CPU/GPU ヘテロジニアス環境における FMM の最適化

福田圭祐[†] 丸山直也[†] 松岡聡[†,††]

1. はじめに

高い性能を要求する HPC 分野において，CPU/GPU ヘテロジニアス環境の重要性が高まっている．東京工業大学学術国際情報センターに設置された TSUBAME2.0 はその身近な例であり，Linpack ベンチマークによる世界ランキング Top500[6] では，上位 5 サイトのうち TSUBAME2.0 を含めて 3 サイトが CPU と GPU を併用したものであった．このような環境では，異種プロセッサの混在を前提とした最適化が必要となる．しかし，GPU 等の特定のプロセッサ環境向けの最適化は移植性に欠け，かつ製品の進歩によってすぐに陳腐化してしまう可能性が高い．

我々は，N 体問題のスケーラブルなアルゴリズムとして期待されている Fast Multipole Method(FMM) に着目し，ヘテロジニアス環境における最適化手法を研究している．FMM は適度に複雑で様々な特性を合わせ持ち，現実の計算でも応用が期待されるアルゴリズムである．このようなアルゴリズムのヘテロジニアス環境上での最適化は今後重要性が高まっていくと考えられるが，現在のところ研究が進められていない．

本稿では最適化手法のための予備評価を行った．FMM の派生アルゴリズムである KIFMM[7] の既存実装を，OpenMP と CUDA を用いて CPU/GPU それぞれについて並列化と最適化を施した．実装したソフトウェアを，TSUBAME2.0 の単一ノードを用いて評価し，大幅な高速化が達成されたことを示す．

2. 関連研究

Lashuk らは KIFMM の GPU 実装に取り組み，大幅な高速化を得ている[4]．本稿での予備評価 GPU 実装については，Lashuk らの成果を参考にした部分が多い．Chandramowlishwaran らは，マルチコア CPU 上での KIFMM を高度に最適化した[2]．

† 東京工業大学
†† 国立情報学研究所

表 1 FMM を構成する計算フェーズ

木構築	空間の再帰的分割と木構造の構築
U-list	部分領域の粒子同士の直接計算
Upward	木構造の上方向への探索
V-list	節同士の近似計算 (多量の小規模 FFT)
X,W-list	木構造の不均衡発生時の処理
Downward	木構造の下方向への探索

3. FMM の概要

N 体問題は直接計算を行うと $O(N^2)$ の計算量を必要とするが，FMM はそれを $O(N)$ 計算量で近似計算するためのアルゴリズムである．空間を再帰的に分割して木構造を作り，それに沿って粒子を「まとめる」操作を行い，遠方の粒子からの影響を近似計算する．

計算フェーズは，空間分割・木構築，U-list フェーズ，Upward フェーズ，V-list フェーズ，X,W-list フェーズ，Downward フェーズの 6 フェーズからなる．それぞれの計算内容は**表 1** の通りである．

4. 実装と評価

本稿では，Ying ら[7] による CPU 上での逐次実装に変更を加え，OpenMP 及び GPU 上での並列化により大幅な高速化を達成できたことを示す．

まず，CUDA による並列化について述べる．本稿では，実行時間の 60%以上を占めている Upward フェーズの一部および U-list フェーズを対象とした．U-list フェーズはまず木構造の葉にあたる部分空間 (box) に属する粒子同士の直接計算を行う．これは，N 体問題の直接計算と同じパターンの計算である．次に，その box に隣接する周囲の box との間で同様の計算を繰返し行い，すべての結果を合計する．**図 1** は，粒子同士の計算をスレッドとスレッドブロックが担当する様子を示している．1 つの box に属する粒子は，パラメータにも依存するが数百〜2000 程度であり，基本的には 1 スレッドが 1 粒子を計算し，粒子数がスレッドブロックの制限を超える場合は複数粒子を担当するようにした．また，Upward フェーズは，U-list の場合とほぼ同等の計算を 1 つの box 内部のみで行う．本稿

