計算機アーキテクチャ 54－1
（1984. 7. 13）

有限要素法専用並列計算のための立方格子状計算機

大澤 暁　齊藤千鶴子　天野英晴　横田隆史　相磯秀夫
（慶應義塾大学理工学部）

第1章 導入

物理現象等を解析する場合、一般には偏微分方程式が用いられるが、これを数学的に解くことは一部の例外を除いて困難である。そこで現在では有限要素法等を用いて離散化し、大型計算機により解かれている。しかし対象とする領域が大きくなったり精度をあげるために領域を密に分割すると、莫大な計算量を必要とし、必ずしもユーザの要求を満たしているとはいえない。このような場合並列計算機による並列処理が適しており、CRAY-1[1] 等のベクトル計算機、ILLLIAC-IV [2] 等の SIMD 計算機、SMS[3]等の MIMD 計算機等の並列計算機も出現している。

2次元領域における偏微分方程式問題を並列計算機で解く場合、

1． 領域を図1.1 のように一様に正方格子状に分割、
2． 三角形要素を用いて離散化、
3． 上下左右の4 Processing Units (PUs) と専用交信路を持った正方格子状計算機を用いて、各 PU に領域の節点を割りあて、反復法を用いて解く

と有効であると一般にいわれている。さらに領域の分割法や境界条件等の処理を考えると PU ごとの計算が異なるため、 MIMD 型の計算機が有利である。例えば、筑波大学の PAX-128[4] 、NASAの FEMマシン[5] がこれにあたる。

この時効率良く並列計算するためには、次の点を考慮する必要がある。

◇ 各 PU は MIMD で動作するが、できるだけ処理が一様であること。

◇ 各 PU の担当する節点数を一様にすること。もしこれにばらつきがあると、全計算時間が最も負荷の重い PU に左右されるため極端な効率低下を招く。なぜならば、すべての節点についてその計算回数はほとんど同じであるからである。

ところでユーザが有限要素法を使う場合を考えると、領域の一部を密に分割して計算の精度を上げたいといった要求が高い。もともと有限要素法は、領域をどのように分割しても離散化可能であるという特徴を持っているので、領域分割において粗密をつけても解くことができる。そこで正方格子状計算機でこのような問題を扱う場合、次のような方法がとられる。

1． 図1.2 のように領域を一様なブロックに分割し、ひとつのブロックに含まれるすべての点を 1 PU に割りあてる。
2． 図1.3 のように領域を密度の一様な部分で分割し、別々に計算を行う Sub-structure法を採用する。

ところが領域分割に粗密をつける場合、どちらの方法をとってもあまり効率が良くない。すなわち1の方法を用いた場合、各 PU の担当する節点数にばらつきが生じるので前述のような効率低下を招く。また2の場合には効率低下は招かないが、Sub-structure ごとの境界に対する考慮や計算結果の後処理が必要となる。

このように粗密のある分割領域を扱う場合、正方格子状計算機では効率よく並列計算することができない。

そこで、正方格子状接続をベースにした新しいアーキテクチャを考える。この時、ユーザが任意に領域分割を行った場合、その分割方法は無限にあり、そのすべてに効率良く対応できるアーキテクチャを考えるこ

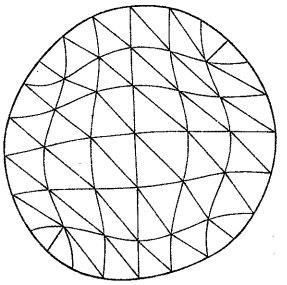


図1.1 正方格子状分割

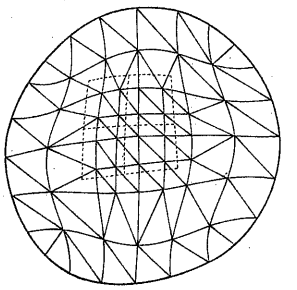


図1.2 ブロック分割による PU の割りあて

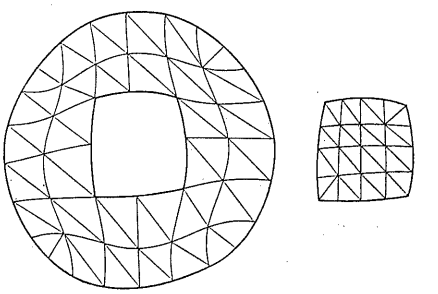


図1.3 Sub-structure 法

とは困難であると考えられるので、ここでは次のような方針をとることにする。

◇　格子状の専用交信路を基本としながらも、密な部分（計算負荷の重い部分）には、粗な部分から PU を論理的にバス等を用いて移動できるような構造とする。

◇　領域の分割のしかたに粗密をつけることは許すが、アーキテクチャにマッピングしやすいように正方格子状分割をベースに、ある程度の制限を設ける。但し、この制限はユーザ側の要求をそこなわない程度のものとする。

そこで先ず粗密のある領域分割法を提案し、次にそれを効率良くマッピングできるアーキテクチャとして立方格子状 PU アレイを提案する。このアレイは正方格子状 PU アレイを層状に並べて、新しくできた次元軸方向にある交信路を用いて順次接続したものである。

本報告では以下２章において粗密のある領域分割法と立方格子状計算機を提案し、３章で領域を PU アレイにマッピングする方法を示す。この場合、実際にデータを交信しなければならない PU が物理的に離れるので、４章においては有限要素法の並列計算の特徴を活かした効果的なデータのルーチング法を示す。さらに５章では GPSS を用いて簡単な評価を行い、その実現性について検討する。

