ハイパフォーマンスコンピューティング 60－7
（1996. 3. 4）

ATM 接続された複数の並列計算機による有限要素法並列化

福盛秀雄　水野裕識
西松研　村岡洋一

広域ATMネットワークの一つであるOLU(On Line University)ネットワークを利用した、複数の並列計算機による有限要素法ソルバの並列化について論ずる。計算法にはネットワーク上におけるデータ転送量を削減するため、階層化されたサブストラクチャ法を採用している。今回は富士通の並列計算機AP1000とNECの並列計算機Cenjuを利用し、性能を評価した。

Parallelization of FEM with Parallel Computers Connected via ATM Network

HIDEO FUKUMORI , HIRONORI MIZUNO , KEN NISHIMATSU
and YOICHI MURAOKA

In this paper, we discuss implementation of a FEM solver on two different parallel computer systems connected via OLU(On Line University) network. To reduce overhead with the network data transfer, this solver uses the substructure method applied in multiple levels as the calculation scheme. We implemneted the solver on the Fujitsu AP1000 and the NEC Cenju and evaluated the performance.

1. はじめに

本発表では、OLU(On Line University)ネットワークによって接続されたヘテロジーニアスな並列計算機上における有限要素法ソルバの実装について論じる。

OLUネットワークは日本全国の大学・研究機関を結ぶ広域ATMネットワークである。各サイトは光ファイバケーブルにより接続されたリング状のトポロジを形成している。

このような広域ネットワークにおける並列計算機環境においては、計算機間で通信を実行する際に、ネットワークの遅延が無視出来ないほど大きくなるという問題が発生する。小さなサイズのデータが、繰り返し転送され、かつ頻繁に同期が要求されるようなようなアプリケーションにおいては、この傾向は特に顕著となる。

また、ヘテロジーニアスな計算機同士を接続し、計算をさせる場合には計算機間の性能差を考慮した適切な負荷分散を実現する必要がある。有限要素法の場合では、節点数による負荷分散の調整などが必要となる。

今回の実装では並列化のためのアルゴリズムとして、階層化されたサブストラクチャ法を採用した。サブストラクチャ法では、計算過程における計算機間のデータ授受は限定された回数のバースト型通信で完了する。そのためネットワークの遅延による影響を最小限にとどめることが出来るという特徴がある。

この階層化されたサブストラクチャ法を基とした並列化有限要素法ソルバをOLUネットワークによって接続された早稲田大学に設置されている富士通の並列計算機AP1000,NEC C& C研究所に設置されている並列計算機Cenju-3上で実行し、単独の並列計算機による実行時間の計測、およびOLUネットワークを使用したデータ転送に要する時間の予測値を元に計算された予想パフォーマンスと比較する。

2. OLUネットワーク

本節では今回の実装の元となったATMネットワークであるOLU(On Line University)ネットワークについて説明する。

OLU(On Line University)プロジェクトは、156Mbps広域ATM網(サイト間の実効帯域幅は67Mbps)を用いた高速コンピュータ通信利用実験であり、現在全国の大学・企業合わせて23の機関がATMネットワークによって接続されている。

OLUネットワークには富士通AP1000、NEC Cenju-3をはじめとした並列計算機が接続されている。これら複数の並列計算機を利用した分散並列環境の構築は本プロジェクトのテーマの一つとなっている。

今回使用したのは早稲田大学と日本電気を結ぶリンクである。途中サイトは5つ、早稲田—日本電気間のネットワーク総延長距離は370km である。

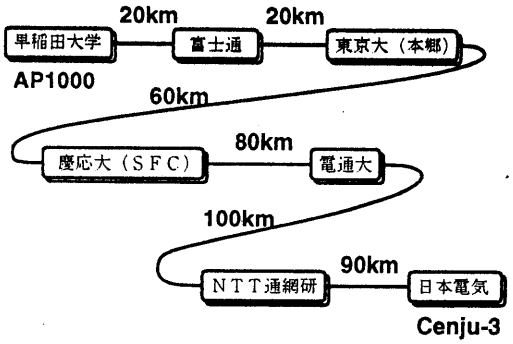

図1　早稲田大学—日本電気間ネットワーク接続

3. 計算アルゴリズム

本実装では階層化されたサブストラクチャ法を計算アルゴリズムとして採用している。この方法は領域分割に基づく直接法を基本としているが、領域分割を複数の段階に分け、各段階で発生する係数行列をサイズの小さな密行列とすることで、計算効率を向上させる点に特徴がある。

