計算機アーキテクチャ　83－13
（１９９０．７．１８）

階層メモリを有するスーパーコンピュータのためのＬＵ分解並列アルゴリズム

妹尾義樹、白戸幸正[1]、西直樹

日本電気㈱Ｃ＆Ｃシステム研究所、[1]日本電気技術情報システム開発㈱

大規模ＬＵ分解を高速に実行する並列アルゴリズムを開発し、実験によりその有効性を確認した。ＬＵ分解の対象は非対称密行列である。実験は通産省大型プロジェクトで開発された、ローカルメモリ、共有メモリ、半導体拡張記憶、磁気ディスクの４階層メモリを有する並列型スーパーコンピュータ上で行い、メモリ階層間のデータ転送を最適化するとともに並列化を行った。32768×32768の密行列（データ量４ＧＢ）の分解を１０時間４０分で実行することができた。また16384×16384行列の分解では４台のプロセッサを用いて１台の2.8倍の高速化が達成できた。さらに磁気ディスクを除いた３階層の実行では１台の3.2倍の並列化加速率が得られ、アルゴリズムの有効性が実証できた。

An LU Decomposition Algorithm for Parallel Supercomputers

Yoshiki SEO, Yukimasa SHIROTO[1], Naoki NISHI

C&C Systems Research Laboratories, NEC Corporation
[1]NEC Scientific Information System Development

4-1-1, Miyazaki, Miyamae-ku, Kawasaki, Kanagawa 213 JAPAN

A newly designed parallel algorithm for solving LU decomposition of huge dense matrices was developed for parallel vector supercomputers with a hierarchy of memory layers(i.e., local memories, shared memory, semiconductor extended storage and magnetic disk). Using four memory layers, an LU decomposition for a 32768x32768 dense matrix was calculated in 10 hours and 40 minutes on the HPP-LHS supercomputer system developed under the MITI(the Ministry of International Trade and Industry) Supercomputer Project. Required memory capacity for the gigantic matrix is 8GB, and the whole matrix data area was allocated to magnetic disk for this calculation. The execution speed with 4 processors was 2.8 times faster than that with 1 processor, even using a magnetic disk, and the algorithm was proved to be effective.

1．はじめに

ベクトル型のスーパーコンピュータはその超高速の演算能力により科学技術計算の分野では実用マシンとしての評価を定着させつつあるが、複雑な自然現象を解析するためにはさらに高速の演算能力と大量のメモリ容量が要求されている。

演算能力の向上に関してはプロセッサは並列化の方向に向かっており、ＮＥＣのＳＸ－３やＣＲＡＹ－２／３、ＸＭＰ、ＹＭＰはいずれもマルチプロセッサ構成となっている[1][2]。また、メモリ容量に関しては半導体高速補助記憶を用いたメモリの階層化が有力な手段であり、ほぼすべての商用機が半導体の拡張記憶をサポートしている。

これらの階層メモリを有する並列型スーパーコンピュータ上で主記憶に入りきらない大量データを扱う大規模数値計算プログラムを高速に実行するためには、メモリ階層間のデータ転送の最適化と並列化が重要となる[3]。

さらに、現在の並列型の商用機はローカルメモリを持たずにすべてのプロセッサが同一の共有メモリだけをアクセスする密結合方式が主流であるが、この方式では共有記憶へのアクセスの集中により結合台数に限界がある。さらに多数のプロセッサを結合しようとするとどうしても各プロセッサがローカルメモリを持たざるを得なくなる。この場合にはローカルメモリ－共有メモリというメモリ階層が存在し、拡張記憶を用いて計算を行う際には３階層のメモリ階層について各メモリ階層間のデータ転送を最適化する必要がある。

本論文では、基本計算として最も重要なＬＵ分解について、メモリ階層間データ転送の最適化と並列化を行う直接法のアルゴリズムを示す。過去においては２階層のメモリ階層を持つスーパーコンピュータ（シングルプロセッサ＋拡張記憶）上でデータ転送を最適化するＬＵ分解アルゴリズムが考案され、９６００元の連立一次方程式が解かれている[4]。また、メモリ階層のない密結合の並列計算機上でＬＵ分解を解く並列アルゴリズムもいろいろと開発されている[5][6]。しかしながら、ローカルメモリを有する並列型スーパーコンピュータ上でしかも３階層以上のメモリ階層を用いて大規模問題を高速に解くアルゴリズムは今回が始めてである。