第２章　立方格子状計算機

２．１　粗密のある領域分割法

対象とする領域を、できるだけ図1.1 に沿って密部も正方格子状に分割するために、例えば４倍密に分割する場合は図2.1 のように、周囲に比べてトポロジィ的に縦線および横線をふやす方法をとる。図2.1 においては、粗と密の部分の境界を除いて正方格子状分割が守られている。このような方法をとることにより２のべき乗に密に分割することが可能であり、ユーザ側にこのような制限を課しても問題はないと考えられる。

２．２　立方格子状計算機

図2.1 のように分割された領域で効率良く並列計算するためには、図2.2 のように中央部は周辺部より約４倍多い PU が必要である。図2.2 のような PU 構成は、ユーザが任意に指定する倍密部でその密度に応じて生ずる。そこで、倍密部における PU をグループ分けする。図では４倍密部においてグループ２，３，４があり、これを含めてすべてのグループの PU が正方格子状に並んでいる（図2.3 ）。そこで正方格子状 PU アレイをあらかじめ複数枚用意し、その各々にグループ１，２，３，４を割りあてる。ところで有限要素法では反復計算の際に、分割された領域の各節点は、直接接続されている隣接節点との間でデータの交信が必要である。従ってグループ分けされた PU アレイ同志でデータの交信が必要となるから、図2.2 において点線で囲まれた４個の PU を基準に図2.4 のように重ね合わせる。これにより、新しい次元軸方向の交信路を用いて効率良くデータの交信を行うことができる。これが立方格子状接続の PU アレイである（図2.5 ）。この PU アレイは各次元においてサイクリックな構造をしており、便宜上新しい次元軸をＺ軸、他の２軸をＸ軸、Ｙ軸と呼ぶことにする。またこのアーキテクチャを用いれば、３次元領域を立方格子状に分割し四面体要素を用いて３次元問題をも解くことができるという利点もある。

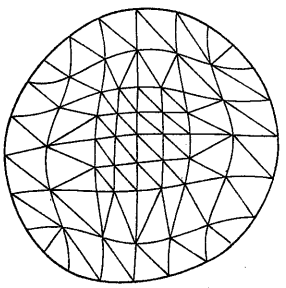


図2.1 粗密のある領域分割法

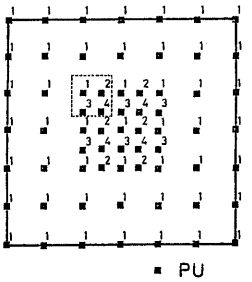


図2.2 必要な PU

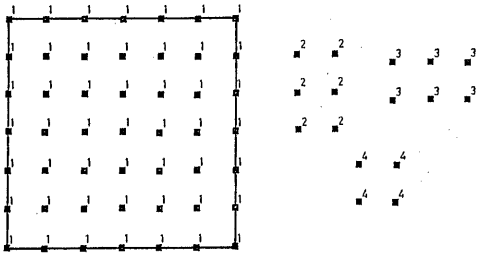


図2.3 PUのグループ分け

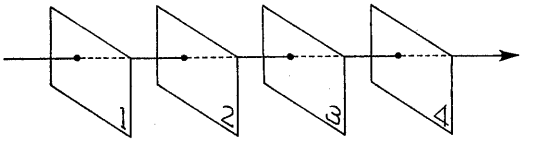


図2.4 次元軸の追加

第3章　マッピング法

３．１　マッピングの概要

２次元領域を３次元 PU アレイにマッピングするために図3.1 に示すように表・裏２枚を１組として互い違いに分割し、ちょうどマットレスを広げるように左右に伸ばす。これにより表・裏２層となった２次元正方格子状PUアレイができあがる。

できあがった２次元正方格子状 PU アレイの表側にグループ１を割りあて、裏側はグループ２以降専用とする。例えば図2.1 の場合は図3.2 のような割りあてとなる。グループ３，４の PU アレイは援助 PU アレイと呼ばれる。立方格子状 PU アレイを用いたのは、Ｚ軸方向にある交信路を用いて密部へ論理的に PU を移動することができるからである。

すべてのマッピングが終了した時点で裏側のアレイに使われていない PU が存在する場合、その PU は計算を行わないことになるので、この PU を活かしてその部分は対応する表側の PU と共に２倍密であると考えることにする。

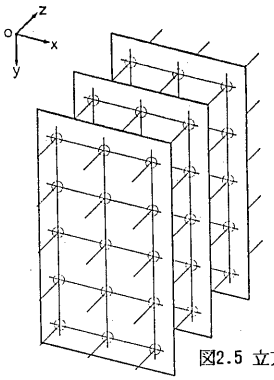


図2.5 立方格子状計算機

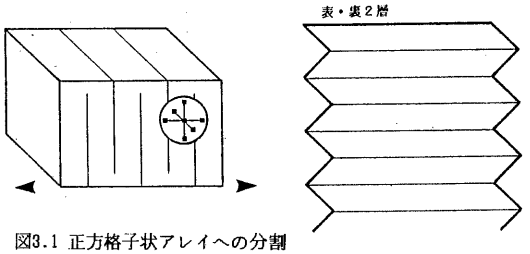


図3.1 正方格子状アレイへの分割

３．２　アルゴリズム

ここでは、分割領域内に存在する節点数に比べ十分なサイズの立方格子状 PU アレイが存在し、“１節点１ PU ”のマッピングができるものとする。

《マッピング・アルゴリズム》

(1)　一番基本的な方法で、領域の上端から順次表側の PU に割りあててゆく方法である（図3.3 ）。

(2)　(1) を採用した場合、図3.4 のように倍密部が２枚のPUアレイにまたがってしまう（以後これをクロッシングと呼ぶ）ことがある。図3.4 においては、倍密部ａ，ｂ共に援助 PU アレイを必要とする。これらのアレイはもとのアレイと物理的に離れてしまう可能性が強く、反復計算においては遠隔 PU アレイ同志でのデータ交信が増える。従って、一点鎖線で囲まれた部分の負荷が重くなってしまう。