階層化されたサブストラクチャ法は以下の手順によって実行される:

(1)　解析しようとする領域を、内部に1節点を持つ、9節点から構成されるサブストラクチャに分解する。これを第1階層のサブストラクチャとする。

(2)　第k階層のサブストラクチャにおいて：

(a)　各サブストラクチャにおける節点は、サブストラクチャ領域内部に存在する節点と、サブストラクチャ外周部に存在する節点の2種類に分けることができる。ここで各サブストラクチャにつき、内部節点を先に、外周節点を後にした新たな番号付けをし、x_iをサブストラクチャ領域内部にある節点についての解、x_bをサブストラクチャ外周部にある節点についての解とおくと、次のような連立方程式が成立する。

$$\begin{bmatrix} A_{ii}^{(k)} & A_{ib}^{(k)} \\ A_{bi}^{(k)} & A_{bb}^{(k)} \end{bmatrix} \begin{bmatrix} x_i^{(k)} \\ x_b^{(k)} \end{bmatrix} = \begin{bmatrix} b_i^{(k)} \\ b_b^{(k)} \end{bmatrix} \quad (1)$$

より、

$$A_{bb}^{*(k)} x_b^{(k)} = b_b^{*(k)} \quad (2)$$

を導く。

(b)　各サブストラクチャは、隣接するサブストラクチャとペアを作り、一段階上の階層を構成するスーパストラクチャをを新たに構成する。これらを新たな階層におけるサブストラクチャと見なすこととする。

(c)　$A_{bb}^{*(k)}$と$b_b^{*(k)}$は、ペアの相手であるサブストラクチャのそれと足し合わされることで、新たに連立方程式$A^{(k+1)}x^{(k+1)} = b^{(k+1)}$を構成する。

(3)　サブストラクチャ数が1になるまで、上の手順を繰り返す。

(4)　サブストラクチャ数が1になるような階層k_{max}に到達したら、式

$$A^{(k_{max})}x^{(k_{max})} = b^{(k_{max})} \quad (3)$$

を解くことにより、$x_i^{(k_{max})}$
を求める。

(5)　第$k-1$階層のサブストラクチャは、式

$$x_i^{(k-1)} = (A_{ii}^{(k-1)})^{-1}(b_i^{(k-1)} - A_{ib}^{(k-1)}x_b^{(k)}) \quad (4)$$

