Vol.2012-HPC-136 No.9
2012/10/3

CPU-アクセラレータ間の負荷分散による分子動力学法の効率化

宇田川 拓郎[1]　関嶋 政和[1,2]

概要：近年，GPU をはじめとするアクセラレータを利用する計算システムが増加している．アクセラレータを用いることで計算が大幅に高速化された例が多数報告されており，その有用性は十分に示されている．しかし，CPU あるいはアクセラレータのいずれかに負荷が過度に集中した場合，リソースが十分に活用されない問題が発生してしまう可能性がある．本研究ではそのような事例を分子動力学法のアプリケーションプログラムである NAMD を用いて提示し，CPU-GPU 間で負荷の分散を行うことで，リソースをより有効に活用できることを示す．

1. はじめに

近年，アクセラレータを利用する計算機クラスタやスーパーコンピュータが増加してきている．TOP500 リスト [1] によると，アクセラレータを利用する計算システムが上位にランクインし，リスト全体に占める割合も増加の傾向にある．中でも特に，GPU (Graphics Processing Unit) の利用が盛んである．GPU は本来画像処理に特化したデバイスであったが，高い並列計算性能と電力効率を持ち，汎用計算に利用されるようになった．NVIDIA 社が CUDA (Compute Unified Device Architecture) を発表してからは，その動きが特に加速されている．

こういったアクセラレータを利用する際には，従来の計算で計算負荷が高かった部分をアクセラレータに任せ，高速で処理させて計算全体を高速化する．アクセラレータの計算は原則 CPU と非同期に実行ができるため，アクセラレータが計算している間は CPU がアイドル状態となる．そのため，アクセラレータと非同期に実行できる計算がある場合にはこの空き時間を利用できるが，そういった計算がない場合や，あってもその量が不十分である場合，あるいはアクセラレータ側の処理が先におわる場合などは，プロセッサがアイドルになる時間が生じる．

そこで，我々は CPU と GPU で同種の計算を行い，CPU-GPU 間で負荷分散を行うことで，計算全体を効率的に行う方法を提案する．このような方法は計算科学シミュレーションのように同じ計算を何度も繰り返す計算に特に向いていると考えられる．本研究では分子動力学法のアプリケーションプログラム NAMD[2] を例として，CPU と GPU の間で負荷の分散を行い，それによって生じる計算速度や電力効率の変化について結果を報告する．

[1] 東京工業大学大学院 情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
[2] 東京工業大学 学術国際情報センター
Global Scientific Information and Computing Center

2. 分子動力学法と NAMD

本節では分子動力学法と NAMD について説明を行う

2.1 分子動力学法

分子動力学法は古典力学に基づいて分子の運動の軌跡を求めるシミュレーション技法である．分子系を構成する原子間の相互作用を微小時間ごとに求め，数値積分法を用いて各原子の Newton の運動方程式を解き，分子系全体の時間発展を観測する．

一般的な分子動力学法の原子間の相互作用は式 (1)　で表される [3].

$$
\begin{aligned}
U(\mathbf{q}) = &\sum_{\text{bonds}} k^b (b - b_0)^2 + \sum_{\text{angles}} k^\theta (\theta - \theta_0)^2 \\
&+ \sum_{\text{torsions}} V_n [1 + \cos^2(n\phi - \phi_0)] \\
&+ \sum_{i,j \in \text{atoms}} \epsilon_{ij} \left[\left(\frac{r_{ij}^0}{r_{ij}} \right)^{12} - 2 \left(\frac{r_{ij}^0}{r_{ij}} \right)^{6} \right] + \sum_{i,j \in \text{atoms}} \frac{q_i q_j}{r_{ij}}.
\end{aligned}
\tag{1}
$$

このうち前半の 3 つの項は化学結合による力で原子数 N に対して $O(N)$ の計算量となる．後半の 2 つの項は分子間力と静電相互作用力で化学結合によらないため全ての原子間に定義される．そのため計算量は $O(N^2)$ となり，実際に