本アルゴリズムは通産省スーパーコンプロジェクトで開発されたＨＰＰ－ＬＨＳスーパーコンピュータシステムを用いて過去に例のない32768×32768という大規模なＬＵ分解を実行し、その有効性が確認できた。この実機評価結果についても本稿で報告する。

2．ＨＰＰ－ＬＨＳスーパーコンシステム

図１にＨＰＰ－ＬＨＳスーパーコンシステムの概要を示す。本システムは４台のマルチプロセッサであり、最大演算性能は１０ＧＦＬＯＰＳである（１ＰＥは４ＧＦＬＯＰＳで、残りの３ＰＥは２ＧＦＬＯＰＳ）。各プロセッサは６４ＭＢのローカルメモリ（ＬＳＵ）を有し、１ＧＢの共有メモリ（ＣＳＵ）に接続されている。ＬＳＵ、ＣＳＵともにワードアドレスでアクセス可能であるが、ＣＳＵに対して高速にアクセスするためにはプログラマが明示的にブロック転送を指定する必要がある。

ＬＨＳは大容量高速記憶装置で、論理的には２の６８乗バイトという大規模な論理空間をサポートする。アクセスの単位は１ＭＢのブロックアドレスである。物理的には４ＧＢの半導体高速記憶部と１５ＧＢの磁気ディスク（バッキングストア）の２階層から構成されるが、ＬＨＳ制御プログラムの提供するワンレベルファイル機能[7]により、プログラマはバッキングストアの容量の単一ファイル空間と考えてＬＨＳを利用できる。つまり、アクセス（ＲＥＡＤ）すべきデータが高速記憶部に存在する場合にはハードウェアで実装されたアドレス変換機構により高速に（１ＧＢ／秒）アクセスが可能であり、高速記憶部に存在しない場合にはＬＨＳ制御プログラムにより自動的にバッキングストアから高速記憶部にデータが転送された後にアクセスされる。この場合にはＬＨＳへのアクセスの実効性能は１．１ＭＢ／秒である。

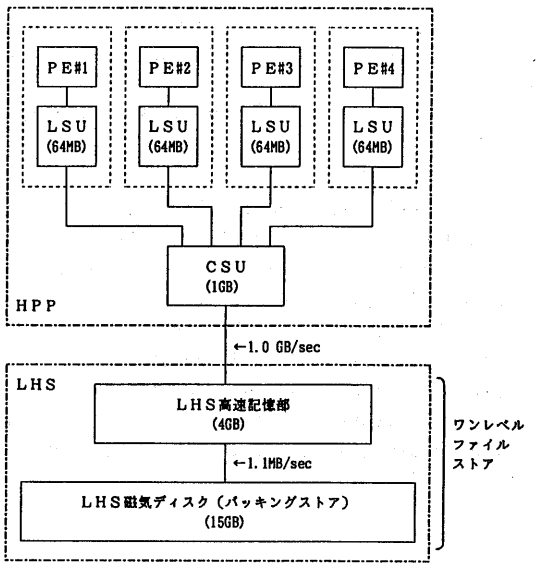

図１　ＬＨＳ－ＨＰＰシステム構成

３．ＬＵ分解のデータ転送／並列化アルゴリズム

ＨＰＰ－ＬＨＳシステムにおいてＬＵ分解を行うことを考える。普通は演算器により近い階層のメモリ（以降メモリの階層が上位であるという）ほど高速のデータ転送が可能であるが容量は小さくなる。したがって下位のメモリ階層ほどデータ転送の影響が大きく、こちらから優先してデータ転送を最適化すべきである。各記憶階層間のデータ転送を最適化するために、再帰型データ分割方式を考案した。これは外積形のガウス消去法アルゴリズムをもとにしてデータ依存関係（定義－参照の順番）が変化しないように実行の順序を入れ換えて問題を分割するものである。この分割により記憶階層間データ転送の最適化と同時に効率の良い並列化をも行うことができる。