より、$x_i^{(k-1)}$を求める。

(6)　上の手順を、k=1になるまで、すなわち最下位のサブストラクチャに到達するまで繰り返す。

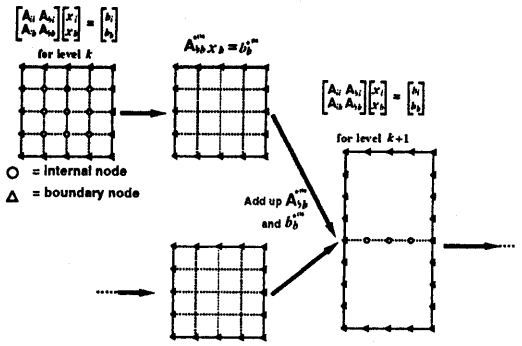

図2　階層化サブストラクチャ法の計算過程

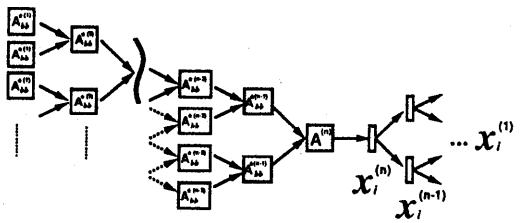

図3　A_{bb}の足し合わせおよびx_iの受け渡し

図3に見られるように、階層化されたサブストラクチャ法の全体の計算過程は二分木構造で表すことができる。階層間でのデータの授受は

- 係数行列の足し合わせ
- 最終解を求める処理

の2個所であるが、いずれも二分木の枝で結ばれた部分のみで発生することが分かる。

また、理想的場合、すなわち計算過程において完全

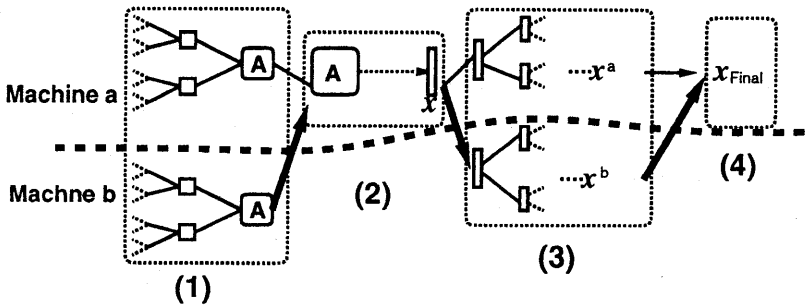

図4　ネットワークを介した階層化サブストラクチャ法の実装

な二分木構造が実現された場合においては、節点数 n に対し、全体での乗算回数のオーダは $O(n^{\frac{3}{2}})$ となる。

4. ネットワークを介した並列化における問題

前節で述べた階層化されたサブストラクチャ法をOLU ネットワークで接続された 2 台の並列計算機に実装する場合のパフォーマンスについての問題についてまとめる。

ネットワークを介した有限要素法の並列化においては通信のオーバヘッドが大きな問題となる。通信における主なオーバヘッドの原因としては、

- ネットワークのレイテンシ
- ネットワークの帯域幅

が考えられる。

有限要素法の並列化の場合、反復法による実装においては、ネットワークのレイテンシが、サブストラクチャ法による実装においてはネットワークの帯域幅が通信のボトルネックになると予想される。

本実装では並列計算機間のデータ送受信を計算過程の一部分に集中し、回数を限定することによって計算中に必要な同期はごく一部分にとどめ、ネットワークのレイテンシによって発生するオーバヘッドを最小限に抑えることにより、実際のパフォーマンスはほぼ帯域幅のみに依存するような状況を実現することとする。

図4にネットワークを介した場合における階層化されたサブストラクチャ法の実装の全体像を示す。サブストラクチャ法の計算処理は大きく以下のように分類される。

処理 (1)　係数行列の計算 (式 (1),(2))
処理 (2)　最上位階層の連立方程式の求解 (式 (3))
処理 (3)　最終解 $x_i^{(k)}$ を求める (式 (4))
処理 (4)　両計算機で得られた解から、全体の解を求める処理

ただしこのうち処理 (4) は代入処理のみであり、処理時間は問題にならない程度に小さい。

1 台の並列計算機αで上記の計算を実行した場合において、処理 (1),(2),(3) に要する時間をそれぞれ T_1,T_2,T_3 とする。

またこれに計算機βを加えた 2 台の並列計算機によって上の計算を並列に実行する場合において、計算機α, βが処理 (1),(2),(3) に要する時間をそれぞれ $T_1^{[\alpha|\beta]}$,$T_2^{[\alpha|\beta]}$,$T_3^{[\alpha|\beta]}$ とする。

α, βの計算性能 (Flops) の比が $a:1$(ただし、$a \geq 1$) であり、負荷分散が理想的に行なわれている場合には、

$$T_1^{\alpha} = T_1^{\beta} = \frac{aT_1}{a+1}$$
$$T_2^{\alpha} = T_2^{\beta} = T_2$$
$$T_3^{\alpha} = T_3^{\beta} = \frac{aT_3}{a+1}$$

という関係が成立する。

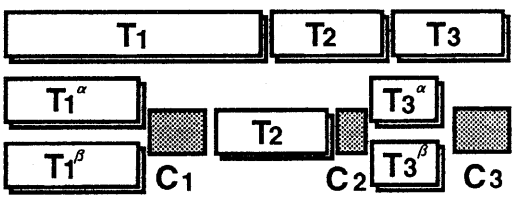

図5　実行時間のモデル

計算機間の通信は以下の 3 個所で発生する。それぞれの通信に要する時間を以下の通り定める。

- 処理 (1)-(2) 間で発生する通信に要する時間 C_1
- 処理 (2)-(3) 間で発生する通信に要する時間 C_2
- 処理 (3)-(4) 間で発生する通信に要する時間 C_3

