仮想テープとメモリ管理

板野肯三（筑波大学電子情報工学系）

1. はじめに

仮想記憶上では、アクセスされるデータの局所性がシステムの効率に大きく影響する。そこで、局所性の高い逐次的にのみアクセスされるデータ構造を仮想記憶を用て実現し、仮想テープ（Virtual tape, V-tape）と呼ぶことにした。"仮想"という言葉をつけた理由は、仮想記憶であるという事の他に、通常の物理的テープではなくて、Turing[1]以来使用されている様な論理的抽象的イメージを与える為である。ここでは、実際に試作された、V-tapeを取扱う機構を持つ計算機に基いて、そのシステムの解析を行なう。

2. 仮想テープの基本的概念とその使用法

2.1 紙テープに逐次的な使用

V-tapeは、論理的には逐次的に配列されたデータセルで構成されていて、相対的、逐次的アクセスは可能であるが、V-tape内のデータセルに対する絶対アドレスの概念はないので、自由にランダムアクセスはできない。V-tapeのデータセルのアクセスにはヘッドが用いられ、右又は左隣りのデータセルに移動できる。V-tapeは左右に完全に対称であり、両端にBOT（Beginning of Tape）, EOT（Ending of Tape）と呼ぶ境界がある。V-tape上の方向は、BOT側を左後向き、EOT側を右前向きとする。

最初に定義された時は、V-tapeは長さ0、すなわち、データセルを含んでいないが、ヘッドが置かれて、V-tape上を移動するのに伴なって自動的に延長される。又、同時に複数本のV-tapeとヘッドを定義することも可能であり、一本のV-tape上に2個以上のヘッドを同時に置いて使用してもよい。

さて、紙テープに逐次的な使用だけであれば、以下に示す様な基本的命令で充分である。

(1) V-tape生成命令（tapeid ← CREATETAPE(erase, l);）
eraseで指定されたモード（後述）を持ち、最大の長さをlで制限されたV-tapeを生成し、このV-tapeを識別するtapeidを与える。lが与えられなければV-tapeの長さに制限はない。

(2) V-tape削除命令（DELETETAPE(tapeid,…);）
tapeidにより指定されたV-tapeを削除する。V-tapeの占有していたリソースはすべて返還される。

(3) ヘッド生成命令（headid ← CREATEHEAD(tapeid, l, m);）
mで指定されたモード（Read only, write only, etc）を持つヘッドを生成し、tapeidで指定されたV-tapeのBOTの右隣りのデータセル上に置く。これがEOTであれば、V-tapeは自動的に1セル分延長されその上にヘッドが置かれる。lは、データセルの物理的長さを指定するパラメータである。

(4) ヘッド削除命令（DELETEHEAD(headid,);）
headidにより指定されたヘッドを削除する。ヘッドの占有していたリソー

スは、すべて返却される。

(5) Headによるアクセス命令 (VTαβδ(headid, data);)

αβδはRWFBEかnull中の1つで、αβδの順序で機能する。RWFBEの機能は以下の通りである。

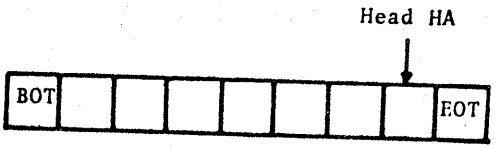
① R：ヘッドが置かれているデータセルのデータを読出す。
② W：ヘッドが置かれているデータセルにデータを書込む。
③ F：右のセルに移動する。
④ B：左のセルに移動する。
⑤ E：ヘッドの置かれているデータセルをV-Tapeより取除く。

e.g. VTRF(h1, d) ：ヘッドh1下のデータをdに読出して前進。
VTBW(h1, d) ：ヘッドh1を後退させてdを書込む。

2.2 Stack と Queue

Stack, Queueは重要かつ有用な逐次アクセス形のデータ構造であり、V-tape上で容易に実現できる。Stackの場合は、ヘッドをstack pointerとして使用し、図.1に示す様にpush downはVTWF命令により行ない、popupはVTBR命令で実行すればよい。queue, dequeの場合は、ヘッドを2個使用する必要がある。queueでは、一方のヘッドはVTWF命令でV-tapeにデータを書込んでいくのに使用し、もう一方のヘッドはVTRF命令でデータを読出していくのに使用する(図.2) deque[9]の場合は、2つのヘッドが、読出し書込みの両方に使用される。LISPのガーベージコレクションに使用するStackの時は、スタック上にもう1つのヘッドをおいて走査する様なscannable stack[4,5,7]が必要である。又、multiple buffering operationを実行する際に役立つ様な、multi headed queue等も容易に実現できる。UNIX[16]等で実現されているprocess間をインターフェイスする"pipe"も基本的にはqueueであり、V-tapeが使用できれば、buffer上に納まらないものでも、自動的に処理が可能である。

