メモリの階層性とベクトル計算機の実効性能†

津田孝夫†† 巽 孝明††

近く国産機も世に現れようとしているベクトル計算機について，主記憶・半導体高速補助記憶という記憶階層のもとで，主記憶に溢れるような大量データの科学技術計算（たとえば空間3次元時間依存偏微分方程式の解法）を行う場合，記憶階層間データ転送がどのようにベクトル計算機の実効性能に影響を与えるかを量的に論じている．実効性能を高く維持するためには，このデータ転送量が可及的最小であることが重要で，アルゴリズム設計，データ構成などソフトウェア技術上特段の工夫を必要とし，その意味で従来の汎用機の場合とは違った新しいプログラミング上の視点が不可欠となることを具体的に示している．

1. まえがき

高速の大規模科学計算を目的とするベクトル計算機がわが国の代表的計算機メーカから発表され，近く登場しようとしている．このタイプの計算機は並列パイプライン処理をもととし，高速に実行可能なベクトル演算命令をもっていて，最適な演算系列に対し500ないし600 MFLOPSにも達する高速計算が可能であるとされている．しかし一般ユーザが，とくにこの種の計算機の特殊機能を意識しないで自分のプログラムをFORTRANで書いた場合でも，ベクトル計算機がそのハードウェア性能を十分に発揮できるためには，現在いくつかの問題が存在すると思われる．

その第一は，FORTRANコンパイラに内蔵されている自動ベクトル化機能が有効に働く範囲が，まだあまり広くなく制限が強いことで，これに関しては現在活発な研究開発が行われている．

第二は，これが本稿において扱う主題であるが，主記憶からデータが溢れる場合の問題である．たとえば128×128×128の3次元格子空間で時間的に変化する現象に対応する偏微分方程式を解く場合，1格子点当り多数個の物理量が扱われる必要があってデータ量がきわめて多くなり（数百MBはかんたんに超える），主記憶と補助記憶との間のデータ転送が避けられなくなることが起こりうる．補助記憶として磁気ディスクを用いると，入出力オーバヘッドのため計算の高速性は失われる．これを避けるため，半導体高速補助記憶（メーカによっては拡張記憶とも呼んでいる）を用い，データ転送の高速化を図る．本論文の表題のメモリ階層とは，このような主記憶・半導体高速補助記憶(以後たんに補助記憶と略す）から成る2階層メモリを指している．主記憶は最大256 MB，補助記憶は最大1 GBまで実装可能であるといわれる．ところで，メモリ階層間データ転送はベクトル化できない部分を増加させる（すなわちベクトル化率を下げる）方向に作用する(後述)．すなわち他の命令と並列に実行できない一連のスカラ命令と等価である．一方，ベクトル計算機の実効性能はきわめて高いベクトル化率に対してはじめて保証され，しかも高いベクトル化率の動作範囲では，わずかなベクトル化率の低下が実効性能を大きく下げてしまうという著しい性質がある．

そこで(a)“かりにベクトル化率100%の数値スキームがあったとき，メモリ階層間データ転送が起きると，ベクトル計算機の実効的な計算速度はどのような影響を受けるか”という問題を量的に論ずる．ベクトル演算が十分高速で，その速度の対スカラ演算速度比が高いほど，実はこれは深刻な問題となる．

次に(a)と関連して，(b)“ベクトル計算機のプログラムを高級言語で書く場合，アルゴリズム設計上，データ構成上，どのような新しい視点が必要となるか”という実際上重要な問題に関し，3次元格子空間上で解く偏微分方程式のモデルについて，メモリ階層間データ転送量を最小化する（すなわちベクトル計算機の実効性能を最大に引き出せる）best possibleなアルゴリズム（格子空間走査法）とデータ転送量の単位やデータ構成法の適正な選択が存在することを示す．

2. ベクトル計算機のモデル

本論文は実際上の問題を扱うことを目的としている

† Effective Performance of Vector Processors in the Presence of Memory Hierarchy by TAKAO TSUDA and TAKAAKI TATSUMI (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

から，当然ベクトル計算機としてはすでに発表されているもの（富士通の VP 100/200，日立製作所の S-810/10，S-810/20，本論文投稿後発表された日本電気の SX-1，SX-2，外国では米国クレイ社の CRAY X-MP など）が念頭にある．できるだけ実際を反映するよう以後の議論に必要な部分だけモデル化する．

2.1 ベクトル化による加速

たとえば

```
      DIMENSION A(100), B(100), C(100)
      ......
      DO 10 I=1, N
   10 A(I)=B(I)*C(I)
      ......
```

のような DO ループは，ループを N 回まわさず，ベクトル {B(I)} とベクトル {C(I)} に対しパイプライン処理を行う．演算結果であるベクトル {A(I)} の各成分は（成分数 N が十分大きいとき），1 マシンサイクルごとに一つずつ得られる．このようなベクトル処理による浮動小数点演算速度は，平均して通常のスカラ演算の速度の **α 倍**であるとする．“平均して”という意味には，次のような条件が含まれている．すなわち，マシンサイクルタイムや並列に走りうる演算パイプの本数などハードウェア性能により定まる上限が α 値には存在する．またある特定のベクトル計算機に着目したとしても，変数の参照関係の相互依存性から判定して 100% ベクトル化可能な高級言語プログラムに対し，プログラムの書き方の任意性やハードウェア資源の有限性やパイプへの入力データ列の長さなどにより α 値は変わりうる．しかしこの α 値は上述の上限を超えない．

