論文 / 著書情報
Article / Book Information

論題(和文)	大規模境界要素法解析における分散並列 FMM の通信最適化
Title(English)	Communication Optimization of Distributed Memory FMM for Large Scale Boundary Element Methods
著者(和文)	横田理央
Authors(English)	Rio Yokota
出典(和文)	シミュレーション, Vol. 35, No. 3, pp. 147--153
Citation(English)	Simulation, Vol. 35, No. 3, pp. 147--153
発行日 / Pub. date	2016, 9

大規模境界要素法解析における分散並列FMMの通信最適化

東京工業大学　学術国際情報センター
横田理央　rioyokota@gsic.titech.ac.jp

概要
単一ノードのメモリに収まらないような大規模な境界要素法解析を行う場合、分散メモリ並列処理が必要になる。このような分散メモリ実装では密行列を反復法で解き、行列・ベクトル積の部分に Fast Multipole Method (FMM) を用いるのが一般的である。よって、FMM の分散メモリ処理における並列化効率が計算全体の速度に大きく影響する。ここでは FMM の並列化効率向上を目的として通信の最適化手法について検討を行う。

キーワード：Fast Multipole Method、境界要素法、分散並列処理、動的負荷分散

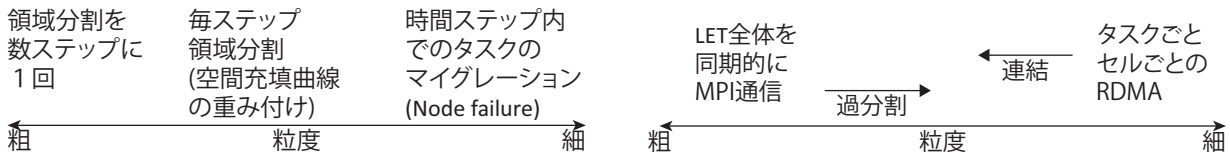
1　はじめに

Fast Multipole Method (FMM) を用いた境界要素法解析の高速化は音響解析 [1, 2]、生体分子 [3]、電磁界解析 [4, 5]、Euler 方程式 [6] や Stokes 方程式 [7] による流体解析、地質学 [8] や地震学 [9, 10] の分野で用いられている。これらの境界要素法解析を用いて単一ノードのメモリに収まらないような大規模な計算を行う場合、Message Passing Interface (MPI) などを用いた分散並列実装を行う必要がある。このような分散並列の境界要素法においては、FMM の分散並列化の部分に最も多くの通信が発生するため、ここがボトルネックとなる。

FMM 自体の分散並列実装は多く存在する。Bédorf ら [11] は複数の GPU 上で動作する Barnes-Hut 型の手法 [12] を実装した。木構造の構築時に Hilbert 曲線を用いて 3 次元 Euclid 空間における局在性を 1 次元のメモリ空間に写像することで領域分割を実現している。具体的には GPU のスレッド 1 つに対して粒子を一つ割り当て、木構造の構築をマスク処理を用いて条件分岐をすることなく GPU 内で行っている。木構造の走査も同様のマスク処理と warp 内の reduction、thread block 間の atomic 処理をうまく用いて条件分岐をすることなく行っている。これにより、280 万粒子の相互作用計算を 1GPU で 1 秒で行うことができ、それを Oak Ridge National Laboratory (ORNL) の Titan の全ノードまでスケーリングさせた。この時の演算性能は単精度・倍精度混合で 24.77 PFlops であった [13]。

Speck ら [14] は時間方向の並列を併用することで 262,144 コアの強スケーリングの計算を実現した。これはノードあたりの粒子数が非常に少ない場合においても時間方向の並列化を生かすことで並列化効率を向上することができることを表している。Jülich 研究所の JUGENE の 262,144 コアを用いた計算では最大 20 億粒子の計算を行った [15]。Lashuk ら [16] は hypercube 型の通信パターンを用いて FMM における全対全通信を $\mathcal{O}(PN^{2/3})$ から $\mathcal{O}(\sqrt{P}N^{2/3})$ に低減した。ただし、この時の P は MPI プロセス数、N は全ノードの粒子数の合計値である。この通信手法を用いることにより 196,608 まで FMM をスケーリングさせることに成功した。Griebel ら [17] は Hilbert 曲線による領域分割の際に木構造の最下層における計算量による重み付けを行った。

Sundar ら [18] は木構造を構築する際に隣接するセルの大きさの比が 2:1 を超えないように制御する手法を開発した。これにより、非一様な木構造の領域分割を行う際の相互作用リストを均