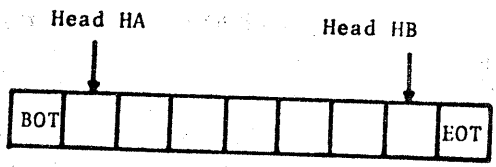
Stack Implementation



write (push down) : VTWF(HA,data)
read (pop up) : VTBR(HA,data)

FIG.1

Queue Implementation



write : VTWF(HB,data)
read : VTRF(HA,data)

FIG.2

2.3 V-tape の erase 機能

Stack上でポインタよりも前方(上)の部分は、論理的には存在しない所であり、又、queueでは、読出しヘッドよりも後方の部分がこれにあたる。これらの場所はヘッドの動きに応じて、自動的にV-tapeより消去(erase)されていく必要があるので、この様なerase機能はV-tapeの属性として指定することができる。一般に、eraseするということは2つのことを意味している。1つは、eraseされたセルは、論理的には存在しないセルであるからアクセスを手エラーとして禁止する必要があるということ。いま1つは、仮想空間の効率的使用の為、使用しないリソー

スを返還するということである。

2.4 PLACE と mark

純粋な逐次アクセスしか行なわない時は、以上に説明した機能で充分であるが、実際的な計算処理への応用では、V-tape上の特定の位置へ、ヘッドを直接的に移動することが必要である。まず、ヘッドの様にリソースを占有しないで、V-tape上のデータセルの位置のみを保持する為に、マークという概念を導入する。そして、ヘッドやマークを、別のヘッドやマークが置かれているデータセル上に直接的に、移動する命令として PLACE という新しい命令を導入する。これらは、後に説明する様なV-tape上で数値計算アルゴリズム等を実現する際に不可欠のものである。マーク、PLACE等の命令についての説明を以下に示す。

(1) マーク生成命令 (markid ← CREATEMARK();)
新しくマークを生成して、そのマークを識別する markid を与える。

(2) マーク削除命令 (DELETEMARK(markid);)
markid で指定されたマークを抹除する。

(3) PLACE 命令 (PLACE (id1, id2);)
id1 で指定されたヘッドやはマークを、id2 で指定されたヘッドやはマークの位置へ移動する。

V-tapeの BOT, EOT は special mark である。

2.5 V-tape の局所的ランダムアクセス

本来、V-tapeは逐次的アクセスしか行なわないデータ構造として導入したのではあるが、実際のプログラミングの際には、少なくとも局所的にはランダムアクセスが可能なほうが有利であるので、2つのタイプのランダムアクセスを導入する。

まず最初のタイプは、ヘッドの移動に関するものである。即ち1回の命令でのセルだけヘッドを移動できる様な命令を考える。VTFX, VTBX の2つがこれであり、

$$VTFX(headid, n) = VTBX(headid, -n)$$

という関係にある。但し、VTFX は現在の位置から前方が、VTBX は後方が n の正の移動の向きである。

2番目のタイプの命令は、ヘッドより相対的に n セルだけ離れたデータセルのアクセスをする命令である。これらの命令は次の様な形式で書く。

$$VTRX(headid, data, n),\ VTWX(headid, data, n)$$

この種のランダムアクセス命令は、当然のことながら、n が大きくなるとデータアクセスの局所性を著しく悪くするので、使用にあたっては、充分その点に注意する必要がある。

3 V-tape システムのインプリメント

3.1 システムの構成

ハードウェアシステムは、試作した CPU と 64KB の主記憶、V-tape の記憶管理及び入出力処理を実行する "intelligent channel"、2.5M バイト disk、他から構

