環状結合型超多重プロセッサシステムによる大次元連立一次方程式の並列計算†

金田 悠紀夫††

複雑な工学問題の解析手法である有限要素法を始めとして，大次元連立一次方程式を解く応用はきわめて多くその計算コストが大きな問題になっている．本論文は環状に結合された超多重プロセッサシステムを用いて並列計算を行い高速にしかも低コストで大次元連立一次方程式を解く計算方法について論じている．

ここでは連立一次方程式の代表的な解法であるガウス消去法，変形コレスキー法，ガウス・ザイデル法（反復法）を用いた場合の並列計算法について解析しており，計算量と比較してみた場合にほぼ無視できる程度のオーバヘッドをともなうデータを環状に結合されたプロセッサ間でフローさせることにより効率よく並列計算することが可能なことを示している．

本システムでは高速の浮動小数点演算機能を持つ高速のマイクロプロセッサと低廉で大容量のメモリユニット群の活用を想定しており，今後ますます発展する技術の有力な応用の1つとして先導的な役目をはたすことが期待される．

1. まえがき

最近の LSI 技術の進歩はめざましく，低廉で高速な（マイクロ）プロセッサや大容量 LSI のメモリユニットが出現してきた．同時にこれらプロセッサやメモリユニットを相互に結合した大規模な超多重プロセッサシステムの開発も現実的な問題になってきている．しかしながら多数のプロセッサやメモリユニットを相互にマトリックス形スイッチによって結合したり木状の構成に結合したりする場合に結合部がきわめて複雑になったり，プロセッサ，メモリユニット，結合機構，ケーブル等の物理的な配線や配置が大きな問題となり超多重プロセッサシステムの実現が難しいこと，またこのような超多重プロセッサシステムの応用面での研究も不足しており利用価値についても疑問があることがその実現へのネックとなっている．

筆者はかつて主メモリ共有形の並列処理システムでの連立一次方程式の並列解法アルゴリズムについて報告をしている[2]が，ここでは**図1**に示すように環状に結合され一列に並んだ超多重プロセッサシステムを用いて高速に大次元連立一次方程式の計算を行う手法を見い出したので報告する．大次元連立一次方程式は複雑な工学問題の解析手法として広く用いられている有限要素法を始めとする多くの数値計算において必要でその応用範囲はきわめて広くその波及効果もきわめて大きいものと考えられる．

2. 環状結合形超多重プロセッサシステムの構成

システムはm台の CPU が図1に示すように環状に結合され，各 CPU は各自対応するメモリユニット (MU) を両隣りの CPU の MU に対して一様にアクセス可能になっており，各 CPU はそれぞれ3つの MU に対してアクセスを行うことができる．逆に各 MU は3つの CPU からアクセスされるようになっている．

CPU 間の情報伝送はデータを MU 間を中継してフロー (flow) させていくことにより行われる．ここで想定しているプロセッサユニット (PU) としては高速のマイクロプロセッサまたはミニコンピュータで命令セットとしては汎用のものと大差ないが，特に浮動小

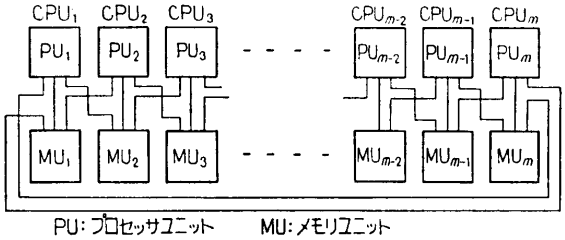

図 1 提案する環状結合形超多重プロセッサシステム
Fig. 1 Structure of the proposed circularly connected highly parallel processor system.

† Parallel Computation of Large Simultaneous Linear Equations under the Circulary Connected Highly Parallel Processor System by YUKIO KANEDA
(Dept. of Systems Engineering, Facuclty of Engineering, KOBE University).