(a) 領域分割における粒度のバランス　　(b) LET 通信における粒度のバランス

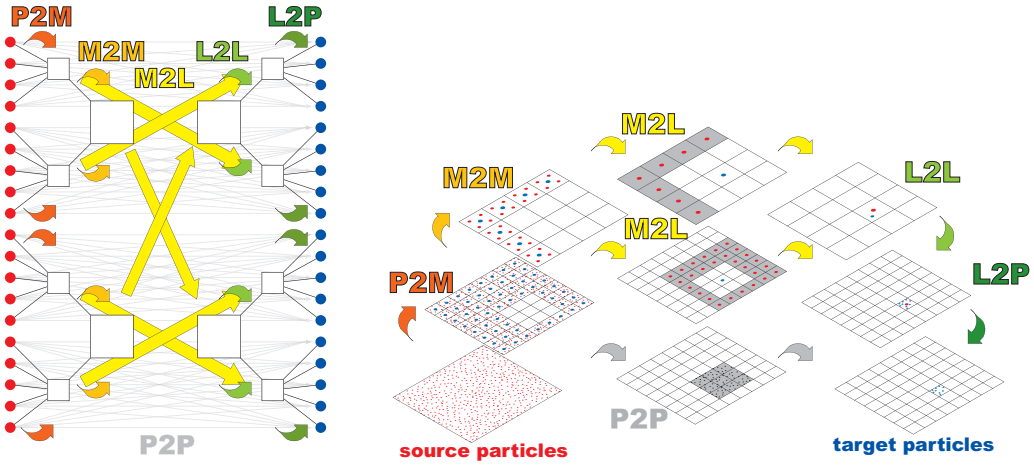
図 1: FMM の通信は大きく２段階に分けることができる。一つ目は領域分割に伴う粒子の移動によるもの、二つ目は LET の通信によるものである。前者は動的負荷分散に関係しており、後者はリモートデータのプリフェッチに関係している。1(a) は領域分割における粒度の種類を、1(b) は LET 通信における粒度の種類をそれぞれ示している。計算とオーバーラップを考慮した MPI の同期的通信は粒度が最大の場合の非同期通信と見ることができる。

等に割り振りやすくなる。Sundar らは Barnes-Hut や ParaDis などでこの仮説の検証を行い、実証することができた。Zandifar ら [19] はランタイムである STAPL 上の skeleton を用いて FMM を実装し、bucket sort や alltoall などを用いて分散している木構造の通信量を最適化した。MPI による実装と比べて同等の性能をランタイムを用いて逐次コードから生成できることを示した。

これらの研究例では MPI による同期的な通信を行っているものがほどんどであり、大型計算機上では同期にかかる時間がボトルネックとなる。また、ランタイムなどを用いて非同期的に実行している研究例では非常にベーシックな FMM を用いており、通信の部分が最適化されている MPI 実装に比べて性能が出ていない。

新たな分散並列化モデルを FMM に適用する場合、既存の分散メモリ型の FMM に動的負荷分散やプリフェッチの機構が存在していることを理解する必要がある。分散メモリ型の FMM ではデータの局在性を最大化しなから、計算負荷を等分割するようにまず領域分割を行う。これは前のステップの計算負荷で空間充填曲線を重み付けしてから、曲線を等分割することで実現できる。また、分散メモリ型の FMM では各プロセスで必要になるリモートプロセスの情報を前もって通信しておくのが一般的である。この通信によって集められてきた部分木のことを local essential tree (LET)[20] といい、最大粒度でのリモートデータのプリフェッチと捉えることができる。

既存の分散メモリ型の FMM の改善すべき点の一つとして、負荷分散とプリフェッチを行う際の粒度の問題が挙げられる。図 1 に FMM の通信手法を粒度ごとに並べたものを示す。図 1(a) は領域分割における粒度の種類を表しており、動的負荷分散の頻度に関連している。既存研究では動的負荷分散を行うタイミングは１時間ステップに１回に固定されている場合が多い。通信の粒度をさらに粗くすることは、領域分割のアップデートを数ステップに１回にすることを意味する。分子動力学のように粒子が振動しながらゆっくり移動する様な計算では数ステップに１回に領域分割の頻度を落としたとしても大きな負荷の不均衡には繋がりにくい。逆に、１ステップに数分かかるような大規模な計算を大型計算機で行っている場合にノードの故障などが起きたとすると、その時間ステップが終了するまで待たずに負荷を再配分できる機構があることが望ましい。このような場合は図 1(a) の右端に示すように時間ステップ内で動的負荷分散を行えるようにすることで耐故障性を有する FMM を構築することができる。図 1(b) には LET 通信における粒度の種類を表している。既存研究では `MPI_alltoallv` を用いて１回の通信で LET を全て送るのが一般的である。`omp sections` を用いて１スレッドを通信専用に割り当てることで通信と計算をオーバーラップすることもできる。これは非同期通信の粒度を最大にした場合に相当する。細粒度の非同期通信では図 1(b) の右端にあるように、木構造の一つ一つ