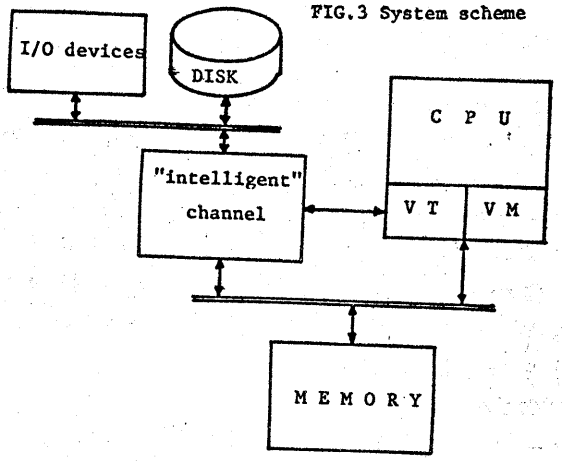


FIG.3 System scheme

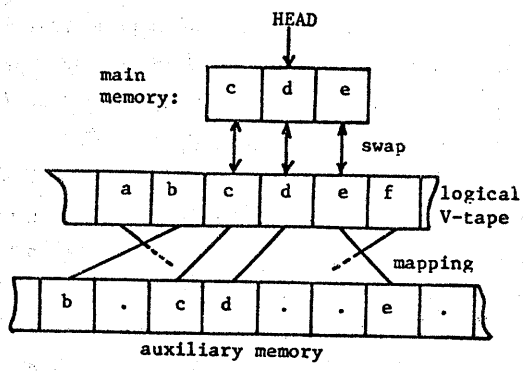


FIG.4 V-tape paging

成されており．この概要を図3に示す。CPU は BCPL の INTCODE machine (後述) を基本にし．V-tape 及び通常の仮想記憶の機構を備えている。又．"intelligent channel" には TOSBAC 40C を使用した。

3.2 V-tape のページング

V-tape の物理的実体の大部分は．ランダムアクセス型の補助記憶内に存在し．ヘッド下のデータセルとその近傍のみを高速記憶上に読出してアクセスする。V-tape は必要に応じて．ダイナミックにその長さが延長．縮小されるので．記憶領域を能率よく管理する為には．V-tape の実体は固定長のページに分割して．これらのページと．高速記憶及び補助記憶上で V-tape に動的に割付ける。図4には．ヘッドとページングの関係が簡単に示されている。ヘッドは．V-tape 上を単純に逐次的に動くことが多いので．前もって．隣りのページも高速記憶上に確保しておけば(look-ahead swapping)，on demand page 型の仮想記憶の場合に起る missing page fault はかなり減少できる。図5には．V-tape のハードウェアによるアドレス変換機構の概要が示してある。この Table の参照は．かなり複雑であるので．高速化を計る為に．head register に高速記憶上の物理アドレスをもたせている。

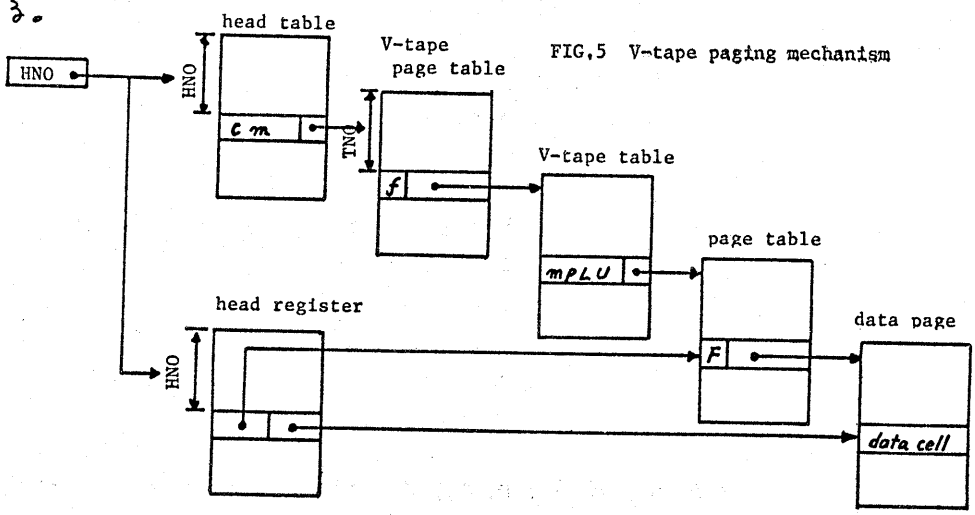


FIG.5 V-tape paging mechanism

4. アルゴリズムとシステムの解析

4.0 測定の環境

以下、V-tape上で実施した各種計算アルゴリズムの性能及びシステムの実測値は、V-tapeおよびデマンドページングシステムの両方で比較した。それぞれ、高速記憶は16KBに制限し、ページサイズは、512, 1024, 2048バイトの3種類について実測した。