すると、2 台の並列計算機による実行時間が 1 台での実行時間を上回るための条件は、

$$\frac{aT_1}{a+1} + C_1 + T_2 + C_2 + \frac{aT_3}{a+1} + C_3 < T_1 + T_2 + T_3$$

となる。これを変形し、整理すると、

$$T_1 + T_3 > (a+1)(C_1 + C_2 + C_3) \qquad (5)$$

という関係が得られる。

次に並列計算機間で発生する通信に要する時間 C_1, C_2, C_3 について考察する。

対象として、有限要素分割された正方形領域を仮定する。全体の節点数を n とした場合、2 つの並列計算

機がそれぞれ担当する領域の境界線上に存在する節点数を$\sqrt{n}$とすると、処理(1)-(2)間で発生する通信においては、担当領域の境界線上に存在する節点についての連立方程式の係数行列の内容が授受され、足し合わされる。この係数行列はサイズ$\sqrt{n} \times \sqrt{n}$の対称密行列である。行列の各要素は倍精度型で表現されるとする。足し合わせ先の行列における行・列の情報をそれぞれ整数型で保持しておくものとすると、処理(1)-(2)の間で送受信されるデータの総量は概算で

$$\frac{\sqrt{n} \times \sqrt{n} \times (8+4+4)}{2} = 8n(Bytes) \qquad (6)$$

となる。

最上位階層では境界部分の節点についての解が求められる(処理(2))。この値からそれぞれの下位階層についての解が段階的に得られる(処理(3))。処理(2)-(3)の間において並列計算機間で送受信されるデータは、境界線上に存在する節点数$\sqrt{n}$に倍精度のデータ表現に要するバイト数を掛け合わせたもの

$$\sqrt{n} \times 8 \qquad (7)$$

となる。

処理(3)と処理(4)の間では、並列計算機βからαへのデータ送信が発生する。マシンα, βの担当節点数の比を$m:1$とすると、αからβへ転送されるデータ量は

$$\frac{8n}{m+1} \qquad (8)$$

となる。ここで節点数nに対する乗算回数のオーダーが$O(n^{\frac{3}{2}})$となることを利用すると、α, βの計算性能(Flops)の比が$a:1$(ただし、$a \geq 1$)の場合、計算により

$$m = a^{\frac{2}{3}} \qquad (9)$$

となる。式(),()より、αからβへ転送されるデータ量は最終的に、

$$\frac{8n}{a^{\frac{2}{3}}+1} \qquad (10)$$

となる。

ネットワークのデータ転送能力をB(bit/s)とすると以上の結果からネットワーク間のデータ転送に要する時間$C1, C2, C3$が求まり、それぞれ

$$\frac{8n}{B} \times 8 = \frac{64n}{B}(sec) \qquad (11)$$

$$\frac{\sqrt{n} \times 8}{8} \times 8 = \frac{64\sqrt{n}}{B} \qquad (12)$$

$$\frac{8n}{a^{\frac{2}{3}}+1} \times 8 = \frac{16n}{(a^{\frac{2}{3}}+1) \times B} \qquad (13)$$

となる。

5. 実装と評価

本節ではOLUネットワーク上ににおける有限要素法並列化プログラムの実装と評価について述べる。

今回はOLUネットワークによって接続された2台の並列計算機(富士通のAP1000およびNECのCenju-3(ともに64プロセッサ構成))に階層化されたサブストラクチャ法を実装した。実際の計算においては、Cenju-3がサーバ、AP1000がクライアントとして動作する。

評価の対象には$0 \leq x \leq 1, 0 \leq y \leq 1$の正方領域におけるラプラス方程式に、128×128の正方メッシュ分割をベースとする一次三角形要素によって有限要素分割したものを用いる。この場合の要素数は32768, 節点数は16641となる。