(a) 木構造間の相互作用　　(b) 階層間のデータの移動

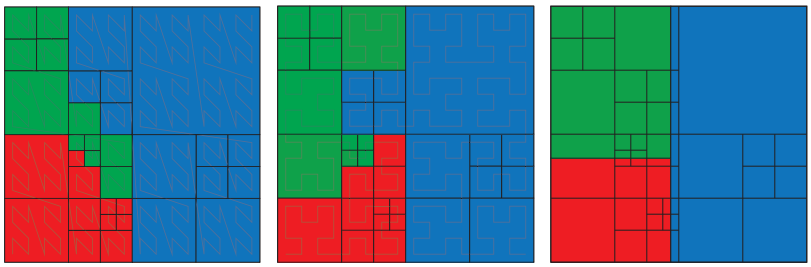
図 2: FMM 計算の全体の流れを示す。図 2(a) は木構造間の相互作用の様子を、図 2(b) は階層間のデータの移動の様子を示す。P2M, M2M, M2L, L2L, L2P, P2P はそれぞれ別の FMM カーネルの計算を表している。

のノードを Remote Direct Memory Access (RDMA) で送信することも考えられる。最適な粒度はこれらのような両極端の場合ではなく、この間のどこかにあることが予想される。

2　FMM における領域分割

FMM に内在する非同期性を抽出するには、FMM の計算内部のデータの依存関係を理解する必要がある。FMM には 6 種類の異なる関数が存在し、それぞれの計算量やデータの依存関係は全く異なる。図 2 に FMM の計算の流れやデータの依存関係を 2 種類の概念図を用いて図示する。図 2(a) は FMM で用いる木構造間の相互作用の様子を示している。FMM は粒子間の相互作用を階層的に計算する手法であるが、その相互作用の際に作用を及ぼす粒子をソース (赤)、作用を受ける粒子をターゲット (青) とする。FMM の計算手順には P2M, M2M, M2L, L2L, L2P, P2P と呼ばれる 6 つのステージがある。図 2(b) にはこれらのステージ及び階層間のデータの依存関係を示す。図 2(a) において、P2P は灰色の線だ表される相互作用に対応する。FMM の重要な特徴として近傍との相互作用は密で遠方との相互作用は疎であるというのがある。これは FFT のデータ依存関係とは対照的であり、分散メモリ型の並列化を行う際に遠方との通信が少なくなることを表している。また、マルチグリッド法などとも異なり、FMM は階層間の同期を行う必要がなく、非同期性が非常に高い。FMM のデータ依存性は郵便のシステムに似ており、P2M は個人から地方郵便局、M2M は地方郵便局から中央郵便局、M2L は航空輸送、L2L が中央郵便局から地方郵便局、L2P が地方郵便局から個人への配達に相当する。また、P2P は近所の人々が直接手渡しでものをやり取りする方法に相当する。このように、個人対個人のやりとりが階層的に効率化される仕組みは郵便システムとの対比で容易に理解できる。

FMM の領域分割法は大きく分けて hashed oct-tree (HOT)[20] と orthogonal recursive bisection (ORB)[21] の二つがある。HOT は空間充填曲線に沿って領域を分割していく方法で Morton 曲線を用いるか Hilbert 曲線を用いるかで、図 3(a) と 3(b) に示すような違いがある。この際に空間充填曲線に沿って振られる数字を一種のハッシュと見ることができることが HOT の名前の由来である。この数字は木構造の階層が深ければ大きな値になり、10 階層を超えた時点で 32-bit の整数には収まらなくなる。64-bit の非負整数を用いても 21 階層までしか表すことができない



(a) HOT (Morton) (b) HOT (Hilbert) (c) ORB

図 3: FMM の様々な領域分割法。(a) Morton 曲線を用いた分割法 (b) Hilbert 曲線を用いた分割法 (c) Orthogonal recursive bisection (ORB) を用いた分割法

ため、超大規模計算においては何らかの工夫が必要である。

ORB では領域全体を粒子数が均等になるように繰り返し割っていき、その際に分割の方向を「x, y, z, x, y, z, ...」と変化させることで 3 次元的な分割を実現する。HOT で作られる各領域が立方体のブロックの集まりのようになるのに対して、ORB では各領域は大きな一つの直方体になる。粒子が非一様に分布している場合、直方体の辺長比が長くなるのを防ぐために「x, y, z, x, y, z, ...」ではなく、常に一番長い辺を分割するように改良する必要がある。プロセス数が 2 の冪乗でない場合には 2 等分の代わりに 3 等分を用いることで任意のプロセス数の領域分割に対応できる。