このクロッシングを防ぐために、領域中の一番大きな倍密部を１枚の PU アレイの上端から割りあてるようにする方法である（図3.5 ）。

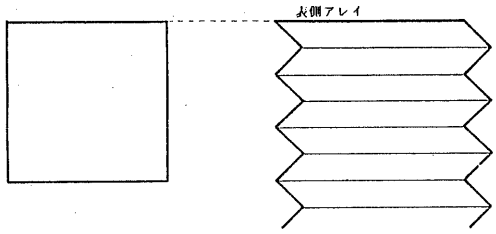
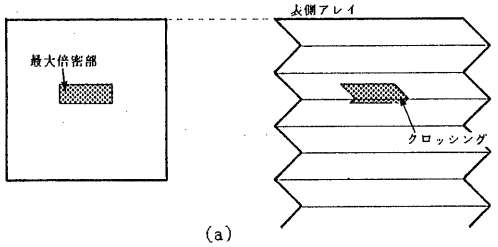
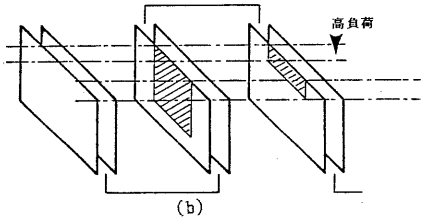


図3.3 方法(1)



(a)



(b)
図3.4 クロッシング

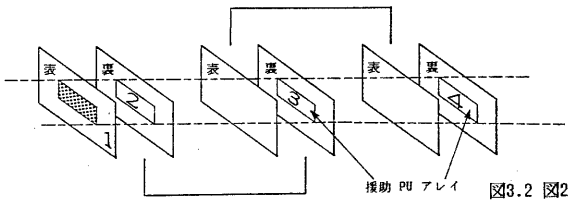


図3.2 図2.1 の場合の割りあて例

(3)　一般に倍密部は領域内に複数個存在するから、最大倍密部を１枚の PU アレイのどの場所に割りあてるかによって別の倍密部におけるクロッシングの様子が変わってくる（図3.6 ）。

そこで(3) は、最大倍密部を１枚の PU アレイに割りあてるにあたり、すべての倍密部を考えて、総クロッシング数が一番小さくなるように考慮する方法である。

《援助 PU アレイ決定法》

倍密部においてさらに必要なPU（援助PU）を決定する方法である。但し、２倍密では援助 PU は必要でない。（例えば図3.2 の場合、グループ３，４のアレイを決定する方法である。）

(a)　この方法は図3.7 のように次の表・裏２層となった PU アレイの裏側から順次援助 PU アレイとして決定してゆく方法である。

(b)　もし PU サイズガ大きくて、表・裏両方共使用していない PU アレイ層があった場合、先ずその裏側のアレイから順次援助 PU アレイとして決定してゆく。このようなアレイが最初からなかった場合、またはなくなった場合、方法(a) をとるというものである。

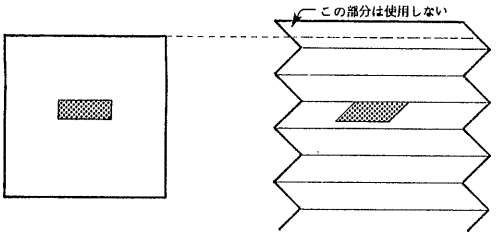


図3.5 方法(2)

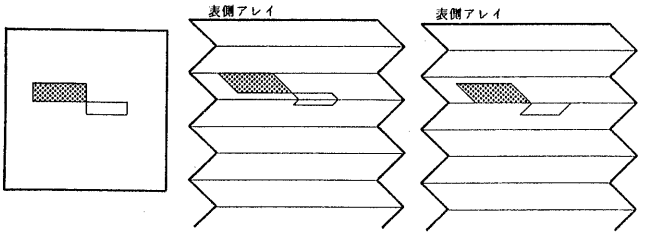


図3.6 方法(3)

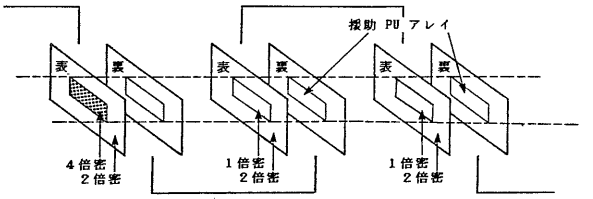


図3.7 方法(a)

３．３　評価

３．２において、マッピング法３種類、援助 PU 決定法２種類、合計６種類の方法を提案しているが、これらは倍密部の数、位置、その密度等により交信路の状況がかなり異ってくる。従ってこれらを定量的に評価するのはむずかしい。そこで本節では次の方針により評価を行う。

