Multi Core PC クラスタにおける省メモリ型有限要素法解析の実装

2007 年 3 月　修了
人間環境学専攻　56776　小野 正法
指導教員　吉村 忍 教授

Instead of driving clock speeds and straight-line instruction throughput ever higher, the major processor manufacturers are turning to multicore architectures. In this paper, we have produced memory efficient FEM solver on Multi Core PC Cluster, using Balancing Diagonal Scaling Domain Decomposition Method, with highly optimized CG algorithm of “Element-by-Element Matrix Storage Free Method (EBEMSFM)”.

Key Words: Domain Decomposition, Iterative Methods, Balancing, Elastic Problems, Element-by-Element, Multimedia Extensions

1　緒言

今日，計算機による物理シミュレーションは様々な人工物の設計や自然物の解析に用いられており，その需要はますます高まっている．結果，より高精度の解析結果が要求されるようになり近年では，解析対象を丸ごと解析する大規模解析が注目を集めている．

大規模問題を解くためには, 高速な演算能力, 膨大な記憶容量が必要であるが，現在，パーソナルコンピュータをネットワークで多数接続した仮想並列計算機，いわゆるクラスタ PC がそのコストパフォーマンスの良さから，主流となってきている．

並列計算機における解析にはそれに適したアルゴリズムが必要であり，有限要素法の並列化として有効な手法の 1 つが，解析対象を有限個の部分領域へと分割する領域分割法である．現在では大規模解析向けオープンソース並列有限要素解析システムであるADVENTUREを用いると，地球シミュレータ上で数億自由度の解析が可能である．[1]しかし，PCクラスタのような汎用的な環境において億単位の自由度をもつ解析を省メモリかつ実用時間内で行いたいというユーザーの声も多い．

そこで，本研究では，数千万から数億自由度を有する 3 次元構造問題を，ロバストな解法を用いつつ, 今後普及が進む Multi Core CPU を用いた PC クラスター上で，省メモリかつ実時間内で解析が可能なシステムの提案と実装を行う．ソルバーとして，ロバストな大規模解析ソルバーである，反復型領域分割法に強力な前処理を施した，Balancing 領域分割法の一種である Balancing 対角スケーリング領域分割法を採用した．領域分割法における，領域の計算を Element-by-Element Matrix Storage Free 法による対角スケーリング CG 法を採用した．そして，上記の手法を今後普及が進む Multi Core CPU 上で高速に行うための高速化手法を紹介した．

2　領域分割法

2.1　原理

領域分割法(Domain Decomposition Method: DDM)とは，解析対象となる領域をいくつかの領域に分割し領域ごとに解析を行うことで，全体領域の解を求める手法の総称である．反復型領域分割法とは，領域内部節点に関する自由度を静的縮約した後に，領域間の節点に関する自由度について CG 法等の反復法により解く手法である．インターフェース問題を CG 法によって解く方法を以下に示す．

2.2　基礎式

領域Ω上で有限要素近似によって得られる次式を考える．

$$\mathbf{Ku} = \mathbf{f} \qquad (1)$$

ただし**K**はn×nの正定値対称な剛性行列，**u**は節点変位ベクトル, **f**は等価節点外力ベクトルである．この領域を領域境界に重なりを持たない

ように分割する. 領域$\Omega^{(i)}$の自由度$u^{(i)}$ (自由度数$n^{(i)}$)により

$$u^{(i)} = R^{(i)t}u, \quad i = 1, \cdots, N \tag{2}$$

$R^{(i)t}$を定義する. (I)部分領域内部節点に関する自由度(自由度数$n_I^{(i)}$)と(B)領域境界上の自由度(自由度数$n_B^{(i)}$)に分けると次式のようになる.

$$u^{(i)} = \begin{bmatrix} u_I^{(i)} \\ u_B^{(i)} \end{bmatrix} \tag{3}$$

同様に，$K^{(i)}$ $f^{(i)}$ $R^{(i)}$について次式が得られる.

$$K^{(i)} = \begin{bmatrix} K_{II}^{(i)} & K_{IB}^{(i)} \\ K_{IB}^{(i)t} & K_{BB}^{(i)} \end{bmatrix}, \quad f^{(i)} = \begin{bmatrix} f_I^{(i)} \\ f_B^{(i)} \end{bmatrix}, \quad R^{(i)} = \begin{bmatrix} R_I^{(i)} & \mathbf{0} \\ \mathbf{0} & R_B^{(i)} \end{bmatrix} \tag{4}$$

よって，各領域における内部節点に関してのつりあいの式は次式で表わされる.

$$u_I^{(i)} = K_{II}^{(i)-1}\left(f_I^{(i)} - K_{IB}^{(i)}u_B^{(i)}\right) \tag{5}$$

領域間境界上自由度$u_B^{(i)}$が得られれば，各部分領域における領域内部自由度の未知数$u_I^{(i)}$を並列に求めることができる.

