ハイパフォーマンス
コンピューティング 74－7
（1998．12．11）

分散メモリ並列計算機上におけるアダプティブ有限要素法のための動的負荷分散アルゴリズム

薗田 勉, 土肥 俊
NEC C&C メディア研究所

要旨

分散メモリ並列計算機上でアダプティブ有限要素法を並列に実行しようとすると、計算の進展に伴って各プロセッサの計算負荷が不均等になり、並列計算の実行効率が著しく低下する。本論文では、この負荷の不均等問題を解決するための動的負荷分散アルゴリズム(PADLOBA 法)を提案する。PADLOBA 法は格子の接続情報からプロセッサの接続グラフを作成し、そのグラフを使い負荷を均等化する計算を行うものである。数値実験によって、PADLOBA 法が動的負荷分散を行わない場合に比べてトータルの計算時間を約 55%短縮させることを検証した。

Dynamic Load Balancing Algorithm for the Adaptive Finite Element Method on Parallel Computers

Tsutomu Osoda, Shun Doi
C&C Media Research Laboratories, NEC Corporation

Abstract

When we execute the adaptive finite element method on parallel computers using domain decomposition technique, calculation load may become different among processors. This load imbalance degrades the efficiency of parallel computing. This paper proposes an algorithm of dynamic load balancing (the PADLOBA method) which solves the problem of the load imbalance. The algorithm of the PADLOBA method is that the processor graph is produced based on the calculation grid and the calculation for equalization of load is made based on the processor graph. Numerical experiment shows that the PADLOBA method shortens the total calculation time approximately by 55%.

1. はじめに

領域分割法は有限要素法の並列処理において広く用いられている。これは有限要素モデルデータである要素（または節点）の集合をプロセッサと同数のブロックにあらかじめ分割し、各ブロックをプロセッサに1対1に割り当てて並列計算を行う方法である。各プロセッサに割り当てられる要素（または節点）数がほぼ同数のとき、各プロセッサの計算負荷が均等になり、効率的な並列計算が行われる（以降では要素をベースにした割り当てを行う）。

高精度の解を効率よく求めるための手法としてアダプティブ法がある。有限要素法の場合、解の精度が必要な部分に要素・節点を追加していく（リメッシュと呼ぶ）h法がよく用いられる。

領域分割法で並列化を行った後h法を実行すると、初期において均等化されていた計算負荷は、リメッシュに伴って各プロセッサ間で不均等になり、並列計算の実行効率が著しく低下する。並列処理の効果を最大限に引き出

すためには各プロセッサの計算負荷を均等化する必要がある。すなわち、図1のフロー図において、(4)の処理を行うことにより、常に負荷の均等化を行いながらアダプティブ有限要素法の並列計算を行う。

本論文では、この負荷の不均等を解決する問題を「各プロセッサに割り当てられている要素数が(リメッシュの結果)不均一であるときに，各プロセッサの要素数が均一になるような要素の再割り当てを求める問題」と捕らえ、それを解決するための動的負荷分散アルゴリズム(PADLOBA(PArallel Dynamic LOad BAlancing)法)を提案する。

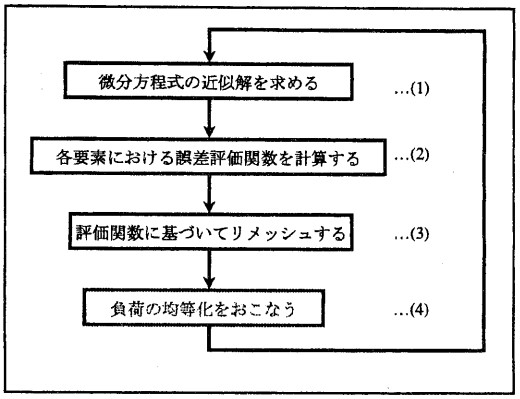

図 1: 分散メモリ並列計算機上におけるアダプティブ有限要素法の実行方法

一般的に、負荷の不均一問題を解決するための動的負荷分散アルゴリズムは以下の要件を満たす必要がある。

1. 格子の再配置後の分割が良質であること。
 *各プロセッサの計算コストが均等になるようにする。
 *分割後の領域間の通信量が少なくなるようにする。
2. 再配置計算のコストが少ないこと。
 *負荷均等化の計算時間が、負荷の均等化によって短縮される計算時間より短い。
 *アルゴリズムが並列度の増大に対してスケーラブルである。