３．１　前提条件

本論文で説明するアルゴリズムでは、ＬＵ分解の諸元とデータの格納およびＬＵ分解を行う基本算法について以下のものを前提としている。

まずＬＵ分解はｎ×ｎの２次元の係数行列に対して行うものとし、係数行列は非対称でかつ密行列であるとする。ＬＨＳにはすべての係数行列を収容できる記憶域が確保されており、ＬＵ分解終了後にはこのエリアに計算結果が格納されるものとする。この際には図２に示すようにＦＯＲＴＲＡＮのデータ格納法と同じく２次元の係数行列がカラムワイズにページ分割されており（各ページがｐ列で全体でｑページ）、ひとつ上位の階層の記憶システムとの間でブロック番号の指定によるブロック転送が可能であるとする。また任意の隣接する記憶階層間で直接にデータを転送することができ、他の記憶階層を経由する必要はないものとする。

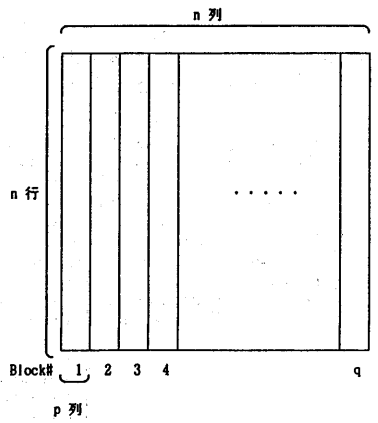

図２　最下位メモリ上の係数行列データ構造

ＬＵ分解の実行は基本的には直接法であるガウス消去法に基づいて行われるものとする。ガウス消去法は外積形でも内積形でもどちらでも用いることができるが、本稿では外積形をベースとして説明を行う。外積形のガウス消去法は図３に示すアルゴリズムであり、係数行列はａijでｎ＊ｎのサイズであるとする。演算の進行する様子を図４に示す。ｋ列に対する処理を

①　ａkkの逆数を計算し、
②　ａki（i=k+1,･･･,n）をこの逆数で割り、
③　②で計算したａkiを用いてk+1カラムからｎカラムまでのデータを更新する。

として、ｋが１からｎまで繰り返せば良い。

```
do k=1,n                          ---(1)
   a_kk=1/a_kk
   do i=k+1,n                     ---(2)
      a_ik=a_ik*a_kk
   end do
   do j=k+1,n                     ---(3)
      do i=k+1,n
         a_ij=a_ij-a_ik*a_kj
      end do
   end do
end do
```

図３　ガウス消去法（外積形）

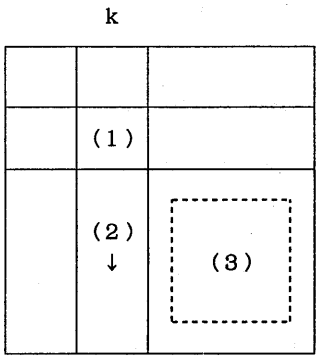

図４　データ参照パターン
(1),(2),(3)は図３のループ処理に対応

３．２　ブロック分割方式

まず、２階層の場合についてデータ転送の最適化について説明する。

上のガウス消去法の順序で演算を進めると、各ピボット列に対する操作の度にピボット列より右側のデータをすべてアクセスする必要があり、非常に多量のデータ転送が必要となる。

そこで処理のローカリティーを高めるために処理を分割する。すなわち、係数行列を複数のブロックにカラムワイズに分割し、全体の消去操作をブロック内消去とブロック間消去に分ける。各ブロックがｐ列でｑブロックに分割された場合のアルゴリズムを図５に示す。(a)がブロックを単位として進行する全体処理を示し、(b)、(c)が対応するブロック間消去、ブロック内消去の処理内容である。上位階層のメモリに２ブロック分の作業エリアが確保できれば、図６のように処理を進めることができる。ブロックｋの更新についてブロックｋを作業エリア２にフェッチし、作業エリア１にブロック１からブロック（ｋ－１）を順次フェッチしてブロック間消去を行う。そしてブロックｋのブロック内消去を行った後にブロックｋを下位メモリに書き戻す。このブロックを単位とする消去の進行は内積形ガウス消去法と同等であり、更新ブロックに対して参照ブロックが走査されるために外積形の処理に比べて走査されるブロック（作業エリア１）の書き戻しが不要であるという利点がある。