◇　いろいろなケースで交信路の状況を調べ、どの方法が良いかを比較する。

◇　そのうえで最良の方法を用い、条件を同じにして、クロッシングがある場合とない場合での交信路の状況を調べる。

倍密は２倍密、４倍密、８倍密とし、データ交信には４章で提案されるルーチング・アルゴリズムを用いた。

《基準》

評価の基準として次のものを求める。

◇　Line conflict :　反復計算において１反復あたり１交信路を通るパケットの数の平均値

◇　Node distance :　送り手 PU と送り先 PU との間に存在する交信路の数の平均値

algorithm		1	2	3
3-b	Line conflict Node distance	0.92 1.34	1.08 1.68	1.88 2.05
3-a	Line conflict Node distance	0.92 1.34	1.03 1.59	1.88 2.07
2-b	Line conflict Node distance	0.92 1.34	1.08 1.71	1.90 2.11
2-a	Line conflict Node distance	0.92 1.34	1.03 1.63	1.90 2.12
1-b	Line conflict Node distance	0.92 1.34	1.03 1.59	不可能
1-a	Line conflict Node distance	0.92 1.34	1.11 1.70	不可能
Region size		10*30	10*30	16*96
PU size		10*8*16	10*8*16	16*16*16
Density		なし	8倍密１ヵ所 4倍密２ヵ所	8倍密１ヵ所 4倍密２ヵ所 2倍密２ヵ所

単位　Line conflict : packets/line

Node distance : lines

表3.1 Line conflict と Node distance

crossing	algorithm	Line conflict	Node distance
あり	3-b	2.30	5.08
あり	3-a	2.00	4.78
なし	3-b	0.65	1.64
なし	3-a	0.51	1.22

Region size : 10*16
PU size : 10*8*32
Density : 8倍密１ヵ所

表3.2 クロッシングによる差異

《結果》

表3.1 に３つのケースにおける Line conflictおよび Node distanceを示す。表中、algorithm 欄の数字はマッピング・アルゴリズムの番号、ハイフンの後の記号は援助 PU アレイ決定法の記号を表す。ケース３で〝不可能〟とあるのはクロッシングが生じた結果、援助 PU アレイが不足しマッピングできなかったことを示している。この表からマッピング法については方法３が、また援助 PU 決定法については大差のないことがわかる。

表3.2 にクロッシングがある場合とない場合についての結果を示す。この結果、クロッシングがおきると、

◇ Ｚ軸方向のデータ交信が増える結果、かなり交信路が混雑する

◇ 相手 PU が遠隔になり、Nonode distance が大きくなる

ということがわかる。

以上から、

“クロッシングがおきないようにマッピングすることが肝要である”

ということがわかった。

第4章 ルーチング・アルゴリズム

４．１ ルーチング・アルゴリズムの概要

これまでの章で、有限要素法専用の立方格子状並列計算機およびそのマッピング・アルゴリズムが提案された。これにより、粗密のある領域を解く場合、 PU を粗な部分から密な部分へ論理的に移動させるので、直接接続していない PU との頻繁なデータ交信が必要になる。そこで、これを効率良く行うためのルーチング・アルゴリズムを提案する。

有限要素法を立方格子状計算機で解く場合、各 PU は新しい反復値 $u^{(n+1)}$ を計算するごとにこれを必要とする PU にブロードキャストする。このとき次のような特徴がある。

◇ データ交信の送り先 PU は 5～8 個程度

◇ 頻繁なデータ交信

◇ データは少量である。すなわち浮動小数点データが１個

◇ 各 PU はデータ駆動型の処理が可能

これらの特徴から、複数の送り先 PU にできるだけ同じパケットでデータを送り、転送途中でどうしても同じパケットで転送できなくなったときにパケットの“COPY”を行う方法をとる。

《パケットの内容》

横幅が 8 bit, 16 bitの２通りのパケットについて、実際に送られるパケットの内容を図4.1 に示す。また図4.1 の語句は次の通りである。

◇ TYPE ： パケットの進む方向

◇ NDEST ： 送り先 PU の数

◇ DESTINATION ADDRESS + TAG ： 送り先 PU の相対アドレス＋送り先 PU が存在する象限

◇ SOURCE ADDRESS ： 送り手 PU の絶対アドレス

◇ DATA ： 実際に送りたいデータ

各パケットのDESTINATION ADDRESS とは、送り先 PU の相対アドレスのＸ，Ｙ，Ｚ座標を象限ごとに次のように変換したものである。

第１象限 (X,Y,Z) → (X,Y,Z) TAG = 1

第２象限 (X,Y,Z) → (Y,-X,Z) TAG = 2

第３象限 (X,Y,Z) → (-X,-Y,Z) TAG = 3

第４象限 (X,Y,Z) → (-Y,X,Z) TAG = 4

そしてまた、

Ｆグループ (X,Y,Z) → (X,Y,Z)

Ｂグループ (X,Y,Z) → (X,Y,-Z)

このようにすると、どの象限でのルーチングも変換された相対座標を用いて同様なアルゴリズムで行うことができる。しかし最初にこのような変換を行ってしまうと、パケット内の DESTINATION ADDRESSがどの象