浮動小数点のスカラ演算速度の平均値は，マシンサイクルの速さと各種演算にかかるサイクル数の平均を乗じて得られる．

以上のように動的に複雑に変化する演算環境を近似的に平均的に表す指標として α 値を用いる．現在のベクトル計算機では，α 値は 10 ないし 30 程度とするのが妥当で，しかし今後ベクトル計算機のハードウェア，ソフトウェア両面の技術進歩で α 値はより高くなろう．α 値が高くなると，後述のようにメモリ階層間データ転送により深刻で鋭敏な影響を実効性能に与えることになる．

ベクトル化率 v を次のように定義する．

v =(あるプログラムのうちベクトル化可能な延べ機械命令ステップ数)/(同プログラムの延べ機械命令ステップ数)　　(1)

“延べ”とは，ループを拡げて実際に逐次実行されるスカラ命令のステップ数を指す．さらにある与えられた FORTRAN プログラムにおいて，ベクトル化手法を用いることにより，スカラ演算の場合より速さが Acc 倍になるとして，**(対スカラ演算) 加速度 Acc** を

$$Acc=(v/\alpha+(1-v))^{-1} \qquad (2)$$

と定義する．ベクトル化率 100% の場合 $v=1$ で $Acc=\alpha$，またベクトル化率 0% のとき $v=0$ で $Acc=1$ となる．次に前述の α 値に対し，あるプログラムをベクトル化率 v で計算するときの**実効性能（実効速度**といってもよい）を

$$r_{eff}=(Acc/\alpha)\times 100\,(\%) \qquad (3)$$

と定義する．いま $\alpha=10$ として v と r_{eff} の関係を図に描いてみるとわかるように，$r_{eff}>50\%$ とするためには，$v>0.9$ という高いベクトル化率が必要となる．一方，v 値の大きいところでは，v 値のわずかな減少が実効性能を急激に下げることがわかる．またこの傾向は，α 値がより大きければ曲線がより急峻になるため，わずかの v 値の減少がなおいっそう鋭敏に実効性能 r_{eff} を劣化させる．(2)は十分よい近似で成り立つ．その理由は次の通り．スカラ処理ユニットはベクトル命令列の実行のための準備を行い，ベクトル処理ユニットに起動をかけた後，いわば busy wait の状態でベクトル命令列の実行が終わるのを待つ．このため FORTRAN 文に直接対応するスカラ命令は，つねにベクトル命令列に対し逐次化されると考えてよい．

2.2 主記憶・半導体高速補助記憶間データ転送*

主記憶と補助記憶との間のデータ転送は，二者間の直接転送命令を機械命令体系に追加し，それを用いて FORTRAN の READ/WRITE 文により行われる[1]．すなわちユーザは READ/WRITE 文中で，対象デバイスとして半導体高速補助記憶を指定し，固定長ブロックの連続した複数個（この個数はユーザ指定）を転送できる．ただしこの転送命令を実行中は，他の命令を実行できないので，メモリ階層間データ転送は CPU 動作とオーバラップできない．この点がチャネルを介して行われる主記憶-磁気ディスク間データ転送と異なる重要な点である（もちろんこの転送命令はベクトル命令の実行ともオーバラップされない）．

上述の転送単位である固定長ブロックを C(kB) としたとき，一つの READ/WRITE 文により連続した

* 本節で述べるデータ転送形式は，日立製作所の S-810/10，S-810/20，最近発表された日本電気の SX-1，SX-2 で採用されている．

nブロックを転送するに要する時間は，スカラ命令のステップ数に換算して

$$a+bnC \qquad (4)$$

となる．ここに定数 a には，実行時ルーチンによりSVC 割込みを起こさせ，スーパバイザモードで上述の直接転送命令を起動するのに要する時間や，最後の転送終了処理に要する時間など，正味の転送時間以外の各種オーバヘッドが含まれる．すなわちソフトウェア上の手順として約500ステップかかるが，それに加えて補助記憶側でコマンド解読，読出し，転送終了処理に約 5 μs 必要であるとしている．定数 b は，命令ステップ数に換算した単位 kB の転送に要する転送時間である．転送速度は1 GB/s としている．かりにいま，マシンサイクルを15 ns とし，1命令を平均3マシンサイクルで実行するとすると，スカラ演算速度は約 20 MIPS となり，a=600, b=20 程度になる．以下数値的な各種推定には，この値を使う．

2.3 データ転送量

本論文で扱う数値計算モデルについては次章で詳しく述べるが，3次元格子空間上で空間3次元時間依存の偏微分方程式を考える．全データ量は主記憶から溢れるほど多いので，補助記憶上に置かれるものとする．

主記憶へのデータ転送量 R を

R=(転送されるデータ量)/(全データ量)

と定義する．Rは後述図1の‘READ’により転送されるデータ量に対応する．処理されるデータは，必ず一度は主記憶に転送されるため

$$R \geqq 1 \qquad (5)$$

である．R=1 は主記憶へのデータ転送量の自明な下界である．主記憶サイズにもよるが，同じ問題に対し不用意な格子点走査法やデータ構成を用いると，容易にRは数十倍に増加してしまうことがある．