内積形ガウス消去法に対して、ブロック分割して演算のローカリティーを高める基本概念は文献［４］で述べられており、この方法を適用して９６００元の連立一次方程式が解かれている。

```
do k=1,q
    do j=1,k-1
      ブロックｊとブロックｋの (b)
      ブロック間消去

    end do

      ブロックｋのブロック内消去 (c)

end do
(a)ブロック分割アルゴリズム（全体構成）

do l=(j-1)*p+1,j*p
    do m=(k-1)*p+1,k*p
        do i=l+1,n
            a_im=a_im-a_il*a_lm
        end do
    end do
end do
(b)ブロックｊとブロックｋのブロック間消去

do l=(k-1)*p+1,k*p
    a_ll=1/a_ll
    do i=l+1,k*p
        a_il=a_il*a_ll
    end do
    do j=l+1,k*p
        do i=l+1,k*p
            a_ij=a_ij-a_il*a_kj
        end do
    end do
end do
(c)ブロックｋのブロック内消去
```

図５　ブロック分割アルゴリズム

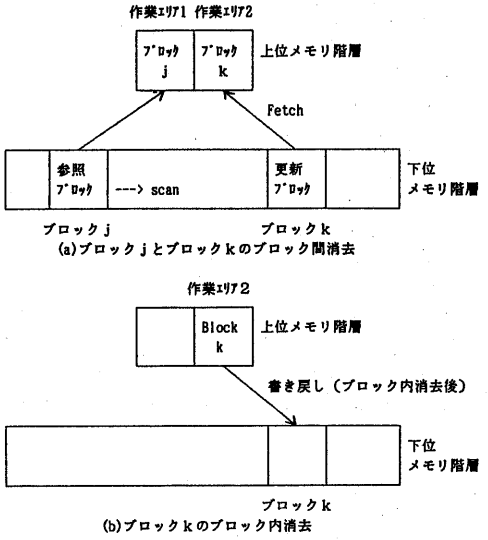

図６　メモリ階層間データ転送の様子

２階層のメモリ階層ではこの方法でデータ転送を最適化することができる。２階層メモリの上位メモリ（小容量で高速の方、たとえば主記憶）の容量がｍであったとすればｍ／２をブロックサイズとして下位メモリ（たとえば拡張記憶）をブロック分割し、上の問題分割法を適用すればよい。

３．３　再帰的ブロック分割

３階層の場合にはこのブロック分割を再帰的に適用すれば下位のメモリ階層間から順にデータ転送を最適化できる。つまり係数行列に対してブロック分割の手法を適用することにより最下位とその一つ上位のメモリ階層間のデータ転送を最適化することができる。このときのブロック内消去については、ブロックをさらにサブブロックに分割することによってさらに上位のメモリとのデータ転送を最適化できる。ブロック間消去操作も簡単にサブブロックへの分割が可能である。これを模

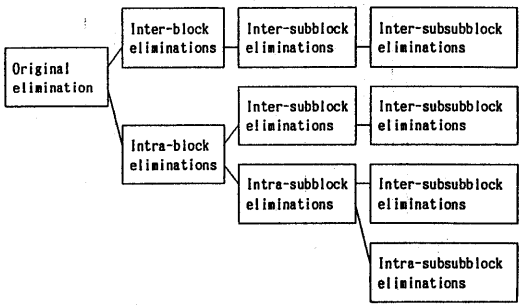

図７　ブロック分割

式的に表すと図７のようになる。４階層の場合にはサブブロックの処理をさらに細分化することにより対応が可能であり、この分割は原理的には分割後の最小のブロックが１列のブロックになるまで適用可能で、４階層以上の場合にも対応できる。

さらに、このブロック分割アルゴリズムでは係数行列の同じ列の要素はすべて同一のブロックに含まれるため、ピボッティングを簡単に行うことができる。

３．４　再帰型データ分割方式を用いた並列化

前節で述べたブロック化を繰り返し用いてブロックの細分化を進めると処理の大半がブロック間消去に帰着される。たとえばｎ＊ｎのサイズの係数行列に対してブロック化を繰り返し、あるレベルで分割されたサブブロックのサイズがｎ行＊ｐ列（ｎ＝ｐ×ｑ）であったとすると、処理すべきブロック内消去の演算量の合計はｎが大きい場合には

$n^3(2/3q^2+1/q-(q+1)/2q^2)$

となる。これはｑ（分割数）が大きくなり、$1/q^2$の項を無視すると$n^3/2q$となる。ガウス消去法における全体の演算量は$2/3n^3$であるから、仮にｑ＝３２とするとブロック内消去に要する演算量は全体のわずか２．３％となる。したがって、主にブロック間消去を対象として並列化を行えば良いことになる。共有記憶のレベルで分割されたブロックに対するブロック間消去の並列化について説明する。

