ハイパフォーマンス
コンピューティング 77－19
（1999. 8. 3）

分散メモリ型並列計算機向けスパースソルバの開発と評価

山本有作†，猪貝光祥§，直野健†
†（株）日立製作所中央研究所
§（株）日立超LSIシステムズ

概要

分散メモリ型並列計算機上での構造解析向け連立一次方程式ソルバとして，直接法に基づくソルバを開発し，並列計算機SR2201上で評価を行った。本ソルバは行列のオーダリングから，行列のプロセッサ間への分割，シンボリック分解，コレスキー分解，前進後退代入までを一貫して行うプログラムであり，計算の中心となるコレスキー分解においては，行列の不規則な非零構造に合わせて局所的に最適なループ展開を行うことにより，RISC向けの最適化を行っている。

本プログラムをSR2201上で評価したところ，7万元の構造解析の問題で単体性能約100MFLOPS（ピークの33%），16プロセッサ時に約10倍の加速率を達成し，実用的な規模の問題で十分な性能が得られることがわかった。

Development and Evaluation of a Sparse Direct Solver for Distributed-Memory Parallel Processors

Yusaku Yamamoto†, Mitsuyoshi Igai§ and Ken Naono†
† Central Research Laboratory, Hitachi Ltd.
§ Hitachi ULSI Systems Corp.

Abstract

We developed a parallel sparse direct solver for the solution of linear simultaneous equations arising from structural analysis on distributed-memory parallel processors. Our solver includes programs for ordering, distribution of the matrix, symbolic factorization, numerical factorization, and solution. The numerical factorization part is optimized for RISC processors by using the optimal loop unrolling pattern for each part of the matrix, according to the local nonzero structure.

We evaluated this solver on the Hitachi SR2201 parallel processors, and obtained single processor performance of about 100MFLOPS, and speedup of 10 on 16 processors for a structural analysis problem of dimension 70,000.

1．はじめに

有限要素法による構造解析は，機械設計や建物設計への利用を初めとして，産業界で広く使われているアプリケーションの一つである。近年，計算の大規模化・精緻化に伴い，構造解析においても，高い演算能力と広いメモリ空間を持つ分散メモリ型並列計算機の利用への要求が高まっており，MARCやNASTRANなど代表的なソフトウェアの並列機への移植が行われている。構造解析の計算では，多くの場合，連立一次方程式の求解が計算の中心であり，計算時間の大部分をこれらのソフトウェアがコールする連立一次方程式ソルバが占める。したがって，全体性能を上げるには効率の良い分散メモリ型並列機向けソルバの開発が不可欠である。

構造解析で現われる行列は多くの場合，対称正定値で条件数が大きい大規模疎行列である。このような行列を係数とする連立一次方程式を安定に解く方法として，スパースソルバと呼ばれる直接法に基づく解法がある[1]。スパースソルバではコレスキー分解後に非零になる要素のみを記憶し，演算を行うことにより，従来のスカイライン法に比べて大幅な演算量削減を実現している。

スパースソルバの並列化の研究は数多く行われており，オーダリング[4][5][6]，シンボリック分解[2]，コレスキー分解[8]，前進後退代入[7]など，ソルバを構成する各部分について，分散メモリ型並列機向けのアルゴリズムが提案･評価されている。しかし，これらすべての処理を一貫して行い，連立一次方程式の求解が行えるプログラムはまだ数が少なく，ScaLAPACK の一部である CAPSS[9]，欧州の共同プロジェクトで開発されている PARASOL[10]などいくつかのシステムが利用可能となっているのみである。

本研究では，行列のオーダリングから前進後退代入までを一貫して行うソルバを開発した。本ソルバの特徴は，(1) オーダリングを単体プロセッサで，シンボリック分解以降を分散メモリ上で行うことにより，扱いやすいインタフェースを保ちつつ分散メモリ型並列機の広いメモリ空間を有効に利用できるようにした点，(2) 計算の中心部であるコレスキー分解部において，スパース BLAS の採用により RISC プロセッサ向けの最適化を行った点の２つである。以下では，まず第２章でスパースソルバの基本的な事項を述べた後，第３章で並列化の詳細，第４章で RISC 向けの最適化について述べる。第５章では，コレスキー分解部分について，分散メモリ型並列機 SR2201 上での評価結果を示す。最後に第６章でまとめと今後の課題を述べる。

2. スパースソルバ

2.1 処理の流れ

スパースソルバでは，オーダリング，シンボリック分解，コレスキー分解，前進後退代入の処理を順次行うことにより，連立一次方程式 Ax = b の解を求める。以下，これらの各処理について簡単に説明する[1][2]。