計算の全体像は図4に示した通りである。図中のmachine aがCenju-3、machine bがAP1000に対応する。

まず解析領域をCenju側担当部分,AP側担当部分に分け、それぞれの領域について別々にサブストラクチャ階層化のための節点番号テーブルを作成する。

両並列計算機は担当部分の節点情報、要素情報、サブストラクチャ階層化テーブルをあらかじめローカルに保持し、それを使って計算を開始する。

- 計算の前半部において、最大サブストラクチャ階層の一つ前の階層k_{max-1}に到達するまでは、両並列計算機は完全に独立に計算を実行する。
- AP1000側からCenju側へ、係数行列$A^{(k_{max-1})}$が送信される。サーバ側ではこれを足し合わせることによって最上位階層のための係数行列$A^{(k_{max})}$を生成する。
- Cenju側において最上位階層における連立方程式(式(4))を解く。
- Cenju側にて得られた解$x_i^{(k_{max})}$をAP1000側へ送信する。
- 両並列計算機は式(4)により、それぞれ全ての階層についてx_i^kを求める。
- AP側で得られた最終解x_i^{AP}をCenju側へ転送する。

次に実測値に基づいた性能の予測と負荷の分散について論ずる。

先の議論では、いずれも計算機の理論的な処理能力としてFlops値を用いていたが、実機での動作においては様々な条件によりパフォーマンスが理論値通りとならないケースが一般的である。

したがって実際のアプリケーションにおける性能比を適切に知るために、実プログラムの実行時間の計測と評価が必要となる。

まず富士通AP1000とNEC Cenju-3(いずれも64プロセッサ構成)にてそれぞれ単独に階層化されたサブストラクチャ法をベンチマークとして実行し、その実行時間を計測した。

この表で得られた実行時間の比の逆比を処理能力比として採用し、式(9)に代入して計算すると、適切な節点数比の概算値として、4225節点の場合には

表1 ratio

分割数	実行時間 (Cenju)	実行時間 (AP)	実行時間の比 (Cenju:AP)
4225	0.241	0.736	1:3.05
16641	1.163	3.187	1:2.73

2.10,16641 節点の場合には 1.95 という値が得られる。128×128 分割の問題を 2 台の並列計算機で実行する場合には、これら二つの節点数比の中間にある適当な値が、適切な節点数の比であると考えられる。今回は単純化した値として、節点数比を 2.0 に設定した。

表 2に各計算過程別にまとめた計算時間の実測値を示す。表 1に示される Cenju あるいは AP1000 の単独動作の場合と比較してのパフォーマンスの低下が見られるが、その理由を以下にまとめる。

- サブストラクチャ計算の前半におけるパフォーマンス低下について、表 1に示されるような並列計算機を単独動作させたような場合には、計算過程で生じる二分木は均等で、かつ常にプロセッサ台数の倍数となるような葉を持つような構造になっている。これに対し解析領域を不均等に分割した場合においては、計算過程における二分木構造が理想的な場合と比べ大きく変化する。現在使用している計算ルーチンは理想的な二分木を対象に最適化されているために不均等な構造を持つ二分木に対し、計算効率が著しく低下する。この部分については早急に改良を加えることとする。まだ同時に、不均等な二分木構造を持つ問題に対しては二分木の葉の数がプロセッサの台数の倍数とならないという状況が発生する結果、負荷分散に不均等が生じ、稼働プロセッサ数が減少してしまうという問題が発生する。この問題についてはさらに解析が必要である。
- 計算機間のデータ転送について、両計算機のネットワークを通じたデータ送受信は AP1000 側はフロントエンドマシン、Cenju 側は第 0 番のプロセッサが全体を代表して行なっている。大きなプロセッサ構成で計算を実行する場合には、このような特定のマシンおよびプロセッサに、他のプロセッサからのデータが集中するという現象が発生する。この部分がネットワークの帯域幅以前のボトルネックとなっている可能性がある。

表2 過程別計算時間実測値

計算過程	所要時間 (s)
サブストラクチャ計算前半 (T1)	3.08
係数行列の転送 (C1)	12.66
最上位階層の求解・結果転送 (T2&C2)	0.056
サブストラクチャ計算後半 (T3)	0.027
最終解の転送 (C3)	8.27
実行時間 (全体)	24.10