3. 数値計算モデル

3次元格子空間上で解く3次元時間依存偏微分方程式を考え，自分自身を含む近傍格子点 s 個に関連づけられている量（旧値）を用いて，その点の解を新値に更新する．これを3次元 **s 点近傍参照モデル**と呼ぶことにする．実用上すぐれた陽的差分法に2ステップ **Lax-Wendroff 法**[2]（**L-W 法**と略称する）がある．たとえば圧縮性電磁流体を扱う場合[3]，この方法では対象とする偏微分方程式系を

$$\frac{\partial}{\partial t}\boldsymbol{U}+\frac{\partial}{\partial x}\boldsymbol{F}(\boldsymbol{U})+\frac{\partial}{\partial y}\boldsymbol{G}(\boldsymbol{U})+\frac{\partial}{\partial z}\boldsymbol{H}(\boldsymbol{U})=\boldsymbol{O} \qquad (6)$$

の形に書く．$\boldsymbol{U}$は多次元ベクトルで，列行列

$$\boldsymbol{U}=(\rho, \rho u_x, \rho u_y, \rho u_z, B_x, B_y, B_z, E_T)^T \qquad (7)$$

となり（右肩のTは転置を表す），ρは密度，u_x 等は流体の速度成分，B_x 等は磁場成分，E_T は単位体積当りの全エネルギである．$\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{H}$ は $\boldsymbol{U}$ と同じ次元数のベクトルで，$\boldsymbol{U}$ を与えれば計算可能な関数である．(6)を $\boldsymbol{U}$ の中間解 $\boldsymbol{U}^{(n+(1/2))}$ を介して次の2ステップで数値的に解く．$\boldsymbol{U}$の右肩の数は時間を，右下の添字は3次元空間の格子点の位置を表す．

$$\begin{aligned}\boldsymbol{U}^{(n+(1/2))}_{i,j,k}=&(\boldsymbol{U}^{(n)}_{i,j,k+1}+\boldsymbol{U}^{(n)}_{i,j,k-1}+\boldsymbol{U}^{(n)}_{i,j+1,k}\\&+\boldsymbol{U}^{(n)}_{i,j-1,k}+\boldsymbol{U}^{(n)}_{i+1,j,k}+\boldsymbol{U}^{(n)}_{i-1,j,k})/6\\&-\frac{\Delta t}{2\Delta x}(\boldsymbol{F}^{(n)}_{i+1,j,k}-\boldsymbol{F}^{(n)}_{i-1,j,k})\\&-\frac{\Delta t}{2\Delta y}(\boldsymbol{G}^{(n)}_{i,j+1,k}-\boldsymbol{G}^{(n)}_{i,j-1,k})\\&-\frac{\Delta t}{2\Delta z}(\boldsymbol{H}^{(n)}_{i,j,k+1}-\boldsymbol{H}^{(n)}_{i,j,k-1}) \qquad (8)\end{aligned}$$

$$\begin{aligned}\boldsymbol{U}^{(n+1)}_{i,j,k}=&\boldsymbol{U}^{(n)}_{i,j,k}-\frac{\Delta t}{\Delta x}(\boldsymbol{F}^{(n+(1/2))}_{i+1,j,k}-\boldsymbol{F}^{(n+(1/2))}_{i-1,j,k})\\&-\frac{\Delta t}{\Delta y}(\boldsymbol{G}^{(n+(1/2))}_{i,j+1,k}-\boldsymbol{G}^{(n+(1/2))}_{i,j-1,k})\\&-\frac{\Delta t}{\Delta z}(\boldsymbol{H}^{(n+(1/2))}_{i,j,k+1}-\boldsymbol{H}^{(n+(1/2))}_{i,j,k-1}) \qquad (9)\end{aligned}$$

ここに $\Delta x, \Delta y, \Delta z$ はそれぞれ x, y, z 軸方向の格子間隔，Δt は時間刻みで，$\boldsymbol{F}^{(n)}_{i,j,k}$ は $\boldsymbol{F}(\boldsymbol{U}^{(n)}_{i,j,k})$ の略記である（他も同様）．各辺を等分割した3次元立方体格子空間を考え，1辺当り格子点がN個で，計 N^3 個の格子点すべてについて時間的に変化する $\boldsymbol{U}$ の値を求める．L-W 法の第2ステップに第1ステップを埋め込むと，格子点データ参照関係は近似的に前述の3次元 s 点近傍参照モデル（s=25）で表される．実際の L-W 法では，第2ステップに第1ステップを埋め込むと，第2ステップで参照する（自分自身以外の）6点は参照する必要がなくなる．

本論文では仮想記憶のメカニズムは考えないが，ユーザが一度に READ/WRITE で転送する量は一定とし，1回分の連続した複数個の固定長ブロック（2.2節参照）を1ページと呼ぶ．このようなページのサイズは，（システムが定めている固定長ブロックの整数倍の範囲で）ユーザが定めることができる．3次元格子空間の1辺長当りのデータ数は，収納の順方向に1ページの整数倍とし，立方体の境界はページ境界と一

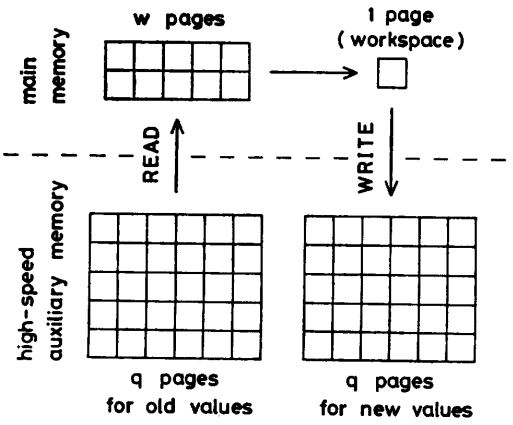

図 1 メモリ階層間データの流れ
Fig. 1 Data transfer between the two levels of memory.