(1) オーダリング

適当な置換行列Pにより係数行列Aの行と列に対して同時置換を行い，$A' = PAP^t$ に変換する。置換行列Pは，コレスキー分解 $A' = LL^t$ を行ったときのフィルイン（分解前に零であって分解後に非零になる要素）の数ができるだけ少なく，かつコレスキー分解における並列性ができるだけ高くなるように選ぶ。オーダリングの主な手法としては，消去の各ステップで生じるフィルインの数を最小にするというアイディアに基づく Minimum Degree (MD) 法系統の手法[3]と，再帰的な領域分割に基づく Nested Dissection (ND) 法系統の手法[4][5][6]がある。ND 法では，有限要素法のメッシュをセパレータと呼ばれる節点集合により２つの部分領域 A と B に分割し，A に属する点，B に属する点，セパレータに属する点の順に番号を付け直す。さらに，この処理を各部分領域に対して再帰的に繰り返すことにより，行列を再帰的縁付きブロック対角行列に変形する。５×５の格子での５点差分の行列に対し，ND 法により置換を行ってできた行列の例を図１に示す。

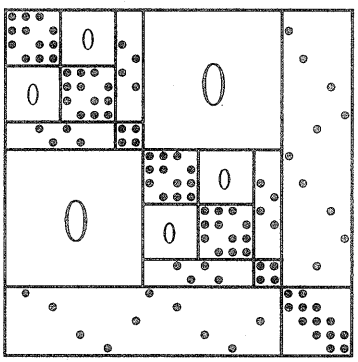


図１ ND法によりオーダリングを行った５点差分の行列

(2) シンボリック分解

コレスキー分解に先立って分解後の非零要素の位置を予め計算し，記憶領域を確保すると共に，非零要素をアクセスするためのインデックスリストを作成する。

(3) コレスキー分解

シンボリック分解で作成したインデックスリストを用いてコレスキー分解 $A' = LL^t$ を計算する。コレスキー分解では，ガウス消去法の中心演算

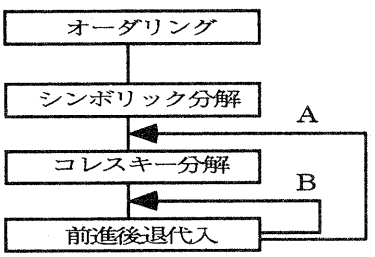
$$a_{ij} = a_{ij} - a_{ik}\, a_{kj} / a_{kk}$$

において，行列の対称性を利用して下三角部分についてのみ演算を行い，かつ分解後に非零となる要素のみを計算する。主な計算方法として，更新を行う列 k に関するループを最外側に持ってくる外積形式，更新される列 j に関するループを最外側に持ってくる内積形式，外積形式において更新を行列に対してすぐに行わずにフロンタル行列と呼ばれる密行列にためておき，後でまとめて更新を行うマルチフロンタル法などがある[2]。

(4) 前進後退代入

方程式 Ly = b と L^tx = y とを順次解くことにより，方程式 Ax = b に対する解を求める。

スパースソルバ全体の処理のフローチャートを図2に示す。シンボリック分解をコレスキー分解から分離したことにより，非線形方程式をニュートン法で解くための反復のように行列の非零構造が変わらず行列要素の値のみが変化する場合には，コレスキー分解以降の処理のみを反復すればよい。また線形問題での時間発展の計算のように，行列が変わらず右辺ベクトルのみが変化する場合には，前進後退代入のみを反復すればよい。



A: 非零構要素の値のみが変わる場合
B: 右辺ベクトルのみが変わる場合

図2　スパースソルバのフローチャート

2.2　消去木と消去演算の並列性

次に，スパースソルバの並列化を行うに当たって便利な消去木について述べる[2]。消去木はコレスキー分解後の行列 L の非零構造を用いて定義される根付き木であり，L の各列が木の節点に対応する。節点間の親子関係は，列 i と j (i>j) に対して，L の第 j 列の対角要素のすぐ下の非零要素が第 i 要素であるとき i が j の親であると定義する。図1の行列に対する消去木を図3に示す。丸の中の数字が列の番号である。行列の再帰的ブロック対角構造と消去木の再帰的な部分木構造

