オペレーティング・システム　25－6
（１９８４．１２．１４）

階層化記憶システムにおける最適リプレース・アルゴリズム

景川　耕宇，武富　敬，末永　正
（九州大学大型計算機センター）

１．まえがき

今日では計算機利用の大半は端末機器からのものであるといっても過言ではない．しかし端末からの計算機利用では，利用方式上，大量の入出力は困難である．これを補う目的で補助入出力機器が常用されてきた．現在では主に，磁気ディスク装置が，この目的のために使用されている．さらに仮想ディスクとしての，ＭＳＳも利用されるようになってきた．データを大量に磁気ディスク装置上に持つことができるようになり通信回線を使用した場合でも比較的容易に大規模なデータ処理が可能となった．このような利用形態が急増するようになると，研究機関の計算センターにおいても，システム評価の際に演算処理能力以上に，磁気ディスクなど補助記憶システムの能力すなわち処理速度，容量，信頼性を配慮しなければならない時代となってきている．計算機システムの規模が大型になるに伴い，データの量が膨大なものとなり，大容量の補助記憶装置が必要になってきた．毎日大量のデータが作られては，消される．頻繁に参照されるものもあるが，中には作成された後，全然使用されず放置され，相当期間の後思い出されたように消去されるものも多い．技術の進歩に伴い，補助記憶装置１台当たりの記憶容量が大きくなり，データ保存のためのコストも大幅に低下していることも事実であるが，利用頻度が小さいからと言ってそのようなデータを，大容量であっても，ＴＳＳの利用には操作性の悪い磁気テープのような記憶媒体に記録せねばならぬ時代でもないことも確かである．ＭＳＳの開発もこのような時代の趨勢によるものであろう．定型業務の多いシステムでは，データの利用頻度を勘案し，システム利用者が記録媒体を選択することによって最適のデータ保存コストを実現することは，比較的容易に実施可能であろうが，大型計算機センターのように不特定多数の利用者が研究目的に利用するシステムでは，記録媒体によって利用方法を変えることを不便と感ずる利用者が多く適当な方法とはいえない．このようなシステムでは，補助記憶装置全体を階層化する方式が考案された．このようなシステムの下では利用者はあたかも，常に磁気ディスク装置を使用しているかのように感ずるが，データは必ずしも最上位のレベルの磁気ディスク装置に格納されているとは限らない．場合によっては下位レベルのＭＳＳなどに格納されている．必要とするデータがどのレベルの記憶媒体に存在させるべきかの決定は，その時点でのデータのそれまでの参照状況，またシステムの状態による適当な判断基準によって判定され，複数のレベルの記憶媒体間のデータの移動はシステムが自動的に行う．このシステムを以後階層化補助記憶システムということとする．特に誤解を招かない場合，単に記憶システム，あるいはシステムと称することとする．文献１），２），３）では，２レベル（磁気ディスク装置，ＭＳＳ）からなる記憶システムを効率よく動作させる各種の処理方式を実際のデータ参照記録を使用し，比較している．この報告では２レベルからなるシステムにおける最適な制御方式を示し，文献１），２），３）で比較された制御方式との関連を論ずる．

２．階層化記憶システムモデルの記述

２．１　使用者が直接利用できる記憶レベル

使用者がデータを利用する方法は一般的には各レベルの記憶媒体の性格によって異なる．このシステムが統合システムとして構成されるためには利用方法を統一化する必要がある．このため利用者が直接利用するデータは，最上位のレベルの記憶媒体にあることを前提としてシステムを構成するのが普通であるが，これは絶対的な条件ではない．むしろ下位レベルの利用方式で使用されている場合もありうる（ディスクキャッシュを付加したディスク装置）．ここで対象とするシステムでは，データを利用する行為（これを以後参照という）は最上位レベルの記憶媒体に対して発生し，さらに参照が完了するためには最上位のレベルの記憶媒体に該当するデータがなければならないものと規定する．従って，最上位レベルの記憶媒体にデータがない場合は下位レベルから上位レベルの記憶媒体へデータを転送することが必要である．以後上位レベル記憶媒体をML_1，下位記憶媒体をML_2と表現する．

２．２　使用者が参照するデータの単位

計算機システムでのデータの処理単位として，レコード，ブロックなどがあり，また磁気記憶装置のハードの面

からはトラック，シリンダーがある．ここではＯＳのデータ処理単位の一つであるファイルあるいはデータセットと呼ばれるものを処理単位として採用する．以後この単位をファイルと呼ぶことにする．ファイルがＭＬ$_1$に占める領域の大きさをファイルのサイズという．サイズの単位は通常ＭＬ$_1$の領域確保の単位（トラック，シリンダー）を採ることが多い．

２．３　各記憶レベル間のデータ転送の単位

データ参照は，ファイルの参照という事象で発生する．ＭＬ$_2$に存在するファイルを参照するためには，ファイルをＭＬ$_1$に転送する必要がある．使用者のデータ参照の単位としてファイルを採ったがこのような事象が発生した場合でも，必ずしも，ファイルをすべて転送する必要はないことが多い．たとえば，複数のメンバーから構成されるファイル（分割形順編成）の１メンバーを参照する場合，他のメンバーを転送する必要がない場合がある．さらに仮想記憶システムでプログラムをページに分割するようにファイルをいくつかの領域に分割し必要な領域のみを転送する方式も考えられる．逆に，転送の契機となるファイルばかりでなく，他のデータも同時に転送することがより効率的であることも考えられるが，ここでは転送の最小単位をファイルとする．主記憶を階層化した仮想記憶システムのデータ転送では，その１回の転送量は固定されているのが普通である．一方ここで定義した転送量は可変であることに注意する必要がある．