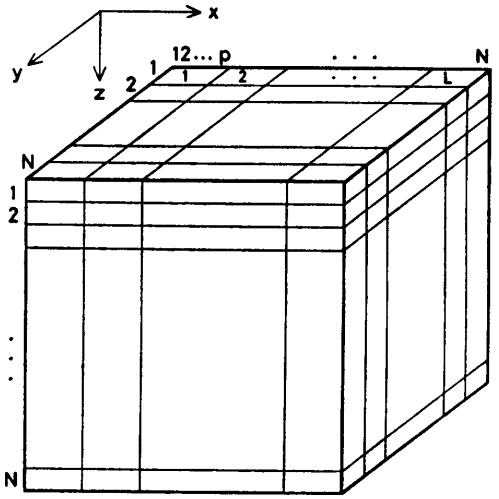

図 2 格子点データのページ分割
Fig. 2 Data at mesh points partitioned into pages.

致するような規則的なデータのページ間分布であるとする.

次章のシミュレーションで扱われるページ参照関係は，実際の数値計算におけるページ参照関係をそのまま扱う．見いだされるデータ転送量Rは，したがって，用いる数値計算アルゴリズムに伴うR値の構築的上界 (constructive upper bound) を与える．3次元 L-W 法ないしは s 点近傍参照モデルのための記憶空間でのデータの流れを模式的に描いたのが**図1**である．一つの小正方形は1ページに相当し，主記憶上に一度にはwページしか読み込むことができないとし，別に1ページの作業領域を設けて，ここで1ページ分の新値を形成して補助記憶へ順次転送する．パラメータwは主記憶の有限サイズを表す．補助記憶は十分広く，旧値を保持するためにqページ，新値を保持するために別にqページの空間をとるとする．これらqページの2群は，1時間ステップごとに互いに新，旧の役割を交換して使うことになる．READ 命令によってフェッチされるデータの総量がRで表され，図1からわかるように実際の延べ転送量は WRITE の1回分を入れて $R+1$ で表される．

4. データ転送量のシミュレーションによる推定

4.1 シミュレーションの目的

全格子点をある定められた順に従って規則的に走査 (scan) し，各点での値を更新していく．参照したいデータが主記憶上になければ（ページフォールトという），そのデータを含むページを主記憶に転送し，計算を進める．その際，主記憶サイズは有限であるから，不必要と思われるページと置きかえる．ただしこの場合，図1からわかるようにページ戻しはしない．アルゴリズム（すなわち格子空間走査法）とデータの構成法（すなわちデータ構造やページサイズなど）はユーザが定義するものだけに，メモリ階層間データ転送の量と回数は大幅に変わりうる．これらが実効性能に与える影響を定量的に調べ，最もよい走査法やデータ構成法が存在することを示すのが本章の目的である．

4.2 シミュレーションのモデル

走査によりある点の値を更新するとき，その点および近傍24点の旧値を含むページ集合が主記憶上で必要となるが，もしそれらのページすべてがないときは，LRU (least recently used) 法により必要なページを主記憶に置く．ページフォールト数を記録する．また立方体の境界面上の格子点を走査するとき，近傍点がもともと存在しない場合はページフォールトとしない．

4.3 格子点走査法によるデータ転送量の差異

ユーザが定義するアルゴリズム上の手順の重要な部分は，全格子点をどのような順に走査し，解を更新するかである．実はこれがデータ転送量Rを大幅に変えるので，ベクトル計算機の実効性能を維持する上で，プログラミング技術上きわめて重要である．

格子点データは $N\times N\times N$ 個が**図2**のようにFORTRAN の3次元配列の格納法に従って，補助記

憶上の1次元記憶空間に収納されているとする．図の x 軸方向にページ分割され，1ページは p 個の格子点データから成り，図に描かれている $p\times1\times1$ の直方体が1ページの内容を構成している．3章の仮定から L を正整数として

$$N=pL \tag{10}$$

とする．3種類の格子点走査法を取り上げてみよう．

4.3.1 格子点走査法

(i) **normal scan 法**　更新したい格子点データが物理的に並んでいる順に走査していく単純な方法である．**図3**(i)の矢印の $1,2,\cdots,N$ 順に走査し，終われば z の増す方向に次の面を選び同様のことを繰り返す．FORTRAN-like な走査法である．

(ii) **switchback scan 法**　normal scan 法では，点 $(N,j,k)(1\leqq j<N, 1\leqq k\leqq N)$ の次に点 $(1,j+1,k)$ を走査する．しかし点 (N,j,k) の値を更新するのに必要なページ集合と，点 $(1,j+1,k)$ の値を更新するのに必要なページ集合とは，一般に共通部分が少ないから，図3(ii)のように走査が境界に達するごとに折り返し，ページフォールト数を減らそうとするもので，これを switchback scan 法という．すなわち格子点 (N,j,k) の次には $(N,j+1,k)$ を走査する．