従来から様々な方法[2,3]が提案されているが、いずれの方法も前の条件を同時に満たすことは難しい。

2章では筆者の提案した動的負荷分散方法(PADLOBA法)について説明をし、3、4章でPADLOBA法を実際の問題に適用し、Cenju-3上で評価を行った結果を示す。

3. PADLOBA法

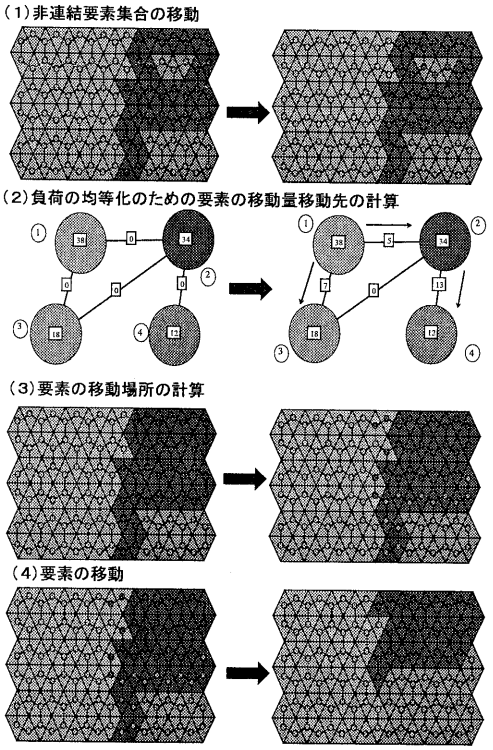

図 2: PADLOBA法による負荷再配置

PADLOBA法[4]のアルゴリズムは以下の4ステップからなる（図2)。

(1)非連結要素集合の移動

各プロセッサが連結しない領域を持っているとき、その領域が連結になるように移動する。

(2)負荷移動量・移動先の計算

格子グラフを基に、プロセッサグラフ(プロセッサの接続状況を示したグラフ)を作成し、均

等化するための移動量を計算する。

(3) 負荷の移動場所の計算

(2)の結果を基に移動すべき要素を決定する。

(4) 負荷の移動

3. 評価プログラム

PADLOBA法の評価を行うために、評価用の並列アダプティブ有限要素法のプログラムを開発した。この章では図1の流れに従い、評価用プログラムの説明をする。

3.1 離散化と線形問題の解法(図1(1))

ポアソン方程式

$$\frac{\partial}{\partial x}\left(K_x\left(\frac{\partial u}{\partial x}\right)\right)+\frac{\partial}{\partial y}\left(K_y\left(\frac{\partial u}{\partial y}\right)\right)=f\,(in\,\Omega)$$

$$u=0\quad (on\,\partial\Omega)$$

$$K_x=1,\ K_y=1$$

を3角形1次要素を使い有限要素法で離散化する。線形問題の解法にはスケーリング前処理つき共役勾配法(SCG法)を用いる。

3.2 リメッシュの方法(図1(2), (3))

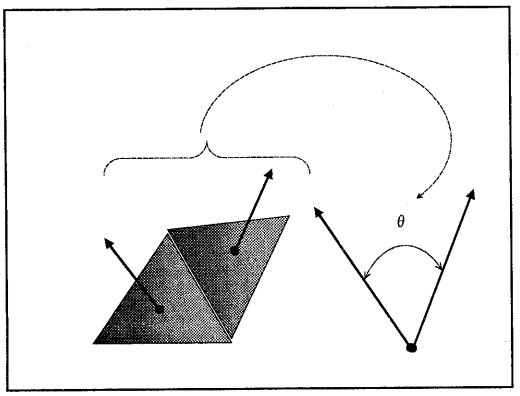

図 3: リメッシュの評価関数。

直前の計算結果からリメッシュのための評価関数を要素の各辺上で計算し(図1(2))、これに基づきリメッシュを行う(図1(3))。評価関数としては、図3に示すように、各要素が(x, y, u)空間で作る平面の単位法線ベクトルを用いて、隣接要素間におけるベクトルの内積を使用した。この値が小さな値のとき、その部分における解の空間変化に対して要素サイズが大きいと判断できる。具体的には内積の値が0.7よりも小さな値のとき、メッシュを細かくするようにした。メッシュの張り直し(図1(3))にはBAMG[1](フランス国立研究所INRIAにて開発)を使用した(ただしこの部分は並列化されていないため4.4章の計算時間には含まれていない)。