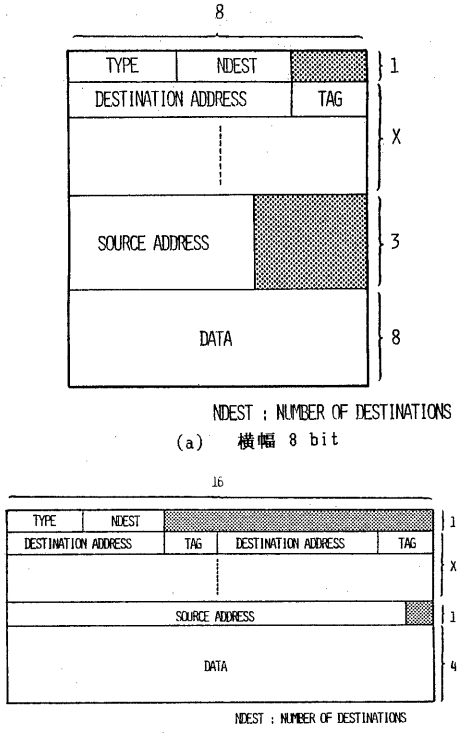


図4.1 パケットの内容

限のものであったかわからないので、その情報を保持するのに TAGが必要である。

《パケット・ルーチング》

パケットを受け取った PU はまずパケット内の DESTINATION ADDRESS (送り先 PU の相対アドレス) を見る。そしてもしその中に自分の相対アドレスすなわち (0,0,0) があればそのパケットのデータを取込む。ここでパケット内の DESTINATION ADDRESSは、常にそのパケットの処理をしている PU に対する相対アドレスとなっている。その後、パケット内の情報から次に直接接続しているどの PU にそのパケットを送ればよいか計算し、送る方向により DESTINATION ADDRESSの相当する座標から1を減じる。そしてこのパケットをその方向に送る。このとき、2個以上の PU にパケットを送らなくてはならない場合パケットの“COPY”が行われ、相当する方向の PU にパケットを送る。またもし DESTINATION ADDRESSに自分のアドレスがなければ、データは取込まず同様の方法でパケットを送る。

《ルーチングの方法》

パケット・ルーチングでは交信路が混雑すると、パケットの転送に時間がかかり、処理時間のネックとなる。そこで、交信路でのパケットの混雑をできるだけ少なくし、また無駄な経路を通らないようにしなければならない。そこで、6方向 (Forward, Backward, East , North , West, South) に別々にパケットを送り、必要が生じたときにパケットの“COPY”を行い分岐する方法をとる。

立方格子状計算機のX，Y，Z軸のうちどれを優先軸にするかによってルーチングの経路が異る。そこで、ここではZ軸を優先軸とした場合についてのみ例を挙げて説明する。また他の軸を優先軸とする場合は、サイクリックにX，Y，Zを置き換えて同様に考えればよい。

まず送り手 PU で、送り先 PU を相対座標(X,Y,Z)によって方向別に6つのグループ(F,B,E,N,W,S) に分け、各グループごとにパケットを生成する。この分類方法は次の通りである。

Z ＞ 0 … F

Z ＜ 0 … B

Z ＝ 0

X ＞ 0 , Y ≧ 0 (第1象限) … E

X ≦ 0 , Y ＞ 0 (第2象限) … N

X ＜ 0 , Y ≦ 0 (第3象限) … W

X ≧ 0 , Y ＜ 0 (第4象限) … S

次に実際のアルゴリズムのフローチャートを図4.2 ,4.3 に示す。

《例》

- SOURCE ADDRESS : (0,0,0)
- DESTINATION ADDRESS : (1,0,0) (1,2,0) (2,1,0) (-1,1,0) (1,-1,1) (1,-1,-1) (1,0,-1)

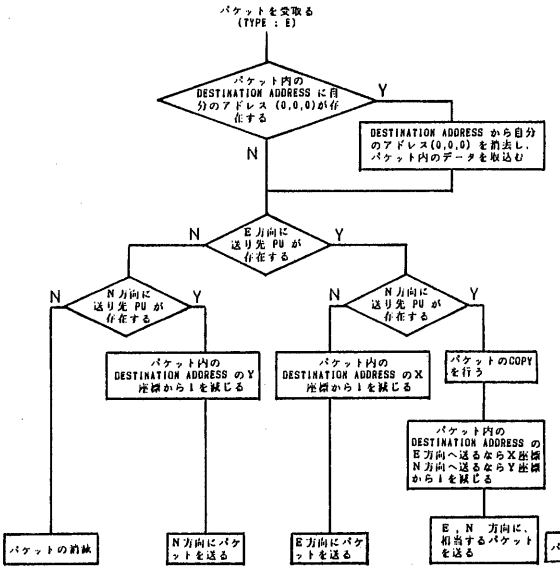


図4.2 Eグループのルーチング・アルゴリズム

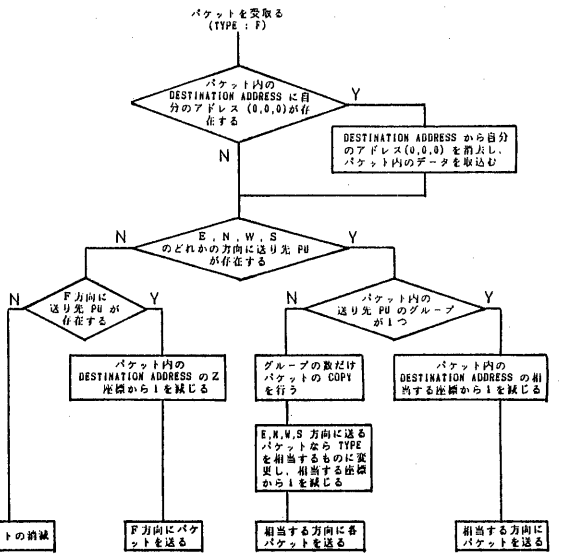


図4.3 Fグループのルーチング・アルゴリズム

この例についてのグループ分けは、

F (Z > 0)　　(1,-1,1)

B (Z < 0)　　(1,-1,-1) (1,0,-1)

E , N , W , S (Z = 0)　その他

となり、E,N,W,S の分け方については図4.4 に示す。この図4.4 を見ると、第２象限は第１象限を90° 回転させたもの、第３象限は 180°、第４象限は 270° 回転させたものである。そこで第１象限のルーチングがわかれば他の象限ついても同様に考えることができるので、第１象限のルーチングについてのみ説明する（図4.5)。

第1.象限ではパケットはまずE方向に送られ、(0,0,0) から (1,0,0)に達する。ここで、この PU は送り手 PU なのでデータが取込まれ、次にこれ以上E方向に送り先 PU が存在しないのでパケットはN方向に送られ、(1,0,0) から (1,1,0)に達する。ここではデータは取込まれないが、E , N両方向に送り先 PU が存在するのでパケットの COPY が行われ、生成された２個のパケットが相当する方向に送られる。これらは (1,1,0)から (2,1,0)および (1,2,0)に達する。これらの２個の PU は両方とも送り先 PU なのでデータが取込まれる。またどちらの PU においても、E , N両方向にこれ以上送り先 PU が存在しないので、ここでパケットは消滅する。

またF方向に送られたパケットは (0,0,0)から (1,-1,1) に達する。そしてこれ以上F方向に送り先 PU が存在しないので、ここでこのパケットは第４象限のパケットとなり、第１象限と同様のルーチングが行われる。

そしてB方向に送られたパケットは (0,0,0)から (1,-1,1) に達する。ここでこれ以上B方向には送り先 PU がないのが、このとき第１象限と第４象限の２つの送り先 PU が存在する。そこでパケットの COPY が行われ、それぞれの象限のパケットとして同様のルーチングが行われる。

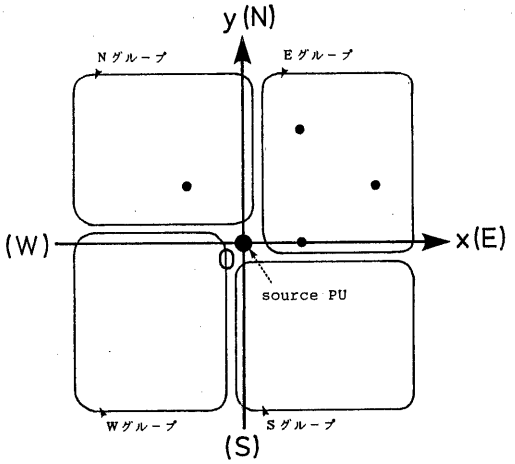


図4.4　E , N , W , Sのグループ分け

４．２　ルーチング・アルゴリズムの評価

提案した“COPY”の方法による Line conflictの減少を優先軸ごとに定量的に調べるために、すべての送り先 PU に対して別々のパケットでデータを送る“NO-COPY ”の方法との比較を行い、最適な優先軸を決定する。またパケット長の比較も行い、これらを考え合わせて“COPY”による効率の上昇を調べる。

《Line conflict 》

定義より、データ交信を円滑に行うためには Line conflictは小さい方がよい。

ここでは２倍密と８倍密の場合について、倍密部とそれへの援助などのために影響を受ける部分すべてについての Line conflictを調べ、それらの平均を求める（表4.1(a)）。表中の語句は次の通りである。

◇　Density : 倍密部の密度

◇　Method　: “COPY” or “NO-COPY ”

◇　*-axis　: 優先軸

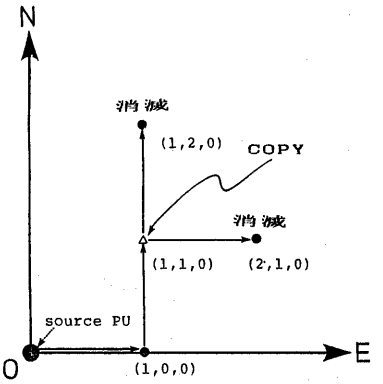


図4.5　Eグループ（第１象限）のルーチング

Density	Method	X-axis	Y-axis	Z-axis
2	COPY	2.21	2.36	1.94
	NO-COPY	2.93	2.93	2.93
8	COPY	3.25	3.15	2.51
	NO-COPY	4.29	4.29	4.29

(a)　Packets / Line

DENSITY	X-axis	Y-axis	Z-axis
2	0.754	0.805	0.662
8	0.758	0.734	0.585

(b)　COPY / NO-COPY

表4.1　Line conflict

表4.1(a)よりZ軸優先の場合が最も Line conflict が小さいことがわかる。

さらに、“COPY”と“NO-COPY ”の比をとった結果が表4.1(b)である。

《パケット長》

パケット長が長いとそれだけ交信路を専有する時間が長くなる。そこで、効率を考える上で1つのパケットの平均パケット長も考慮する必要がある。

図4.2 において、“NO-COPY ”では送り先 PU が1個なので X=1であるが、“COPY”では X=1又はそれ以上である。そこで“COPY”の平均パケット長を求め“NO-COPY ”との比をとった結果が表4.2 である。

表4.2 からわかるように、“COPY”の方が“NO-COPY ”に比べて平均パケット長が長くなる。

《効率》

“COPY”によってどれだけ効率が上昇するかを検討するには、 Line conflictとパケット長を考え合わせなくてはならない。そこで、効率を以下の式によって求めその結果を表4.3 に示す。

（効率）＝1／

{(Line conflict の比) ＊ (パケット長の比)}

表4.3 より、“COPY”によって効率がかなり上昇することがわかる。

第5章　アーキテクチャの実現性

5．1　PUモデル

各 PU は反復計算のような演算処理に加えて、4章で述べたルーチング処理を行う必要がある。そこでルーチング処理によって実際の処理が妨害されるのを避けるために、各 PU はルーチングのみを行う“ルーチング・プロセッサ”と、実際に計算を行う“演算プロセッサ”とに分ける（図5.1)。各 PU に送られたパケットはまずルーチング・プロセッサでルーチング処理が行われる。そしてルーチング・プロセッサでデータが取込まれたときのみ、そのデータが演算プロセッサに送られデータ駆動で反復計算が行われる。

