Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

FMMを用いたペタスケール乱流解析

横田理央[†1]　成見哲[†2]
L.A. Barba[†3]　泰岡顕治[†4]

Fast Multipole Method (FMM) は従来粒子の N 体問題の高速化手法として発展してきたが，近年その応用の幅を広げる研究が多くなされている．本研究では，大規模 GPU システム向けに開発された FMM を用いて 2048^3 規模の乱流解析を行い，同様の計算条件のもとでスペクトル法との比較を行った．ただし，今回の解析に用いた手法は Treecode と FMM の長所を組み合わせたハイブリッド型になっており，GPU 上で高い Flops が出る treecode の特長をさらに高速なアルゴリズムである FMM で実現している．TSUBAME2.0 上で 4096 GPU を用いた計算において 74%の並列化効率を得た．また，このときの演算性能は 1.01PFlops であった．

Petascale Turbulence Simulation Using FMM

Rio Yokota,[†1] Tetsu Narumi,[†2] Lorena Barba[†3]
and Kenji Yasuoka[†4]

Fast multipole methods (FMM) were originally developed for accelerating N-body problems in astrophysics and other particle based methods. A recent trend in HPC has been to use FMMs in unconventional application areas. We have performed a 2048^3 turbulence calculation using an FMM designed for large scale GPU systems. The proposed method uses a hybridization of the treecode and FMM, and combines the data-parallel treecode with the O(N) FMM. The run on TSUBAME 2.0 using 4096 GPUs achieved 74 % parallel efficiency, and the sustained performance reached 1.01 PFlops.

1. はじめに

重力多体問題や分子動力学にみられるような N 体問題の直接計算は N 個の粒子に対して $\mathcal{O}(N^2)$ の計算時間を要するため大規模な計算は困難である．このため，遠方の粒子をまとめて近似的に計算する階層型の高速化手法の研究が 80 年代後半から盛んに行われてきた．近年，このような階層型の N 体ソルバが大規模 GPU システムで高い性能を実現していることから，[1),2)] ポストペタスケールにおける重要なアルゴリズムとして注目されている．特に分子動力学や流体力学の分野では，従来 FFT によって高速化されてきた particle mesh Ewald 法[3)] やスペクトル法[4)] と同等のアプリケーションが N 体ソルバでも解析可能であるため，今後アーキテクチャの変遷に伴い最適なアルゴリズムが変わる可能性がある．

階層型の N 体問題の高速化手法には大きく分けて 2 種類ある．一つは treecode[5)] で，遠方のソース粒子をまとめて計算することで計算時間を $\mathcal{O}(NlogN)$ に低減する手法である．もう一つは fast multipole method(FMM)[6)] で，近傍のターゲット粒子と遠方のソース粒子の両方をまとめて計算することで計算時間をさらに $\mathcal{O}(N)$ まで低減する手法である．これらの二つの手法は tree 構造を用いるという点は共通しているが，独立して発展してきた経緯があり，それぞれに長所と短所がある．Treecode は主に宇宙物理学の分野で盛んに研究されてきたため，天体の非一様な分布に対応できるようアダプティビティに重点をおいた手法となっている．[7)] また，重力多体問題で必要な計算精度がさほど高くないことから，多重極展開の近似の次数を固定した実装が多い．一方，FMM は分子動力学や流体解析，音響解析などのように，領域や表面に一様に分布した計算点の解析に用いられることが多く，一般的な実装のアダプティビティは treecode に比べて低い．逆に，FMM の多くのアプリケーションでは高い計算精度が求められるため，多重極展開や局所展開の次数が大きい場合を想定した高度な実装が多くなされている．[8)]FMM と treecode の手法間の差異は，GPU への実装効率を左右する要因ともなる．例えば，treecode の多重極展開から粒子への変換

†1 KAUST
King Abdullah University of Science and Technology
†2 電気通信大学
University of Electro-Communications
†3 ボストン大学
Boston University
†4 慶應義塾大学
Keio University

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

は FMM の多重極展開から局所展開への変換に比べてテータあたりの演算量が多く，データの並列性も高いため，GPU 上で高い演算性能を得ることができる.[9] つまり，ヘテロジニアスなアーキテクチャでは treecode と FMM のそれぞれの長所を合わせたハイブリッド型のアルゴリズムが必要となる．