HOT、ORB いずれの分割法においても、各領域で粒子数（演算量）が等しくなるようにバランスすることは容易ではない。HOT の場合は Morton/Hilbert 曲線に沿って粒子をソートする必要がある。ただし、ここで行うのは空間充填曲線によって振られる番号を bin として用いた bucket sort や radix sort などの比較なしのソートである。ORB においても座標の大小で粒子を振り分ける作業が必要になる。これは sort と言うよりは n 番目の要素を探す `nth_element` の操作に近い。これらを分散メモリで行う場合にはサンプリングを用いた手法が有効であり、既存研究では ORB[22] にも HOT[23] にも既に適用されている。粒子から木構造を構築する操作は粒子をソートする操作に似ている。特に、ORB は分割統治型の merge sort に似ており、HOT は 1 ビットずつ見ていく点で radix sort に似ている。よって、分散メモリ用のソーティングにおいて有効なサンプリングを用いる手法は分散メモリ上で木構造を構築するのにも有効であることがわかる。サンプリングを用いた分散メモリ用のソートは 2 段階に分けることができる。一つ目は、領域同士の境目に来る粒子番号の特定である。二つ目が、領域の境目が全て分かったところで実際に粒子をそれぞれのプロセスに送ることである。図 4 にこのようなサンプリングを用いた分散メモリ用のソートの概念図を示す。ここではまず、粒子を 8 つのグループに分けることにする。この 8 つに分ける基準は ORB の場合は分割したい方向の座標の値、HOT の場合は空間充填曲線の通し番号の値である。ただし、全てのプロセスで共通の値でグルーピングしなくてはならない。グループごとにローカルのヒストグラムを生成し、そのヒストグラムの値を `MPI_Allreduce` で足し合わせるとグローバルな中央値がどのグループにあるかが特定できる。次に、この中央値があると特定されたグループのみをさらに 8 つに細かく分割し、同様の操作を繰り返す。この操作を数回繰り返すと中央値が正確に特定でき、ORB も HOT もそれを領域分割に用いることができる。領域の境目が全て分かると粒子を `MPI_Alltoallv` で交換することができる。FMM では図 2 に示される計算を何ステップも行うが、領域分割の通信時間は初回が最も多く 2 ステップ目以降では大きく減少する。

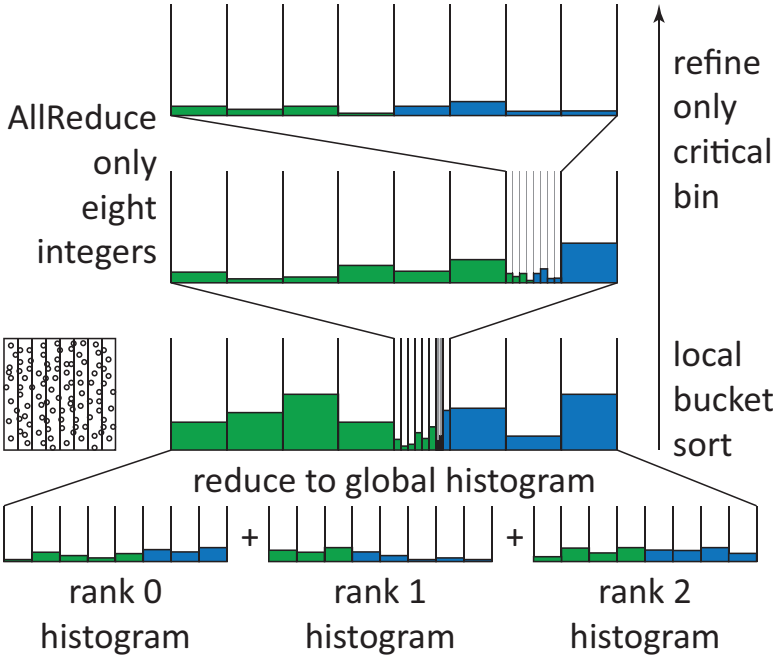


図 4: ヒストグラムを用いた分散メモリ用のソート

上述の領域分割法を用いることで粒子数を各プロセスに均等に割り振ることはできるが、FMM では粒子数と計算量は必ずしも比例しない。これは、一定の近接要素としか相互作用しない有限差分法、有限要素法、Smoothed Particle Hydrodynamics (SPH) 法などとは異なり、遠方の要素とも相互作用する FMM の場合、粒子数が同じでもそれらと相互作用する粒子の数が異なれば計算量は異なる。よって、粒子の分布が偏っているような問題では、粒子数のヒストグラムを用いたのでは負荷分散は実現されない。この問題に対処するために最もよく用いられるのが、前のステップの計算時間による重み付けである [20]。これは元々HOT の負荷分散のために開発された手法であるが ORB にもそのまま適用できる。前のステップの理論的な計算量ではなく計算時間の実測値を用いているところも一つのポイントである。純粋に計算量を用いて重み付けする場合は通信時間を無視していることになるが、通信時間のばらつきは計算時間に現れる。並列プロセス数を上げていくと通信が隠蔽できなくなり、通信量の相対的な増加により通信時間が最終的には支配的になる。このような場合でも、前のステップの計算時間にはこの通信時間も含まれるため、計算時間による重み付けで十分であることが分かる。