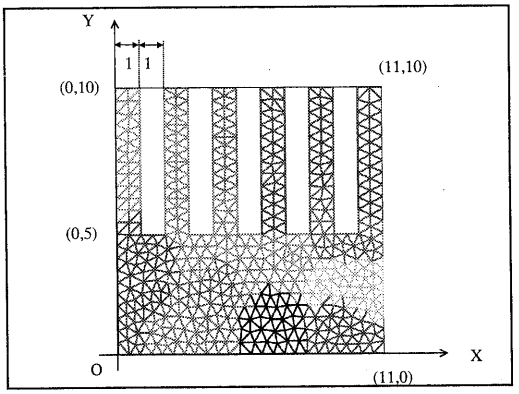

図 4: 評価例題を定義した領域と初期メッシュ。初期メッシュは各プロセッサに均等に分割されている。(初期節点数、要素数はそれぞれ513, 814。)

3.3 評価例題

領域Ωとして図4に示すような形状を用いる。初期節点数を513、要素数は814とする。またポアソン方程式のソース項fは以下のように定義する。

$$f=\frac{10000}{100^{\{(x-2)^2+(y-3)^2\}}}$$

SCG法は、相対残差の2ノルムが1e-8以下となったとき収束したものとみなす。

4. 数値実験

この章では前述の評価例題をCenju-3上(プロセッサ台数は16台)で評価した結果を示す。

4.1 均等度の定義

均等度を以下の式により定義する。この指標が1のとき、負荷は均等になっており、0に

近づくに従い負荷は不均等になっている。

$$\text{均等度}:=\frac{\sum_{i=1}^{P} Ni}{P \max_{i=1} Ni \times P}$$

Ni = 各プロセッサが持っている数(要素数、節点 数)

P = プロセッサ台数

4.2 評価関数とリメッシュの妥当性

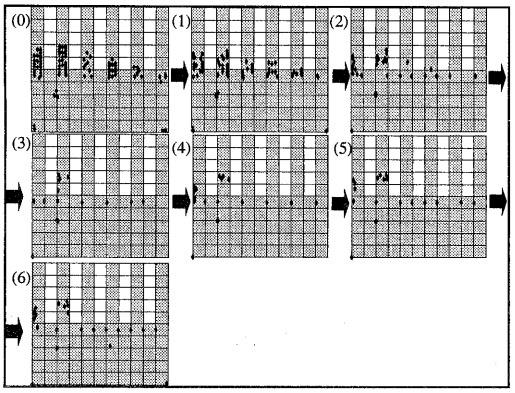

図 5: 評価関数値の大きな領域を点によって示した図。図中の番号はメッシュをはり直した回数。リメッシュするにつれ、評価関数値の大きな地点が少なくなってきている。

図5は3.2章の評価関数値の大きな領域を示したものであり、図の(0)から(6)はリメッシュの回数を示している。図中の黒丸は評価関数の大きな領域を表している。

図5から以下のことがわかる。初期メッシュでは評価関数値の大きな領域が疎らに分散していたが、リメッシュの回数を増やすにつれ、評価関数値の大きな領域が狭くなってくる。したがって領域全体の評価方法は収束の方向にあり、今回評価に用いた評価関数、リメッシュ方法は現実的にも妥当性を持っているということができる。

実問題においては、領域全体で評価関数値が十分に小さくなったところで計算を打ち切る。(今回の実験では6回のリメッシュの後、動的負荷分散を行わない場合、特定のプロセッサの要素数が急増し、メモリ容量の制約から、実行不可能になった。次節の評価はリメッシュの回数をこの6回までとした。)