FMM と treecode のハイブリッド化に関する研究も僅かながら行われている．FMM と treecode の最も大きな違いは，FMM が多重極展開から局所展開へのセル同士の計算を行うのに対して treecode は多重極展開から粒子へのセル対粒子の計算を行うことである．Warren&Salmon[10] は従来の treecode 枠組みの中でセル同士の計算を行う手法を提案した．Dehnen[11] はこのセル同士の相互作用リストの構築を木構造の双対走査 (dual tree traversal) によって実現し，$\mathcal{O}(N)$ かつアダプティブな木構造の走査法を考案した．さらに，Wang[12] は実球面調和関数を用いて treecode に高次の多重極展開を実装し，従来の treecode の弱点であった高精度の計算を可能にした．上記のような treecode に FMM の長所を導入する研究に加え，FMM に treecode の長所を導入する研究も行われている．Cheng[8] らは FMM のセル同士の作用のリストをよりアダプティブなものに拡張し，さらに FMM の枠組みの中でセル対粒子の計算も選択できる仕組みを考案した．ただし，従来の FMM の固定された相互作用リストを無理矢理アダプティブなものに拡張したこの手法は，Dehnen の双対走査法に比べて複雑であるという欠点がある．また，上述の全てのハイブリッド化に関する工夫を組み合わせることは可能であるが，そのような研究はまだ例がない．

本研究では超大規模でヘテロジニアスな計算機を想定した FMM と treecode のハイブリッド化手法を提案する．具体的には，FMM と treecode に関する様々な工夫を並列性，Flop/Byte，メモリ消費量などの観点から分析し，最適な組み合わせを自動的に実現する仕組みを考案した．また，本手法を用いて FMM としては最大規模となる 640 億粒子を用いた計算を行い，TSUBAME 2.0 で 4096 台の GPU を用いることで 1.01PFlops を記録した．さらに，FMM と FFT の実アプリケーションコードでの性能比較として一様等方性乱流の解析を行った．4096 プロセスを用いたときの並列化効率は FMM が 74%であったのに対し FFT は 14%であった．

2. FMM と treecode のハイブリッド化

2.1 木構造の双対走査

本研究で用いる木構造の走査方法は Dehnen[11] の提唱したターゲットの木構造とソースの木構造の同時走査（シリアル CPU 用）を MPI+CUDA 用に拡張したものである．図 1 に木構造の双対走査の手順を示す．以下の説明では木構造の節点をデータ構造上は「ノード」，物理空間においてそれに対応するものを「セル」と呼び分けるが，これらは基本的には同じものを指すと考えていただきたい．まず，図の左側に示す初期操作ではターゲット粒子とソース粒子に対してそれぞれ木構造を構築する．もとの Dehnen の手法ではターゲットとソースが同じであることを前提としているが，本手法では分散メモリ並列化を前提としているためこれらの二つの木構造を別のものとみなす．次に，これら二つの木構造の根ノードのペアをスタックに Push する．続いて，図の右側に示す逐次操作を行う．ただし，ここには逐次操作が数回行われた後の状態が図示してあり，初期操作の直後にはスタックに根ノードのペアのみが存在している状態から始まる．逐次操作ではまず，スタックの最上位にあるペアを Pop し，ターゲットかソースかどちらか一方のセルを分割する．次に，二つのセルの大きさの和 $r_t + r_s$ とセル同士の中心からの距離 R の比が多重極展開が収束するのに十分小さいかどうかを判別する．セル同士の距離が十分遠い場合はその場で相互作用が計算され，そのペアに対する操作はそこで終了する．セル同士の距離が近すぎる場合はそのペ

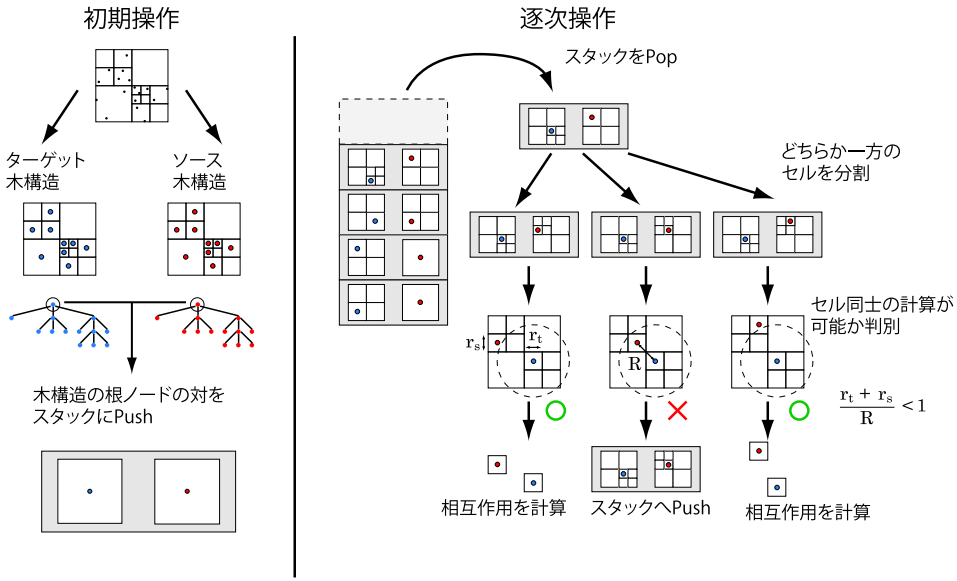

図 1 木構造の双対走査の模式図
Fig. 1 Schematic of the dual tree traversal.