また，領域間境界上自由度に関する釣り合いの式は次式で表わされる.

$$\sum_{i=1}^{N} R_B^{(i)} K_{IB}^{(i)t} u_I^{(i)} + \left\{\sum_{i=1}^{N} R_B^{(i)} K_{BB}^{(i)} R_B^{(i)t}\right\} \cdot u_B = \sum_{i=1}^{N} R_B^{(i)} f_B^{(i)} \tag{6}$$

u_Bについて整理すると次式が得られる.

$$\left\{\sum_{i=1}^{N} R_B^{(i)} \left\{K_{BB}^{(i)} - K_{IB}^{(i)t}\left(K_{II}^{(i)}\right)^{-1} K_{IB}^{(i)}\right\} R_B^{(i)t}\right\} \cdot u_B = \sum_{i=1}^{N} R_B^{(i)} \left\{f_B^{(i)} - K_{IB}^{(i)t}\left(K_{II}^{(i)}\right)^{-1} f_I^{(i)}\right\} \tag{7}$$

この式(u_Bに関する連立１次方程式)はインターフェース問題と呼ばれ，領域分割法において領域間の連続性を満たすために扱う式となる. 通常構造問題において式(7)の係数行列は対称正定値となるため，CG 法などの反復法を用いて解くことができる.

領域間境界上自由度の解を求め，得られた解を式(5)に代入し領域内部自由度の解を求めることで領域全体の解を得ることができる．次節において，インターフェース問題に CG 法を適用し領域全体の解を求めるアルゴリズムを示す.

2.3 CG 法による解法

以下に従い CG 法によって式(7)を求める.

Step 0:領域境界上の初期節点変位u_Bを与える.

Step 1:領域間境界上の節点変位を Dirichlet 境界条件として, 領域ごとに有限要素解析を行う.

Step 2:Step 1 の結果から部分領域間境界上に課せられた反力を求め，それらの領域間境界上での釣り合いを調べる．反力の和が十分ゼロに近ければ収束とみなし，計算を終了する.

Step 3:領域間境界上に与えられた強制変位を更新し Step 1 へ戻り，領域ごとに計算を行う.

このCG法において得られるのは領域間境界上の節点変位u_Bなので，これを用いて領域ごとに有限要素解析を行うことで領域全体の解を得る.

3 Balancing 対角スケーリング領域分割法

3.1 概要

領域分割法は，領域数の増加に伴って，ある領域の情報が他の領域に伝わる速度が遅くなることにより収束性が悪化してしまう．そこで，解析領域に対するコースグリッドを用いて収束性をよくするマルチグリッド(MG)法の考えを取り入れた，Balancing前処理法の一種である Balancing 対角スケーリング領域分割法(BDD-DIAG)を用いた．Balancing前処理は，コースグリッドにおいて情報交換を行うことで，情報が全体に伝播されるので，領域数に依存しない前処理となっている．[2]

3.2 BDD-DIAG 計算量見積もり

次に BDD-DAIG を用いたときの計算量の見積もりを行う. Coarse 行列においては解析対象のモデルが一定の場合不変となり，BDD 反復ごとの Balancing 処理においては右辺項のみが変化する右辺問題となることから，解析開始時に 1 度だけ LDL 分解しておくことがよく行われる．本研究でも同様の手法を用いた．領域計算においては，解析時間を短縮するために，全領域においてローカル剛性行列を LDL 分解しておくことが用いられるが，PC クラスタにおいて大規模解析を行う場合，規則容量の観点から自ずと解析規模の上限が 1 千万自由度クラスに限定されてしまう．そこで BDD-DIAG の反復ごとに全領域において要素剛性行列を作成するアルゴリズムの演算量および演算時間の見積もりを行った.