(iii) **partitioned normal scan 法**　(ii)の主旨と同様に，さらに工夫することによりデータ参照の局所性を維持しようとする方法である．主記憶は $w\geqq 25L$ が成り立つ程度に大きいとしよう．そして $w\geqq L\times M\times5$ (この右辺の因数 $L,M,5$ はこの順に x, y,z 各軸方向の'幅'に対応している．以下このような記法を使う) を満たす最大の正整数 M を見いだし，格子空間を図3(iii)で示すように，y 軸方向に M の幅で z-x 平面に平行な平面で分割する．ただしこのとき，各分割の区切り部分を y 軸方向に4点ずつ重複させる．もちろんこの4という値は，s 点近傍参照モデルの s 値に依存し，図3(iii)の斜線で示す重複部分の格子点は，相隣る二つの partition に含まれることになる．このように $N\times M\times N$ のスライスに分割された各 partition について，前述(i)の normal scan 法により格子点走査を行う．すなわち走査の範囲は第1の partition に関し $1\leqq i\leqq N$, $3\leqq j\leqq M-2$, $1\leqq k\leqq N$ で，この場合第1 partition の各ページすべて1回の主記憶への転送ですむことに注意すべきである．次に隣りの第2の partition をやはり(i)の方法で $1\leqq i\leqq N$, $M-1\leqq j\leqq 2M-6$, $1\leqq k\leqq N$ の範囲につき走査する．以下同様，要するにこの方法は多少の主記憶へのデータ転送量を犠牲にして，データ参照の局所性の向上を狙うものである．主記憶があまり大きくなく，$w<25L$ となる場合は，先ほどとちがって

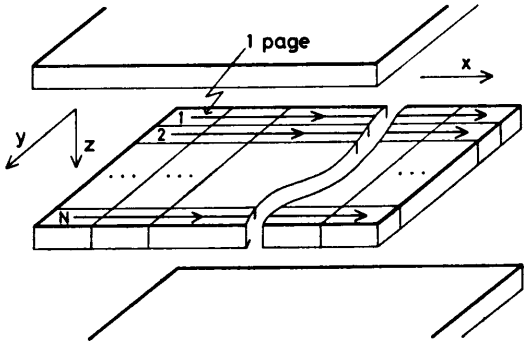

図3(i)　normal scan 法
Fig. 3(i)　Normal scan.

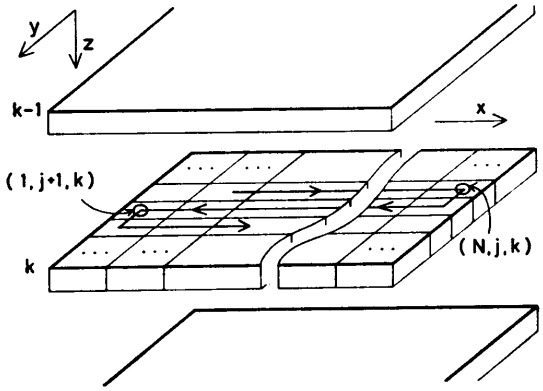

図3(ii)　switchback scan 法
Fig. 3(ii)　Switchback scan.

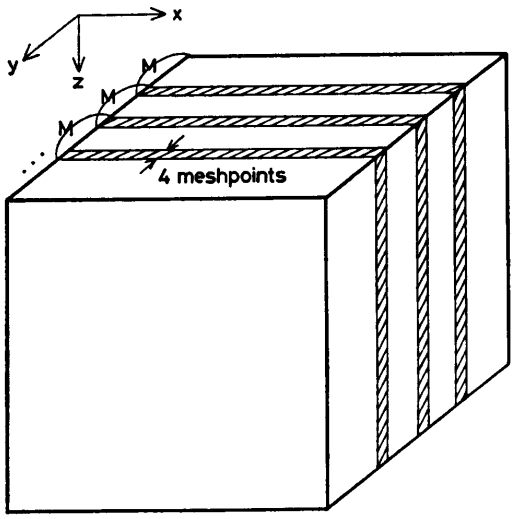

図3(iii)　partitioned normal scan 法
Fig. 3(iii)　Partitioned normal scan.

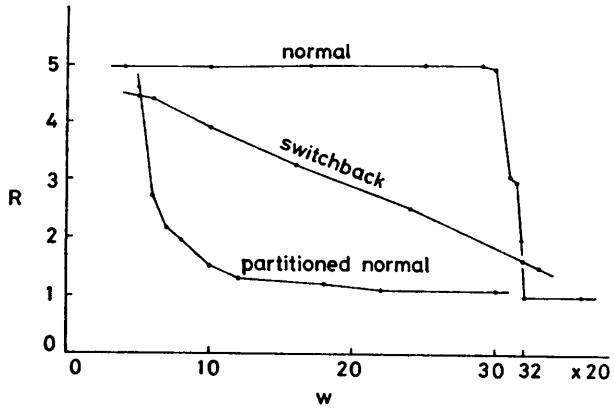

図 4 走査法とR値（シミュレーションによる）
Fig. 4 The amount of data transfer R versus three types of scan (by simulation).

x方向に分割し，区切り境界のデータをやはり重複させて partition を作ればよい．詳細は省略．

4.3.2 データ転送量の理論的予測

$w \geqq 25L$ の場合について，主記憶へのデータ転送量Rの予測値を次に示そう．結果のみ示すが証明は比較的容易である（Rは1ページ当りのページフォールト数になっていることに注意）．

（i） normal scan 法 $L\times5\times5 \leqq w < L\times N\times5$ の場合，$R=5$. $L\times N\times5 \leqq w$ の場合，$R=1$.