†† 神戸大学工学部システム工学科

数点演算命令が豊富かつ高速で1演算時間が約1 μsec以下程度の計算能力を持ちアドレススペースは数Mバイト以上の広いものを想定している．また各 MU はアクセス時間数 100 nsec で容量は数十K～数M語を想定しており，容量上からのシステムボトルネックの発生は生じないことを想定している．また各CPU間の通信は隣接 CPU に対して相互に割込を行う機能を持つものと想定しており，各 CPU は自分自身のアドレススペース内に制御プログラム，連立一次方程式計算プログラム，係数行列の一部や未知数ベクトル等のデータを持ちいわゆる MIMD 方式で全 CPU は独立に並行して計算を進めることが可能となっている．なおプログラム格納用のメモリを各 CPU に対してローカルなものとすることも考えられる．ここで提案しているシステムと類似したものに TOPSTAR[7),8)] がある．多数台のマイクロプロセッサとメモリユニット (TOPSTAR では P モジュールと C モジュール) を結合して高い並列処理をねらっている点は共通しているが，TOPSTAR が，

1 P, C モジュールは n 隣接結合で n は 8～16.

2 汎用データフローコンピュータを指向している．

3 データフローコンピュータの論理的かつ汎用性ある制御方式の研究を重視している．

4 C モジュールに通信制御機能がある．

5 各 P モジュールには一般には異なったプロシージャが割付けられていて，処理が必要となったデータが token として送り込まれる．

一方環状結合形の本システムは，

1 両方向2隣接プロセッサ間のみメモリの共有をしており環状に結合されている．

2 行列形の数値計算の高速並列計算を指向している．

3 行列計算を強く意識した制御構造を追求している．

4 メモリモジュールに通信制御機能はない．

5 各 PU で実行されるプロシージャはほぼ同一で，同期を互いにとりながら行列データに対して演算する，という形式をとっている．

3. 環状結合形超多重プロセッサによる並列計算法

ここでは連立一次方程式の代表的な解法であるガウス消去法，変形コレスキー法，ガウス・ザイデル法を取り上げ並列計算法について述べる．

3.1 ガウス消去法による場合

解くべき n 元連立一次方程式を

$$Ax=b$$

とする．A は係数行列，$[a_{ij}]$，x は未知数ベクトル $[x_i]$，b は定数項ベクトル $[b_i]$ である．

ガウス消去の手順は前進消去と後退代入とに分けられるが，多量の計算時間を要するのは前進消去である．前進消去の手順は，i, j, k を整数変数として，

B0
- $k=1$ から $n-1$ まで1ずつ増加させながら B1 の計算をする．
  - 第 k 式の両辺を a_{kk} で割る．
  - B1
    - $i=k+1$ から n まで1ずつ増加させながら B2 を計算
    - B2
      - $j=k$ から n まで1ずつ増加させながら B3 を計算
      - B3[新 a_{ij}=旧 a_{ij}−(旧 a_{ik})a_{kj}
      - 新 b_i=旧 b_i−(旧 a_{ik})$\cdot b_k$

となる．

後退代入の手順は，

$x_n=b_n/a_{nn}$ を計算し，つぎに

$i=n-1$ から1まで1ずつ感じながら

$$x_i=(b_i-\sum_{j=i+1}^{n} a_{ij}x_i)/a_{ii}$$

を計算する．

MU_j には A の列ベクトルで mod $(i-1, m)+1=j$ $(i=1\sim n)$ となる a_i 列群のデータを格納し，a_n 列が格納された MU の右隣りの MU に b 列のデータを格納する．各 CPU_i は自分のメモリユニットである MU_i に格納されている列ベクトルに対するガウス消去計算を担当する．このように A の列ベクトルを各 MU に飛び飛びの順に格納するのはガウス消去過程で各 CPU の負荷ができる限り均衡することが望ましいためである．**図2**のように第 k 列の消去過程を進める場合を考える．もし各 CPU_j $(j=1\sim m)$ が a_k 列のデータに対してアクセス可能ならば MU_j に割当てられ格納された列群に対してシークエンシャルに B1 の消去操作をほかの CPU と独立に並行して実行することができる．したがってこの a_k 列を各 CPU が直接アクセスできるように MU 群に分配するために（データ）フローさせてやればよいことになる．a_k 列は MU mod $(k-1, m)+1$ に存在するが，この a_k 列データを**図3**に示すように1つおきの MU に両方向に向かって順次フローさせていき，全 CPU がアクセス可能

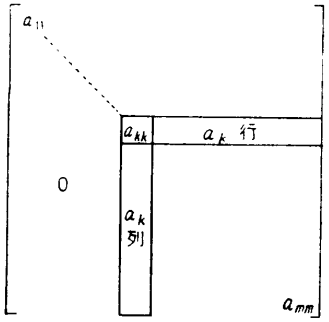

図 2 Gauss 消去における第 k 列目の消去
Fig. 2 Gaussian elimination of the k th column of the coefficient matrix.

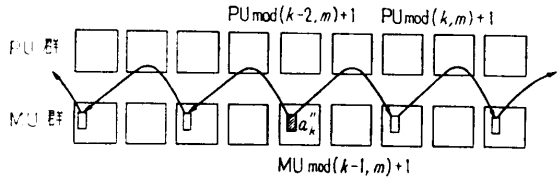

図 3 係数行列の列ベクトル a''_k 列のデータフロー
Fig. 3 Data flow of a''_k column vector of the coefficient matrix.

になるようにする．各 CPU はこのデータの中継と担当部分の列に対する Gauss 消去計算を行えばよく B^0 の計算において $k=1\sim n-1$ までシリアルに進めれば全 CPU はほぼ idle 状態になることなくガウス消去の前進消去の計算を進めていくことになる．

一方後退代入は前進消去がオーダ $O(n^3)$ の計算量であるのに対して $O(n^2)$ の計算量であり計算量は少ないが以下の手順で前進消去で得られた上三角行列の行および列の入替を行うと計算ずみの x_i の値を次々とほかの CPU にフローしてやることにより並列に計算できる．前進消去が終った段階では各 MU に得られた上三角行列の列ベクトルが分散して存在することになり，$x_n\sim x_1$ へと順に値を求めていく後退代入の手順を効率よく行うには困難な配置となっている．つまり x_i を求めるには $x_j, a_{ij}(j=n\sim i+1)$ の値を必要とし行方向の係数ベクトルに対して CPU がアクセスできることが必要となる．したがって各 CPU が行方向の係数ベクトルを持つように係数行列の入替を行う．MU_j に $\mathrm{mod}(i-1,m)+1=j(i=1\sim n)$ となる i 行ベクトル群が格納されるように，各 CPU は自分の MU 中の MU_j へ転送すべき行要素に MU_j への行先きラベルを付け CPU_j に近い方の隣接 PU を順に介して MU_j へフローしてやる．全行列の入替は全CPUが並行して行うが，入替が終了すると，$x_n, x_{n-1}\cdots x_1$ の順の未知数 x の値を求める計算を進めていく．この過程で求めた x_i の値は直ちに各 CPU にアクセス可能となるように両方向に向って 1 つおきの PU を介してフローする．各 CPU は未知数値がアクセス可能となるたびに後退代入の計算を進め最後に x_1 の値が求められたところで全計算が終了する．

3.2 変形コレスキー法を用いた場合

有限要素法では係数行列 A が正定値で対称であり変形コレスキー分解法がよく採用される．

ガウス消去法における前進消去と同様に連立一次方程式 $Ax=b$ を $A'x=b'$ に変形し A' を上三角行列にする手法である．

計算の手順は行列 A'' を導入して

$$a_{ij}=a_{ij},\ a''_{ij}=a'_{ij}/a'_{ii}(j=1,2,\cdots n)$$

を計算しておき，

$$\mathrm{B}_0\begin{cases}-k=2 \text{ から } n \text{ まで 1 ずつ増加させながら } \mathrm{B}_1 \text{ の計算をする．}\\ \mathrm{B}_1\begin{cases}-j=k \text{ から } n \text{ まで 1 ずつ増加させながら } \mathrm{B}_2 \text{ の計算をする．}\\ \mathrm{B}_2\begin{cases}-a'_{kj}=a_{kj}-\sum_{i=1}^{k-1}a''_{ik}a_{ij}\\ -a''_{kj}=a'_{kj}/a'_{kk}\end{cases}\\ -b'_k=b_k-\sum_{i=1}^{k-1}a''_{ik}b'_i\end{cases}\end{cases}$$

となる．

ガウス消去の場合と同一の方式で各 CPU に A の列ベクトル群を割当て格納して計算を進める．A', A'' の計算結果を入れる領域は A と同一領域を用いる．

図 4 に示すように a'_k 行の値を求める計算の場合を考える．a''_k 列のデータが各 CPU にとりアクセス可能となると全 $a'_{kj}(j=k\sim n)$ は全 CPU が並行して各 a_j 列のデータを用いて計算できる．したがって a''_k

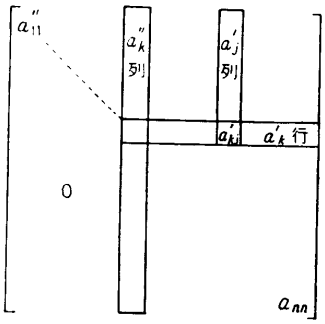

図 4 変形コレスキー法における係数行列の第 k 行の計算
Fig. 4 Computation of the k th row of coefficient matrix under modified choleski method.

列のデータを各 CPU がアクセスできるように1つおきの MU にガウス消去の場合と同様に両方向に向ってフローして計算を進めていけばよいことになる.

後退代入の手順はガウス消去と同一手順で並列計算することができる.

3.3 ガウスザイデル法を用いた場合

反復法の代表的なものにガウスザイデル法がありその公式は, $x^{(k)}{}_i$ を未知数 x_i の第 k 近似値とすると,

$$x_1^{(k+1)}=a_{11}^{-1}\{b_1-(a_{12}x_2^{(k)}+a_{13}x_3^{(k)}+\cdots+a_{1n}x_n^{(k)})\}$$
$$x_2^{(k+1)}=a_{22}^{-1}\{b_2-(a_{21}x_1^{(k+1)}+a_{23}x_3^{(k)}+\cdots+a_{2n}x_n^{(k)})\}$$
$$x_n^{(k+1)}=a_{nn}^{-1}\{b_n-(a_{n1}x^{(k+1)}+a_{n2}x_2^{(k+1)}\cdots+a_{nn-1}x_{n-1}^{(k+1)})\}$$

となる. 各 MU_i は A の j ($j=i+m\cdot k\leq n, k=0\sim[n/m]$ なる j) 行群の係数ベクトルを持ち CPU_i は $x_j^{(k+1)}$ 群の計算を担当するとする. ある j に対して $x_j^{(k+1)}$ の計算が終了すると CPU_i は直ちに両隣りの CPU に割込をかけて $x_j^{(k+1)}$ のほかの CPU へのフローを依頼する. 割込を受けた CPU は実行中の計算を中継し直ちに割込を受け, データを受取るとともに必要な場合には隣接の CPU を介してデータをさらにフローさせる. また受取ったデータは $x_j^{(k+1)}, j, k+1$ の値を3つ組として queue の形で記憶しておき, 中断していた計算を続行する. 各 CPU の担当している $x_j^{(k+1)}$ 群の計算では queue から順次 $x^{(k)}$ または $x^{(k+1)}$ の値を被乗数として取出し, 担当する $x_j^{(k+1)}$ 群の計算で対応する項について順次 (列単位で) 計算を行う手法をとる. 値の求まった $x_j^{(k+1)}$ は直ちに $j, k+1$ との3つ組の形で両方向の CPU を介してフローしていく. このような形で, 計算された近似値を CPU 間でフローさせながら $\boldsymbol{x}^{(k+1)}$ の反復計算を k についてシリアルに実行していくことによりガウスザイデル法による近似計算が並列に実行される.