分子動力学法のシミュレーションの計算時間の大部分はこの力の計算が占め，様々な高速化のアルゴリズムが開発されている．最も単純な方法は非結合力が距離の逆乗で収束することを用いて，ある距離より遠方の原子の相互作用を無視するカットオフを利用する方法である．特に分子間力は距離の 12 乗，6 乗で収束する項を含んでいるため，極めて速く収束する．静電相互作用力は分子間力に比べ収束が遅いため，異なる高速化のアルゴリズムが適用される．代表的なものは PME(Particle Mesh Ewald) 法 [5] で，周期境界条件下の静電相互作用を周期関数で表し，電荷分布をグリッドに内挿する形に書き換え，FFT を用いて高速に計算する方法である．PME 法により静電相互作用の計算量は $O(NlogN)$ まで削減される．

2.2 NAMD

NAMD は分子動力学法を行うプログラムでイリノイ大学の研究チームが公開している．Charm++[4] で記述され，特に巨大分子系のシミュレーションで高い性能を発揮するようデザインされている．

NAMD ではシミュレーションを行う空間を patch と呼ばれる単位に分割し，プロセッサに分配する．patch の分割は隣接する 26 patch のみで化学結合，分子間力や静電相互作用の直接計算部が計算できる大きさになるように patch の大きさを定める．力の計算は計算する力ごとに compute object を定義して行われる．compute object はそれぞれ利用する patch が定められており，それに応じてプロセッサに分配される．この compute object はプロセッサをまたいで，移動させることが可能であり，計算の途中必要に応じて移動させられる．

NAMD のコードは既に GPU の利用もされている [6]．式 (1)　の後半 2 項にあたる部分を NAMD では GPU で計算させるようにしており，CPU のみの場合に比べ大幅に計算速度を向上させている．

3. プログラムの実装と電力測定方法

3.1 CPU-GPU 間の負荷分散

上述の通り，NAMD では長距離力の計算を GPU を用いて行っている．そこで，本研究では我々の提案手法に従い長距離力の計算の一部を CPU で行った．NAMD は，シミュレーションの初期化の際に各プロセスに割り振られた patch から長距離力の compute object を作る．その際に CPU で処理させる patch と GPU で処理させる patch に区別を行い，CPU で処理させる patch に起因する長距離力の compute object は CPU で計算を行い，それ以外の compute object は GPU で計算させる．CPU に割り当てる patch の割合は現段階では静的に決定しており，実験ではこの割合を変化させ，それによる計算効率の変化を観測した．

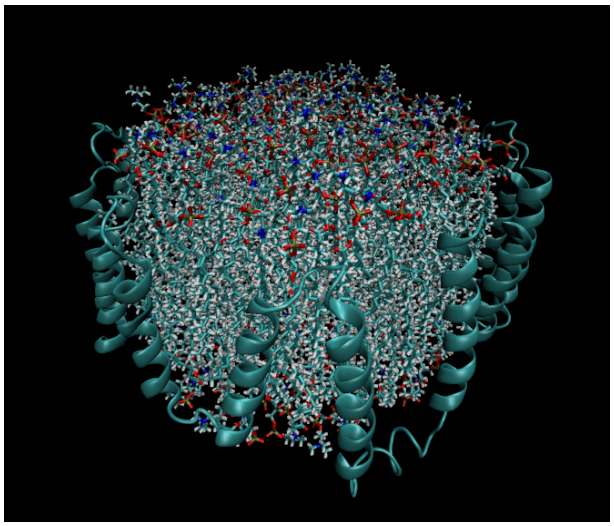


図 1　ApoA1

3.2 電力測定

本研究で用いた実験環境は東京工業大学学術国際情報センターが保有する TSUBAME2.0 スーパーコンピュータである．TSUBAME2.0 は日本初のペタフロップス級のスーパーコンピュータであり，初登場である 2010 年 11 月の TOP500 では世界 4 位，2012 年 6 月発表の最新の TOP500 リストでも 14 位と世界屈指の性能を誇っている．

TSUBAME2.0 のノードの大半を占めるのは Thin ノードと呼ばれ，Hewlett Packard 社製の HP Proliant SL390s G7 server に Intel 社製の CPU Xeon 12 コア，NVIDIA 社製の GPU Tesla M2050 を 3 枚搭載したノードである．Proliant SL390s G7 server には管理システム HP iLO (Integrated Lights-Out) 3 が備わっており，各時点のノードの消費電力を計測することが可能である．分子動力学法は同じ計算を繰り返すシミュレーション技法であるため，各ステップにおける計算パターンはほぼ同一であり，消費電力も一定であると考えられる．従って我々はこのシステムから周期的に消費電力を求め，計算全体の平均消費電力を求めることとした．