共有メモリ上に参照ブロックと更新ブロックがロードされ、このブロック間消去が並列化される様子を図８に示す。更新ブロック、参照ブロックをさらにサブブロックに分割し、各ローカルメモリに２つのサブブロック用の作業エリアが確保できるとする。まず、更新ブロックの先頭から４つの連続するサブブロックをローカルメモリにフェッチし、これに対して参照ブロックを先頭から順次同一サブブロックを各プロセッサの作業エリア１にブロードキャストする。更新サブブロックの参照サブブロックを用いた更新が終了すれば更新サブブロックを共有記憶に書き戻す。以上の処理を順次、更新ブロックの残りのサブブロックについて繰り返す。ここでも参照サブブロックの走査を内側ループとすることが、共有記憶への書き戻しデータ量を減らすために重要となる。

上でブロック内消去の演算量が少ないことを述べたが、さらにプロセッサ台数が増えた場合の並列化においてはブロック内消去の並列化も重要となる。本稿では省略するがブロック内消去についても並列化が可能であり、以下で示す実験についてはこの部分も並列化を行っている。

また、並列化はすべて係数行列の列を単位として行われるため、一列の内部の処理はベクトル化によりベクトル演算器での高速処理が可能である。

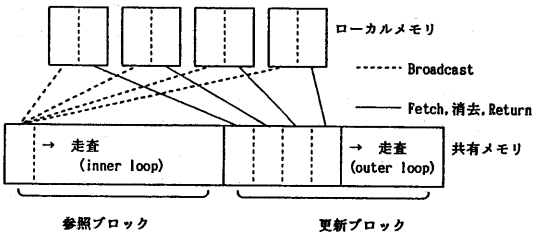

図８　ブロック間消去の並列化

４．実験

４．１　評価の前提条件

まず、ＬＵ分解の実測についての前提条件を示す。係数行列（初期データ）はプログラム中で生成しており、ディスク等から読み込んだ訳ではない。初期データをディスクにおいてスタートするとすれば、ディスクからＬＨＳ高速メモリ部へのデータ転送が余分にかかることになるが、ステージング制御によるプリロードを行えば、今回行った評価とほぼ同等の時間で演算を実行できる（別ジョブとしてのプリロード処理に時間はかかる）。実験に用いた係数行列$A_{i,j}$は次式により生成した。

$A_{i,j}$＝if（i<>j）　then　i　else　i+1

また、プログラムはスーパーコンプロジェクトで開発されたＰｈｉ１言語により記述した。ＬＨＳのアクセスはシステムで用意された関数で記述し、並列化はＰｈｉ１言語が提供する非同期サブルーチン呼び出しを用いて記述した。

４．２　実測結果

ＬＨＳ－ＨＰＰシステムを用いてＬＵ分解を行なった場合の結果を表１に示す。16384×16384と32768×32768の係数行列に対して実測評価を行なった。32768×32768の実行においてはローカルメモリ各３２ＭＢ、共有記憶７６８ＭＢ、ＬＨＳ高速メモリ部４ＧＢ、ＬＨＳバッキングストア８ＧＢを係数行列のためのメモリ領域として用いた。16384×16384の実行に際してはそれぞれの領域を４分の１として測定している。

表１により32768×32768の実行を１０時間４０分で実行できたことが分かる。このうちＬＨＳのアクセスに要した時間は約２時間１９分であり、全体の２２％である。この内の大半がＬＨＳのディスク－高速記憶部のデータ転送とＬＨＳの初期化で占められている。ＬＨＳの初期化とはあるブロックへの最初のアクセス時に要するアドレス変

換テーブルへの登録処理などである。演算精度の評価は分解されたＬ＊Ｕと分解前のＡの行列式の差をＡの行列式で割った値で行い、７桁以上の精度が得られている。

次に図８に16384×16384の実行における並列化加速率（対１ＰＥ性能向上比）を示す。実線は表１と同一条件で行った場合であり、４台で２．８倍の速度向上が得られている。データ転送を工夫すれば磁気ディスク装置を用いても４台のプロセッサを有効に利用できることが分かる。