3 LET の非同期通信

領域分割が終わると、木構造の構築を各プロセスごとに割り振られた粒子から行う。ただし、木構造は全プロセスで共通のものを用いており、各プロセスはその部分木を構築する。各プロセスの部分木はリモートプロセスの部分木の情報の一部を必要とし、各プロセスが必要なリモートプロセスの部分木の共通集合を local essential tree(LET) と呼ぶ。この LET を各プロセスが構築するためにはリモートプロセスの部分木の一部を通信する必要がある。全プロセスが全プロセスからそれぞれ異なる情報を必要とするため `MPI_Alltoallv` のような通信が必要になる。ここで課題となるのが、どのプロセスに何を送れば良いかの特定方法とそれを効率的に送受信するための方法である。リモートプロセスの木構造の様子は見えないため、送って欲しい情報が存在しない場合もありうる。逆に、何を送れば良いかの判定はリモートプロセスの木構造の詳細がわからずとも、領域分割の際に用いた領域の境目の情報だけから概ね特定できる。粒子

の分布が非一様な場合も、ローカルの部分木の様子は完全に分かっているので、これから何を送れるかの判定は容易にできる。このような送信側主体の LET 通信は FMM ではよく用いられている [24]。受信側主体の手法では一度、送信側に送るべき情報を伝えてから、送信側がそれを送るという手順にならざるをえないため二度手間になる。送るべき多重極展開の係数が一つであるような重力多体問題では送るべき情報のインデックスリストが実際に送る多重極展開の係数の情報と同じになり通信量は倍になる。

LET の通信を行っている間に、各プロセスに既に存在する部分木の情報だけでできる計算を済ませてしまうこともできる。これは MPI3 の非ブロッキング型 `MPI_Alltoallv` を用いるやり方と OpenMP の section を用いて通信専用のスレッドを作るやり方や MPI の 1 対 1 通信 (`MPI_Isend`,`MPI_Irecv` や `MPI_Put`,`MPI_Get`) を用いてより細粒度で行う方法などが考えられる。プロセス数が少ない場合は同期的に通信を行ったとしても、このような通信と計算のオーバーラップにより通信を完全に隠蔽できる。非同期通信が重要になるのはローカルな情報を用いてできる計算が LET の通信を隠蔽するのに不十分なプロセス数に達した場合に有効になる。この全対全通信を行う際にプロセス数 P が 2 の冪乗である場合、$\log_2 P$ 次元のハイパーキューブの各辺同士で通信を行うことで、$P \times P$ の通信を $\log_2 P$ 回の 1 対 1 通信に低減することができる。一回に通信するデータ量は増えるものの、レイテンシや輻輳制御の観点から非常に有効な手法であるといえる。

粒子数 N、プロセス数 P とした時の LET の通信量の理論的な上限に関する研究も行われており、前述の方法を用いることで改善できることが証明されている。表 1 に様々な LET 通信の手法ごとの通信量の粒子数 N とプロセス数 P に対する依存性（通信複雑性）を示す。FMM の通信量に関する理論的な上限値を計算したのは Teng[25] が初めてで、これは木構造同士のデータの依存関係をグラフ化してグラフ理論を用いて算出した通信量である。これは P 個のプロセスがそれぞれ $(N/P)^{2/3}(\log N)^{1/3}$ の通信を行うということからきている。$(N/P)^{2/3}$ は典型的な halo 通信に見られる値であり、$(\log N)^{1/3}$ は木構造の非一様性による近接通信の増加に起因する値である。次の Lashuk ら [26] と Teng の式を比べると P が $\sqrt{P}$ になっており、$(\log N)^{1/3}$ の項が消えていることがわかる。P と $\sqrt{P}$ の違いは Teng[25] の 650 ページの下にある $\sum_{i}^{\log P-1} 2^i = \mathcal{O}(P)$ が Lashuk ら [26] の 7 ページ右側の $\sum_{i}^{\log P-1} min(2^{\log P-i-1}, 2^i) = \mathcal{O}(\sqrt{P})$ に変化したためである。これは前述のパイパーキューブ型の全対全通信によるものである。$(\log N)^{1/3}$ の項は粒子の数密度に極端な偏りがある場合にあるリーフセルの横に $(\log N)^{1/3}$ 個の小さいリーフセルが隣接する可能性を示している。Lashuk ら [26] は隣接するリーフセルの大きさの比を 2:1 に制限することで、近接相互作用をするセルの数を $\mathcal{O}(1)$ に抑えたためこの項を消すことができた。Lashuk

表 1: Communication complexity of FMM.