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

アを再びスタックへ Push する．ただし，ペアの両方が葉ノードとなった場合はセル内の粒子の直接相互作用を計算する．その後再び逐次操作の最初に戻り，スタックが空になるまで Pop，分割，判別，Push を繰り返す．このとき，必ず大きい方のセルが分割されるものとし，大きさが同じ場合はターゲットのセルを分割する．大きい方のセルを分割することで，セルのペアをほぼ同じ大きさのセル同士に制限することができる．これにより，任意のターゲットセルと相互作用するソースセルの個数を $\mathcal{O}(1)$ に制限できるため全体として $\mathcal{O}(N)$ で木構造の双対走査を実現できる．

木構造の双対走査が従来の FMM の走査とどのような相違があるかを明確にするために，ここでは一般的な FMM で用いられる走査方法と照らし合わせて解説する．主要な FMM のコード[2),13)–15)] では木構造の走査は行わず，各ターゲットセルに対して「親ノード」の「隣接セル」の「子ノード」という関係を用いて相互作用リストを作成する．ただし，親ノード，子ノードは木構造の上下に隣接するノードを示し，隣接セルは物理空間で隣接する 26 個のセルを指す．したがって，隣接セルは木構造の情報から直接求めることはできない関係にある．このため，FMM のセルには空間充填曲線に沿って番号をふっておき，隣接セルを参照する際にはセル番号から空間的位置関係を毎回計算する必要がある．このような方法では，空間的に隣接する 26 個のセルが多重極展開が収束するための距離を確保するのに用いられているため，セルは常に等方的でなければならず treecode のように直方体のセルを用いることはできない．これは treecode が FMM よりもアダプティブである要因の一つである．また，treecode のように調節可能な近傍領域を定義するためには隣接セルの候補を効率的に探す方法が必要となる．隣接セルの候補を木構造の走査によって求めた場合 $\mathcal{O}(NlogN)$ の手法になるばかりでなく，従来の FMM よりも遥かに非効率な方法になってしまう．本研究で用いる木構造の双対走査は，まさにこの隣接セルの候補を効率的に探す方法に他ならない．つまり，図 1 のスタックに積まれるセルのペアは，ターゲットの親から子へと継承される，途中まで走査済みのソースの木構造（隣接セルの候補）を表していると考えることができる．このように，隣接セルの候補を親から子へと継承することで，毎回根ノードから木構造を走査する必要を解消し，treecode と同様のアダプティビティを有しながらも $\mathcal{O}(N)$ で走査を行う手法が実現できる．

2.2 FMM の自動最適化

FMM，treecode ともに近傍場では粒子対粒子の直接計算を行う．遠方場は treecode ではセル対粒子，FMM ではセル対セルの計算を行う．このとき，粒子対粒子の計算もセル対粒子の計算も全てセル対セル単位で行われるため，FMM の最適化は上記の双対走査で求めたセルのペアに対してどの演算を行うのが最適かという問題に帰着する．多重極展開，局所展開の次数とカーネルの種類が決まれば，粒子対粒子，セル対粒子，セル対セルの計算量は一意に決まるので，これらの計算時間をあらかじめ測定することで最適化の指標として用いることができる．特にヘテロジニアスなシステムにおいては，粒子対粒子やセル対セルのルーチンの相対的な演算性能が変化したとしても，それが最初に測定した計算時間に反映されるため，どのアーキテクチャにおいても常に最適な組み合わせが選択されることになる．

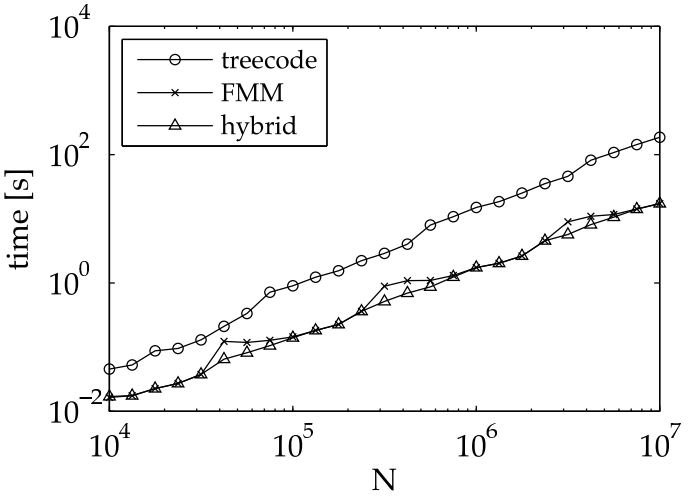

図 2 CPU 上での treecode と FMM とハイブリッドの比較
Fig. 2 Comparison of treecode, FMM, and hybrid on a single CPU.