4. 評価実験

4.1 計算機環境

本研究で用いた計算機環境は TSUBAME2.0 の Thin ノードである．各ノードは HP Proliant SL390s G7 server にヘキサコアの Intel Xeon X5670 2.93GHz が 2 ソケット，NVIDIA Tesla M2050 GPU が 3 枚搭載されている．メモリは 54GB，OS は Suse Linux Enterprise Server 11 SP1 である．

4.2 シミュレーションの設定

実験に用いたターゲットは NAMD のベンチマーク測定に用いられている ApoA1 タンパク質，脂質，水からな

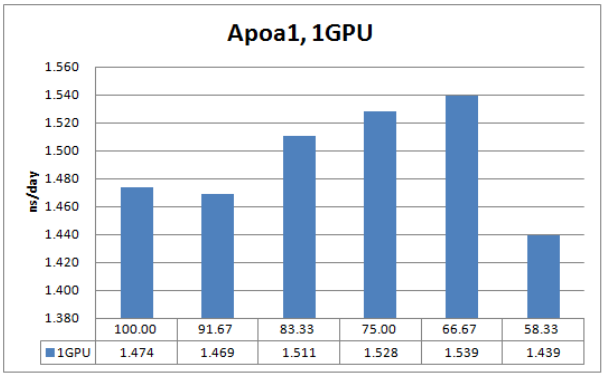


図 2 GPU1 枚利用時の計算速度の推移.

る原子数 92,224 の系である．シミュレーションの時間幅は 1femto 秒，カットオフ距離は 12Å とし，境界条件は周期境界とする．静電相互作用の長距離力の計算には PME (Particle Mesh Ewald) 法を用いる場合とカットオフ距離内の相互作用のみを考慮する場合の 2 つの場合について性能を測定した．

5. 実験結果

5.1 PME 使用時の結果

はじめに CPU12 コアと GPU1 枚を使い，PME を使用した場合の結果を示す．図 2 は計算速度の結果を示している．図の最も左のグラフが GPU に 100%タスクを割り振った場合の計算速度を表わし，右に寄ると，CPU に割り振るタスクの割合が増加する．縦軸は計算の速度を表わし，上に伸びるほど高速であることを意味する．CPU に仕事を割り振ると計算速度の向上を得ることができ，33%程度 CPU にタスクを割り振ったところで速度が最大となり，約 4.4%の速度の向上がみられた．

次にこの時の実行時のタイムラインを表示する．図 3 は Projections を用いてシミュレーションの最中に CPU が実行していた命令を時間軸に沿って可視化した図となっている．図の白い領域が CPU がアイドルになっている状態に対応し，青い領域では CPU が力の計算を行っている状態に対応する．図の上側が GPU に 100%仕事を割り振った場合の結果である．白い領域，すなわち CPU がアイドル状態になっている時間が非常に大きいことがみてとれる．これに対し，CPU 側に 3 割程度仕事を割り振った下の図では，白い領域が減り，青い領域が増えていることが見て取れる．従って，CPU のアイドル領域を利用して計算を行うことができている．数値で比較してみても，GPU が 100%の時はアイドル状態が全体の 74.8%を占めていたのに対し，CPU に 3 割割り振った場合には 45.2%にまでアイドル状態が減少していた．

消費電力のグラフをみると，100%GPU に割り振った場合と 3 割程度 CPU に割り振った場合とで，消費電力に大きな差がないことがみてとれる．平均の消費電力と電力効率をみてみると，GPU100%の方が 605.1W, 0.00243ns/W に対し，GPU67%の方が，599.5W，0.00257ns/W という結果になり，電力効率の面でも CPU に仕事を割り振った場合の方が有利である結果となった．