Reference	Communication complexity
Teng (1998) [25]	$\mathcal{O}\left(P(N/P)^{2/3}(\log N)^{1/3}\right)$
Lashuk *et al.* (2009) [26]	$\mathcal{O}\left(\sqrt{P}(N/P)^{2/3}\right)$
Yokota *et al.* (2014) [27]	$\mathcal{O}\left(\log P + (N/P)^{2/3}\right)$

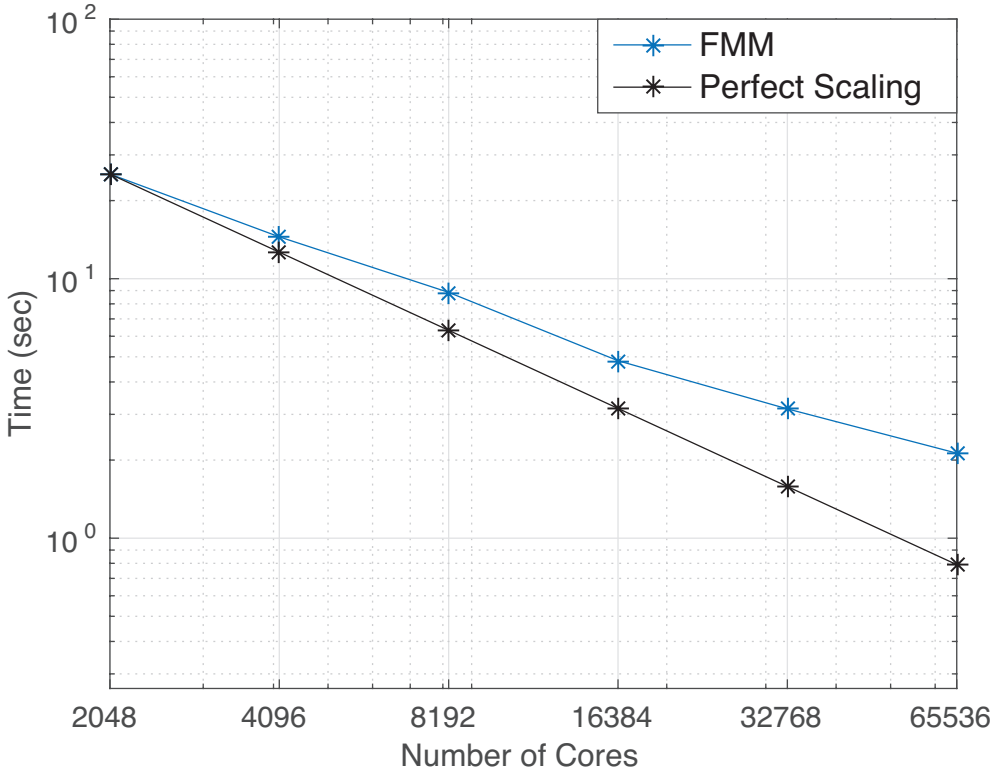


図 5: N=10^9 における強スケーリング。縦軸は FMM の 1 ステップにかかる計算時間を表す。

ら [26] と Yokota ら [27] の違いは、$\sqrt{P}$ が $\log P$ になっていることと、$\sqrt{P}$ との積から $\log P$ との和に変化していることである。$\sqrt{P}$ から $\log P$ への変化は木構造の上層部での情報の冗長性を利用することで実現できる。積が和になっているのは木構造をローカルな部分とグローバルな部分に分解し、それぞれで異なる通信の方法を用いることで実現できる。本研究では Yokota ら [27] の方法を採用している。

4 計算結果

計算は全て King Abdullah University of Science and Technology (KAUST) の Shaheen2 (Cray XC40) を用いて行った。Shaheen2 は Intel Xeon E5-2698v3 16 コア 2.3GHz が 12,288 ソケットあり、それらが Cray Aries の相互結合網で接続されている。High Performance LINKPACK (HPL) のスコアは 5,536.99 TFlop/s であり、2016 年 6 月時点で世界 10 位のスパコンである。粒子の分布は境界要素法の計算点を模擬した球面に均等に分布したものを用いた。FMM のカーネルは Laplace 方程式を用い、球面調和関数による展開を 9 次まで行った。木構造には八分木を用いており、領域分割には Hilbert ベースの HOT を、LET 通信には Yokota ら [27] の手法を用いた。通信と計算はオーバラップし、通信は非同期的に行った。1 ノード 1 プロセスを割り当て、ノード内の 16 コアに対してはスレッド並列化を行った。

図 5 に粒子数 $N = 10^9$ のときの強スケーリングを 2048 コアから 65,536 コアまでの間で示す。横軸はコア数を表しておりプロセス数はこれを 16 で割った値となっている。縦軸は FMM を 1 ステップ行うための計算時間を秒単位で表したものである。理想的な場合を「Perfect Scaling」で示しているが、FMM の並列化効率はコア数が増加するにしたがって徐々に悪くなる様子が見てとれる。ただし、32,768 コアから 65,536 コアにかけてまだわずかに計算時間は短縮されて