破線はLHS高速メモリ部の容量を２ＧＢとして、ＬＨＳバッキングストアにアクセスが行かない条件で測定を行ったものである。この場合はシリアル処理であるＬＨＳアクセス時間が減少し、４台で３．２倍の速度向上が得られた。

５．まとめ

ＬＨＳ－ＨＰＰシステムを用いた実測評価において、過去に例のない３万２千元余りの膨大な元数のＬＵ分解を１０時間４０分で実行することができた。高速実行のポイントは４階層メモリの各階層間データ転送量の最適化と、並列実行にある。今回新たに開発したアルゴリズムはこれら２点のポイントを満たすものとして今後重要な役割を果たすものと考える。

係数行列のサイズ		16384*16384	32768*32768
ELAPSE time (秒)		7586	38387
HPP内処理時間 (秒)		5505	30062
LHSアクセス (HIT時)	転送量 (GB)	69	274
	時間 (秒)	66	263
	速度 (MB/秒)	1040	1040
LHSアクセス (MISS HIT時)	転送量 (GB)	1.9	7.8
	時間 (秒)	1741	6965
	速度 (MB/秒)	1.1	1.1
LHS 初期化時間 (秒)		274	1097
Error Ratio for determinant		1.3×10^{-6}	5.2×10^{-6}

表１　実測結果（４ＰＥ実行）　'90 １月　富士通(株)沼津工場にて実測
LHSｱｸｾｽ(MISS HIT)はＬＨＳバッキングストアー高速メモリ部間のデータ転送量、時間、データ転送速度を示す。

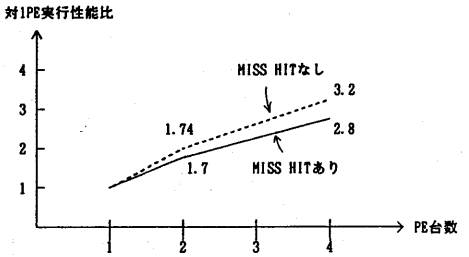

図９　16384×16384実行での並列化加速率

謝辞

本研究は通商産業省工業技術院大型プロジェクトの一環として新エネルギー・産業技術開発機構（ＮＥＤＯ）から委託を受けて、実施したものである。

ＬＵ分解プログラムの開発において様々な助言を頂いた京都大学の津田教授と岡部助手に感謝致します。またＬＵ分解プログラムのデバッグ、実機評価でお世話頂いた橋本氏を始めとする富士通(株)の方々に御礼申し上げます。

本研究の実施にあたりご指導頂いた、C&Cｼｽﾃﾑ研究所の石黒所長、大野主管研究員、ｺﾝﾋﾟｭｰﾀ・ｼｽﾃﾑ研究部の小池部長、中崎課長に感謝します。また、本プロジェクト遂行にあたり様々なご協力を頂いた関係諸機関各位に深く御礼申し上げます。

参考文献

[1] J. L. Larson, "Multitasking on the Cray X-MP-2 Multiprocessor", IEEE COMPUTER, Vol. 17, No. 7(July 1984),pp.62-69.

[2] T. Watanabe and H. Matsumoto and P.D.Tannenbaum, "Hardware Technology and Architecture of the NEC SX-3/SX-X Supercomputer System", Proceedings of Supercomputing '89 (Reno,Nevada,November 1989), pp.842-846.

[3] T. Tsuda and Y. Seo, "Supercomputing External Multidimensional FFT", Journal of Information processing, Vol.11, No. 2, 1988.

[4] T. Tsuda and Y. Okabe, "Use of Semiconductor Extended Storage as Extended Main Storage for Large-Scale Supercomputing'', Proceedings of the Second International Conference on Supercomputing Vol.1 of three volumes,(Santa Clara,Calif.,May 1987),pp.176-183.

[5] I. Bar-on, "A Practical Parallel Algorithm for Solving Band Symmetric Positive Definite Systems of Linear Equations, ACM Trans. Math. Softw., Vol.13, No.4 (December 1987),pp.323-332.

[6] V. Conrad and Y. Wallach, "Iterative Solution of Linear Equations on a Parallel Processor System", IEEE Trans. on Computers, Vol.C-26, No.9 (September 1977), pp.838-846.

[7] 西ほか「大容量高速記憶装置（ＬＨＳ）の基本方式」信学会CPSY90-6　1990 ６月