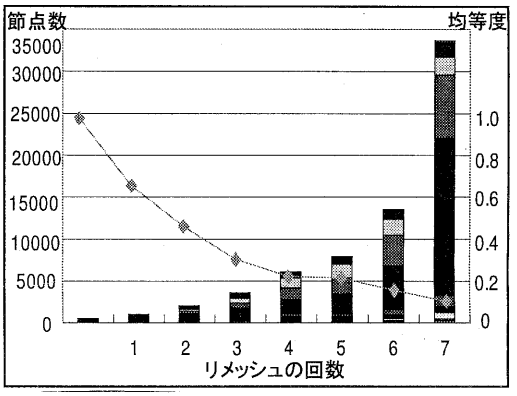

図 6: メッシュをはり直したときの節点数の変化とプロセッサ間の節点数の均等度の変化。縦軸は節点数および節点数の均等度。横軸はリメッシュの回数。棒グラフが各プロセッサに割り当てられた節点数。傍線が節点数の均等度

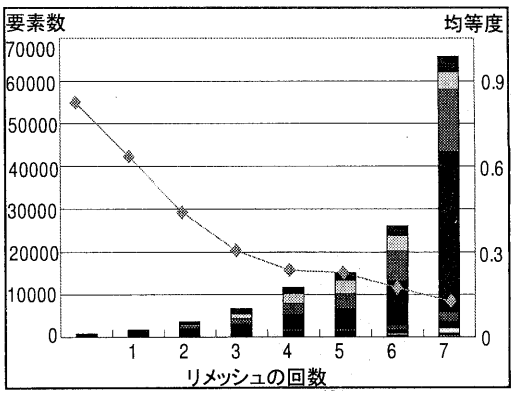

図 7: メッシュをはり直したときの要素数の変化とプロセッサ間の要素数の均等度の変化。縦軸は要素数および要素数の均等度。横軸はリメッシュの回数。棒グラフが各プロセッサに割り当てられた要素数。傍線が要素数の均等度。

4.3 動的負荷分散を行わないときの均等度の変化

動的負荷分散を行わずに評価例題を計算し

たときの各プロセッサの節点数・節点数の均等度、要素数・要素数の均等度の推移を、それぞれ図6、7に示す。図6、7の横軸はリメッシュの回数である。図6、7の縦軸はそれぞれ節点数・節点数の均等度、要素数・要素数の均等度を示している。

図6、7からアダプティブ有限要素法の計算の進行とともに節点数と要素数はともに増大し、プロセッサ間における節点数、要素数が急速に不均一になっていることがわかる。たとえば節点数を例に取ると、計算の初期段階において、均等度は約97%(最大節点数=33, 最小節点数=31)であったが計算の進行とともに低下し、最終的には均等度が約11%(最大節点数=18689, 最小節点数=30)になっている。この不均一な状態を視覚的に表現した図が図8である。図8は最終的にはられたメッシュを示している。評価関数値が大きな領域にメッシュが集中し、その領域を受け持つプロセッサに高い負荷が局所的に生じていることがわかる。

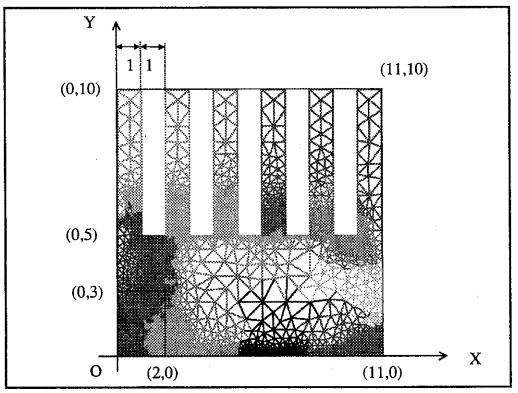

図 8: リメッシュを繰り返していったときのメッシュ。この図はリメッシュの回数が6回の後のメッシュを示している。リメッシュの回数が増大するとあるプロセッサで局所的にメッシュが集中する。

4.4 動的負荷分散による計算時間の短縮

並列計算機上において例題を評価した結果を図9に示す。縦軸に計算時間、横軸にリメッシュの回数を示している。棒グラフが動的負荷分散を行ったときの計算時間で、折れ線が動的負荷分散を行わなかったときの計算時間を示している。

図9から以下のように考察できる。

➢ 動的負荷分散を行わない場合よりも、動的負荷分散を行ったほうが全体の計算時間が短くなった。---図6、7に示したように、動的負荷分散を行わない場合、アダプティブ有限要素法の計算の進行とともプロセッサ間での負荷が不均一になり、並列計算機の性能を十分に発揮していないためである。