6. まとめ

本発表では ATM ネットワーク上で接続された 2 台の異なる並列計算機による有限要素法ソルバの並列化について論じた。実行時間の測定の結果、負荷の分散とデータ集中によるボトルネックによると思われるパフォーマンスの低下が見られた。この問題への対策、今後の方針を以下に述べる。

- 並列要素分割の利用。確実な負荷分散と並列による要素生成を同時に実行することによって、全体のスループットを向上させる。
- 個別のプロセッサによるネットワーク通信のサポート。データを 1 個所に集中させることなく転送することによりボトルネックの回避を実現する。ただし通信が分散することにともない、ネットワーク遅延の影響が問題となる可能性がある。この点についてはさらに検討が必要である。

謝辞

本研究を進めるにあたり、計算機環境、ネットワーク環境、その他多くのご支援を提供いただいた NEC C & C 研究所 並列処理センター、(株) 富士通研究所、および OLU コンソーシアムの皆様に深い感謝の意を表します。

参考文献

1) 村岡洋一: On-line University プロジェクトについて, 第 2 回 JAIN CONSORTIUM Symposium 論文集, pp. 1-10 (1994).
2) 天野良治ほか: 長距離超高速インターネットの特性解析, 第 2 回 JAIN CONSORTIUM Symposium 論文集, pp. 11-18 (1994).
3) 西村 浩二: OLU-net の現状と課題、そして今後
4) 加納 健, 中田 登志之, ほか: 並列マシン Cenju 上の有限要素法による非線形変形解析, 並列処理シンポジウム JSPP'92, pp.399-406, 1992
5) 加納 健, 中田 登志之, 他: "並列マシン Cenju2 上の有限要素法による非線形変形解析, 並列処理シンポジウム JSPP'93, pp.379-386, 1993
6) 杉原 光太, 藤原 秀洋, ほか: 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, 情報処理学会研究報告 95-HPC-59, pp.31-36, 1995
7) 鶴見敬之: 有限要素法ソルバアルゴリズムの分散メモリ型並列計算機 AP-1000 への実装方法の研究, 村岡研究室 1993 年度修士論文, 1993

8) G.Yagawa, N.Soneda and S.Yosimura: A Large Scale Finite Element Analysis Using Domain Decomposition Method on A Parallel Computer, Computers & Structures Vol.38, No.5/6, pp.615-625, 1991
9) 矢川元基, 曽根田直樹: 計算力学と CAE シリーズ 7 パラレルコンピューティング, 培風館, 1991
10) G.Fox, M.Johnson, et al: Solving Problems on Concurrent Processors (Volume1), Prentice Hall, 1988
11) I.St.Doltsinis and S.Nolting: Studies on parallel processing for coupled field problems, Computer Methods in Applied Mechanics and Engineering vol.89, pp.497-521, 1991
12) Hideo Fukumori and Yoichi Muraoka: Parallel FEM Solution Based on Substructure Method, PCW'93 Proceedings of Fujitsu Second Parallel Computing Workshop, P1-K, 1993
13) F.J.Peters: Sparse Matrices and Substructures, Mathematisch Centrum, 1980
14) 河野 他：分散メモリ型並列計算機上の有限要素法のための要素自動分割, 情報処理学会第 48 回全国大会,1-135,1994.
15) 福盛秀雄, 河野洋一, 西松 研, 村岡洋一: 階層化サブストラクチャ法の適用による有限要素法の並列化とその実装, 情報処理学会研究報告 95-HPC-57, pp.79-84, 1995
16) H. X. Lin and H.J. Sips: Parallel direct solution of large sparse systems in finite element computations, Proceedings of 1993 Int. Conf. on Supercomputing, pp.261-270, 1993
17) Hideo Fukumori et al.: "Parallelization of FEM with multi-level substructure method", Proceedings of HPC'ASIA 1995, 1995
18) Michel T. Heath et al.: "A cartesian parallel nested dissection algorithm", SIAM J. MATRIX AMAL. APPL. vol.16, No.1, pp.235-253, 1995