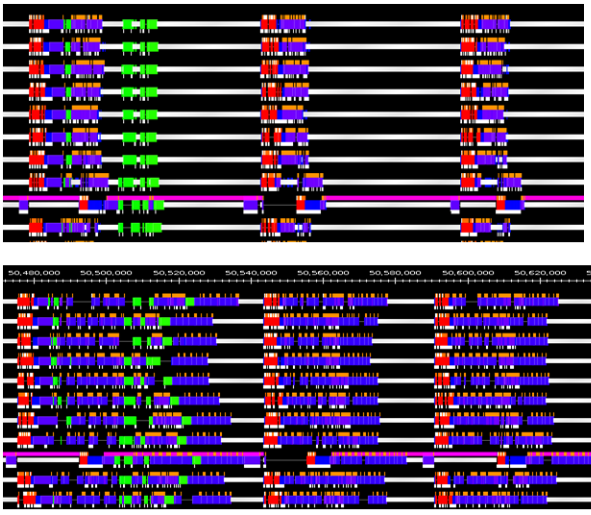


図 3 GPU 1 枚利用時のタイムライン

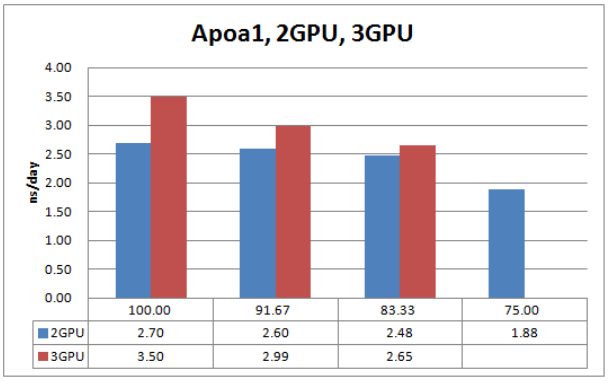
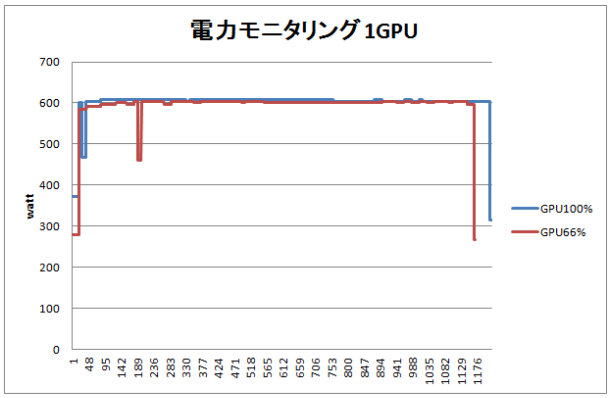


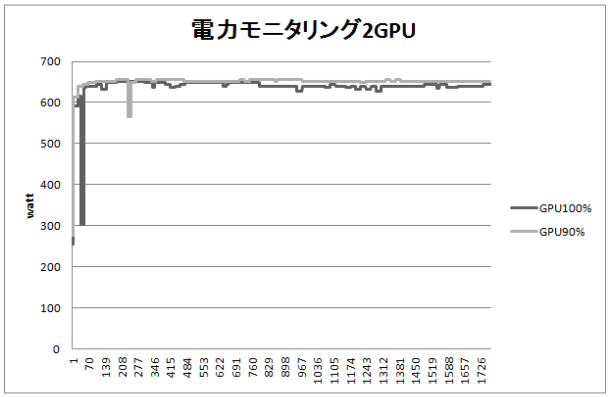
図 4 GPU2 枚，3 枚利用時の計算速度

次に，GPU の枚数を増やした際の結果を提示する．図 4 は GPU を 2 枚，3 枚利用した際の計算速度の推移のグラフである．青いグラフが GPU2 枚の結果に対応し，赤いグラフが 3 枚の場合に対応する．一番左側の GPU に 100%タスクを割り振った場合の結果が最も高速となり，CPU に割り振るタスクの量を増やすにつれ計算速度が低下する結果となった．CPU の利用率はアイドル状態の時間がいずれの場合も 8%程度減少し，改善していたが，GPU のタスクが先に終わり，CPU の計算の終了を待つ時間が生じてしまったため，計算速度の向上にはいたらなかったと考えられる．消費電力についても GPU1 枚の場合と同様，CPU にタスクを割り振っても目立った差異はみられなかった．