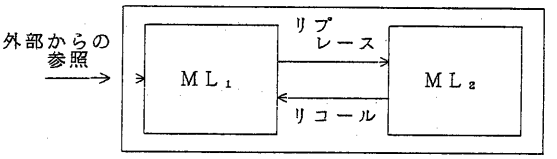

図　１　階層化記憶システム概念図

２．４　各記憶レベル間データ転送の契機と時点

２．４．１　ＭＬ$_2$からＭＬ$_1$へデータ転送

（リコール：ｒｅｃａｌｌ）

データの参照が発生した段階で該当ファイルがＭＬ$_1$に存在しない時（データ・フォールトの発生時）に該当ファイルの転送が行われる（デマンド・リコール）．

２．４．２　ＭＬ$_1$からＭＬ$_2$へのデータ転送（リプレース：ｒｅｐｌａｃｅｍｅｎｔ）

リコールの発生に応じＭＬ$_1$の領域を空けるためにリプレースを行うこともできるが，ここでは１日の中で比較的利用の少ない時間帯にまとめて行うこととする．利用者の参照と密接な関係はないので，転送は緊急を要しないからである．このため，同時に多量のデータを処理することになる．リプレースすべきファイルを判定し選択するアルゴリズムをリプレース・アルゴリズムという．

２．５　階層化記憶システムの効率

階層化記憶システム効率については関連する機能を使用した場合の利用性能とその機能を使用した場合の全システムの総合処理能力に与える影響の両面から，考慮する必要がある．前者は，ファイルの参照要求に対して，リコールを伴う場合に転送が完了するまでの待ち時間，利用可能な階層化記憶システムの容量で代表される．後者には，各記憶レベル間で行われるデータ転送のためのチャネル装置使用量，ファイルの状態量の収集，データ転送を判定するアルゴリズムのために使用される演算時間などがある．

３．階層化記憶システムの解析

３．１　ファイル参照履歴と参照確率

ファイルが期間（ｔ，ｔ＋Ｔ）に少なくとも１度は参照される確率を考え，これを参照確率と定義する．ファイルの参照はシステムの使用者数により増減し，利用者数は１日の時間帯によって著しく異なるのが普通である．従って，参照確率はｔ，Ｔによって異なる．しかし参照頻度が時期によらず一様に分布しているものと仮定すると，参照確率は区間長Ｔに関する増加関数となり，ｔを無視してよいことは容易にわかる．仮定を少し緩め，毎日同じ程度の利用者がファイルを参照しているものとし，区間長Ｔの単位として日（ｄａｙ）を採ると，参照確率はｔによらない．このように定義された参照確率は次のような性質を持つものと予想される．

① ファイルの領域確保量が大きければ大きい程参照されずに放置されている可能性が少ない（参照確率は大）．ファイルの利用料金がその領域確保量に比例して算定されていたり，利用者１人用の合計使用量に上限がある時には，特にこの傾向が顕著に表れる．

② ①と逆に領域確保量の少ないファイルは作成された後参照されずに放置されるものが多い（参照確率小）．

③ 参照されて日の浅いファイルは参照され易く（参照確率大），長期間参照されていないファイルは参照される可能性が少ない（参照確率小）．

④ ファイルが参照されないと参照確率の値が小になる．

⑤ 分割形順編成のファイルは多くのメンバーを含んでいるので，順編成のものより比較的参照され易い（参照確率大）．

以上により，参照確率はそれまでの参照履歴，ファイルのサイズ，ファイルの使用用途，データ編成などのファイルの性格によって異なり，そのファイルの性格によってある程度予想が可能であろう．このことから，参照確率は区間長Ｔばかりでなく，参照履歴Ｈ，サイズＳ，クラスＫ（参照履歴およびサイズを除くファイルの性格を包含したもの）によって定まると考えることができよう．ここで区間長Ｔを１に固定し，参照履歴の１要素としてＣＬＵ（Ｃｏｕｎｔ　Ｓｉｎｃｅ　Ｌａｓｔ　ｒｅｃｅｎｔｌｙ　Ｕｓｅｄ）を導入する．以後，特に断わらない限り参照確率の区間長を１日としている．ＣＬＵはファイルが参照された時に０にリセットされ，リプレース処理完了の直後に１が加算される．従って，前回のリプレース処理以後に参照されたファイルのＣＬＵは，次にリプレース・アルゴリズムが動作する時には０となっている．毎日リプレース処理を行う場合は，最近参照日以後の経過日数（Ｄａｙ　Ｓｉｎｃｅ　Ｌａｓｔ　Ｕｓｅｄ）と同じとなる．ここで，参照履歴をＣＬＵで代表させると，参照確率は次のようなものとなる．P_i を前回のリプレース処理時にＣＬＵ＝ｉであったファイルが今回のリプレース処理時にