3　提案手法

SHAKE 法の計算を高速化するための手法として，**表 3** に示す 4 つの手法を実装した．また，これらの提案手法について考えられるメリットとデメリットを**表 4** に示した．SHAKE 法の計算は異なる原子組の間で計算の依存関係は無く，独立に計算することができる．そのため並列計算と相性がよく，大規模並列計算が可能な GPU を用いることで高速に計算を行えると考えられる．

2.3.1　提案手法 1: 原子組ごとの CPU による並列化

原子組間での計算の独立性を利用して並列化を行う．CPU 上での並列化の実装には OpenMP [14] を用いる．4 つの手法の中で最も実装コストが低いが，最大でも CPU の利用可能なスレッド数が高速化率の上限となる．**図 3** に処理の概念図を示す．

2.3.2　提案手法 2-1: 原子組ごとの GPU による並列化

提案手法 1 の CPU による並列化と同様に，原子組間で計算の依存関係がないことを利用して並列化を行った．GPU の 1 スレッドを用いて 1 つの原子組の計算を行うよう実装した．GPU 上での計算の実装には CUDA 6.5 [15] を利用した．**図 4** に処理の概念図を示す．

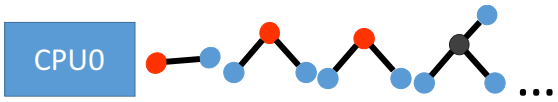

図 2　改良前の SHAKE 法実装の概念図

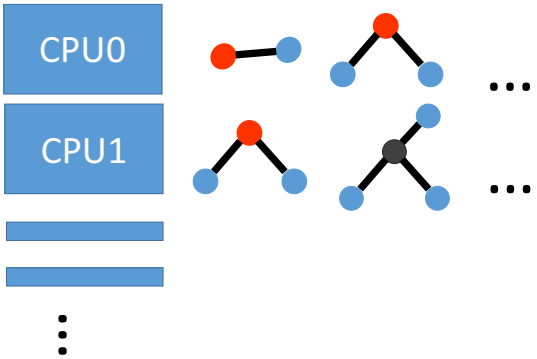

図 3　提案手法 1: CPU による並列化の概念図

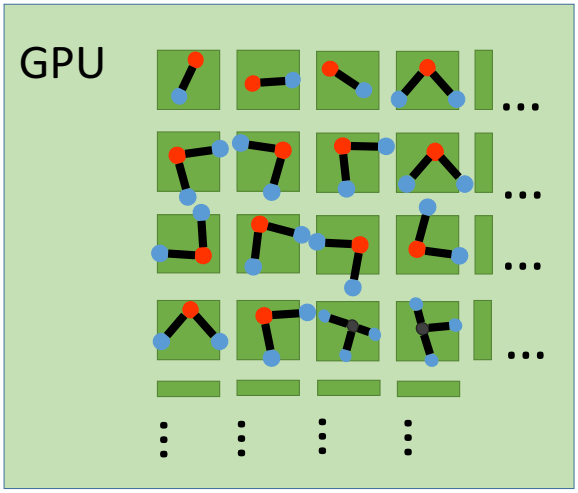

図 4　提案手法 2-1: GPU による原子組ごとの並列化の概念図

GPU による並列化では次のような段階を経て SHAKE 法の計算が行われる．

(1) 座標入力（CPU→GPU メモリ転送）
　CPU のメモリで持つ，原子の座標データを転送する．

(2) 原子組の計算
- 2 原子組の計算
- 3 原子組の計算
- 4 原子組の計算

　これらは GPU 側の計算資源が許す限り，オーバラップして計算される．

Vol.2016-MPS-108 No.17
Vol.2016-BIO-46 No.17
2016/7/5

表 5　実験に用いた系の詳細

タンパク質	Trp-Cage	RNAP
アミノ酸残基数	20	3,278
水分子数	4,316	205,139
総原子数	13,277	670,957
計算ステップ数	100,000	1,000
2 原子組数	50	9,012
3 原子組数	4,354	210,437
4 原子組数	8	2,281

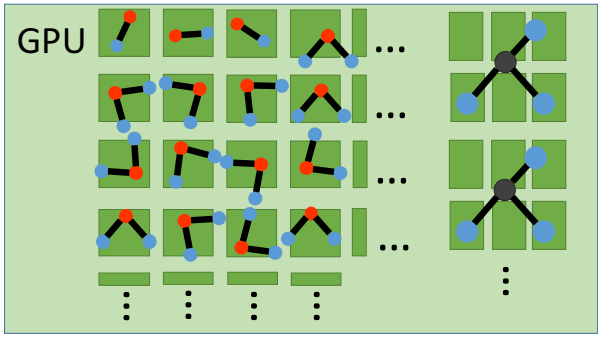

図 5　提案手法 2-2: GPU による原子組内での並列化の概念図

(3) 結果出力（GPU→CPU メモリ転送）
GPU 側で計算された新しい原子の座標データを CPU 側へ転送する．

このように CPU と GPU 間で座標情報を転送する必要があり，真に SHAKE 法の計算を行う以外の処理も掛かるため，計算速度が低下することが考えられる．

2.3.3　提案手法 2-2: 原子組内での GPU による並列化

表 5 に示すように，4 原子組はほかの 2 原子組，3 原子組と比べて数が少なく，最大でも約 1%である．提案手法 2-1 では原子組が多数存在することを利用して並列化を行っているため，存在数が比較的少ない 4 原子組では，十分な並列化ができない可能性がある．また，4 原子組について SHAKE 法を適用する場合，6 個の運動方程式を解く必要があるため，この計算を GPU 1 スレッドで行った場合は逆に低速化の原因になりうる．これを解決するために，複数の GPU スレッドを用いて 1 個の原子組について SHAKE 法を適用する実装を提案する．運動方程式を解く場合に，他の運動方程式とは独立に解くことができるため，運動方程式の数と同じだけのスレッド数を用いる．すなわち 3 原子組の場合に 3 スレッド，4 原子組の場合は 6 スレッドとなるが，3 原子組は十分に存在数が多いため，原子組ごとの並列化のみであっても効果があると考え，4 原子組に対してのみ実装を行った．図 5 に処理の概念図を示す．

2.3.4　提案手法 3: CPU と GPU の両方を使用する並列化

提案手法 2-1，2-2 について事前に実験を行った結果，計算方法にかかわらず 4 原子組の計算がボトルネックになっていることがわかった．この問題の解決策として，GPU で 2 原子組と 3 原子組の計算をすると同時に CPU で 4 原子組の計算を行う．一般に MD シミュレーションの行われる系では水溶媒が選択されることが多く，3 原子組の存在数が多い．よって，CPU 上で行われる 4 原子組の計算がボトルネックになる可能性は低いため，有効な手法と考えられる．図 6 に処理の概念図を示す．

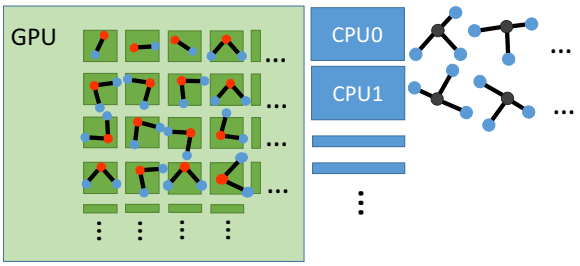

図 6　提案手法 3: CPU と GPU 両方による並列化の概念図

3.　評価実験

3.1　評価方法

実験に使用した計算機環境と，myPresto/Omegagene に入力したパラメータは表 2 に示したとおりである．シミュレーションの対象となる系として本研究では表 5 に示す，大小の 2 種類の系を用意した．小さな系としてトリプトファンケージ [16,17]（Trp-Cage），大きな系として RNA ポリメラーゼ（RNAP）を用いた．表 5 に示すステップ数の MD シミュレーションを myPresto/Omegagene を用いて行った．計算時間については，SHAKE 法の計算にかかった時間のみを 3 回測定し，その中央値を示した．

3.2　結果

3.2.1　SHAKE 法の計算時間

図 7 に SHAKE 法の計算時間と高速化率を示す．Trp-Cage，RNAP のどちらの系に対しても，提案手法 1 の CPU を用いたスレッド並列化が最も良い結果となった．

3.2.2　計算時間の内訳

GPU を用いた場合，どの方法においても CPU を用いた場合よりも高速化率は向上しなかった．この原因を特定するために各手法について，計算時間の内訳を測定する実験を行った．図 8 に計算時間の内訳を示す．図 8(A) Trp-Cage については 100,000 ステップで測定された時間を 100 で割り，1,000 ステップあたりの計算時間を示した．

なお，提案手法 2-1, 2-2, 3 について座標入力にかかる時間が N/A となっている．これは GPU での計算時間の内訳を計測する際，座標入力のみで SHAKE 全体の約 98%を占める現象が発生し，正しく測定できなかったためである．座標入力の転送量は結果出力の約 2.3 倍であるため，約 4,600 ミリ秒になることが予想されるが，この 50 倍以

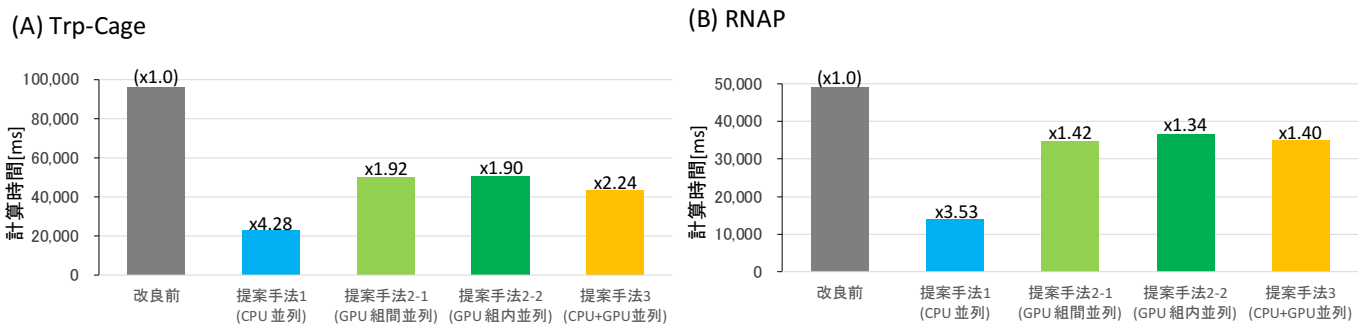

図 7　各実装ごとの SHAKE 法の計算時間．(A) Trp-Cage，(B) RNAP．グラフ内の数値は改良前に対する速度向上率．

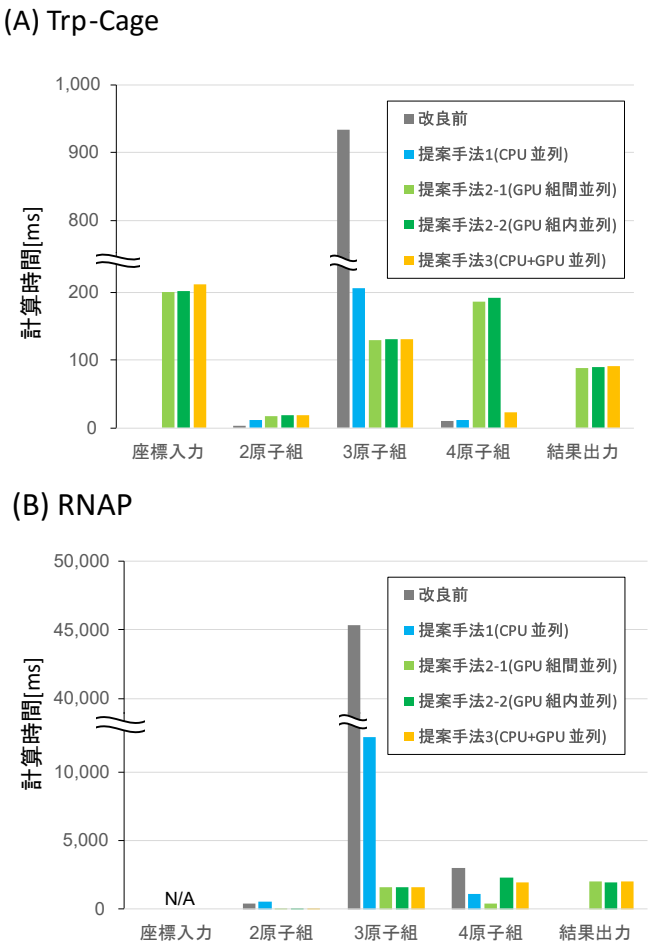

図 8　各実装ごとの SHAKE 法の計算時間の内訳．(A) Trp-Cage，(B) RNAP．N/A は正しく測定できなかったデータを表す．

上の異常な値が測定された．これは無効な値とし図中では N/A と表した．

4. 考察

提案手法 1 は提案手法の中で最も実装コストが低く，効果のある手法となった．以下に SHAKE 法の計算方法の，各段階ごとの考察を述べる．

4.1 座標入力および結果出力について

図 8 によると，GPU 上での計算は図 8(A) Trp-Cage の場合で約 45%，図 8(B) RNAP の場合は約 75%がメモリ転送にかかっている状態であった．現在の実装では SHAKE 法の適用が本来必要でないデータ，すなわち表 5 に示すどの原子組にも含まれない原子の座標データも転送している．しかし，Trp-Cage では全 13,277 原子中，13,194 原子が，RNAP では 670,957 原子中，649,447 原子が SHAKE 法の対象となるため，転送データの最適化によって転送時間は減ったとしても僅かであり，転送のメモリアクセスがランダムなアクセスになってしまう影響により，低速化してしまうと考えられる．myPresto/Omegagene は CPU で行った場合に，計算時間が長くなる一部の処理のみを GPU で行っている．その他の処理は CPU で行っているため，CPU と GPU の間でのデータ転送を行っている．これについて，全ての処理を GPU で行うことができれば，CPU と GPU の間でのデータ転送の必要がなくなるため，メモリ転送にかかる時間は 0 となる．この場合 Trp-Cage で約 68%，RNAP で約 46%の計算時間となる．

4.2 2 原子組，4 原子組について

提案手法 2 で計算した 4 原子組を除いて，どの手法でも改良前が最も良い結果となった．提案手法 3 のような，原子組内での並列化はすでに報告されている [21, 22] が，今回の実験では高速化にはつながらなかった．同じ原子組を計算する，6 スレッドの間でデータを交換する部分がボトルネックになってしまったことが原因だと考えられる．原子組内での並列化を行わなかった場合，RNAP では高速化の効果があったが，Trp-Cage では効果が無かった．Trp-Cage の 4 原子組は 8 組しか存在せず，CPU と GPU 両方において十分な並列数が得られなかったことが原因と考えられる．