（ii） switchback scan 法 $L\times5\times5 \leqq w$ の場合，

$$R=5-\frac{4w}{5L}/N.$$

（iii） partitioned normal scan 法 $L\times8\times5 \leqq w < L\times N\times5$ の場合，$w/(5L) \triangleq M$ として，生ずる partition 数 N_p は

$$N_p=\lceil (N-4)/(M-4) \rceil$$

で与えられ，

$$\begin{aligned} R&=[2\times\{(N_p-1)\times L\times4\times N\} \\ &\quad +1\times\{(N-(N_p-1)\times4)\times L\times N\}] \\ &\quad /(L\times N\times N) \\ &=1+4(N_p-1)/N \end{aligned}$$

となる．上式最初の右辺の分子で，1番目の中カッコは重複部のページ総数で，その係数2は2回転送されることを示し，同2番目の中カッコは重複しないページの総数で，その係数1は1回しか転送されないことを示す．分母の $L\times N\times N$ は全データのページ総数である．wが以上の場合より小さい場合は，

$w=L\times5\times5$ のとき

$$R=[10+5(N-4)]/N$$

$w=L\times6\times5$ のとき

$$R=[6+3(N-4)]/N$$

$w=L\times7\times5$ のとき

$$R=\left[7+\frac{7}{3}(N-5)\right]\Big/N$$

となることが個別的に示せる． □

以上三つの走査法(i), (ii), (iii)について，たとえば $N=128$, $w=240$, $L=4$ とすると，(i) $R=5$, (ii) $R=4.63$, (iii) $R=1.46$ となり，partitioned normal scan の転送量は格段に少ない．以上の推定値を一括して相互に比較するとよくわかるように，partitioned normal scan 法は2.3節で述べた主記憶へのデータ転送量の下界 $R=1$ をパラメータwの相当広い範囲にわたってほぼ実現しており，その意味においてほぼ best possible なアルゴリズムである．次項でさらにこれについて，実際のデータ参照関係を用いたページフォールトに関するシミュレーションにより検証してみよう．

4.3.3 各走査法に関するシミュレーション

前項の理論的推定は，格子点の絶対数が十分大きいときにももちろん成り立つ．しかしシミュレーションでは実際のデータ（すなわちそれの属するページ）参照関係を抽出してページフォールトを検証し，しかも何度も各パラメータを広範囲に変化させて調べる．それで適当な計算時間に実行を終わらせるため，モデルの物理サイズを小さくし

主記憶サイズ：$w>L\times3\times5$
格子点数 (N^3)：$32\times32\times32$
ページサイズ (p)：8データ/ページ

とした．シミュレーションは3次元25点近傍参照モデルについて行い，走査される各格子点の値を更新するために必要な近傍点（自分自身を含む）のデータが含まれるページが主記憶上にあるか実際に探索する．前項の推定値に対応するシミュレーション結果を図4に示す．少々の量的なちがいが生じているが，理論的推定をよく裏づけている．このシミュレーションにおけるデータ参照関係は，実際の数値計算の場合と同じで，観測されるデータ転送量のR値はその構築的上界を数値的に与える．これにより partitioned normal scan 法は，やはりほぼ best possible な結果 $R \fallingdotseq 1$ を与えることが検証されたといえる．

4.3.4 ページサイズを変えた場合のページフォールト数およびデータ転送量

データ転送量Rはデータ構成法に影響される．そこ

表 1 ページサイズの影響
Table 1 Page faults and value R versus page sizes and main memory pages (with $w \times p$ kept constant).

Page size p	Number of main memory pages w	Number of page faults (×32)	R
2	800	2,560	5
4	400	1,280	5
16	100	320	5
64	25	80	5
128	13	40	5
200	8	25	5
228	7	23	5
229	6	1,257	305
400	4	4,828	1,873

でデータ構成の一側面であるページサイズについても，前項と同様の手法にしたがってシミュレーションを行った．格子点数は $128\times128\times128(N=128)$，データ格納法は図2の FORTRAN-like なものとし，走査法は 4.3.1 項で述べた normal scan 法で，いままで同様ページ置換えは LRU 法に従うとする．主記憶容量 $w\times p$ を一定とし，これらの条件のもとで行ったシミュレーションの結果を**表1**に示す．ページに格子点データが格納されている順方向(図3(i)参照)に走査するから，メモリ参照の逐次性から，ページサイズ p を大きくするとフォールト数はしだいに減り，しかしページサイズが極端に大きくなるとスラッシング(thrashing)を起こしてしまうことが観察される．本論文で考えているような補助記憶では，転送のオーバヘッドは磁気ディスクの場合に比しきわめて小さいが，それでも 2.2 節で述べたようなソフトウェア上のオーバヘッドのため，転送上の効率からはスラッシングが生じない程度にページサイズは大きいほうがよい．

また表1から次のこともわかる．すなわちページサイズを増していくとフェッチ回数はしだいに減るが，スラッシングが起こるまではデータ転送総量 R は一定である．いったんスラッシングが起こると，フェッチ回数も転送量もきわだって増大する．このことから，スラッシングが起きない限り，図4で示したような w-R 特性はページサイズ p と直接関係がないといえる．

5. ベクトル計算機の実効性能の評価

5.1 1格子点1要素データと1格子点多要素データ

本論文で扱う数値計算モデルは，3次元 L-W 法あるいはその近似である s 点近傍参照モデルであるが，実際問題の場合，問題を解きたい物理空間上の1（格子）点に複数個の量が対応する．たとえば(6)～(9)の場合には，(7)で示されるように，1格子点に対し密度，フラックス，磁場，単位体積当りのエネルギなど8個の物理量が対応し，その時間変化を求める．このような場合を**1格子点多要素データ**と呼ぶ．各点に1種類のデータのみを考えればよい場合を，**1格子点1要素データ**と呼ぶ．

