TSUBAME2.0 の全 GPU ノードを用いたペタフロップス規模の乱流解析

[1]泰岡　顕治、[2]成見　哲、[3]Lorena Barba、[4]横田　理央

1 慶應義塾大学　理工学部　機械工学科

2 電気通信大学　情報理工学部　情報・通信工学科

3 ボストン大学　機械工学科

4 アブドラ王立科学技術大学　数理情報工学科

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、2010 年の Gordon Bell peak performance 賞受賞論文に見られるようにその応用の幅は広い。本課題では、大規模 GPU システム向けに開発された ExaFMM ライブラリを用いて 4096^3 規模の乱流解析を行った。ただし、今回の解析に用いた FMM は 2010 年の Gordon Bell peak performance の FMM とは異なり、Treecode と FMM の長所を組み合わせたハイブリッド型の手法になっている。これにより、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 で実現した。TSUBAME2.0 のフルノードにおける実施では 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 であり、FMM の演算性能としては世界新記録であった。

Keywords: FMM, GPU, 自動最適化, 乱流、ペタスケール

1． 背景

乱流は我々の身の回りに広く存在し、エネルギーや環境に関する様々な工学上の問題に現れる。乱流解析は支配方程式の非線形性により空間的にも時間的にも高い解像度を要するため大規模な計算資源が必要となる。近年、計算機の発達により Navier−Stokes 方程式の直接数値解析が多くの場面で利用可能になってきている。

多くの工学的アプリケーションは複雑な境界条件や様々な物理的過程との連成が必要となる一方で、このような複雑な現象から孤立した素過程を抽出し解析することで乱流の普遍的な性質が明らかになることもある。一様等方性乱流はこのような素過程の一つであり、平均せん断流や壁面の影響を含まない高レイノルズ数乱流の基本的な分析が可能となる。

一様等方性乱流の直接数値解析にはこれまで擬スペクトル法が用いられてきた[1]。最大規模のものでは地球シミュレータ上で 2002 年に行われた 4096^3 格子点、Re_λ=1130 の計算が挙げられる[2]。石原ら[3]はこの大規模直接数値解析結果から、高次の乱流統計量は高レイノルズ数で異なった挙動を示すが、構造関数のスケール指数は普遍的である可能性が高いことを示した。このような結論は最高性能のスーパーコンピュータを利用して初めて得られるものであり、ハイパフォーマンスコンピューテイングの有用性を象徴しているといえる。

地球シミュレータから京コンピュータにかけて LINPACK ベンチマークにおける演算性能が約 300 倍向上したにも関わらず、上述の 4096^3 格子点の計算規模の記録は現在でも破られていない。これは以下の二通りの解釈が可能である。a) FFT はトーラスネットワークを用いた分散メモリ型の大規模計算機で性能を発揮できない。b) LINPACK ベンチマークは実アプリケーションの性能を予測する指標としては適当でない。前者の解釈では、次世代計算機で性能を発揮できるアルゴリズムの開発が必要であるということになる。後者の見方では、LINPACK に変わる新たなベンチマークの採用を促進し、ハードウェアとソフトウェアの協調設計に用いることが求められる。

本研究では一様等方性乱流の計算を行うための新たな手法を提案し、TSUBAME2.0 上で良好なスケーラビリティが得られることを示す。ただし、スケーラビリティはパフォーマンスとは直接関係のない指標であることに注意されたい。寧ろ計算の遅いコードほど通信を隠蔽することができるため、良好なスケーラビリティを得やすい。このため、本計算ではスケーラビリティだけでなく計算時間も比較し、次世代のハードウェアとアルゴリズムの協調設計に役立つような指標を提供するよう心がけた。

2． 手法

2．1 渦法

「渦法」と称される数値解析手法には様々なものがある[4]。離散化手法で大別すれば、格子を一切用いない Lagrange 型のものと、格子と粒子を併用する semi−Lagrange 型の手法がある。支配方程式の定式化で分けると渦度−流れ関数によるものと渦度−速度によるものが挙げられる。いずれの場合も渦度方程式を解くことになるが、このとき粘性拡散項の粒子上での扱いにも様々な手法が提案されている。全ての「渦法」に共通の特長としては，対流項が Lagrange 的に扱われるため、数値拡散や数値分散の問題から解放されるという点が挙げられる。さらに、圧力でなく渦度をベースとする定式化を行うことで計算領域を渦度がある場所に限定できる。これは渦輪や翼端渦のような渦運動が支配的な外部流において計算点の大きな節約につながるため、演算とメモリ使用量を大きく削減することができる。