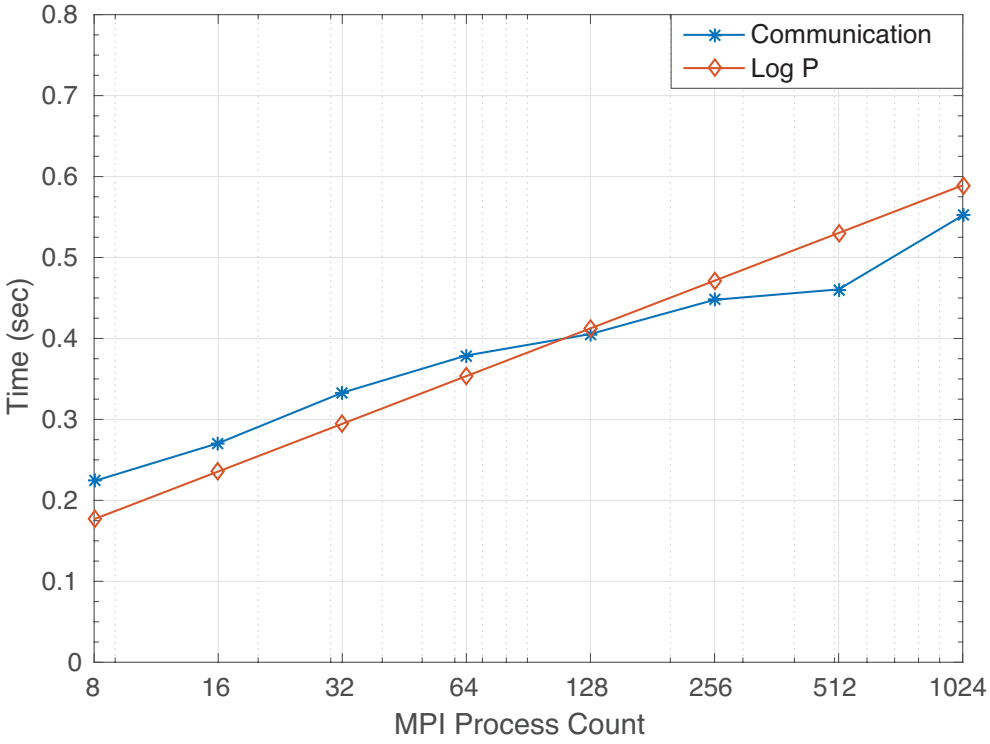


図 6: プロセスあたり $N = 10^8$ の弱スケーリング。縦軸は FMM の 1 ステップにかかる計算時間を表す。

おり、強スケーリングとしては比較的良好な結果であると言える。

図 6 にプロセスあたりの粒子数を $N = 10^8$ とした時の弱スケーリングを示す。横軸はプロセス数を表し、縦軸は通信時間を表している。参考のために理想的な $\log P$（P はプロセス数）を「Log P」に示す。Shaheen2 の Aries 結合網の Dragonfly トポロジーも手伝って、ここでは理想に近い通信時間を実現できている。8 プロセスから 1024 プロセスにかけてほぼ理想的に通信時間が弱スケーリングする FMM を示したのは著者らの知る限りこれが初めてである。

5 結言

本研究では大規模境界要素法解析においてボトルネックとなる FMM の通信について、既存研究の中でも特に有用な手法を取り上げ、それをさらに発展させた独自手法による通信最適化を行った。領域分割法については ORB と HOT の両方の可能性を検討しヒストグラムを用いた分散メモリ用の最適化手法を開発した。また、領域分割を粒子数が均等になるように分けるのではなく、前の時間ステップ（もしくは反復ステップ）の計算＋通信時間で重み付けされた値のヒストグラムを用いることで、反復しながら動的負荷分散を行う方法を実現した。LET の通信に関しては送信側主体の通信を行うことで非一様な分布においても遠方の木構造の部分木を正しく見つけて送信することができた。また、全対全通信を非同期的に行うことでローカルな情報の計算と通信のオーバーラップだけでなく、先に届いたリモートの情報の計算と通信もオーバーラップさせることができた。また、理論的な通信量の上限の漸近挙動を著しく改善することに成功し、実験結果ではこの理論と一致する性能を確認することができた。

謝辞

本研究は JSPS 科研費 16H05859,16H02827 の助成を受けたものです。

参考文献

[1] W. R. Wolf and S. K. Lele. Aeroacoustic integrals accelerated by fast multipole method. AIAA Journal, **49**-7, 1466/1477 (2011)