図 2 に単一 CPU 上での treecode と FMM とハイブリッド法の粒子数 N に対する計算時間を示す．粒子は $[0,1]^3$ の立方体にランダムに分布させ，N は 10^4 から 10^7 まで変化させた．また，多重極展開，局所展開の次数は $p=5$ であった．ただし，このときの treecode は FMM と同様の球面調和関数を用いた多重極展開を用いており，デカルト座標系による多重極展開の実装と比べて性能は低いことに注意されたい．このため，図 2 では FMM に比べて treecode の方が大分遅いように見える．興味深いのは，ハイブリッド化を行うことで FMM の N の増加にともなう波打つような挙動が見られなくなることである．FMM の計算時間が波打つのは，粒子対粒子の計算とセル対セルの計算のバランスが突然変化するためである．ハイブリッド法では粒子対粒子，セル対粒子，セル対セルの間で最適なバランス

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

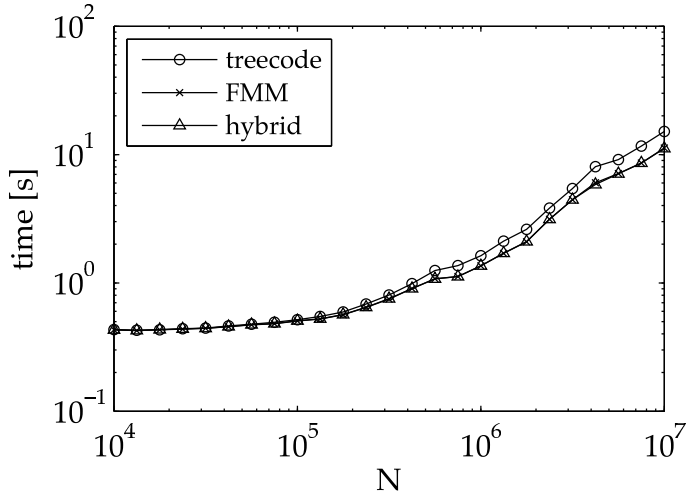

図 3 GPU 上での treecode と FMM とハイブリッドの比較
Fig. 3 Comparison of treecode, FMM, and hybrid on a single GPU.

が保たれるため，N の増加に伴う急激な変化は起きないものと思われる．

図 3 に単一 GPU 上での treecode と FMM とハイブリッド法の計算時間を示す．計算条件は図 2 と全く同様であった．$N < 10^5$ では演算量が少なすぎるため GPU の演算性能を引き出すことができず，N によらないオーバーヘッドに計算時間が支配されている．一方，treecode の計算時間を図 2 の CPU の場合と比較すると，FMM やハイブリッドに比べて大幅に加速していることが分かる．これは treecode のセル対粒子の計算が FMM のセル対セルの計算に比べて GPU 上で高い演算性能が得られるためである．また，ハイブリッド法はセル対粒子の計算を選択できるにも関わらず FMM と同様の加速率にとどまっている．これは，GPU 上で各ルーチンの計算時間を測定して自動最適化に用いる方法に問題があると考えられる．本手法では，各ルーチンの計算時間を測定するのに単一のセル同士の計算時間から実際の計算時間を外挿しており，CPU 上ではこれがある程度正確な予想として機能しているものの，GPU 上では問題の規模によって演算性能が著しく変化するため予想が大きく外れている可能性が高い．今後の課題として，GPU 上での自動最適化の精度を高めるために，計算量と計算時間の関係を現在の線形近似から多項式近似へと変えることが考えられる．

2.3 スペクトル法

スペクトル法の計算には FFTW を用いたオープンソースコード hit3d[⋆1] を用いた．ただし，hit3d は 2/3 法によるエイリアシング除去を行っており，時間積分には 2 次の Adams-Bashforth 法を用いている．また，hit3d には forcing の機能は備わっているが，今回は渦法計算と条件を一致させるためあえて用いなかった．