4.1 行列の積

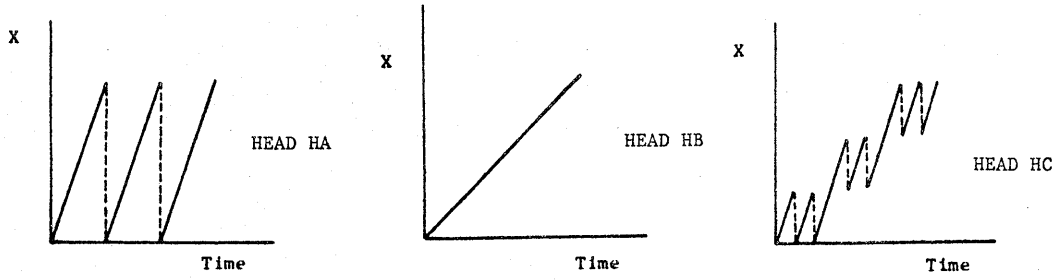
行列データのV-tape上の格納形式は、行ごとにデータをつめて一次元的に格納する"packed row storage"方式を用いた。行列A, Bの積Cは $c_{ij}=\sum_k a_{ik}b_{kj}$ で定義され、N次の正方行列である場合は、プログラムM1の様にして計算できる。しかし、このプログラムは階層構造をもつ記憶上では効率よく実行できない。そこで、ネストしたloopの入れ換え[11]をすることにより、プログラムM2の様に変換すれば、データのアクセスの局所性がよくなり、効率が改善される。プログラムM1では全体の計算に必要なステップはN^4に比例し、M2ではN^3に比例する。従って、大中にステップが軽減される。この様な変換は、最近のオプティマイズを行なうコンパイラでは、行なっている。さて、M2におけるデータのアクセスは局部的には、ほとんど逐次的アクセスであり、そのままV-tapeを用いて実行できる。このアルゴリズムでは、ヘッドが一定期間ごとに、V-tapeの特定の位置に戻る

```
program M1.
          DO 20 J=1,N
          DO 10 I=1,N
       10 C(I,J)=0
          DO 20 I=1,N
          DO 20 K=1,N
       20 C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

```
program M2.
          DO 20 J=1,N
          DO 10 I=1,N
       10 C(I,J)=0
          DO 20 K=1,N
          DO 20 I=1,N
       20 C(I,J)=C(I,J)+A(I,K)*B(K,J)
```