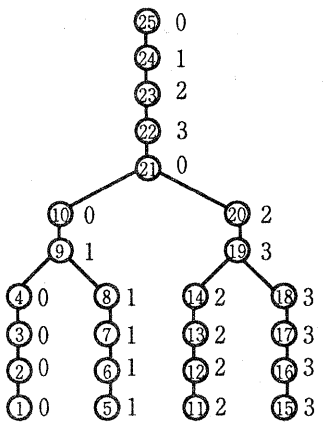


図3　図1の行列に対する消去木

とが対応し，対角ブロックが部分木に，セパレータが分岐と分岐の間の鎖の部分に対応する。

消去木の性質として，「行列の第 j 列による更新が第 i 列に影響を及ぼすのは，消去木において節点 i が節点 j の祖先であるときに限る」ということが成り立つ。従ってコレスキー分解に内積形式を用いる場合，ある列 i の更新に使う列はすべて消去木上での節点 i の子孫であり，節点を共有しない複数の部分木の更新演算は独立に行える。

このことを用いて subtree-to-subcube 割り当て[2]という行列データの分割法が提案され，広く使われている。subtree-to-subcube 割り当てでは，2のべき乗個のプロセッサに対し，消去木の根から始めて各節点をサイクリックにプロセッサに割り当て，木の分岐点に来たらプロセッサ群を半分ずつ2個の部分木に割り当てて，それぞれの部分木で再びサイクリック割り当てを繰り返す。図3の消去木に対して subtree-to-subcube 割り当てにより4台のプロセッサを割り当てたときの担当プロセッサ番号を，各節点の右に数字で示してある。

subcube-to-subtree 割り当てでは，1台のプロセッサに割り当てられた部分木の更新演算を通信なしに行える。したがって，演算の並列性を高めるには，なるべく多くの節点がこれらの部分木に含まれるように(すなわちセパレータがなるべく小さくなるように) し，かつ各部分木の大きさがなるべく均等になるようにすればよい。

3. スパースソルバの並列化

3.1 並列化対象の処理

本章では，スパースソルバの各部分の並列化手法について述べる。

前章で述べたスパースソルバの各処理のうち，本研究で開発したプログラムでは，オーダリング部分のみを1プロセッサで行い，その後にユティリティプログラムにより行列を複数のプロセッサに分割し，シンボリック分解以降の処理は完全に分散メモリ上で行う方式を採用した。その理由は次の通りである。

(a) 現状では，有限要素法のプログラムからソルバに対して分割された形の行列を渡すためのインタフェースが統一されておらず，行列全体を渡す方がユーザにとって使いやすい。

(b) 分解前の非零要素数は分解後の非零要素数に比べて1/10程度であるため，プロセッサ数が十数台までの並列化であれば，分解前の行列を1プロセッサのメモリに格納することは，それほど大きなメモリ上の制約にならない。

これにより，扱いやすいインタフェースを保ちつつ，分散メモリ型並列機の広いメモリ空間を有効に利用できる。なお，本プログラムでは各処理が独立なプログラムとなっているため，オーダリングと行列分割ユティリティのみを変更すれば，オーダリングから求解までのすべてを分散メモリ上で行うソルバへの変更も可能である。

3.2 各部分の並列化

(1) オーダリング

オーダリングには，ND法の一種であるMultilevel ND (MND) 法[4][5]を採用した。MND法は，入力メッシュを粗いメッシュで近似し，粗いメッシュ上でのセパレータを求め，それを再び元のメッシュ上に引き戻す，という処理を再帰的に行うことによりセパレータを求める手法である。MND法は，セパレータが小さく部分木の大きさが均等な高品質の分割を得られるだけでなく，セパレータを求めるための演算量もSpectral ND法[6]など他のND法系統のアルゴリズムに比べて小さいという特長を持つ。

本ソルバでは，MND法によりメッシュをプロセッサ台数分の部分領域へ分割した後，各部分領域内部をMD法系統の手法[3]によりオーダリングし直すことで，さらに演算量の削減を図っている。なお，本ソルバで採用したオーダリングの詳細については，別稿で報告する予定である。

(2) 行列の分割

オーダリング後の行列に対して消去木を作成し，subtree-to-subcube 割り当てにより行列の各列をプロセッサに割り当てる。なおサイクリックに割り当てを行う部分では，通信オーバーヘッド削減のため，ブロックサイズ12のブロックサイクリック分割を採用した。

(3) シンボリック分解

各プロセッサに分割された行列の非零要素の情報を用いて，通信を行いながら分解後の非零要素位置を計算し，非零要素をアクセスするためのインデックスリストを作成する。インデックスリストは，Lの各列の非零要素の行番号として持つ。

(4) コレスキー分解

コレスキー分解には，通信量が比較的少ない内積形式を採用した。内積形式では，2.2節で述べたように，1プロセッサが担当する部分木においては更新演算を通信なしに独立に行う。処理が進み，消去木を上に辿って分岐を通るごとに，協調して更新演算を行うプロセッサの数は2台，4台と増え，一番上の分岐から根までの間では，すべてのプロセッサが協調して更新演算を行う。