図 4 に 3 次元 FFT の計算手順を示す．まず，x 方向に領域分割されたデータ上で y,z 方向の 2 次元 FFT を行い，Alltoall 通信によって z 方向の領域分割へとデータを転置し，x 方向の 1 次元 FFT を y 方向の点の数だけ行う．このとき領域分割の転置の際の通信は，格子点数を N^3，プロセス数を P とすると，プロセスあたりの送信量は $N^3/P \times 4\text{Byte}$ になる．つまり，後に示すウィーク・スケーリングのように $N = P = 4096$ のとき約 67MB の送信量となる．これに対し，$N = P = 4096$ の時の FMM はプロセスあたりセルが 11144 個，粒子が 512×6536 個送信され，セルには 105 個の多重極展開の係数が含まれ，粒子には 6 個の変数が含まれる．よって，プロセスあたりの送信量は $(11144 \times 105 + 512 \times 6536 \times 6) \times 4\text{Byte}$(約 85MB) となり，一見 FFT とほとんど同じ程度の通信料に見える．しかし，スペクトル法は 1 タイムステップあたり FFT と逆 FFT を速度 3 成分に対して行うため，結果的に FMM の 5 倍のデータ量が送信されることになる．また，上記のような合計の送信量からは見えないが，FFT は全てのプロセスのペアについて同じ量のデータを送受信するのに対して，FMM は遠方場に割り当てられたプロセスほど通信料が少なくなるため，フルバイセクションのネットワークでなくとも性能がでる仕組みになっているといえる．

3. FMM による乱流解析

3.1 渦　　法

渦法は速度と渦度による定式化用いた粒子ベースの非圧縮流体解析法である．[16] 渦法では，Poisson 方程式の代わりに N 体問題を解くことで質量保存則を満たす．また，渦法は粒子ベースの流体解析法の中でも質量を離散化する SPH[17] などとは異なり，渦構造の相互作用を渦粒子を用いて直接解いているため，乱流解析に向いている手法であるといえる．速度と圧力による定式化を行う通常の非圧縮流体解析法とは異なり，渦法は速度と渦度による定式化を用いるため，速度の Poisson 方程式と渦度方程式を交互に解くことで質量保存と運動量保存を満たす．速度の Poisson 方程式は積分型の定式化を用いて以下のような N 体問

⋆1 http://code.google.com/p/hit3d/

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

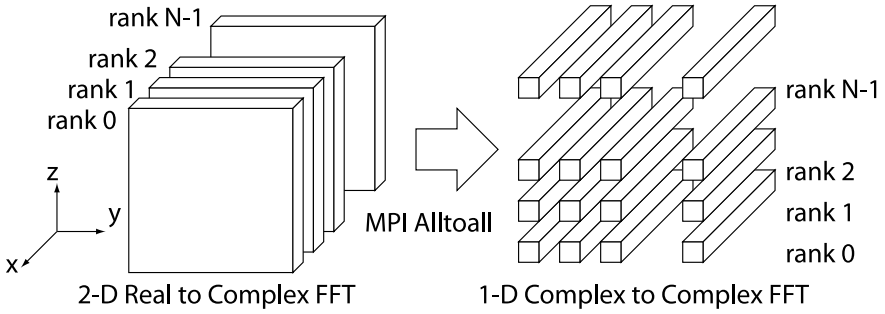

図 4 3次元 FFT の計算手順
Fig. 4 Flow of 3-D FFT calculation.

題として解くことができる．

$$\mathbf{u}_i = \sum_{j=1}^{N} \boldsymbol{\alpha}_j g_\sigma \times \nabla G \tag{1}$$

ただし，α は渦要素の持つ渦強度，g_σ はカットオフ関数，G は Laplace 方程式の Green 関数である．渦度方程式の伸張項は

$$\frac{D\alpha_i}{Dt} = \sum_{j=1}^{N} \boldsymbol{\alpha}_j \nabla(g_\sigma \times \nabla G) \cdot \boldsymbol{\alpha}_i \tag{2}$$

によって計算される．また，渦度方程式の拡散項は reinitialized core spreading 法により計算される．[18] 渦法では式 (1) と (2) が計算時間の大部分を占め，これらは FMM を用いて計算される．

図 5 に渦法計算のフローチャートを示す．MPI コード上では各プロセスが異なる粒子を保有しているが，計算負荷と通信負荷をバランスするためにまず最初に八分木構造に沿った直交再帰二分法を用いて領域分割を行う．粒子が 1 ステップの間に移動する距離は微小であるため，最初のタイムステップを除けば領域分割に伴う通信はごく僅かである．続いて，グローバルな木構造の中から各プロセスに必要な部分 (local essential tree) のみの送受信を行う．つまり，遠方の領域を保有するプロセスからは木構造の上層部のみが送信される．送受信されるべきノードの効率的な探査方法に関しては Dubinski[7] を参照されたい．また，本手法では local essential tree の通信とローカルな木構造の計算をオーバーラップさせるため，local tree と local essential tree の残りの部分を分けて計算する．これにより，2.1 節で述べた木構造の双対走査を行う際に，local tree の場合はターゲットとソースが同じなるため作用反作用を同時に計算し，local essential tree の残りの部分に関してはターゲットとソースが異なるため作用のみを計算できる．FMM の計算が終わると，渦要素の座標 $\mathbf{x}$，渦強度 $\boldsymbol{\alpha}$，渦核半径 σ をローカルな情報のみを用いて更新する．また，数ステップに一回，渦要素の再初期化を行う必要がある．これには渦核半径が大きくなりすぎるのを防ぎ，渦要素の分布を一様に保つ効果がある．今回の計算では，5 ステップに一回再初期化を行い，この際に用いる反復解法[19] の収束条件は相対残差 1e-5 とした．