4.3 3 原子組について

どの手法においても 3 原子組の計算時間は改善前よりも良い結果となった．特に，GPU を用いた並列化では大き

い系である RNAP で約 27 倍，小さい系である Trp-Cage で約 7 倍の高速化を達成した．3 原子組数が多い RNAP に比べると Trp-Cage の高速化率が低く，さらなる高速化の余地があると考えられる．

結果のプロファイルによると，3 原子組の計算は使用レジスタ数が原因で並列計算効率は理想値の 1/8 にとどまっている．これは 3 原子組の計算を行う際の GPU 内使用レジスタ数が多いことに原因がある．実装を改善してレジスタ使用数を減少させることで，計算効率を高めてさらなる高速化が達成できると考えられる．

5. まとめ

5.1 本論文のまとめと結論

本研究では，分子動力学シミュレーションソフトウェア，myPresto/Omegagene の改良を SHAKE 法の計算に着目することで行った．SHAKE 法を適用する原子組の計算は独立であることを利用して，CPU と GPU の両方で並列化を行い，その評価を行った．本研究のまとめを以下に示す．

- SHAKE 法の計算は提案手法 1 の CPU による並列化が良い結果となった．ただし，2 原子組に関しては低速化してしまったため，3 原子組と 4 原子組を原子組ごとに並列化すると最も計算時間が短くなる．
- GPU による並列化はいずれの方法もメモリ転送が計算時間の大半を占めているため，現状の myPresto/Omegagene の処理では高速化率が CPU よりも低い．しかし，SHAKE 法の計算を真に行っている時間は短いため，myPresto/Omegagene の他の処理が GPU 上で行われるようになれば，効果を発揮すると考えられる．

5.2 今後の課題

今後の課題として次の 2 つが挙げられる．

- **CPU 上での処理の GPU 化**
  SHAKE と同等に CPU での計算時間がかかっていたエネルギー計算や，原子の座標更新などの処理を GPU 側で行うことで，その処理自体の高速化と，CPU と GPU の間でのデータ転送にかかる時間の削減により，計算速度の向上が期待できる．
- **3 原子組計算の使用レジスタ数削減**
  もっとも出現する組数の多い，3 原子組の GPU 上での計算効率を高めるためには，使用レジスタ数を削減する必要がある．これにより，どのようなサイズの系に対しても GPU の利用効率を向上させることができる．

謝辞 本研究の一部は JST CREST 「EBD: 次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」の支援を受けて行われた．

参考文献

[1] M. Karplus, J. Kuriyan: “Molecular dynamics and protein function”, *Proc. Natl. Acad. Sci.*, (2005), **102**, 6679–6685.