1格子点 m 要素データ（$m>1$）の場合，データの物理的格納法としては2通り考えられる：(i) 1格子点に関係する m データは，すべて一定の順に物理的に隣接して並べる．このひとまとまり自体が3次元配列の1要素であるので，これを**3次元ベクトル配列**と名づけよう；(ii) m 要素の各要素ごとに別々に合計 m 個の3次元配列を用意する方法で，これを**多重スカラ配列**と呼ぶ．

5.2 計算およびデータ転送に必要な時間(命令ステップ数換算)の推定

圧縮性電磁流体の3次元 L-W 法をより具体的に検討してみよう．この方法のステップ1（((8)式）をステップ2（((9)式）に埋めこむと，プログラムは図2のメモリ空間を用いて，次の形となる．

DO 10 $i, j, k=1, N$

$$\boldsymbol{U}^{(n+1)}_{i,j,k}=\boldsymbol{f}(\boldsymbol{U}^{(n)}) \qquad (11)$$

10 CONTINUE

データ転送は別として，いま計算のみに着目すると，(11)は8本のスカラ方程式（代入文）と同じで，その右辺はすべてもはや変更を受けない旧値のみで定まるから，このループは完全にベクトル化可能である．しかし(11)のままでベクトル処理されるためには，並列に走る8本のパイプが必要で，現状では(11)の8個の方程式はそれぞれ別々にベクトル化され，

DO 10 $i, j, k=1, N$

$$(\boldsymbol{U}^{(n+1)}_{i,j,k})_l=f_l(\boldsymbol{U}^{(n)}) \qquad (12)$$

10 CONTINUE

の形で $l=1, 2, \cdots, 8$ につき順次実行されるとするほうが妥当であろう．(7)の場合につき FORTRAN で(12)の部分のプログラムを書き，コンパイラで機械語に落とすと，ステップ1すなわち(12)の右辺で，

$(\boldsymbol{F}^{(n+(1/2))}_{i+1,j,k})_l$, $(\boldsymbol{F}^{(n+(1/2))}_{i-1,j,k})_l$,
$(\boldsymbol{G}^{(n+(1/2))}_{i,j+1,k})_l$, $(\boldsymbol{G}^{(n+(1/2))}_{i,j-1,k})_l$,
$(\boldsymbol{H}^{(n+(1/2))}_{i,j,k+1})_l$, $(\boldsymbol{H}^{(n+(1/2))}_{i,j,k-1})_l$

の6項の評価には，平均1項当り85命令程度必要で

ある．またこれら6項が与えられたとき，ステップ2に従い $(U^{(n+1)}_{i,j,k})_t$ を1個計算するのに約75命令必要である．したがって(12)の1行を1回実行するためには約 580 (≒6×85+75) 命令が実行されなければならない．(12)の最内側ループが100%ベクトル実行されると，(12)のようにタイトな多重ループになっているためのソフトウェア・オーバヘッド（スカラ処理）はベクトル命令列とオーバラップされ，全体として実効的なベクトル化率は落ちないと仮定できる．

5.3 ベクトル化率

次に(12)を含むループ全体を計算する場合の実効的なベクトル化率を算出する．主記憶へ転送するのに必要な延べデータ転送量は，すでに定義したとおり全データ量のR倍である．5.1節で述べた多量スカラ配列を用いるとする．$\{U^{(n+1)}\}$ のあるページに着目し，ページ当りのデータ数pを単位に考えると，1ページ分の解の更新をするための命令ステップ数は5.2節から $580p$ で，これらのスカラ命令はすべてベクトル化可能である．一方この1ページ分の解の更新に必要な往復のデータ転送は，R回の READ と1回の WRITE，すなわち計 $(600+20C')(R+1)$ 命令ステップ数かかる．ただし C' は1ページ分のデータの kB 数で，$C'=nC$（(4)式参照）．これによりベクトル化率 v は(1)から

$$v=580pq/[580pq+(600+20C')q(R+1)]$$
$$=1\Big/\left[1+\frac{600(R+1)}{580p}+\frac{1}{29}\cdot\frac{C'}{p}\cdot(R+1)\right] \quad (13)$$

となる．データ長は8Bで一定とすれば C'/p は定数となるから，ベクトル化率を上げるにはデータ転送量

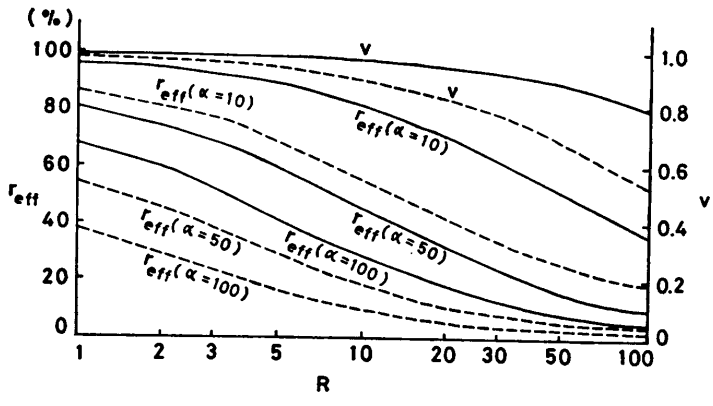

図5 $p=125$(破線) と $p=500$(実線) の場合の r_{eff} と v の値
Fig. 5 Effective performance r_{eff} and vectorization ratio v for the cases $p=125$ (shown by dashed lines) and $p=500$ (shown by solid lines)

表2 r_{eff} の値
Table 2 r_{eff} versus page sizes and types of scan.