4. むすび

3章で示したように本システムではデータを各 CPU間でフローさせながら計算を進めるのでそれにともなうオーバヘッドが問題となる. また CPU が計算を進めるのに必要なデータ群の伝搬を待つ必要が発生したときの待ち時間による遅れが問題となる. したがってまず割込処理を含めたデータフロー機構の高速化 (ハードウェア化またはファームウェア化) と各CPUのロードバランスがきわめて重要となる.

係数行列が密行列の場合には非対称行列に対するガウス消去に必要な演算回数は $O(n^3/3)$ となり, 正定値対称行列に対する変形コレスキー法では $O(n^3/6)$ の演算回数となる. またガウスザイデル法では1回の反復計算に $O(n^2)$ の演算回数となる. 一方計算を進めるため必要なデータフロー量(単一データを1つ先の CPU へ中継するための操作) はガウス消去法および変形コレスキー法では $O(n^2\cdot m/4)$ となる. またガウス・ザイデル法においては1回の繰返し計算を行うために必要なデータフローの量は $O(u\cdot m/2)$ となる. したがって $n/m\gg 1$ なる場合はデータフローによるオーバヘッドはほぼ無視できると考えられる. 次に実用上重要な帯行列の場合について検討してみる. 帯幅を $2l+1$ ($l\ll n/2$ なる整数) とすると, ガウス消去の場合は $O(nl^2)$ 変形コレスキー法では $O(nl^2/2)$ ガウス・ザイデル法では1回の反復に $O(2nl)$ 回の演算回数になる. 今 $l>m$ とするとガウス消去, 変形コレスキー法では $O(nlm/2)$ 回のデータフローが必要となり, またガウス・ザイデル法では $O(m\cdot n/2)$ 回のデータフローとなる. (演算回数)/(データフロー回数)$\gg 1$ の場合は計算の効率が高くなることが期待できるが, $\gg 1$ の条件が成立しない場合には単位データ当りのデータフローによるオーバヘッドが効率に大きな影響を与えるものと考えられる. このように文献 2) での結論でも述べているように問題が大形化し, n, l の値が m より十分に大きい場合は高い効率が得られることが予想される.

本システムを実現するためには詳細なハードウェア構成, CPU 間通信方式の確立, 各 CPU で実行される計算プログラムおよび制御プログラムの設計, システム効率の定量評価等の検討を十分に加える必要があるが, 本システムは見方を変えればデータフロー形計算機システムの典型的な1例であり, 高速の数値演算機能を持ったマイクロプロセッサと大容量低廉なメモリの実用が可能となればきわめて実現性が高く, かつ広範な応用を持つシステムになると考えられ現在その実現について検討を進めている.

謝辞 本研究を進めるに当り有限要素計算に対する有益なご助言と計算方式に関するご検討をいただいた神戸大学工学部システム工学科教授瀬口靖幸博士に深謝します.

参考文献

1) 金田: 並列処理システムの動向, 電子技術総合

研究所調査報告，第174号（昭48年2月）．
2) 金田：並列処理システムによる連立一次方程式と楕円形偏微分方程式の数値計算法，情報処理，Vol. 16, No. 2（昭50年2月）．
3) 金田：並列処理システムによる線形計画計算と実対称行列の三重対角化計算，情報処理，Vol. 19, No. 1（昭53年1月）．
4) 戸川：マトリクスの数値計算，オーム社．
5) Dennis, J. B. et al.: A Preliminary Architecture for Basic Data Flow Processor, Proc. 2nd Annual Symposium on Computer Architecture, (Jan. 1975).
6) Rumbaugh, J. E.: A Data Flow Multiprocessor, IEEE Trans. on Computer, Vol. C-26 (Feb. 1977).
7) 元岡，喜連川，鈴木：A High Level Data Flow Machine～hardware～，情報処理学会第20回全国大会講演論文集（昭54）．
8) 栗原，鈴木，元岡：High Level Data Flow Machine (TOPSTAR) のシステムプログラム，電子通信学会電子計算機研究会資料，EC 79-56（昭55）．

（昭和55年3月10日受付）
（昭和55年6月19日採録）