➢ 計算時間の差は、節点数・要素数が増大するほど、大きくなった。---今回使用した評

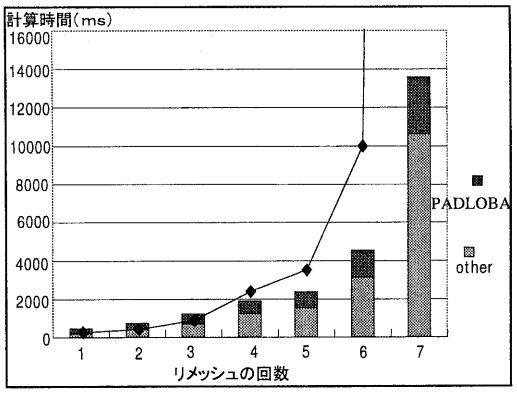

図 9: 計算時間。縦軸は計算時間。横軸はリメッシュの回数。折れ線は動的負荷分散を行わないときの計算時間で、傍線は動的負荷分散を行ったときの計算時間。

価例題では、評価関数値の大きな領域が移動する傾向はなく、節点数・要素数を増大させればさせるほど、要素・節点があるプロセッサに集中する傾向にあった(図8)。そのためリメッシュするたびに負荷のバランスが悪くなったことが原因と考えられる(図5、6)。

➢ 動的負荷分散を行わない場合、特定のプ

ロセッサの要素数が急増し、メモリ容量の制約から実行不可能になった。動的負荷分散を行った場合には各プロセッサの要素数の増加は均等化されていて、同じ要素数でも計算が可能であった。——原因は上記項目とおなじである。

➢ 6回のリメッシュの後、領域全体の節点数が13526、要素数が26084となった。そのときに動的負荷分散を行った場合と行わない場合を比較し約2.2倍の速度向上を得た。——4.1章にて説明したように実問題では領域全体の評価関数が十分に小さくなるまでメッシュを張り直す。一方、今回の評価ではメモリ容量の制約からリメッシュの回数を6回で打ち切った。したがって約2.2倍という数値は、実問題という点ではあまり意味のあるものではなく、実問題に適用した場合には、この章で考察に示したように、さらに大きな速度向上が期待できる。

5 スケーラビリティに関する考察

アルゴリズムが大まかに4つの操作からなることは2章で説明した。そのうち(2)でプロセッサ間の接続を示すグラフを作成するが、そのグラフの節点数（つまりプロセッサ台数）をN、枝数をMとすれば、(2)の計算量はは最悪O(MN)となることが理論上証明できる。(1),(3),(4)の操作は要素数に比例した操作になるが、要素数はM,Nに比べ非常に大きいため(1),(3),(4)の時間、とりわけ(3)の時間が支配的になる。一方、(3)の操作は各プロセッサ同時に並行実行可能なためスケーラビリティが高い。

6. まとめ

この論文では、アダプティブ有限要素法を分散メモリ並列計算機上で実行するために必須の動的負荷分散問題に対して有効なアルゴリズム(PADLOBA法)を提案した。また、数値実験によって、PADLOBA法は、評価例題に対してトータルの計算時間を約55%に短縮し、すぐれた動的負荷分散の手段を提供していることを確認した。今回は評価例題としてポアソン方程式を選び、また、負荷の不均一な状態をシミュレーションするために、人為的なソース項を用いた。実際の流体計算などでは、メッシュの疎密差により大きな問題が実際に現れる（たとえば衝撃波捕獲の問題など）ため、本評価例題以上の計算時間短縮効果を期待できる。考察において、スケーラビリティは高いと予測したが、今後はプロセッサ台数が16台以上の場合についても数値実験をおこない、アルゴリズムが高いスケーラビリティを持つことを検証していきたい。

参考文献

[1]http://www.rocq.inria.fr/gamma/.

[2]Johan De Keyser, Kurt Lust, and Dirk Lose. Run-time load balancing support for a parallel multiblock euler/navier-stokes code with refinement on distributed memory computers. Parallel Computing, 20:1069-1088, 1994.

[3]George Karypis and Vpin Kumar. Multilevel k-way partitioning scheme for irregular graphs. Technical report, University of Minnesota, Department of Computer Science, August 1995.

[4] 襲田,土肥. 分散メモリ並列計算機上におけるアダプティブ有限要素法のための動的負荷分散. 計算工学講演会論文集, Vol.2 ,No.1,pp81-84