本計算では完全メッシュフリーの離散化手法と渦度−速度による定式化を採用した[5]。このためにまず渦度場を Gauss 分布を基底関数にもつ渦粒子の重ね合わせによって表す。渦度−速度による定式化では速度の Poisson 方程式

$$\nabla^2\mathbf{u} = -\nabla\times\boldsymbol{\omega}$$

と渦度方程式

$$\frac{\partial\boldsymbol{\omega}}{\partial t} + \mathbf{u}\cdot\nabla\boldsymbol{\omega} = \boldsymbol{\omega}\cdot\nabla\mathbf{u} + \nu\nabla^2\boldsymbol{\omega}$$

を交互に解く。速度のPoisson方程式はGreen関数を用いた積分方程式に変換するとBiot-Savart 式になる。これは全ての粒子間の相互作用を伴うN体問題に帰着し、高速多重極展開法(FMM)を用いて解くことができる。渦度方程式の各項は個別に解くが、更新する変数が各項で異なるため部分段階法には相当しない。対流項は粒子の座標を更新することで正確に解くことができる。伸張項の計算は速度にBiot-Savart 式を代入することでN体問題に変換できる。拡散項は基底関数であるGauss分布の標準偏差sを増大させることで考慮した。Lagrange 型の離散化手法が収束するためには基底関数が十分重複する必要があるため、本計算では数ステップに一回渦粒子の座標を等間隔に再配置し、基底関数のsを再初期化し、渦粒子の強度は元の渦度場を再現するように調節した。このとき渦強度を調節する計算は動径基底関数を用いた補間によって行った。ただし、動径基底関数には渦要素の基底関数に合わせてGauss分布を用いた。

2．2 高速多重極展開法(FMM)

Biot-Savart式と渦度方程式の伸張項の計算は全ての粒子同士の相互作用の計算になるため直接解くと $O(N^2)$の計算量になる。高速多重極展開法(FMM)を用いることでこのときの計算量を $O(N)$に低減することができる。

Biot-Savart 式は Poisson 方程式から導出できるため Laplace 型の FMM カーネル関数を用いることができる。Biot-Savart 式の右辺と渦度方程式の伸張項にはそれぞれ回転と勾配の空間微分演算子が存在するため Laplace カーネルの二回微分までの項が必要となる。FMM で用いる多重極展開と局所展開にはこれらの微分が既に含まれているため、この情報をそのまま用いることができる。

2．3 負荷分散

FMMを並列化する際に問題となるのはN体問題に内在するデータの大域的な依存性と粒子のダイナミックな性質である。格子法とは異なり、N体問題では領域分割法を用いて粒子群を分割したとしても問題を局在化することはできない。これは、それぞれの部分領域にある粒子が必要とする情報が結局全領域にまたがっているためである。FMM では大域的な木構造を用いた領域分割法が良く用いられるが、粒子の分布が非一様で木構造に偏りがある場合には領域を等分割すると逆に負荷は偏ってしまう。さらに、粒子が毎ステップ移動するため木構造を構築し直す必要があり、高速に木構造を生成、負荷分散する機構が不可欠となる。

FMMの負荷分散の問題を解決する巧妙な手法として前のステップの負荷のばらつきを元に現ステップの領域分割を更新するものが挙げられる。このような手法は90年代前半から共有メモリ[8]、分散メモリ[9]の負荷分散にそれぞれ用いられてきた。前のステップの情報を元に負荷分散を微調整する発想自体は Orthogonal recursive bisection (ORB)[7]、Morton/Hilbert key の分割法[9]、グラフ理論を用いた領域分割法[10]などの基本的な負荷分散の手法と併用することができる。本計算に用いる FMM コードは ORB、Morton key の分割の両方から最適なものを選ぶことができる仕組みになっている。