小規模 PC クラスタを前提にノード数 16，ノードあたりのメモリ量は 2GB，ノードあたりの

演算性能は 25GFlops とした．なおコース行列と一時的に保存しておくローカル剛性行列は skyline 形式で保存するとする．演算量削減のために搭載されている 2GB のほとんどをコース行列保存に用いるとしたので領域数は 8000 とした．

上記を前提に 1 ノードあたりに 1BDD 反復にかかる演算時間の見積もりを行った結果，Fig. 1 となった．図からわかるとおり，領域の自由度が 3000 万自由度近辺から演算時間が指数関数的に延びてしまっていることがわかる．そこで本研究では，領域の計算にＣＧ法をベースとした，Multi Core CPU に適したアルゴリズムを採用した．

4　CG 法による領域計算

4.1はじめに

これまでスカラーCPU の演算能力は半導体技術の微細化技術の発展に伴って，飛躍的に向上してきた．しかし近年，消費電力や熱密度の問題から，1)動作周波数の向上 2)動作周波数あたりに実行できる命令数の向上，などの古典的 CPU の性能向上に限界ができてきている．そこで近年は，1 つのＣＰＵのソケットに比較的単純な回路を実装した core を多数集積することで演算性能の向上を図る，いわゆる chip multi processor 化（または muti core 化）が主流となっている．

Multi Core CPU では多数の core が同一のメモリバスを共有することから，メモリアクセス競合がおきやすく，メモリアクセス頻度の高いアルゴリズムは高速化できない．メモリアクセスを少しでも減らすために，キャッシュ(core の中に実装されているローカルなメモリ)を有効利用することが必要となる．

そこで, 本研究では BDD-DIAG における領域計算にキャッシュ効率の良くすることが可能なアルゴリズムとして Element-by-Element Matrix Storage Free 法による対角スケーリング CG(EBEMSF-DSCG)法を用いる．

EBEMSF法とは有限要素解析を行う際に全体剛性行列や要素剛性行列をメモリに格納せずに解析を行う手法の総称である．[3]

この手法と CG 法とを組み合わせる場合, CG 反復ごとに，要素において要素剛性行列は作る必要がある．省メモリではあるが，メモリアクセス量に対する必要演算量が多いアルゴリズムとなっている．キャッシュを有効利用することで，メモリアクセス頻度を減らし，Multi Core PC 環境において良好にスケールするチューニングを行った．

4.2高速化(計算量の軽減)・最適化

あらかじめ計算量を減らすために体積座標によって要素剛性行列を作成した．数値積分を用いた場合に比べて計算量を約 1/5 にできた．

また、本研究では CPU に実装されているストリーミング SIMD 命令を有効活用するためキャッシュ効率を良くするための最適化を行った. コンパイラとして Intel Compiler を用いた. 具体的には要素 12 個分を 1 単位として，コネクティビティ情報と，座標データをメモリから読み出し，要素剛性行列作成部分をベクトル化している．こうすることで 2 度目に呼び出される座標データは，ほぼキャッシュ上にのっているので高速に演算が可能となった．

結果，CG 法 1 反復にかかる時間は，全体剛性行列の非ゼロ項だけを保存する非ゼロ形式の CG 法の場合が 72.8ms，EBEMSF 法は 46ms と非ゼロ形式の CG 法と比べて，1.5 倍の高速化が実現できた．また，8core 搭載されているマルチコア環境において，EBEMSF-DSCG 法は良好なスケーラビリティがあることが確認できた. そこで, EBEMSF-DSCG を BDD -DIAG 中における領域計算に用いた結果を次節に示す.

4.3 EBEMSF-DSCG による BDD-DIAG 領域計算

BDD-DIAG における領域計算を EBEMSF-DSCG 法を用いた場合，領域間境界上に課される Dirichlet 境界条件によって, CG 法の収束条件の悪化が懸念される．そこで Fig. 3 のようなモデルを用意し BDD-DIAG の領域計算に EBEMSF-DSCG を用いたときの演算時間を計測した．領域分割を 1x10x10 とし，領域の要素分割については数種類用意した．得られた結果をもとに，再び領域数を 8000 として各全体自由度における計算自由度の見積もりを行った結果を Fig. 1 と比較したグラフが Fig. 2 である．また，Fig. 3 と同様な形状・境界条件でアスペクト比が 1:100:100 である 1400 万自由

度問題の解析を，領域分割数を 6300 として行った．結果 BDD 反復回数が 101，1BDD 反復に 88 秒, 反復に要した時間は 2 時間 30 分であった．この自由度の規模では直接法と比べて差はないが，より大規模な問題において本手法は威力を発揮することがわかった．

