ハイパフォーマンス
コンピューティング 59－6
（1995．12．11）

有限要素法の並列アルゴリズムの開発と並列計算機 Cenju-3 上での性能評価

杉原 光太、藤尾 秀洋、村松 一弘、土肥 俊

NEC C&C 研究所 コンピュータ・システム研究部

代数的領域分割法に基づく有限要素法の並列アルゴリズムを開発し、NEC 製並列計算機 Cenju-3 上に実装、評価する。線形方程式の求解には並列 Scaled Conjugate Gradient(SCG) 法を採用する。平面応力問題を用いて評価した結果、プロセッサ数 64 台で 50 倍の速度向上 (節点数 10000 程度の問題) を実現した。1 プロセッサでの直接法 (Skyline 法) による計算と比較すると 71 倍の速度向上にあたる。筆者らが開発している有限要素解析のための数値シミュレーション言語 FEEL に今回開発した並列アルゴリズムを組み込み、有限要素法の並列プログラム自動生成を目指している。

DEVELOPMENT OF A PARALLEL COMPUTATION ALGORITHM FOR THE FINITE ELEMENT METHOD AND ITS PERFORMANCE EVALUATION ON CENJU-3 PARALLEL COMPUTER

Kouta Sugihara, Hidehiro Fujio, Kazuhiro Muramatsu, and Shun Doi

Computer System Research Laboratory, C&C Research Laboratories, NEC Corporation

4-1-1, Miyazaki, Miyamae-ku, Kawasaki-shi, Kanagawa, 216 Japan

We have developed a parallel algorithm for finite element analysis, and evaluated it on the NEC Cenju-3 parallel computer. This algorithm is based on an algebraic domain decomposition and uses the Scaled Conjugate Gradient method for the solution of linear systems. A plane stress problem is used for evaluation. It is shown that, for 64 processing elements, speedup factor is 50.3 when the number of nodes is about 10000. This parallel algorithm is now being incorporated in the FEEL (Finite Element Equation Language) FORTRAN program generator for finite element analysis, so that a user can easily obtain parallel FEM code.

1 はじめに

マイクロプロセッサの高速化・低価格化に伴って、それらを多数接続した並列計算機が高速計算の有効な手段として注目されている。実際、大規模なCPUリソースを必要とする科学技術計算の世界では、このような並列計算機を利用するための並列アルゴリズムの研究が近年極めて活発である。しかし一方で、並列計算機の性能を十分に引き出すプログラムの開発は高度のスキルと多くのプログラミング労力を要し、そのことが並列計算機の普及のさまたげにもなっている。

筆者らは、従来より数値シミュレーション言語FEELの研究開発を行っている[3]。これは、解くべき現象を支配する方程式(偏微分方程式)やそれに付随する条件などの入力から、有限要素法による解析のためのFORTRANプログラムを自動生成するものであり、それによって数値シミュレーションプログラムの開発に要する労力を軽減しようとするものである。現在、FEELは、2次元プロトタイプ版が完成しており、ワークステーション(WS)用のFORTRANプログラム自動生成が実現されている。

本研究は、FEELによる問題記述から、その問題に対する並列プログラムを自動生成することを目指すものである。本報告は、この目的のために開発した有限要素法の並列アルゴリズムの概要を述べるとともに、同アルゴリズムの並列計算機Cenju-3上での性能評価を行うものである。

採用した並列アルゴリズムは、有限要素法により離散化された方程式(主に連立一次方程式)の処理を全ての未知数について反復法を適用し、並列化する手法である。即ち、有限要素メッシュをPE(Processing Element)数と同数の部分領域に分割し、それぞれ対応するPEに割り当てる。離散化方程式(連立一次方程式)の係数行列作成は、各PEが、割り当てられた有限要素メッシュについて、他と独立に行う。離散化方程式の求解は、各PEがそれぞれに割り当てられた未知数に関して、他のPEと協調して並列に解く。具体的には、スケーリング前処理を施した共役勾配法系の反復解法を用いる。なお、「有限要素メッシュをPE数と同数の部分領域に分割」する分割の方法については、Simonらによって有効性が報告されているRecursive Spectral Bisection法を採用した[1][2]。

本論文の第2章では有限要素のアルゴリズムについて述べる。第3章は試作した有限要素法の並列アルゴリズムを詳述する。第4章では平面応力問題を取り上げ、試作した並列アルゴリズムをCenju-3上で評価し、その結果を報告する。

2 有限要素法

有限要素法は構造、流体、電磁場解析など広範な分野に適用されている数値計算手法である。複雑な領域の取扱いが容易であり、また境界条件の処理が容易であるといった点が優れている。本章では、有限要素法のアルゴリズムについて述べる。有限要素法のアルゴリズムを図1に示す。即ち、