```
program VTM2.
// matrices A, B are on V-tapes TA, TB, and
   matrix C is on V-tape TC.
          HA=CREATEHEAD(TA)
          HA2=CREATEHEAD(TA)
          HB=CREATEHEAD(TB)
          HC=CREATEHEAD(TC)
          HC2=CREATEHEAD(TC)
          DO 10 J=1,N
          CALL PLACE(HA,HA2)
          CALL PLACE(HC2,HC)
          DO 20 K=1,N
       20 CALL VTWF(HC,0)
          DO 10 I=1,N
          CALL PLACE(HC,HC2)
          DO 30 K=1,N
                CALL VTRF(HA,DA)
                CALL VTR(HB,DB)
                CALL VTR(HC,DC)
                DC=DC+DA*DB
       30       CALL VTWF(HC,DC)
       10 CALL VTF(HB)
          CALL DELETEHEAD(HA,HA2,HB,HC,HC2)
```

FIG.6 Head movement



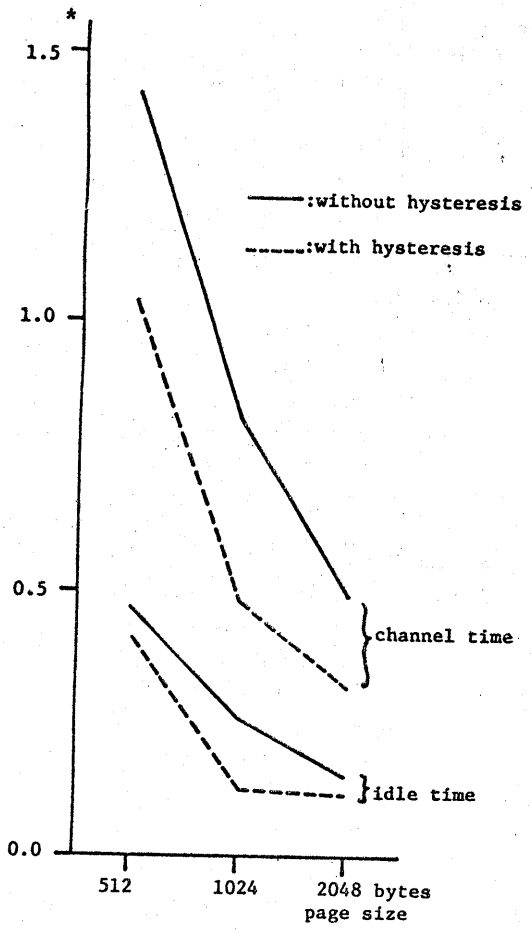


FIG.7 Hysteresis effect

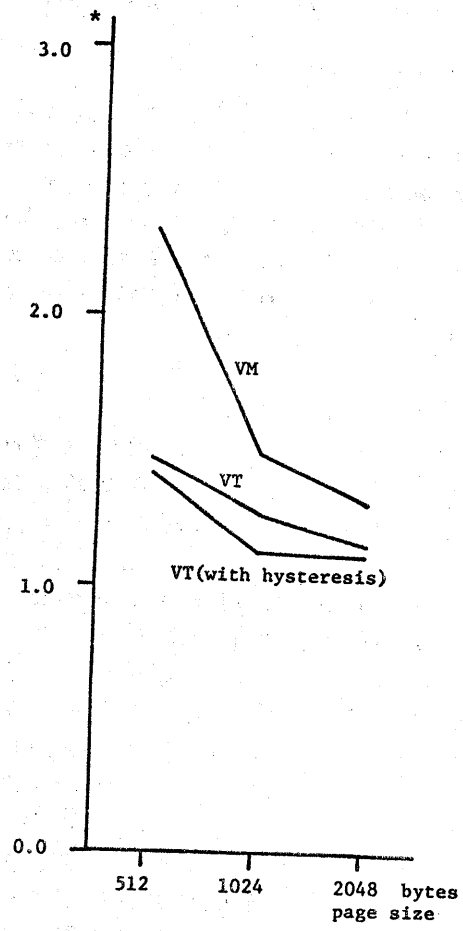


FIG.8 Speed of matrix multiplication
*normalized by cpu time

てくるので．そこにmarkの代りにヘッドを置けば更にspeedを上げることができる。このプログラムはVTM2として示す。VTM2を実行中のヘッドHA,HB,HCの動きを表わしているのが図6である。図8のVM,VTは．それぞれ．デマンドページング仮想記憶と．V-tapeとで実行した行列積の実行時間である(この計算だけ実行した時のelapsed time)。

V-tapeシステムでは、ヘッドに必要なページはlook-aheadしてswapしているので．ヘッドがページの境界でたびたび振動すると．余分なswap requestが発生して．システム全体に悪影響を及ぼす。これを防止する為には．swapのrequest発生とヘッドの動きの間にヒステリシスを入れてやればよい。VTM2をヒステリシスがあるときと．ない時で実行して．swapに消費された時間と．cpuのidle timeを測定したデータが図7に示してある。ヘッドが単純に一方向のみに進んでいる時は．ヒステリシスがあることは不利であるが．少し複雑な動きをする時は効果があることがわかる。

4.2 行列の対角化

連立一次方程式を解く際には．行列を対角化することが必要である。そこで．

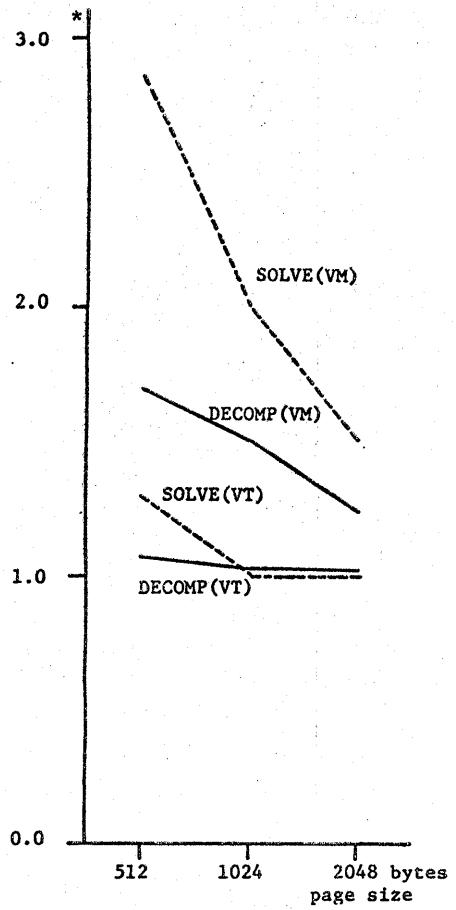


FIG.9 Speed of Linear Equation Solving

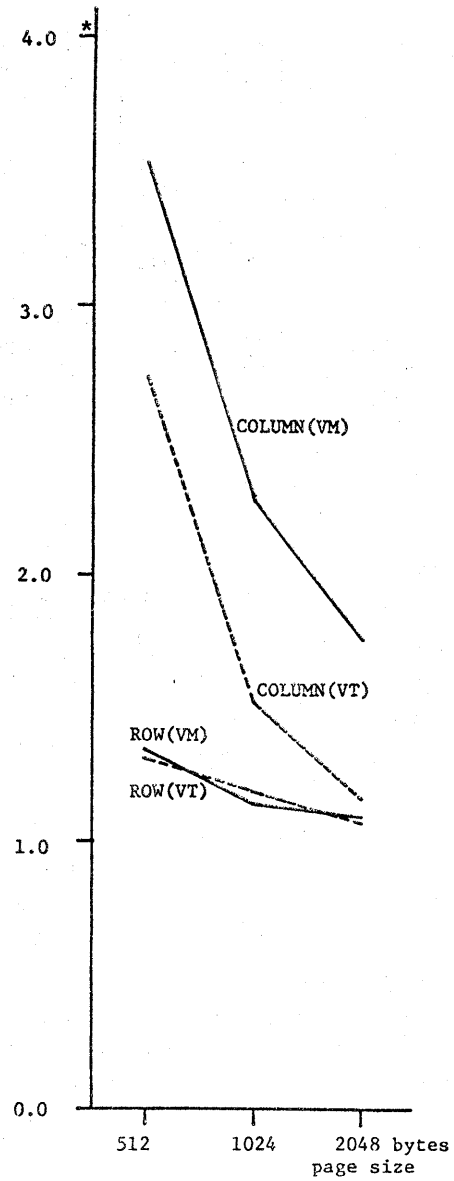


FIG.10 Speed of FFT

V-tape上で、Gauss の消去法によって行列を三角化し、対角化することを考える。この時は、完全ピボット変換は困難であるので、部分ピボット変換を行なうことにした。測定結果は、三角化のプログラム(DECOMP)と、後退代入のプログラム(SOLVE)について別々に行ない、図9に示してある通りである。反復法による解法や、帯状方程式の解法にもV-tapeは利用できるであろう。