5．2　アーキテクチャの検討

これまでに提案した立方格子状計算機のアーキテクチャの実現性を示すために、 GPSS によるシミュレーションを行う。またその結果から最適なアーキテクチャの検討を行う。

《基準》

まず評価の基準として次のものを求める。

◇　全実行時間　：稼動しているすべての PU が少なくとも10回の反復計算を行うのにかかる時間

◇　有効稼動率　：演算プロセッサが有効な反復計算を行っている時間の全実行時間に対する割合

◇　平均 Queue長：ルーチング・プロセッサに処理されるのを待っているパケットの数の平均

《パラメータ》

上記の基準を求めるためのパラメータとして、次のものを考える。

◇　HOP　：パケット1個を1交信路転送するのに要する時間

◇　RTG　：パケット1個のルーチング処理に要する時間

◇　EXEC：データ1個の演算時間

《結果》

“HOP ：RTG ”を一定にした場合、グラフ5.1(a)より EXEC が増加するほど全実行時間も増加することがわかる。ただし、 EXEC が変化してもあまり大きな変化はない。またグラフ5.2(a)より EXEC が増加するほどルーチング PU の平均 Queue長が急激に減少することがわかる。そしてグラフ5.3(a)より EXEC が増加するほど有効稼動率が上昇し、しだいに一定値に近づくことがわかる。

次に、“HOP ：EXEC”を一定にした場合、グラフ5.1(b)より RTGが増加すると全実行時間も増加することがわかる。またグラフ5.2(b)より RTGが増加するほどルーチング PU の平均 Queue長が急激に増加することがわかる。そしてグラフ5.3(b)より RTGがある程度以上になると有効稼動率は急激に減少する。

Density	パケット長
2	1.04
8	1.06

表4.2　パケット長

Density	効率
2	1.45
8	1.61

表4.3　効率

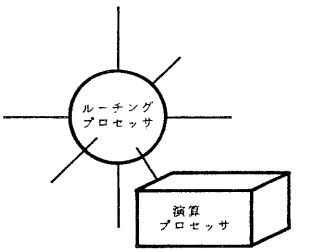
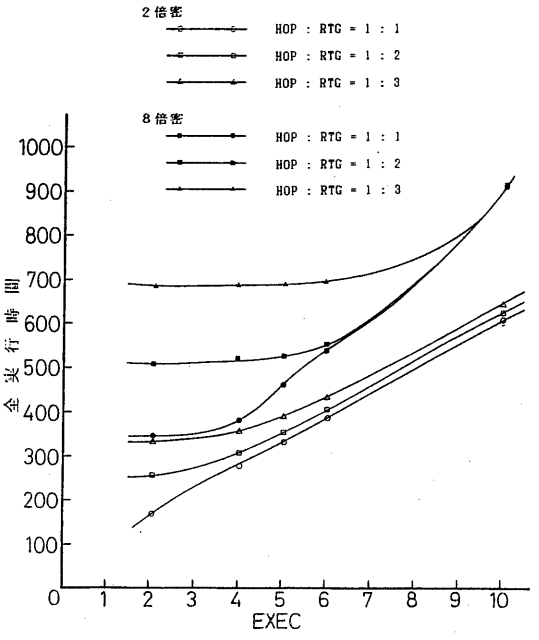
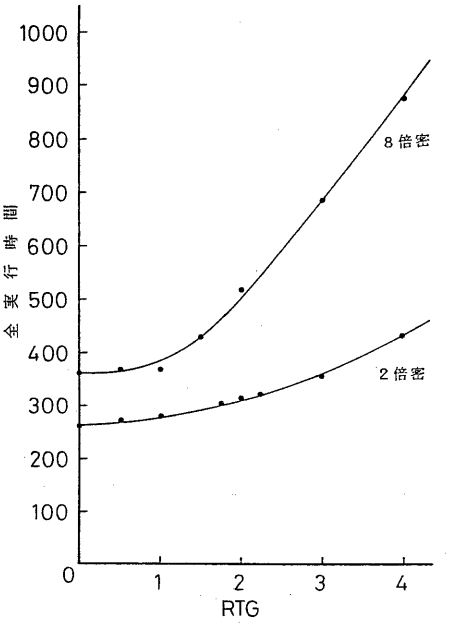