Step 1 計算領域を三角形などのm個の有限要素$e_i(i=1,...,m)$に分割する。(図1(a))

Step 2 解析対象の偏微分方程式を、各有限要素e_i毎に離散化近似して、要素行列A_i、要素荷重項b_iを作成する。(図1(b))

Step 3 式

$$(1) \qquad A=\sum_{i=1}^{m} A_i$$

$$(2) \qquad b=\sum_{i=1}^{m} b_i$$

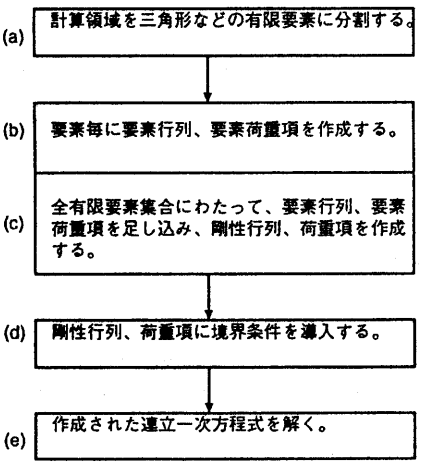


図1: 有限要素法のアルゴリズム

に基づいて全体剛性行列 A、荷重項 b を作成する。(図 1(c))

Step 4 全体剛性行列 A、荷重項 b に境界条件を導入する。(図 1(d))

Step 5 連立一次方程式 $Au = b$ を解き、解 u を求める。

3 有限要素法の並列アルゴリズム

この章では試作した有限要素法の並列アルゴリズムについて説明する。まず並列アルゴリズムを図 2 に示す。並列アルゴリズムにおける処理は、並列処理を行うためのデータの準備である「前処理」と、実際の「並列処理」の二つのステップに分けられる。前処理では以下の処理を行う。

Step 1 n 台の PE を使用する場合、全体要素の集合 $E = \{e_i | i = 1, ..., m\}$ を、互いに重複のない n 個の部分要素集合 E_k $(k = 0, .., n-1)$ に分割する。次に各 $E_k (k = 0, .., n-1)$ を $\mathrm{PE}_k (k = 0, .., n-1)$ に割り当てる。(図 2 (a)、(b))

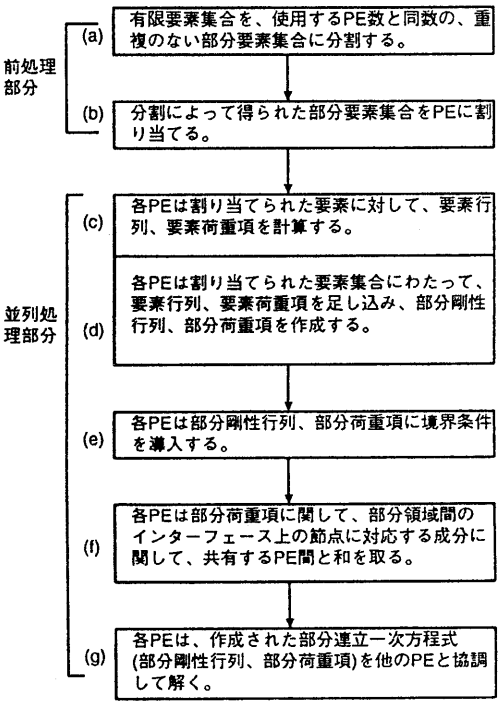


図 2: 有限要素法並列処理系のフロー

次に各 PE に部分要素集合を割り当てた後の並列処理は、次の五つのステップから成る。

Step 2 $\mathrm{PE}_k (k = 0, .., n-1)$ は、各々割り当てられた部分要素集合 $E_k (k = 0, ..., n-1)$ に属す要素 e に関して、要素行列 A^e、要素荷重項 b^e を計算する。(図 2 (c))

Step 3 $\mathrm{PE}_k (k = 0, .., n-1)$ は、式

$$A_k = \sum_{e \in E_k} A^e \tag{3}$$

$$b_k = \sum_{e \in E_k} b^e \tag{4}$$

に基づいて、部分剛性行列 A_k、部分荷重項 b_k を作成する。(図 2(d))

Step 4 $\mathrm{PE}_k (k = 0, .., n-1)$ は、作成した部分剛性行列 A_k、部分荷重項 b_k に境界条件を課す。(図 2 (e))

Step 5 $\mathrm{PE}_k (k = 0, .., n-1)$ は、部分荷重項 b_k に関して、部分領域間のインターフェース上の節点に対応する成分に関して、共有する PE 間と和を取る。(図 2 (f))

Step 6 各 PE は、作成された部分連立一次方程式 (部分剛性行列、部分荷重項) を他の PE と協調して解く。(図 2 (g))