4.3 2次元高速フーリエ変換(Fast Fourier Transform, FFT)

一般に1次元のFFTは、データ量がそれほど多くないので、real memory内で処理できるが、画像情報等の2次元FFTは、データの量が莫大であり、取扱に注意を要する。V-tape上で行なったアルゴリズムは、ページを2個用いた、inplaceの変換である。実際に実行したデータは、256×256点(64bit/点)のデータであり、測定は、行変換と列変換を別々に行なった。この結果は、図10に示してある。

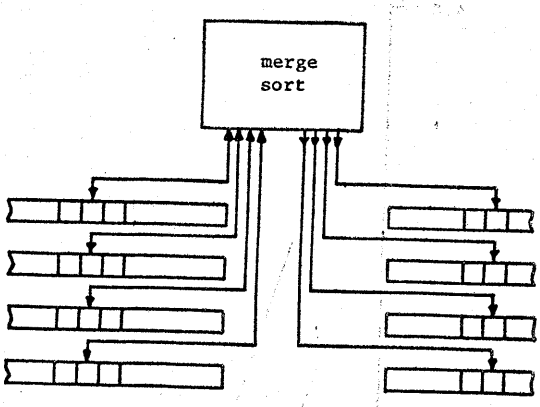


FIG.11 Merge sorting on V-tapes

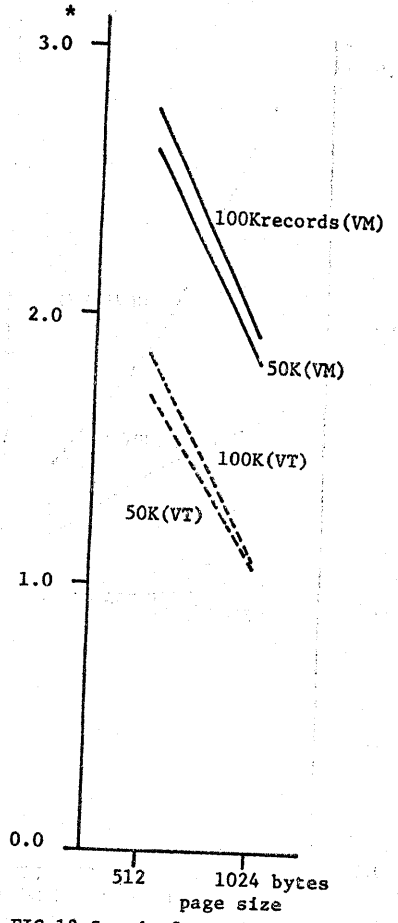


FIG.12 Speed of sorting
*normalized by cpu time

4.4 ソーティング

merge sorting は、最も単純でかつ能率のよいソート技法の1つであり、補助記憶を用いたシステム上で、多くのアルゴリズムが考案されている。2次のバランスマージソートをV-tapeを用いて実行するには、2本のV-tapeと4個のheadがあればよい。マージソートの能率は、マージの次数を上げれば良くなる。磁気テープの場合は、システムによって物理的制限がある為で、カスケードマージやポリフェイズマージ等の複雑なアルゴリズムの使用により次数を上げている。これらのアルゴリズムでは、1時期には1本のテープが出力用に使用され、残りのテープは入力に使用する様な工夫がされている。しかしながら、V-tapeでは、高速記憶の容量の許す限りの本数のテープが使用できるばかりでなく、ヘッドの数をテープの数よりも少なくすることで、バランスマージソートでも、マージの次数が上がらない。これらのことは、V-tapeのそれが物理的なテープよりも優位であることを示している。次に、V-tape上のソーティングをページング方式の通常の仮想記憶上でのそれと比較してみよう。仮想記憶上では、大きさSのデータを4次のバランス併合によりソートすると、2M個の配列が必要であり、2S～2MSの領域が必要である。しかし、V-tapeを用いると、ヘッドが読み終ったデータはeraseすればよいので、ソーティングに必要な領域は、ほぼSだけあればよい。従って、仮想空間を大幅に節約できる。又、ソーティングでは、V-tapeから消去された部分はswap outの必要がなく、又新しく書き込む領域はswap inする必要がないので、必要なスワップの量をオンデマンドページ型仮想記憶での量の約2/3に軽減できる。V-tapeによる4次のマージの概念図を図.11に、100000レコード、50000レコードのソーティング時の実測データを図.12に示す。レコードのサイズは16ビート、キーは乱数を使用した。

4.5 BCPL コンパイラ

BCPL[2]の INTCODE machine は、比較的単純な stack oriented な計算機である。ここでは、V-tape をファイルシステムの中に利用して、コンパイラで使用した結果を示す。BCPL コンパイラは、図13.に示す様に、4つのフェイズから成り、AE tree 以外は、すべて逐次形のファイルでよいので V-tape が使用できる。コンパイラと入出力部分の速度を測定した結果が、表2に示してある。ここには、TOSBAC40 のソフトウェアで処理(インターフリタ及び入出力処理)とハードウェアで処理した場合の結果が比較してある。表1は、INTcode machine の代表的な命令の実行速度を測定した結果が示してある。

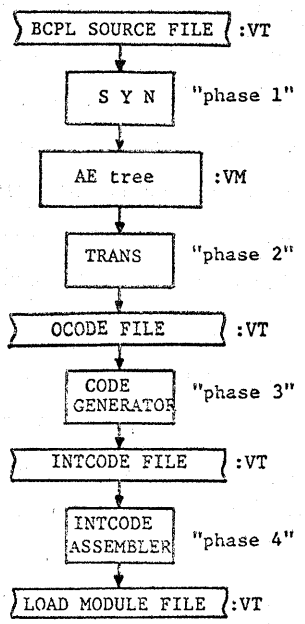


FIG.13 Phases in BCPL compiler

5. おわりに.