上述の方法で粒子の領域分割が行われると、それぞれの領域で木構造が生成される。次に、この局所的な木構造の一部を各プロセスが相互に送信しながら各部分領域にある粒子が必要な大域的な木構造(Local Essential Tree)を構築する必要がある。この LET の構築に伴う通信は大域的な通信になるため、大規模な分散メモリの計算ではここがスケーラビリティのボトルネックとなる。LET の通信の最適化手法として ORB の二分木の各階層を次元に見立てた超立方体に沿った通信パターンを用いるものがある。これは treecode[11]にも FMM[12]にも適用することができる。本計算で用いたFMMコードでは上記の手法をプロセス数が 2 の冪乗でない場合に拡張したものを用いた。さらに、周期境界 FMM に上記の通信方法が使えるようにも拡張した。

2．4 Dual Tree Traversal

粒子を領域分割し、各プロセスで局所的に木構造を生成し、LET を通信するところまでは treecode と FMM の手順は全く同様である。これら二手法間の差異は LET を走査する方法にある。ただし、以下の説明では粒子間相互作用の作用される側をターゲット、作用を及ぼす側をソースと定義する。また、木構造の末端にあるセルを葉セルと呼ぶこととする。 木構造の生成時には、葉セルあたりの粒子数ができるだけ均一になるようにする。Treecode ではターゲットの木構造の葉セルに関するループをまわし、各葉セルに対してソースの木構造を走査する。一方、FMM では通常木構造の走査は行わず、ターゲットセルに関するループをまわして各ターゲットセルについて作用するソースセルのリストを逐次生成する。粒子の分布が不均一で木構造に偏りがある場合にはこのソースセルのリストの生成が煩雑になる。このため、FMM では通常 multipole acceptance criterion (MAC)の概念は用いず、影響距離の固定されたソースのリストを使用する。

Dual tree traversal[13]を用いることで上記のソースのリストの生成を簡略化・最適化することができる。これは、FMM でよく用いられる U,V,W,X リスト[12]に MAC の概念を付加したものと考えることもできる。図1に dual tree traversal の概念図を示す。基本的な手順はターゲットとソースの双方の木構造を同時に走査するだけである。この際に生じるターゲットとソースのセルのペアに対してMACを適用し、セル同士の計算を行うのに十分な距離にあるかを判定する。十分でなければ大きい方のセルを分割し、再びターゲットとソースのセルのペアに対して MAC による判定を行う。これにより双対な木構造の走査を行い、ターゲットとソースの両方が葉セルに達した時点で距離が未だに十分遠くなければ直接計算を行う。Dual tree traversal はターゲットの親セルから子セルへソースの候補を継承することで、各階層で排他的なMACベースのソースリストを構築する最適な手法を実現しているといえる。