5.2 カットオフ内のみを考慮した結果

静電相互作用力の計算に PME を用いず，カットオフ内の計算のみを考慮した場合の結果について示す．GPU1 枚利用時から 3 枚利用時までの計算速度は CPU にタスクを





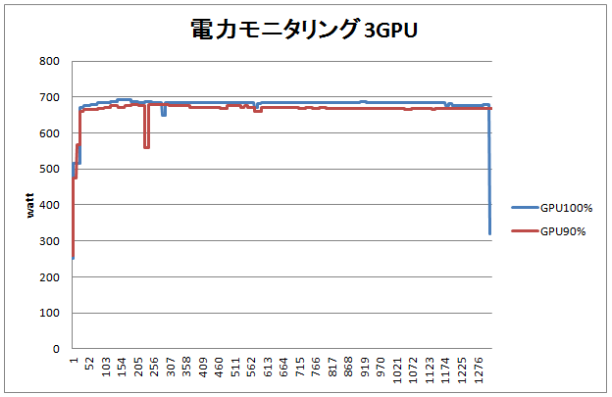


図 5 消費電力のモニタリング

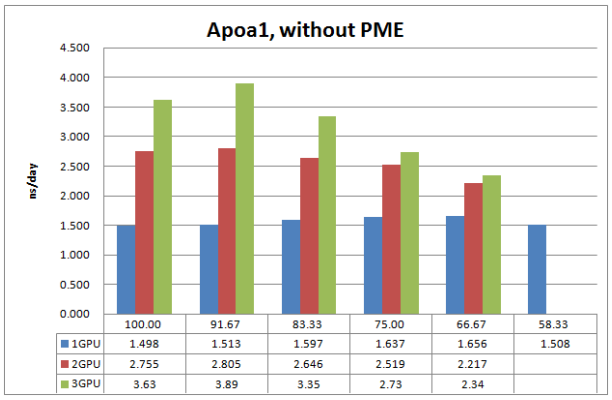
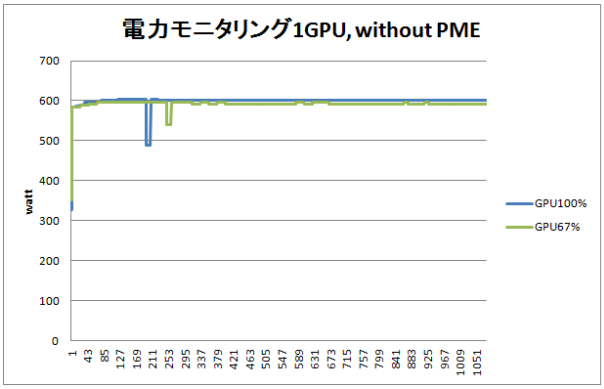
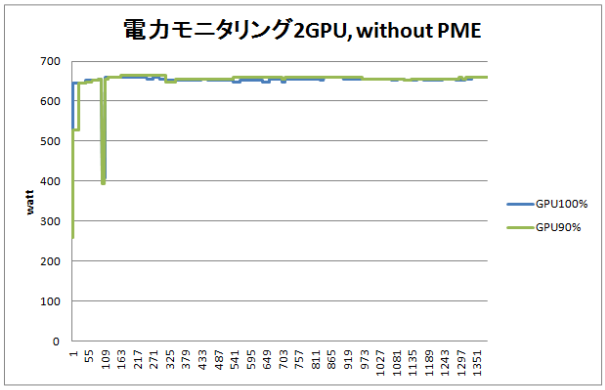


図 6 カットオフ内の計算のみを考慮した場合の計算速度





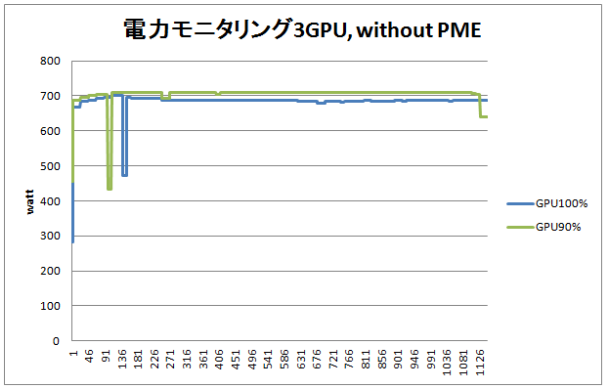


図 7 カットオフ内の計算のみを考慮した場合の電力消費

割り振ることでいずれも向上がみられた (図 6)．GPU1 枚利用時では GPU のタスクの割合が 67%程度のときに性能が最大となり，9%程度の速度の向上が得られた．GPU2 枚，3 枚利用時ではいずれも GPU のタスクの割合が 90%程度の時に速度の向上がみられ，それぞれ 2%，7%程度高速化した．