以上の応用に限らず、一般的なファイル入出力処理、更に Relational data base 等にも応用することが考えられる。補助記憶装置をすべて、仮想記憶という概念で取扱うには、segment という概念と V-tape という概念があれば、すべて統一的に取扱えるのではないかと思われる。

一般に計算処理を V-tape を用いて実行する場合は、アルゴリズムを変換する必要があり、やや複雑となるが、cpu の idle time 及び elapsed time はマルチプログラムの無い小さなシステムでもかなり減少できる。V-tape システムはページ方式仮想記憶の性能を改善する為のテープというデータ構造と、先行スワップを指定できる命令セットをもつシステムとみなすことができる。V-tape の概念を大型計算機に応用すれば、データスワップによる cpu の idle time を減少させることが可能であり、プロセス間の制御の移動が少なくなるので、OS のオーバーヘッドが減少する。マルチプログラムシステムにおいても、プログラムの多重度を下げることができるであろう。

TABLE.1 Speed of instruction

load	4.3	5.1	10.9	10.9
store	12.4	12.8	15.9	15.9
add	4.7	5.1	11.0	11.0
VTR	22.8			
VTRF	26.0			

all data in micro-second

TABLE.2 Speed of BCPL compiler

	phase 1	phase 2	phase 3	phase 4	total
cpu time	11.4 / 149.1	6.2 / 79.0	5.3 / 64.5	3.9 / 52.6	26.8 / 345.2
I/O routine (with idle)	0.1 / 12.3	0.1 / 4.1	0.5 / 13.3	0.1 / 7.9	0.8 / 37.6
channel	1.1 / 1.2	1.4 / 1.3	3.5 / 3.6	0.9 / 0.8	6.9 / 6.9

unit sec. BCPLmachine+VT / Soft.interpreter+soft.I/O

References:

1. Turing, A. M. On computable numbers with an application to the Entscheidungproblem. Proc. London Math. Soc., (1936).
2. Richards, M. A tool for compiler writing and system programming. SJCC,(1966),557-556.
3. Goto, E., Itano, K., and Mateev, L. A. Propositions for birtual multihead multitape processor. Proc. of 15th Annual Conference of IPSJ, Kyoto.(Nov.1974).
4. Sassa, M., and Goto, E. "V-tape", a virtual memory oriented data type, and its resource requirements. Research Report no.C-9, Tokyo Institute of Technology,(Jan.1977).
5. Mateev, L. A. A proposal for virtual multihead and multitape processor. M.S. Thesis, Department of Physics, Univ. of Tokyo, Japan, (Jan.1975).