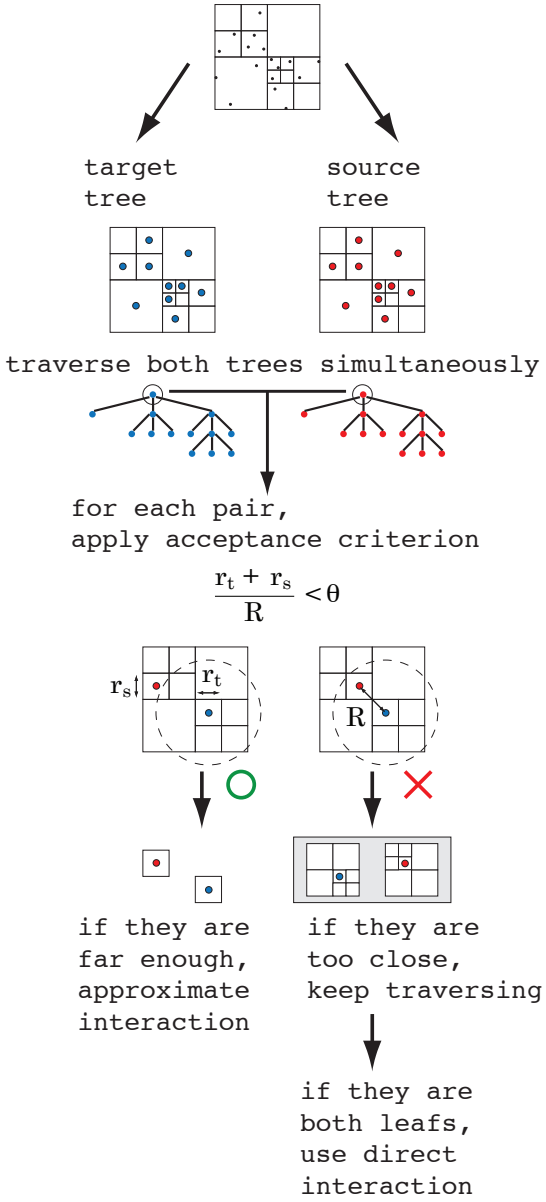

図1　Dual tree traversal の概念図

2．5　CPU/GPU 上での自動最適化

Treecode や FMM などの階層的な N 体問題の高速化手法では近傍場の particle-particle (P2P)カーネルと遠方場の multipole-to-local (M2L)や multipole-to-particle (M2P)カーネルが計算時間の大半を占める。また、この際の近傍場の P2P と遠方場の M2L/M2P の計算時間のバランスが重要である。CPU と GPU が混在する環境ではこれらのカーネルの加速率にばらつきがあるため[14]バランスをとることは容易ではない。さらに、多重極展開や局所展開自体にも様々な基底のものが存在し、計算機のアーキテクチャによってそのカーネルの性能は異なる。

これらのカーネルの中から最適なものを選ぶ簡単な方法として、各カーネルの処理時間を予め計測し実行時にその情報をもとに自動最適化を行うものがある。Dual tree traversal 中に MAC を満たすセルの各ペアについて P2P (direct)、M2P (treecode)、M2L(FMM)のカーネルから最適

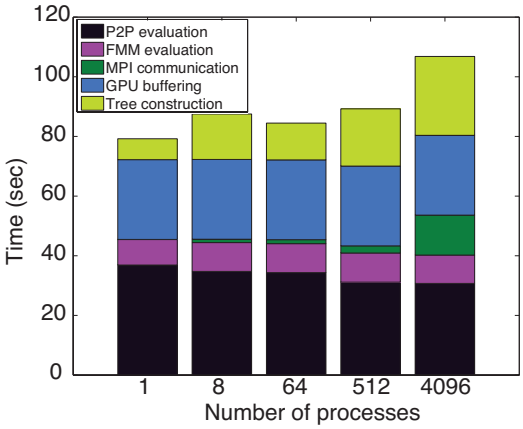

図2　GPU あたり N=256^3 粒子を用いた 4096 GPU までの FMM の weak scaling

なものを選ぶことでいずれの単独の手法よりも常に高速な実装となる。このことは本コードを用いた性能試験で実証している[15]。

3．　グランドチャレンジ制度で実施した計算内容

本実施では FMM コードにより高速化された渦法と FFTW による擬スペクトル法を用いた Re_∞=500 の一様等方性乱流の解析を行った。計算領域は$[-\pi,\pi]^3$ の周期的な立方体で格子点および粒子の数は 4096^3 であった。FMM に関しては 27^3 の周期鏡像を置くことで周期境界条件を考慮した。また、FMM の級数展開の次数は p=14 とした。初期条件は指定したエネルギースペクトルにランダムな位相を持たせたものを波数空間の速度場として生成した。擬スペクトル法はこの情報をそのまま初期条件として用いた。渦法では波数空間の速度場をまず実空間へと変換し各セルの中心点(スタガード格子における圧力の計算点)における渦度を計算した。次に、渦粒子を同じくセルの中心点に配置し、動径基底関数を用いた補間法によって渦度場を再現するように渦粒子の強度を計算した。渦粒子の渦核半径はオーバラップ比が1となるように$\sigma=\Delta x$ とした。

4．　使ったプログラムについて

本計算では著者らによって開発されたオープンソースの FMM ライブラリである ExaFMM を用いた. ExaFMM は C++ベースで OpenMP, CUDA, MPI による並列化が可能である. ソースコードは mercurial によりバージョン管理されており bitbucket 上に置かれている. [1]ExaFMM には2節で述べた手法が全て実装されているだけでなく, 他にも様々な手法の中からコンパイル時や実行時に最適なものを選択できるようになっている.