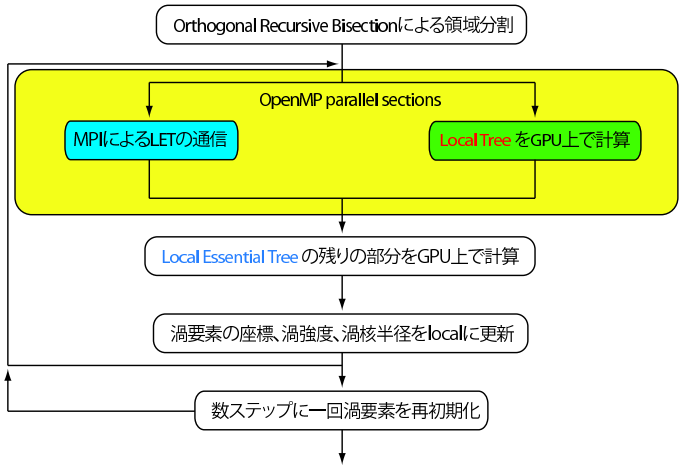

図 5 渦法計算のフローチャート
Fig. 5 Flow chart of vortex method calculation.

3.2 スケーラビリティ

渦法計算における FMM の並列性の高さを示すために，TSUBAME 2.0 においてプロセスあたり $N = 4096^2$ の粒子を用いた場合の 4096 プロセスまでのウィーク・スケーリングとその内訳を図 6 に示す．ただし，凡例の “Local evaluation” は粒子対粒子の直接計算，“FMM evaluation” はセル対セルもしくはセル対粒子の計算，“MPI communication” は領域分割と local essential tree の通信，“GPU communication” はホストから GPU への通信，“Tree construction” は木構造の生成にかかった時間を表す．ただし，“MPI communication” は “Local evaluation” や “FMM evaluation” とオーバーラップしているため，ここでは “Local evaluation” の時間から引くことで棒グラフ全体の高さが実際の wall clock time と一致するようにした．512GPU までは通信時間の割合はごく僅かであるが，4096GPU では全体の

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

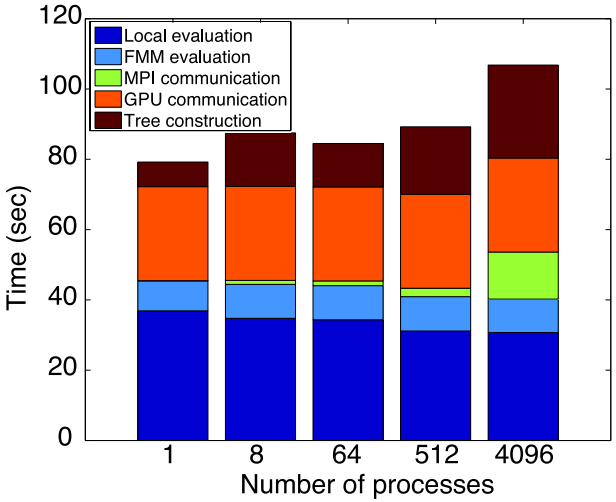

図 6 4096GPU までのウィーク・スケーリングとその内訳
Fig. 6 Breakdown of weak scaling up to 4096 GPUs.

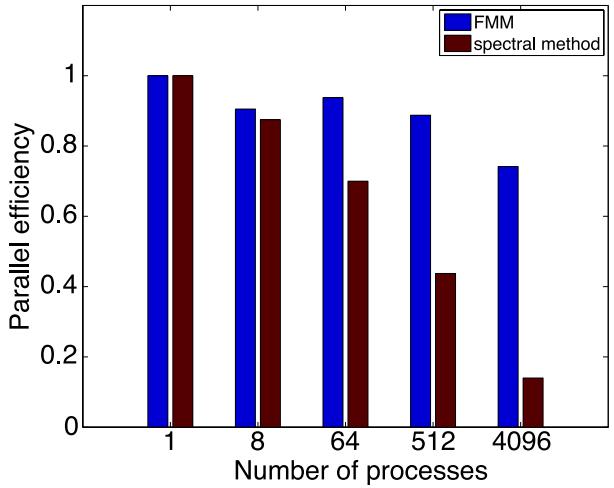

図 7 FMM とスペクトル法の並列化効率
Fig. 7 Parallel efficiency of FMM and spectral method

10%以上を占めている．また，木構造の生成にかかる時間もプロセス数とともに増加しており，4096 では全体の 20%以上を占めている．最大規模の計算は $N = 4096^3$ であり，このときの演算性能は $1.01PFlops$ であった．今後の課題としては，通信のさらなる効率化と，木構造生成の並列化効率の向上が挙げられる．