[2] S. Hao, P. G. Martinsson, and P. Young. An efficient and highly accurate solver for multi-body acoustic scattering problems involving rotationally symmetric scatterers. Computers and Mathematics with Applications, **69**, 304/318 (2015)

[3] R. Yokota, J. P. Bardhan, M. G. Knepley, L. A. Barba, and T. Hamada. Biomolecular elec- trostatics using a fast multipole BEM on up to 512 GPUs and a billion unknowns. Computer Physics Communications, **182**,1272/1283 (2011)

[4] E. Darve and P. Havé. A fast multipole method for Maxwell equations stable at all frequencies. Philosophical Transactions of the Royal Society of London A, **362**, 603/628 (2004)

[5] Z. Gimbutas and L. Greengard. Fast multi-particle scattering: A hybrid solver for the Maxwell equations in microstructured materials. Journal of Computational Physics, **232**, 22/32 (2013)

[6] D. Willis, J. Peraire, and J. White. FastAero – a precorrected FFT-fast multipole tree steady and unsteady potential flow solver. http://hdl.handle.net/1721.1/7378 (2005)

[7] 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 Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 1/12 (2010)

[8] A. Verde and A. Ghassemi. Fast multipole displacement discontinuity method (FM-DDM) for geomechanics reservoir simulations. International Journal for Numerical and Analytical Methods in Geomechanics, **39**-18, 1953/1974 (2015)

[9] S. Chaillat, M. Bonnet, and J.-F. Semblat. A multi-level fast multipole BEM for 3-D elastodynamics in the frequency domain. Computer Methods in Applied Mechanics and Engineering, **197**, 4233/4249 (2008)

[10] D. R. Wilkes and A. J. Duncan. A low frequency elastodynamic fast multipole boundary element method in three dimensions. Computational Mechanics, **56**, 829/848 (2015)

[11] J. Bédorf, E. Gaburov, and S. Portegies-Zwart. A sparse octree gravitational N-body code that runs entirely on the GPU processor, Journal of Computational Physics, **231**-7, 2825/2839 (2012)

[12] J. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. Nature, **324**, 446/449 (1986)

[13] J. Bédorf, E. Gaburov, M. S. Fujii, K. Nitadori, T. Ishiyama, and S. Portegies-Zwart. 24.77 Pflops on a gravitational tree-code to simulate the Milky Way Galaxy with 18600 GPUs, in Proceedings of the 2014 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, 1/12 (2014)

[14] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon. A massively space-time parallel N-body solver, in Proceedings of the 2012 ACM/IEEE International Conference on High Performance Computing, Networking, Storage and Analysis, 1/12 (2012)

[15] M. Winkel, R. Speck, H. Hubner, L. Arnold, R. Krause, and P. Gibbon. A massively parallel, multi-disciplinary Barnes-Hut tree code for extreme-scale N-body simulations, Computer Physics Communications, **183**-4, 880/889 (2012)

[16] 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, Communications of the ACM, **55**-5, 101/109 (2012)

[17] M. Griebel and M. A. Schweitzer. A particle-partition of unity method part iv: Parallelization, in Meshfree Methods for Partial Differential Equations, 161/192, Springer (2003)

[18] H. Sundar, R. S. Sampath, and G. Biros. Bottom-up construction and 2:1 balance refinement of linear octrees in parallel, SIAM Journal on Scientific Computing, **30**-5, 2675/2708 (2008)

[19] M. Zandifar, M. AbdulJabbar, A. Majidi, D. Keyes, N. M. Amato, and L. Rauchwerger. Composing algorithmic skeletons to express high-performance scientific applications, in Proceedings of the 29th ACM on International Conference on Supercomputing, 415/424 (2015)

[20] M. S. Warren and J. K. Salmon. A parallel hashed oct-tree N-body algorithm, in Proceedings of the 1993 ACM/IEEE conference on Supercomputing, 12/21 (1993)

[21] J. K. Salmon. Parallel hierarchical N-body methods, Ph.D. dissertation, California Institute of Technology (1991)

[22] J. Makino. A fast parallel treecode with GRAPE, Publications of the Astronomical Society of Japan, **56**, 521/531 (2004)

[23] E. Solomonik and L. V. Kalé. Highly scalable parallel sorting, inProceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, 1/12 (2010)

[24] J. Dubinski. A parallel tree code, New Astronomy, **1**, 133/147 (1996)

[25] S.-H. Teng. Provably good partitioning and load balancing algorithms for parallel adaptive N-body simulation, SIAM Journal on Scientific Computing, **19**-2, 635/656 (1998)

[26] 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, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis (2009)

[27] R. Yokota, G. Turkiyyah, D. E. Keyes. Communication Complexity of the Fast Multipole Method and its Algebraic Variants, Supercomputing Frontiers and Innovations, **1**-1, 63/84 (2014)