6. Itano, K., Ida, T., Ikawa, M., and Ishihata, K. Virtual multihead multitape processor. Proc. of 16th Annual Conference of IPSJ, Tokyo, (Nov. 1975). (in Japanese)
7. Itano, K. Realization of a Processor with virtual tapes and its evaluation. Doctor of Science dissertation, Univ. of Tokyo, (Jan. 1977).
8. Cheney, C. J. A non-recursive list compacting algorithm. Comm. ACM 13, 11(Nov.1970), 667-678.
9. Knuth, D. E. The art of computer programming I, Addison-Wesley, (1968).
10. Goto, E. Ida, T., and Gunji, T. Parallel Hashing algorithms, Information Processing Letters, 6, (1977), 8-13.
11. Elshoff, J. L. Some programming techniques for processing multi-dimensional matrices in a paging environment. NCC, (1974).
12. Forsythe, G. and Moler, C. B. Computer solution of linear algebraic system. Prentice-hall, (1967).
13. Cooley, J. W., and Turkey, J. W. An algorithm for the machine calculation of complex Fourier series. Math. Comput. 19(1965), 297-301.
14. Singleton, R. C. A method for computing the fast Fourier transform with auxiliary memory and limited high speed storage. IEEE Trans. AU-15, 2(June 1972).
15. Lorin. H. Sorting and sort system. Addison-Wesley, (1975).
16. Ritchie, D. M., and Tompson, K. The UNIX Time-Sharing System, CACM,vol.17,(1974), 365-375.

APPENDIX List of V-tape instructions

1. tapenumber := CREATETAPE(erasemode)
2. DELETETAPE(tapenumber,...)
3. headnumber := CREATEHEAD(tapenumber)
4. DELETEHEAD(headnumber,...)
5. VTαβγ : head instructions
   αβγ are one of the letters:
   R(read), W(write), F(forward), B(backward)
   E(erase), or a void.
6. Random access extension
   VTFX, VTBX, VTRX, VTWX
7. marknumber := CREATEMARK()
8. DELETEMARK(marknumber,...)
9. PLACE(head/marknumberl, head/marknumber2)