図 7 に FMM とスペクトル法のウィーク・スケーリングにおける並列化効率を比較したものを示す．FMM，スペクトル法ともに 1 プロセスを基準とし，4096 プロセスまでの並列化効率を示す．4096GPU を用いた場合 FMM は 74%の並列化効率が得られているのに対してスペクトル法では 4096CPU コアを用いた場合 14%であった．このときの計算時間は FMM，スペクトル法ともに 100 秒程度であった．ただし，FMM の計算はノードあたり 3GPU，スペクトル法の計算はノードあたり 3CPU コアを用いた．TSUBAME2.0 のようなフルバイセクション，ノンブロッキングの fat tree ネットワークにおいて，このような大きな違いが得られたことは意義深い．今後，ポストペタスケールのアーキテクチャを考える上で FMM と FFT のような代表的なアルゴリズムの性能を実アプリケーションを通して評価することがますます重要になると考えられる．

3.3 一様等方性乱流の解析結果

ここでは，FMM をベースとする渦法と FFT をベースとするスペクトル法を用いて同じ $Re_\lambda \approx 500$ の一様等方性乱流の解析を行う．計算領域は $[-\pi, \pi]$ までとし，2048^3 の格子（粒子）を用いて計算を行った．FMM の周期境界条件には周期的な鏡像を 27^3 個配置し，多重極展開の次数は $p = 14$ とした．初期条件は並列のスペクトル法のコード上で，指定されたエネルギースペクトルにランダムな位相を持たせた速度場を波数空間で生成した．スペクトル法ではこの初期条件をそのまま用いるが，渦法ではこれを物理空間へ変換し格子状に配置された粒子の渦強度を計算する必要がある．ただし，スペクトル法と渦法の計算点はスタガート状にずれた配置になっている．つまり，一般的な有限差分法におけるスタガート格子の圧力の位置に渦度を配置していることになる．これらの点での渦度が求まれば，この渦度から各粒子の渦強度を求めることができる．初期の渦核半径は粒子の間隔と等しくなるよう $\sigma = \Delta x$ とした．

図 8 に t/T=2 における速度勾配テンソルの第二不変量の等値面を示す．ただし，T は大規模渦の回転の時間スケールを表す．微細な構造は多数見られるものの，計算時間が短いせ

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

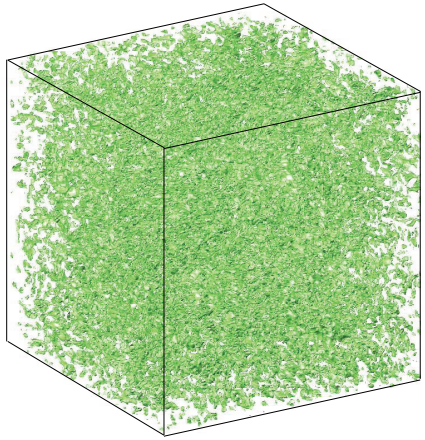

図 8 速度勾配テンソルの第二不変量の等値面
Fig. 8 Isosurface of the second invariant of the velocity gradient tensor

いか大規模な渦構造を明確に確認することはできない．今回の計算ではマシンの占有時間の制約から十分な時間計算を行うことができなかったが，計算自体の健全性を確認するには十分であったといえる．

図 9 に t/T=2 における渦法とスペクトル法のエネルギースペクトルを示す．高波数成分において僅かな差異が見られるものの，二手法間でスペクトルは概ね定量的に一致しているといえる．渦法とスペクトル法の比較に関する過去の研究[18),20)] と比較すると，このような良好な一致を得る条件としては，FMM の多重極展開の次数を向上すること ($p = 14$)，FMM の周期境界の鏡像を十分多くとること (27^3)，粒子の再初期化を頻繁に行うこと (5 ステップに 1 回)，粒子の再初期化の際の反復解法の収束条件を厳しくすること (相対残差 1e-5) などが挙げられる．ここには示していないが，これらの 4 つの条件が全て満足されない限り今回のようなスペクトルの良好な一致は得ることができないことが確認されている．

4. 結　　論

大規模 GPU システムで高い演算性能を実現できるアルゴリズムとしての fast multipole method(FMM) の可能性を定量的に検討するため，TSUBAME 2.0 上で 4096GPU を用いたときの実アプリケーションにおける FMM と FFT の性能比較を行った．ウィーク・スケーリングは 4096 プロセスにおいて，FMM では 74%の並列化効率が得られているのに対してスペクトル法では 14%であった．また，最大規模の計算は $N = 4096^3$ でありこのときの持続演算性能は 1.01$PFlops$ で FMM としては著者らの知る限り過去最高である．アプリケーションには一様等方性乱流の解析を用い，FMM をベースとする渦法と，FFT をベースとするスペクトル法の間でエネルギースペクトルの定量的な一致が得られた．