[2] T. Bastug, S. Kuyucak: “Molecular dynamics simulations of membrane proteins”, *Biophys. Rev.*, (2012), **4**, 271–282.

[3] D. E. Shaw, P. Maragakis, K. Lindorff-Larsen, *et al.*: “Atomic-Level Characterization of the Structural Dynamics of Proteins”, *Science*, (2010), **330**, 341–346.

[4] D. van der Spoel, E. Lindahl, B. Hess, *et al.*: “GROMACS:Fast, Flexible, and Free”, *J. Comput. Chem.*, (2005), **26**, 1701–1718.

[5] B. Hess, C. Kutzner, D. van der Spoel,*et al.*: “GROMACS4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation”, *J. Chem. Theory Comput.* , (2008), **4**, 435–447.

[6] S. Pronk, S. Pall, R. Schulz, *et al.*: “GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit”, *Bioinformatics*, (2013), **29**, 845–854.

[7] D. A. Case, J. T. Berryman, R. M. Betz, *et al.*: AMBER 2015, University of California, San Francisco.

[8] J. C. Phillips, R. Rraun, W. Wang, *et al.*: “Scalable moleculer dynamics with NAMD”, *J.Comput.Chem.*, (2005), **26**, 1781–1802.

[9] K. J. Bowers, E. Chow, H. Xu, *et al.*: “Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters”, *Proceedings of the ACM/IEEE Conference on Supercomputing (SC06)*, (2006), Tempa, Florida, November 11–17

[10] Y. Fukunishi, Y. Mikami, H. Nakamura: “The filling potential method: A method for estimating the free energy surface for protein-ligand docking”, *J. Phys. Chem. B.*, (2003), **107**, 13201–13210.

[11] J. RyckAert, G. Ciccotti, H. J. C. Berendsen: “Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of n-Alkanes”, *J. Comput. Phys.*, (1977), **23**, 327–341.

[12] T. Mashimo, Y. Fukunishi, N. Kamiya, *et al.*: “Molecular Dynamics Simulations Accelerated by GPU for Biological Macromolecules with a Non-Ewald Scheme for Electrostatic Interactions”, *J. Chem. Theory Comput.*, (2013), **9**, 5599–5609.

[13] S. Pall, B. Hess: “A flexible algorithm for calculating pair interactions on SIMD architectures”, *Comput. Phys. Commun.*, (2013), **184**, 2641–2650.

[14] “OpenMP” http://openmp.org/wp/

[15] “NVIDIA CUDA” http://www.nvidia.co.jp/object/cuda-

jp.html

[16] J. W. Neidigh, R. M. Fesinmeyer, N. H. Andersen: "Designing a 20-residue protein", *Nat. Struct. Biol.*, (2002), **9**, 425–430.

[17] C. D. Snow, B. Zagrovic,V. S. Pande: "The Trp Cage: Folding Kinetics and Unfolded State Topology via Molecular Dynamics Simulations",*J. Am. Chem. Soc.*, (2002), **124**, 14548–14549

[18] I. Fukuda, Y. Yonezawa, H. Nakamura: "Molecular dynamics scheme for precise estimation of electrostatic interaction via zerodipole summation principle", *J. Chem. Phys.*, (2011), **134**, 164107.

[19] I. Fukuda, N. Kamiya, Y. Yonezawa, *et al.*: "Molecular dynamics scheme for precise estimation of electrostatic interaction via zerodipole summation principle", *J. Chem. Phys.*, (2012), **137**, 054314.

[20] N. Kamiya, I. Fukuda, H. Nakamura: "Application of zero-dipole summation method to molecular dynamics simulations of a membrane protein system", *Chem. Phys. Lett.* , (2013), **568-569**, 26–32.

[21] R. Elber, A. P. Ruymgaart, B.Hess: "SHAKE Parallelization", *Eur. Phys. J. Sperial Topics*, (2011), **200**, 211–223.

[22] A. P. Ruymgaart, R. Elber: "Revisitng Molecular Dynamics on a CPU/GPU system:Water Kernel and SHAKE parallelization", *J. Chem. Theory Comput.*, (2012), **11**, 4624–4636.