なお、Step1 における全体要素の集合を PE 数と同数の部分要素集合に分割手法として、Simon らによって提案されている Recursive Spectral Bisection 法を採用した [1][2]。

また Step6 における連立一次方程式の並列解法には、前処理として対角項によるスケーリングを使う SCG (Scaled Conjugate Gradient) 法を採用した。

4 数値実験と考察

4.1 Cenju-3 [4]

Cenju-3 は VR4400SC を要素プロセッサとし、これらを多段接続網で結合した並列計算機である。各プロセッサは、64M バイトのローカルメモリを装備している。(図 3 参照) ネットワークは 4×4 のクロスバースイッチを用いた多段パケット交換網であり、通信距離の点で有利な構成になっている。本研究では、並列プログラミングは、FORTRAN と並列通信ライブラリ MPI (Message Passing Interface) を用いた。

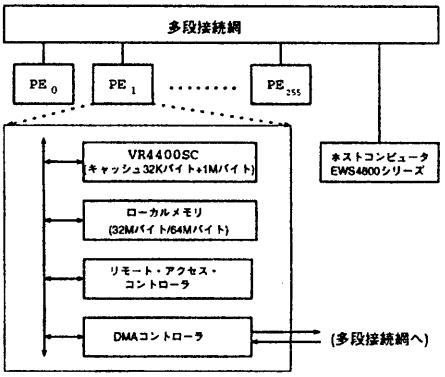


図 3:Cenju-3 システム構成図

4.2 例題

試作した並列処理アルゴリズムを図 4 で示す平面応力問題に適用した。数値実験は、表 1 に示す要素、ならびに節点数、要素数で行った。また連立一次方程式の並列解法の収束判定条件は、打ち切り誤差が 10^{-12} 以下とした。

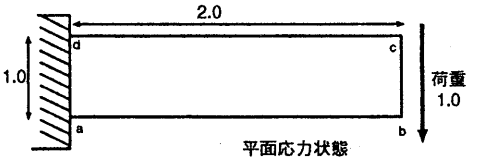


図 4 : 評価対象問題

表 1: 数値実験に用いた要素、節点数、要素数

	用いた要素	要素数	節点数
例 1	四辺形二次要素	1785	5494
例 2	四辺形二次要素	3656	11161

4.3 数値実験結果

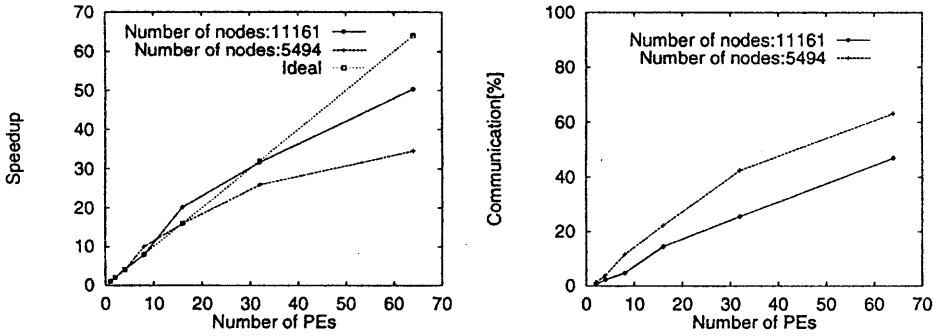
並列処理部分の実行結果

並列処理部分における並列化の効果を示す。図 5 の (a) に、例 1(節点数 5494) と例 2(節点数 11161) の場合の並列処理部分の速度向上率 S_n を示し、図 5 の (b) に例 1(節点数 5494) と例 2(節点数 11161) の場合の並列処理部分における通信時間の割合を示す。ここで速度向上率 S_n は

$$S_n = \frac{\text{PE 数 } n \text{ 台での実行速度}}{\text{PE 数 1 台での実行速度}}$$

で定義する。

図 5: 並列処理部分の実行状況



(a) 並列処理部分の速度向上率　　(b) 並列処理部分における通信時間の割合

Skyline 法と SCG 法の比較結果

次に Skyline 法と SCG 法の比較を行う。一般に有限要素法の汎用プログラムでは連立一次方程式の解法として多くの場合、直接法が用いられる。これは、板やシェルの解析の場合、反復法では計算が収束しないなど安定性に欠けることによる。

今回扱った平面応力問題 (節点数が 5494 の場合と節点が 11161 の場合) に関して、直接法と反復法の性能比較を行うため、NEC 製ワークステーション EWS4800/360AD での直接法 (Skyline 法) の実行時間と Cenju-3 での反復法 (SCG 法) の実行時間の比較結果を表 2 に示す (EWS4800/360AD と Cenju-3 の 1PE の性能はほぼ同じ)。

表 2: 並列 SCG 法 (Cenju-3) と Skyline 法 (EWS4800/360AD) の実行時間の比較

(a) 例 1(節点数 5494、要素数 1785)

PE 数	1	2	4	8	16	32	64
並列 SCG 法 [sec]	151.73	72.04	39.76	15.48	9.91	6.18	4.75
直接法 [sec]	150.57	—	—	—	—	—	—

(b) 例 2(節点数 11161、要素数 3656)

PE 数	1	2	4	8	16	32	64
並列 SCG 法 [sec]	440.32	219.81	108.27	56.25	22.19	14.39	9.03
直接法 [sec]	624.64	—	—	—	—	—	—

4.4 考察

並列処理部分の実行状況の解析

図 5 から以下のことがいえる.