この時，シミュレーション実行時の消費電力をモニタリングした結果が図 7 である．GPU を 1 枚利用した際には計算実行時の平均消費電力が，GPU のタスクの割合が 100%の場合 598W，GPU のタスクの割合が 67%の場合 592W と，やや減少する結果が得られた．対して，GPU を 2 枚利用した際には GPU のタスクの割合が 100%の場合 652W，90%の場合 653W，GPU を 3 枚利用した際には GPU のタスクの割合が 100%の場合 684W，90%の場合 703W と増加する傾向がみられた．

6. 関連研究

CPU と GPU の計算負荷を考慮し，計算の高速化を図る研究は大島 [7] らの研究や小田嶋 [9] らの研究があげられる．大島らは CPU と GPU の演算能力を有効に活用する数値計算ライブラリを開発し，HPL ベンチマークにおいて計算速度の向上を達成した．小田嶋らの研究では並列言語 XcalableMP[8] の GPU 向けの拡張の開発を行い，その予備評価として N 体問題と行列積の計算を CPU/GPU の協調計算で行い，CPU のみ，GPU のみの場合に比べ高速化を達成した．本研究はライブラリやベンチマークのみではなく，既に広く活用されているアプリケーションソフトにおいても，CPU と GPU による負荷分散による計算が，計算速度，計算効率の向上を実現できることを示した．

7. まとめ

本研究では分子動力学法のアプリケーションプログラム NAMD を例に，CPU とアクセラレータの負荷に偏りが存在するとリソースが十分有効に活用されない事例があることを示した．その解決法として我々は従来 CPU 若しくはアクセラレータのみで行われる計算の一部を負荷分散する手法を提案し，実際に NAMD の計算速度を向上させることに成功した．現段階では CPU とアクセラレータの仕事量の比率は恣意的かつ静的に定めているが，今後は問題サイズやノードの構成に応じ，自動的に割合を決定する機構を開発する予定である．

参考文献

[1] http://www.top500.org/
[2] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kale, and Klaus Schulten. “Scalable molecular dynamics with NAMD”. Journal of Computational Chemistry, 26:1781-1802, 2005.
[3] Wang, J., Wolf, R. M.; Caldwell, J. W.;Kollman, P. A.; Case, D. A. “Development and testing of a general AMBER force field”. Journal of Computational Chemistry, 25, 2004, 1157-1174.
[4] L. V. Kale and Sanjeev Krishnan. “Charm++: Parallel Programming with Message-Driven Objects”. In Gregory V. Wilson and Paul Lu, editors, Parallel Programming using C++, pages 175-213. MIT Press, 1996.
[5] Darden T, York D, Pedersen L “Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems”, Journal of Chemical Physics, 98, 10089 (1993).
[6] James C. Phillips, John E. Stone, and Klaus Schulten. “Adapting a message-driven parallel application to GPU-accelerated clusters”. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC08), 2008.
[7] 大島聡史，吉瀬謙二，片桐孝洋，弓場敏嗣. “CPU と GPU を用いた並列 GEMM 塩山の提案と実装”. 情報処理学会論文誌，コンピューティングシステム，Vol.47， No.12，pp.317-328，2006-09-15.
[8] 李珍泌, 朴泰祐, 佐藤三久. “分散メモリ向け並列言語 XcalableMP コンパイラ の実装と性能評価”，先進的計算基盤システムシンポジウム SACSIS2010 論文集，pp.175-182，2010.
[9] 小田嶋哲哉，チャントゥァンミン，李珍泌，朴泰祐，佐藤三久.“並列言語 XcalableMP の GPU 向け拡張”， 2011-HPC-129, pp.1-7, 2011.