5　結論

本研究では Balancing 対角スケーリング領域分割法をベースに，領域の計算に EBEMSF 法による CG 法を実装した．小規模な PC クラスタにおいて，従来は使用メモリの限界から，膨大な演算時間がかかっていた数千万～数億自由度規模の問題，本手法を用いることで，実用時間内で行うことが可能であることを示すことができた．

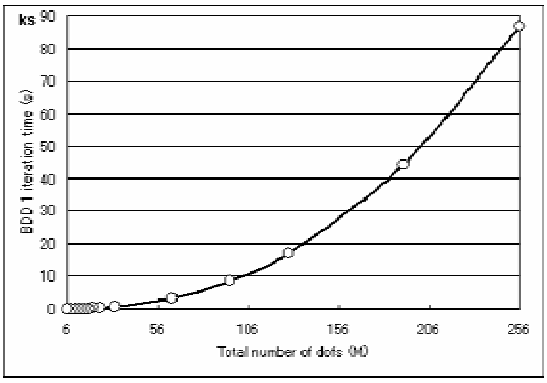

Fig.　1 1BDD iteration time / node vs. total number of dofs (M)

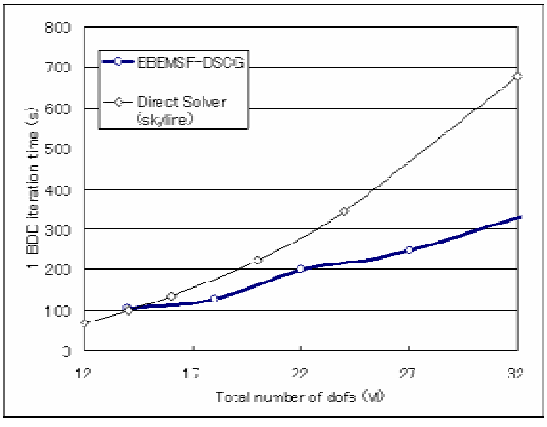

Fig.　2 1 BDD iteration time / node vs. total number of dofs (M)

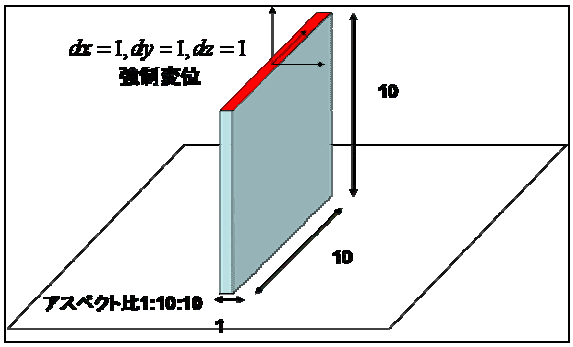

Fig.　3 Analysis model and its boundary conditions (aspect ratio=1:10:10)

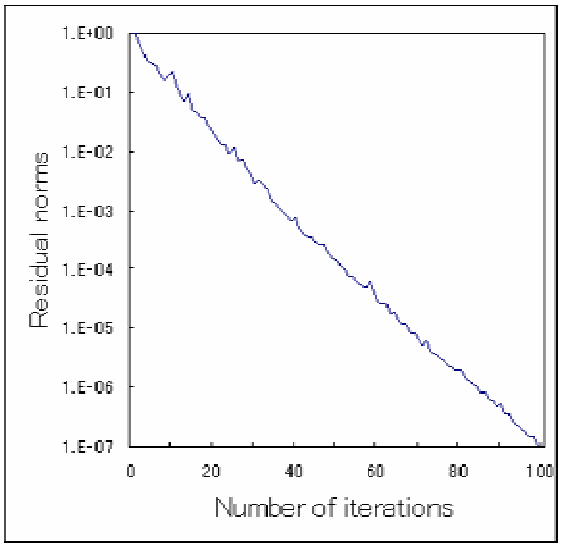

Fig.　4 History of residual norms

参考文献

1) Masao Ogino, Ryuji Shioya, Hiroshi Kawai, and Shinobu Yoshimura, Seismic Response Analysis of Nuclear Pressure Vessel Model with ADVENTURE System on the Earth Simulator, *Journal of the Earth Simulator, Vol.***2**, 2005, pp41-54.

2) Mandel, J., Balancing Domain Decomposition, *Communications on Numerical Methods in Engineering*, Vol.**9**, 1993, pp233-241.

3) Yagawa G, Nakabayashi Y, Okuda H (1997) Large-scale finite element fuid analysis by massively parallel processors, *Parallel Computing* **23**,1365-1377