図5.1　PUモデル

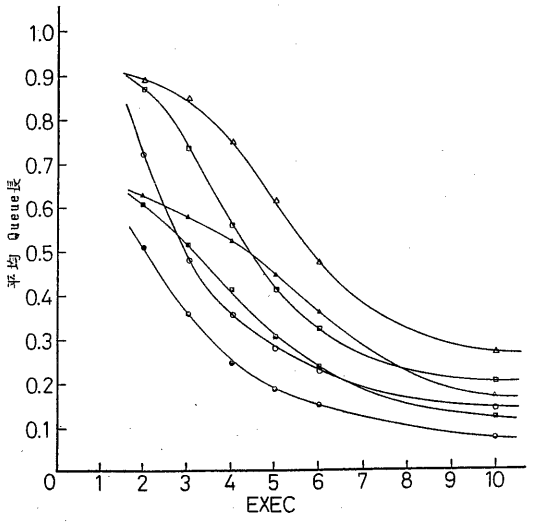


(a)　HOP : RTG が一定

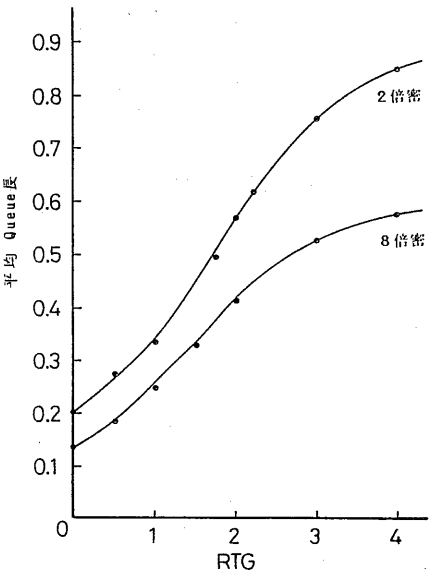


(b)　HOP : EXEC が一定

グラフ5.1　全実行時間

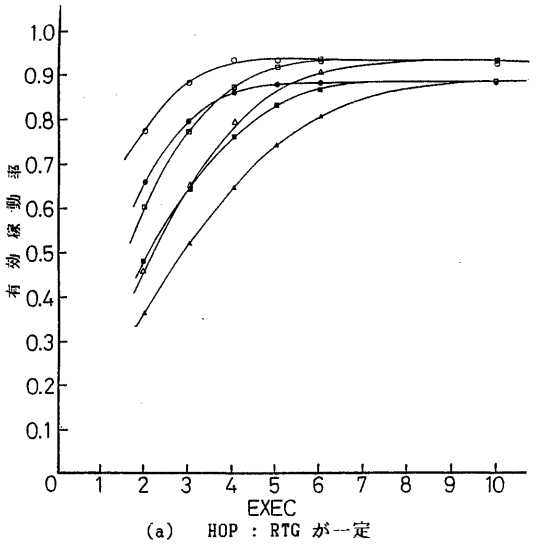


(a)　HOP : RTG が一定

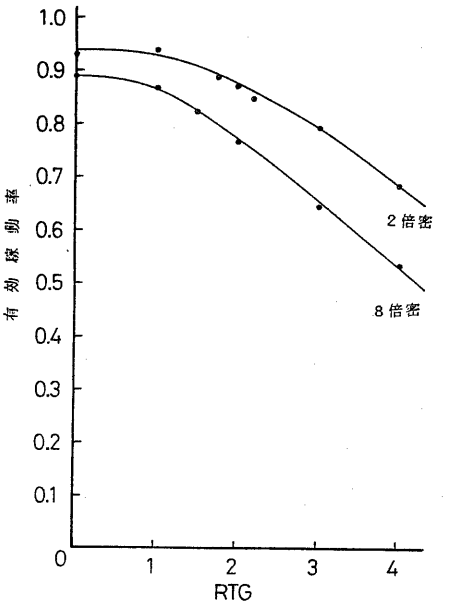


(b)　HOP : EXEC が一定

グラフ5.2　平均 **Queue**長



(a) HOP : RTG が一定



(b) HOP : EXEC が一定

グラフ5.3 有効稼動率

これらの結果から最適な HOP : RTG : EXEC を求めると次のようになる。

◇ HOP : RTG : EXEC = 1 : 2 : 4

そこで現在簡単に入手できるチップとして1 μsec/floating operationを考えた場合、先程のパラメータの値を求めると、次のようになる。

EXEC : 2 μsec

RTG : 1 μsec

HOP : 0.5 μsec

ここでは転送速度(HOP) が最も問題になる。しかしデータ交信路を full duplexにしたり、横幅 16 bit であるパケット長の短いパケットを用いたりすることによって転送速度は上昇する。またルーチング処理についても、ルーチング・プロセッサをハードウェア化し、6方向から入ってくるパケットを同時に処理できるようにすれば不可能ではない。これらから、提案したアーキテクチャは十分実現可能であることがわかる。

第6章 結論

本報告では、粗密のある領域を正方格子状をベースに分割する方法と、分割領域を効率良くマッピングすることのできるアーキテクチャとして立方格子状計算機、およびそのマッピング・アルゴリズムを提案した。さらにこのアーキテクチャを用いると、実際に反復計算においてデータの交信を行わなくてはならないPUが物理的に離れる可能性があるので、データの交信法として、有限要素法やアーキテクチャの特徴を活かしたパケット・ルーチングの方法を示した。

そして、これらのアルゴリズムを実際に実装し簡単な評価を行った。その結果、マッピングについてはクロッシングをおこさないようにすることが大切であり、データ・ルーチングについてはZ軸方向にまずパケットを送るのが最も効率の良くなることがわかった。さらに交信路の混雑状況を GPSS によりシミュレーションした結果、

HOP : RTG : EXEC = 1 : 2 : 4

とすると実現性のあることがわかった。

これからますます増大してゆくマルチプロセッサ・システムにおいて、立方格子状計算機はその構造が均質であるので VLSI 化することも比較的簡単に行えると考えられる。今後は、粗密のある3次元分割領域を効率良く扱うことのできるアーキテクチャを開発することが課題となろう。

References

1. R. M. Russell, "The CRAY-1 Computer System," Comm. ACM, vol. 21, no. 1, pp. 63-72, Jan. 1978.

2. W. J. Bouknight and et al., "The ILLIAC-IV System," Proc. IEEE, vol. 160, pp. 369-388, Apr. 1972.

3. R. Kober, "The Multiprocessor System SMS201 Combining 128 Microprocessors to a Powerful computer," Digest of Papers Compcon Fall, 1977.

4. T. Hoshino, T. Shirakawa, and et al., "Highly Parallel Processor Array PAX For Wide Scientific Applications," Proc. of 1983 Int. Conf. on Parallel Processing, Aug. 23-26, 1983.

5. H. F. Jordan, "A Special Purpose Architecture for Finite Element Analysis," ICASE Report, No. 78-9, March 1978.