(5) 前進後退代入

消去木を用いて考えると，前進消去は消去木の葉の部分から根に向かって順に解を求めてゆく計算，後退代入は逆に根から葉に向かって順に解を求めてゆく計算となる。

前進消去においてもコレスキー分解と同様に，中心演算 $y_i = y_i - L_{ij}y_j$ においてjに関するループを内側に持ってくる内積形式と，iに関するループを内側に持ってくる外積形式とがあり，本ソルバでは，行列Lの非零要素を各列jの非零要素の行番号iのリストにより管理していることから，内側ループがiの外積形式が適している。しかし外積形式による計算をそのまま行うと，各ステップで求めた解yjを用いて，消去木上で祖先に当たる解をすべて更新することになる。これらの解は一般に他のプロセッサが担当しているため，各ステップjごとに通信が必要になってしまう。

そこで本ソルバでは，消去木上で祖先に当たる解への更新を直接には行わず，更新分を各プロセッサのワーク配列にため込んでおき，自分が担当

する部分木中の解の計算がすべて終了した時点で，他のプロセッサに送る。これにより，各部分木での計算は通信なしに実行できる。

一方，後退代入では中心演算が $y_i = y_i - L_{ji} y_j$ となるため，内側ループが j の内積形式が適している。内積形式では，各ステップ i での解の計算のため，消去木上で節点 i の祖先にあたる解が必要となるが，各プロセッサが担当する部分木中の解の計算を始める前に，それまでに求めた解をすべてそのプロセッサに転送しておくことにより，各部分木での計算は通信なしに実行できる。

4. RISC 向け最適化

前章で述べたスパースソルバの処理のうち，もっとも計算量の多いのはコレスキー分解であり，その中心演算は，図4に示すように小行列 A_k とベクトル $\mathbf{b}_k$ との積を計算してベクトル $\mathbf{c}_k$ から引く演算 $\mathbf{c}_k = \mathbf{c}_k - A_k \mathbf{b}_k$ である。ただし実際には，分割のブロック化に伴ってコレスキー分解の多段化を行っているため，$\mathbf{b}_k$, $\mathbf{c}_k$ は幅がブロックサイズと等しい行列 B_k, C_k となり，演算は行列乗算 $C_k = C_k - A_k B_k$ となる。

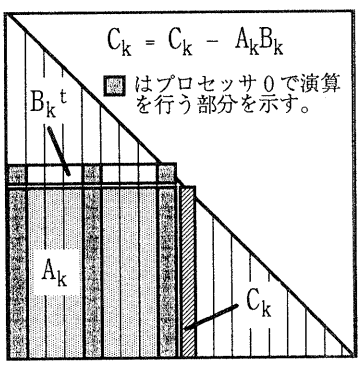


図4　コレスキー分解の中心演算

この計算において，A_k, B_k, C_k はすべて疎行列である。そのため，インデックスリストを介してアクセスを行わなくてはならず，リストベクトルのための機構を備えていない RISC プロセッサでは，密行列演算の場合に比べて単体性能が低下する。さらに，ロード／ストアを削減して性能を上げるためのループ展開についても，ループの各繰り返し毎に非零要素の位置や数が異なるため，通常のループ展開を機械的に適用すると，本来零である要素に対しても余分な演算を行ってしまうことになり，かえって演算効率が落ちる。

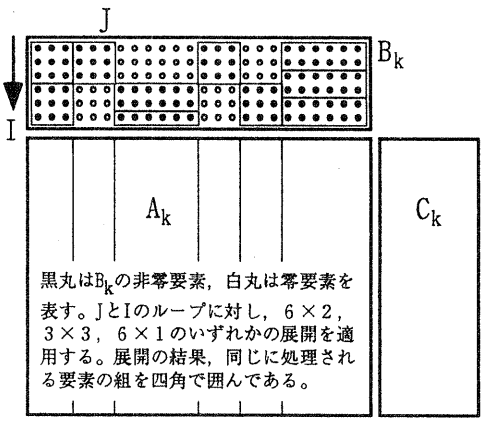


図5　局所的な非零構造に基づくループ展開

そこで本ソルバでは，行列の不規則な非零構造に合わせて局所的に最適なループ展開を行うことで，RISC プロセッサ上での性能向上を図った。3次元構造解析では行列の非零構造が3×3のブロック単位で現れる。そこで，図5に示すように行列 B_k の非零構造を6×2，3×3，6×1の3種類のブロックの直和に分割し，それぞれに応じた局所的なループ展開を適用すれば，演算量を増やさずにループ展開が行える。この3種類の展開はこの順に性能が高いため，6×2のブロックをできるだけ多く取り，残りを3×3と6×1に分割した。なお，この分割処理はシンボリック分解のフェーズを利用して行うため，同じ非零構造を持つ行列を複数回解く場合には，分割のためのオーバーヘッドは無視できる。