5．　スケーラビリティ

本計算は全て TSUBAME 2.0 上で行った。TSUBAME 2.0 の thin-node は Intel Xeon5670 を 2 ソケット、NVIDIA M2050 を 3 カード、54GB の RAM、120 GB

[1] https://bitbucket.org/exafmm/exafmm

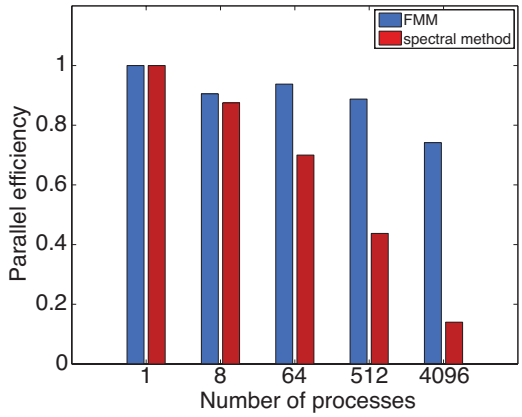

図３ FMM とスペクトル法の 4096 プロセスまでの並列化効率

のローカル SSD を有するノード 1408 台からなる。ノード間はフルバイセクション・ノンブロッキングの fat-tree ネットワークにより接続されており、各ノードは dual link の QDR Infiniband でリンクされている。

図２に GPU あたり $N=256^3$ 粒子を割り当てたときの 1 から 4096GPU までの weak scaling とそのときの計算時間の内訳を示す。本計算では GPU あたり MPI プロセスを１つ割り当てたため、MPI プロセス数と GPU 数は等しい。計算時間は Biot-Savart 式と渦度方程式の伸張項の両方を含んでいる。 凡例の「P2P evaluation」は近傍場の P2P カーネルの計算時間、「FMM evaluation」はその他の FMM カーネルの合計値、「MPI communication」は全ての通信時間の合計値、「GPU buffering」は GPU 周りのデータのバッファリングと GPU との送受信の
合計、「Tree construction」は木構造の生成と領域分割、負荷分散の時間を表している。ただし、MPI の通信はローカルな P2P カーネルの計算とオーバーラップしており棒グラフからはこの重複部分は引かれている。このため、棒グラフの合計の高さは全体の実行時間と一致している。最も大きい 4096GPU を用いた計算では $N=4096^3$ の粒子の計算を 100 秒程度で行い 1.01 Pflop/s を達成した。

図２を見ると「GPU buffering」に時間がかかっていることが分かる。ただし、これは FMM のカーネルを GPU 上で効率的に実行するために必要であり、実行時間の合計値を低減する効果があることが分かっている。また、この部分は非常に良くスケーリングするため、FMM 全体のスケーラビリティには全く影響しない。FMM のスケーラビリティに効いてくるのは「MPI communication」と「Tree construction」である。「Tree construction」の中にも MPI 通信があり、これがスケーラビリティを低下させている。2.3 節で述べた超立方体に沿った階層的な LET の通信を行う手法は、TSUBAME 2.0 のネットワーク上では単純な MPI_Alltoallv に比べて低速であった。このため、図２では MPI_Alltoallv を用いたときの通信時間を示している。

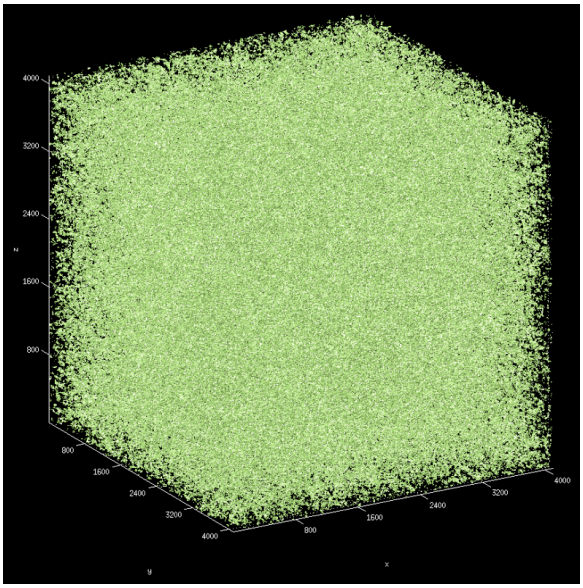

図４ 渦法の計算結果から得られた速度勾配テンソルの第二不変量の等値面

擬スペクトル法に関しても同様な weak scaling の試験を行い、図３にこのときの並列化効率を FMM と比較したものを示す。ただし、本計算を行った時点では CPU 上の FFTW と GPU 上の CUFFT の計算速度に（CPU-GPU 間の通信時間を含めると）大きな差異が見受けられなかったため、擬スペクトル法の計算はノードあたり３プロセスを用いて CPU 上で行った。4096 プロセスを用いたときの並列化効率
は FMM では 74%でスペクトル法では 14%であった。計算時間は両手法ともに１タイムステップあたり 100 秒程度であった。