α	r_{eff}(%)	
	Large page size; partitioned normal scan	Small page size; normal scan
10	84.1	29.9
20	71.4	16.8
30	62.2	11.7
50	49.3	7.2
100	32.5	3.7

Rを下げ，ページサイズ p を大きくしなければならない．(13)により(3)から実効性能 r_{eff} を計算して描いたのが**図5**で，$p=125$(破線) と $p=500$(実線) の場合につき，実効的なベクトル化率と種々のα値に対する実効性能を示してある．

5.4 走査法，データ構成法の最適化の効果

実効性能はRとpに依存し，Rは格子点走査法によって大きく変わる．すでに見たとおり1格子点1要素データの場合，normal scan 法で $R=5$ となる主記憶容量の範囲では，必ず partitioned normal scan 法は $R≒1$ を与える．

前節にひき続いて，多重スカラ配列を用いて $\{U^{(n)}\}$ (qページ)から $\{U^{(n+1)}\}$ (qページ) を求めることを考える．$\{(U^{(n+1)})_t\}$ は $q/8$ ページを占め，その1ページを作るために約 $580p$ 命令ステップの計算が必要である．normal scan 法の場合，$\{(U^{(n+1)})_t\}$ を作るため $\{U^{(n)}\}$ の q ページを5回 READ し，結果を $q/8$ ページ WRITE する．したがって都合合計の往復のデータ転送量は $\{(U^{(n+1)})_t\}$ の1ページ当り $(5q+q/8)/(q/8)=41$ となり，これが(13)における $(R+1)$ に相当する．partitioned normal scan ではこの値は $(1q+q/8)/(q/8)=9$ となる．図5から次のことがわかる．$p=125$ で normal scan をすると，$\alpha=10$ のとき $r_{eff}=29.9\%$ であるが，partitioned normal scan 法を用いれば $r_{eff}=60.7\%$ に上がる．さらに同じ α 値で同じく partitioned normal scan 法のまま $p=500$ とすれば，$r_{eff}=84.1\%$ にまで改善される．このようにアルゴリズム（格子点走査法）とデータ構成法（ページサイズすなわち転送単位の調整）の選択いかんによって，実効性能がどのように影響されるかを示したのが**表2**である．ベク

トル演算による高速化が進歩すればするほど，α 値は増す．α 値が大きいとさらにユーザが用いるプログラミング手法の上述のような諸因子が，きわめて鋭敏にベクトル計算機の実効性能を左右することがわかった．

6. 結　論

主記憶・半導体高速補助記憶間のデータ転送が，ベクトル計算機の実効性能に与える影響について，具体的な，しかし典型的と思われる実例を用いて量的に論じた．将来的な問題を含めて，得られた結論を要約すると次のとおりである．

（1） ことにベクトル化率の高い優良な計算スキームにおいて，わずかなデータ転送量の増加は実効性能を著しく阻害する．これはベクトル演算がスカラ演算にくらべて速い（α 値が高い）ほど，深刻な問題となる．

（2） 主記憶から溢れるような大量データの問題に関しある計算目的が設定された場合，それに伴うデータ転送量・転送回数の下界理論が必要である（これについては別に研究が進行中である[4]）．すなわちプログラムを書き下すとデータ転送量・回数の構成的上界が得られるが，それが理論下界と照合して best possible かどうかが重要となる．

（3） 本論文で考察の対象とした陽的解法の3次元 L-W 法，ないしはそれの近似的表現である s 点近傍参照モデルに対し，通常行われる normal scan 法 (FORTRAN-like な走査法) は転送量が多く，その意味でベクトル計算機との整合性がきわめて悪い．そのため partitioned normal scan 法と称する新しい方法を考案したが，これは上記(2)の意味において best possible である．

（4） コンパイラのもつ自動ベクトル化機能がかりに理想的で，十分ベクトル化率の高い目的コードが生成されるとしても，大量データの数値計算では，プログラミング上，いままでの汎用機と違った新しい視点が必要である．これを具体的に本論文で示した．

謝辞 本研究の一部は，京都大学大型計算機センター数値解析研究委員会の研究計画にもとづき行われ，シミュレーションは同センター FACOM M 382 を用いた．記して謝意を表する．

参 考 文 献

1) 和田英夫，後藤二三男，面田耕一郎，田部井隆：ベクトル計算機の階層記憶方式，情報処理学会第26回（昭和58年度前期）全国大会講演論文集（I），pp. 169-170 (1983. 3).

2) Richtmyer, R. D. and Morton, K. W.: *Difference Methods for Initial-Value Problems*, 2nd ed., pp. 354-368, Interscience Publishers, New York (1967).

3) Ugai, M. and Tsuda, T.: Magnetic Field-Line Reconnexion by Localized Enhancement of Resistivity. Part 1. Evolution in a Compressible MHD Fluid, *J. Plasma Phys.*, Vol. 17, Part 3, pp. 337-356 (1977) ; Tsuda, T. and Ugai, M.: Magnetic Field-Line Reconnexion by Localized Enhancement of Resistivity. Part 2. Quasi-Steady State, *J. Plasma Phys.*, Vol. 18, Part 3, pp. 451-471 (1977).

4) Tsuda, T., Sato, T. and Tatsumi, T.: Minimizing Page Fetches for Permuting Information in Two-Level Storage. Part 1. Generalization of the Floyd Model, *J. Inf. Process.*, Vol. 6, No. 2, pp. 74-77 (1983).

(昭和58年3月31日受付)
(昭和58年6月20日採録)