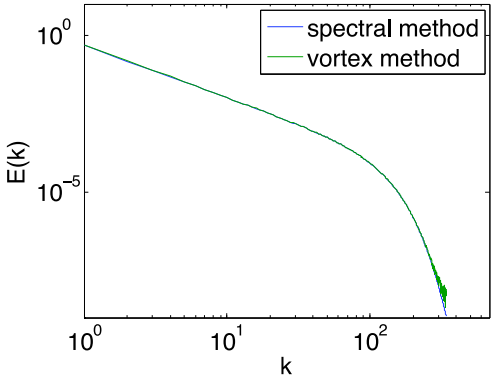

図 9 渦法とスペクトル法のエネルギースペクトルの比較
Fig. 9 Comparison of the energy spectrum between vortex method and spectral method

謝辞 本研究の計算機環境は TSUBAME2.0 の平成２３年秋期グランドチャレンジ大規模計算制度によるものである，ここに感謝の意を表する．

参 考 文 献

1) T.Hamada, R.Yokota, K.Nitadori, T.Narumi, K.Yasuoka, M.Taiji, and K.Oguri. 42 tflops hierarchical N-body simulations on GPUs with applications in both astrophysics and turbulence. In *SC '09 Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009.
2) A. Rahimian, I. Lashuk, K. Veerapaneni, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, J. Vetter, R. Vuduc, D. Zorin, and G. Biros. Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures. In *SC '10 Proceedings of the 2010 ACM/IEEE International*

Vol.2011-ARC-197 No.29
Vol.2011-HPC-132 No.29
2011/11/29

Conference for High Performance Computing, Networking, Storage and Analysis, 2010.

3) T.Darden, D.York, and L.Pedersen. Particle mesh ewald: An nlog(n) method for ewald sums in large systems. *Journal of Chemical Physics*, 98(12):10089–10092, 1993.
4) R.S. Rogallo. Numerical experiments in homogeneous turbulence. NASA Technical Memorandum 81315, NASA Ames Research Center, 1981.
5) J.Barnes and P.Hut. O(NlogN) force-calculation algorithm. *Nature*, 324:446–449, 1986.
6) L.Greengard and V.Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.
7) J.Dubinski. A parallel tree code. *New Astronomy*, 1:133–147, 1996.
8) H. Cheng, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithmin three dimensions. *Journal of Computational Physics*, 155(2):468–498, 1999.
9) Rio Yokota and L. A. Barba. Treecode and fast multipole method for N-body simulation with CUDA. In Wen-Mei Hwu, editor, *GPU Computing Gems Emerald Edition*, chapter9, pages 113–132. Elsevier/ Morgan Kaufman, 2011.
10) M.S. Warren and J.K. Salmon. A portable parallel particle program. *Computer Physics Communications*, 87:266–290, 1995.
11) W.Dehnen. A hierarchical O(N) force calculation algorithm. *Journal of Computational Physics*, 179(1):27–42, 2002.
12) Q.X. Wang. Variable order revised binary treecode. *Journal of Computational Physics*, 200(1):192–210, 2004.
13) H.Cheng, W.Y. Crutchfield, Z.Gimbutas, L.F. Greengard, J.F. Ethridge, J.Huang, V. Rokhlin, N. Yarvin, and J. Zhao. A wideband fast multipole method for the helmholtz equation in three dimensions. *Journal of Computational Physics*, 216:300–325, 2006.
14) Q.Hu, N.A. Gumerov, and R.Duraiswami. Scalable fast multipole methods on distributed heterogeneous architectures. In *SC '11 Proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011.
15) E. Darve, C. Cecka, and T. Takahashi. The fast multipole method on parallel clusters, multicore processors, and graphics processing units. *Comptes Rendus Mecanique*, 339:185–193, 2011.
16) P. D. Koumoutsakos and G.-H. Cottet. *Vortex Methods -Theory and Practice-*. Cambridge University Press, 2000.
17) R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: Theory and application to non-sperical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.
18) R.Yokota, T.K. Sheel, and S.Obi. Calculation of isotropic turbulence using a pure lagrangian vortex method. *Journal of Computational Physics*, 226:1589–1606, 2007.
19) R. Yokota, L. A. Barba, and Matthew G. Knepley. PetRBF — a fast radial basis function interpolation algorithm in parallel. *Comp. Meth. Appl. Mech. Engrg.*, 199:1793–1804, 2010.
20) R.Yokota, T.Narumi, R.Sakamaki, S.Kameoka, S.Obi, and K.Yasuoka. Fast multipole methods on a cluster of GPUs for the meshless simulation of turbulence. *Computer Physics Communications*, 180:2066–2078, 2009.