6. 一様等方性乱流の解析

図4に t/T=2 における渦法の計算結果から得られた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 は大規模渦の回転時間である。微小な渦構造が多数見受けられるが、大規模はコヒーレント渦構造は計算時間が不十分なため確認することができなかった。計算時間は TSUBAME 2.0 のフルノードを占有できる時間の制約により僅かなものに限られた。ただし、渦法の計算精度やスケーラビリティの違いを定量化するためにはこのような短時間の計算で十分であるといえる。

図5に t/T=2 における擬スペクトル法と渦法の速度場から得られるエネルギースペクトルを示す。ただしここでは、擬スペクトル法の結果が正しいものとして渦法の計算精度を検証することを目的とする。高波数成分において僅かな差異はみられるものの両手法間のスペクトルは定量的に一致していることが見てとれる。過去の一様等方性乱流の渦法による解析例[5][16]と比べて高い精度が得られている要因として以下のことが挙げられる。今回の計算では FMM の級数展開の次数が高く(p=14)、周期鏡像の数も多い(27^3)。また、再初期化の頻度が高く(5 ステップに1回)、再初期化の際の動径基底関数の補間の収束判定を厳しく設定した($|L|^2$=1e−5)。もちろん、膨大な数の渦粒子(4096^3)を用いていることも高い精度を実現できた要因の一つである。逆に、これらの条件を全て満たさない限り

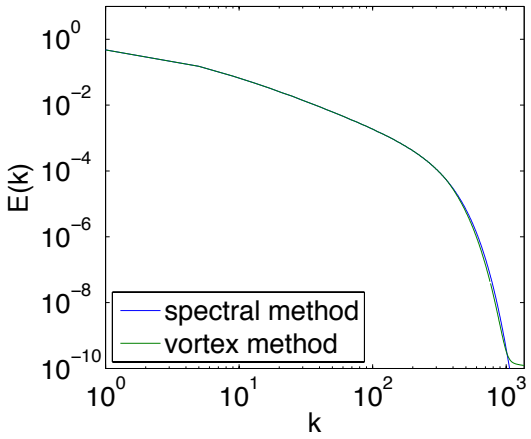

図5 擬スペクトル法と渦法の t/T=2 におけるエネルギースペクトル

高精度な渦法計算は実現できないことが分かった。

7. 得られた結果のインパクト

本計算で達成した持続演算性能 1.01Pflop/s は FMM としては著者らの知る限り最速であった (2010 年の Gordon Bell Peak Performance 賞は 0.7Pflop/s であった). 4096^3 粒子を用いた計算も著者らの知る限り粒子系の流体解析としては最大規模のものであった. また, FMM と FFT を同等のアプリケーションについて比較し, スケーラビリティに関する定量的な差異を確認できたことは意義深い. 今後, ポストペタスケールの大規模計算機において FMM と FFT の相対的な計算速度を比較していくことは, アーキテクチャの変遷に伴う最適なアルゴリズムの変化の観点からますます重要になると考えられる.