5. 性能評価

スパースソルバで計算の中心部分となるコレスキー分解の部分について，並列計算機 SR2201 上で次元数 N=70000 の3次元構造解析の実例題を用いて評価を行った。プロセッサ台数 p は1，2，4，8，16の5通りについて実行した。

前章で述べた RISC 向け最適化を行った場合と行わなかった場合について，性能評価結果を図に示す。最適化を行った場合，1プロセッサでの性能は約 100MFLOPS と，最適化なしの場合に比べて

約20%向上した。また，16プロセッサでの実行時には，1プロセッサに比べ，約10倍の加速が得られた。

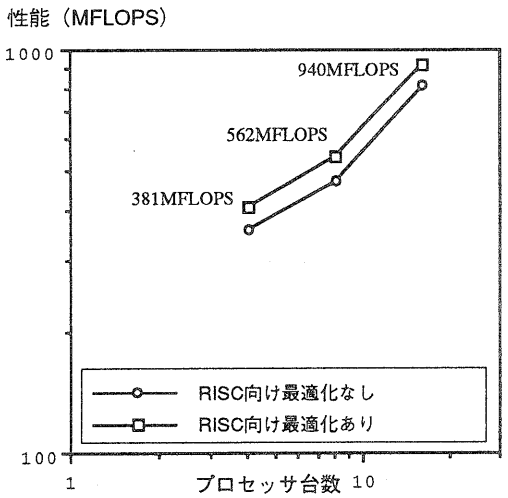


図6　3次元構造解析での評価

6．終わりに

本研究では，分散メモリ型並列機向けのスパースソルバを開発し，評価を行った。本ソルバは行列のオーダリングから，プロセッサ間への分割，シンボリック分解，コレスキー分解，前進後退代入までを一貫して行うプログラムであり，計算の中心となるコレスキー分解においては，局所的なループ展開により，RISC向けの最適化を行った。SR2201上の評価では，7万元の構造解析の問題で単体性能約100MFLOPS（ピークの33%），16プロセッサ時に約10倍の加速率を達成し，実用的な規模の問題で十分な性能が得られた。

今後の課題としては，シンボリック分解や前進後退代入を含めたスパースソルバ全体の最適化と性能評価，並列計算機SR8000向けの最適化と性能評価，CAPSS[9]，PARASOL[10]など他のソルバとの性能比較が挙げられる。

参考文献

[1] I. S. Duff, A. M. Erisman and J. K. Reid: "Direct Methods for Sparse Matrices", Oxford University Press, 1986.

[2] K. A. Gallivan et. al.: "Parallel algorithms for Matrix Computations", SIAM, 1990.

[3] T. A. Davis, P. Amestoy and I. S. Duff: "An Approximate Minimum Degree Ordering Algorithm", SIAM Journal on Matrix Analysis and Applications, Vol. 17, No. 4, pp. 886-905 (1996).

[4] E. G. Boman and B. Hendrickson: "A Multilevel Algorithm for Reducing the Envelope of Sparse Matrices", Technical Report SCCM-96-14, Stanford University (1996).

[5] G. Karypis and V. Kumar: "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs", Technical Report TR95-035, Department of Computer Science, University of Minnesota (1995).

[6] A. Pothen, H. Simon and K. Liou: "Partitioning Sparse Matrices with Eigenvectors of Graphs", SIAM ournal on Matrix Analysis and Applications, Vol. 11, pp. 430-452.

[7] A. Gupta and V. Kumar: "Parallel Algorithms for Forward and Backward Substitution in Direct Solution of Sparse Linear Systems", Proceedings of the Supercomputing '95, Dec. 1995.

[8] A. Gupta, G. Karypis and V. Kumar: "Highly Scalable Parallel Algorithms for Sparse Matrix Factorization", IEEE Transactions for Parallel and Distributed Systems, Vol. 8, No. 5, pp. 502-520 (1997).

[9] M. T. Heath and P. Raghavan: "Performance of a Fully Parallel Sparse Solver", The International Journal of Supercomputer Applications and High Performance Computing, Vol. 11, No. 1, pp. 49-64 (1997).

[10] P. Amestoy and I. S. Duff: "The PARASOL Project and the Multifrontal Parallel Solver for Sparse Systems", Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, March, 1999.