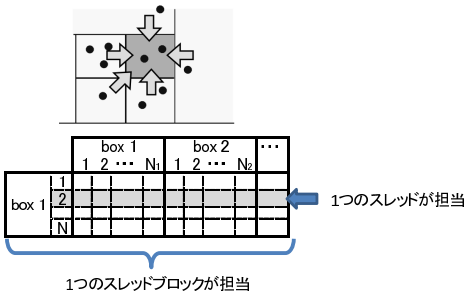

図 1　U-list フェーズの CUDA スレッド計算分担

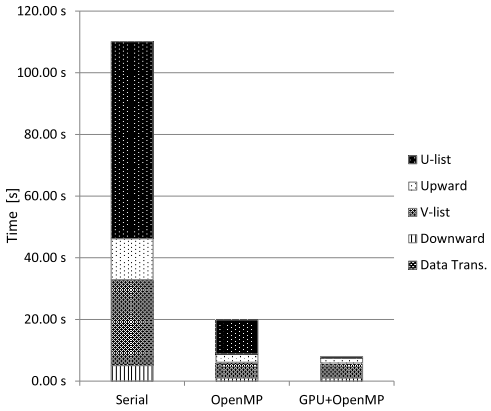

図 2　OpenMP および GPU+OpenMP による高速化

の実装の範囲内では，データ変換・転送のコストは非常に小さく問題にならなかった．

次に，OpenMP を用いた並列化は，計算フェーズ内の依存性に留意しつつプログラムに存在する for ループを並列化した．実験環境上では 6 つの物理コアに 1 つずつのスレッドを割り当てた．

測定環境は**表 2** の通りである．粒子の初期配置は，100 万粒子の均一分散を用いた．我々の OpenMP 実装を 1 スレッドで実行したものを逐次実装とし，OpenMP 実装に CPU1 ソケット，OpenMP+CUDA 実装に CPU と GPU それぞれ 1 ソケットを用いた評価結果を**図 2** に示す．逐次実装に対して，それぞれ 5.55 倍，13.8 倍の高速化となり，特に並列度の高い U-list フェーズはそれぞれ 5.9 倍，41.1 倍となった．

今回並列化を行わなかったフェーズは逐次実装の実行時間に占める割合が低かったが，さらなる高速化のためにはこれらも並列化していく必要がある．特に V-list は多数の小規模 FFT を含み，GPU での効率的な実装方針は明らかではなく，今後の課題である．

表 2　評価環境

	CPU	GPU
Type	6 core Xeon X5670 × 2 (評価では 1 ソケット使用)	Tesla M2050 × 3 （評価では 1 ソケット使用）
Freq	2.93GHz	1.15GHz
Cores	6 x 2 (with HT)	14SM/448 cores
Memory	54GB	3GB

5. まとめと今後の課題

本稿では，CPU/GPU ヘテロジニアス環境上での FMM の最適化を目標とし，その予備評価として既存の逐次 FMM 実装に対して CUDA および OpenMP を用いた並列化を行い，逐次実装に比べ OpenMP 版が 5.55 倍，GPU+OpenMP 版が 13.8 倍の大幅な性能向上が得られることを確認した．

今後は，CPU･GPU 上での高性能な並列化実装を実装し，それを利用して CPU/GPU ヘテロジニアス環境上での最適なタスク割り当て戦略の研究を進めていく．ヘテロジニアス環境上での計算の最適化としては，遠藤らの Heterogeneous LINPACK[3] に代表される手動によるものや，テネシー大学による MAGMA[5] や INRIA による StarPU[1] などの自動化に関する研究がある．現時点では，これらは線形代数などの数値計算を主なターゲットとしており，FMM に直接に適用可能であるかどうかは今後の研究課題である．

参　考　文　献

1) Cédric Augonnet, et al. Euro-par 2008 workshops - parallel processing. chapter A Unified Runtime System for Heterogeneous Multi-core Architectures, pp. 174–183. Springer-Verlag, Berlin, Heidelberg, 2009.
2) A. Chandramowlishwaran, et al. Optimizing and tuning the fast multipole method for state-of-the-art multicore architectures. In *IPDPS 2010*, pp. 1 –12, April 2010.
3) T. Endo, et al. Linpack evaluation on a supercomputer with heterogeneous accelerators. In *IPDPS '10*, pp. 1 –8, April 2010.
4) I. Lashuk, et al. A massively parallel adaptive fast-multipole method on heterogeneous architectures. SC '09, pp. 58:1–58:12, New York, NY, USA, 2009. ACM.
5) Rajib Nath, et al. Accelerating gpu kernels for dense linear algebra. VECPAR'10, pp. 83–92, Berlin, Heidelberg, 2011. Springer-Verlag.
6) Top500. Top500 supercomputing sites, September 2010. `http://www.top500.org/`.
7) L. Ying, et al. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *Journal of Computational Physics*, Vol. 196, No. 2, pp. 591 – 626, 2004.