8. まとめ

本研究では一様等方性乱流を例にとって、二つの異なる計算手法による解析を同等の計算条件の元で行い、計算精度と計算速度、スケーラビリティに関する比較を行った。FMM をベースとする渦法による解析と FFT をベースとする擬スペクトル法の解析を 4096^3 の粒子・格子点を用いて行い TSUBAME 2.0 上で最大 4096 GPU まで用いた計算を行った。このとき FMM の並列化効率は 74%であったのに対して FFT の並列化効率は 14%であった。両手法とも計算時間は1ステップあたり 100 秒程度であり、4096^3 を用いた渦法計算は 1.01Pflop/s の持続演算性能に相当する。渦法と擬スペクトル法のエネルギースペクトルは定量的に一致し、本手法の計算精度を検証することができた。ただし、TSUBAME 2.0 全体を占有できる時間が限られていたため、高次の乱流等計量を得るのに十分な計算時間を確保することができなかった。今後、同様の計算資源が安価に利用できるようになれば、FMM を用いた乱流解析の有用性が明らかになると期待される。

参考文献

[1] S. A. Orszag and G. S. J. Patterson: Numerical Simulation of Three Dimensional Homogeneous Isotropic Turbulence. *Phys. Rev. Lett.* Vol. 28, pp. 76-79 (1972)

[2] M. Yokokawa, K. Itakura, A. Uno, T. Ishihara, and Y. Kaneda: 16.4-Tflops Direct Numerical Simulation of Turbulence by a Fourier Spectral Method on the Earth Simulator, *Proc. ACM/IEEE Supercomput. Conf.*, Washington DC, pp. 50-66 (2002)

[3] T. Ishihara, T., Gotoh, and Y. Kaneda: Study of High-Reynolds Number Isotropic Turbulence by Direct Numerical Simulation, *Annu. Rev. Fluid Mech.,* Vol. 41, pp. 165-180 (2009)

[4] G. H. Cottet and P. Koumoutsakos: Vortex Methods, *Cambridge University Press* (2000)

[5] R. Yokota, T. K. Sheel, and S. Obi: Calculation of Isotropic Turbulence Using a Pure Lagrangian Vortex Method, *J. Comput. Phys.,* Vol. 226, pp. 1589-1606 (2007)

[6] H. Cheng, L. Greengard, and V. Rokhlin: A Fast Adaptive Multipole Algorithm in Three Dimensions, *J. Comput. Phys.,* Vol. 155, pp. 468-498 (1999)

[7] M. S. Warren and J. K. Salmon: Astrophysical N-body Simulation Using Hierarchical Tree Data Structures, *Proc. ACM/IEEE Supercomput. Conf.*, pp. 570-576 (1992)

[8] J. P. Singh, C. Holt, J. L. Hennessy, and A. Gupta: A Parallel Adaptive Fast Multipole Method, *Proc. ACM/IEEE Supercomput. Conf.*, pp.54-65 (1993)

[9] M. S. Warren and J. K. Salmon: A Parallel Hashed Oct-tree N-body Algorithm. *Proc. ACM/IEEE Conf. Supercomput.,* pp. 12-21 (1993)

[10] S.-H. Teng. Provably Good Partitioning and Load Balancing Algorithms for Parallel Adaptive N-body Simulation. *SIAM J. Sci. Comput.,* Vol. 19, pp. 635-656 (1998)

[11] T. Hamada, K. Nitadori, K. Benkrid, Y. Ohno, G. Morimoto, T. Masada, Y. Shibata, K. Oguri, and M. Taiji: A Novel Multiple-walk Parallel Algorithm for the Barnes-Hut Treecode on GPUs - Towards Cost Effective, High Performance N-body Simulation, *Comput. Sci. - Res. Dev.*, Vol. 24, pp. 21-31 (2009)

[12] I. Lashuk, A. Chandramowlishwaran, H. Langston, T.-A. Nguyen, R. Sampath, A. Shringarpure, R. Vuduc, L. Ying, D. Zorin, and G. Biros: A Massively Parallel Adaptive Fast Multipole Method on Heterogeneous Architectures, *Proc. Conf. High Perform. Comput. Netw.*

Stor. Anal., (2009)

[13] W. Dehnen: A Hierarchical O(N) Force Calculation Algorithm, *J. Comput. Phys.,* Vol. 179, pp. 27-42 (2002)

[14] R. Yokota and L. A. Barba: Treecode and Fast Multipole Method for N-body Simulation with CUDA, *GPU Computing Gems*, Morgan Kaufmann (2011)

[15] R. Yokota and L. A. Barba: Hierarchical N-body Simulations with Auto-tuning for Heterogenous Systems. *Comput. Sci. Eng.*, Vol. 14, pp. 30-39 (2012)

[16] 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. *Comput. Phys. Comm.,* Vol. 180, pp. 2066-2078 (2009)