- 節点数約 1 万の例 2 では、プロセッサ数 64 のとき速度向上率が 50 となり、トータルとしてはほぼ台数に見合う向上が得られているといえる (図 5(a))。
- プロセッサ台数が同一のところで例 1・2 の速度向上率を比較すると、(8 プロセッサ数の場合を除いて) 例 2 の方が常に例 1 を上回っている。これは，問題の規模 (節点数) が大きくなる程、全処理時間に占める通信時間

の割合が少なくなるためである。従って、より大規模な問題(例えば10 ~ 100万節点)の問題では、更に多数のプロセッサによる並列処理でも十分の並列化効果が得られると期待される。

- 図5(a)を見ると、例1(例2)ではプロセッサ数16(8)において速度向上率が理想値を上回る"スーパースケーラブルな"性能向上が得られている。これはよく知られているように、キャッシュヒット率向上によるプロセッサの(演算)性能向上によるものである。実際の通信時間は、図5(b)を見ると、64プロセッサでは40%(例1)から60%(例2)に達している。上記の50倍という速度向上や、スーパースケーラブルな性能は、実際はこのようなキャッシュ効果による相殺の結果である。より通信時間を短縮することによって更に性能を向上する余地があるといえる。

Skyline法とSCG法の比較

表2について考察する。

- プロセッサ1台のときのSCG法とSkyline法の性能差を見ると、例1ではほぼ同一であるのに対し、例2ではSCG法がSkyline法よりも約1.4倍高速である。仮に行列の帯半幅が一定(m)とすると、Skyline法の演算量は$O(nm^2)$(nは行列サイズ)であることが知られている。一方SCG法の場合演算量のオーダは$O(nk)$(kは収束までの反復数)である。上の結果は、n(あるいはm^2)の増加に較べkの増加が十分少ないことを意味している。問題が大規模になる程反復法が有利になると期待される。
  マルチグリッド法は計算のオーダが$O(n)$(反復数が問題の規模に依存しない)となることが知られている。従って、この方法を用いることによって、更に計算時間が短縮されることが期待される。

- 1プロセッサでのSkyline法の計算と、64台でのSCG法の計算とを比較すると、結局71倍の速度向上を得たことになる(例2)。

5 まとめ

本研究では開発した有限要素法の並列アルゴリズムの概要とその性能評価の結果を述べた。

開発した並列アルゴリズムを平面応力問題に適用し、Cenju-3上に実装、性能評価を行った。評価の結果、問題の規模が大きくなる程、演算量に比べ通信のオーバーヘッドが相対的に減少するため、より高い速度向上率が実際確認された。実際、PE数64台を使用したとき、節点数が5494で速度向上率が34.5倍であったのに対し、節点数が11161のとき、速度向上率は50.3倍であった。SCG法とSkyline法との比較でも、SCG法を用いた方が高速であり、その傾向は問題の規模が大きくなる程顕著である。

今回開発した並列アルゴリズムは、筆者らが開発中の有限要素法のためのシミュレーション言語FEELに組み込み中である。FEELは、ユーザによる解析対象の方程式の入力から有限要素法のプログラムを自動生成するシステムである。これにより、ユーザが方程式を記述するだけで並列計算機用のプログラムが自動生成され、ユーザは並列計算機上でのプログラミングを意識せずに数値シミュレーションを行うことが可能となる。

参考文献

[1] Alex Pothen, Horst D. Simon and Kang-Pu Liou : *Partitioning Sparse Matrices with Eigenvectors of Graphs*, SIAM J. Matrix Anal. Appl. 11 (1990) 430-452.

[2] H. D. Simon : *Partitioning of Unstructured Problems for Parallel Processing*, Computing Systems in Engineering 2 (1991) 135-148.

[3] 藤尾、村松、土肥：有限要素法のための数値シミュレーション言語，情報処理学会第49回全国大会講演論文集(1) (1994) 141-142.

[4] 丸山 他：並列コンピュータ *Cenju-3* のアーキテクチャとその評価，電子情報通信学会論文誌 Vol.J78-D-I,No.2 (1995) 59-67.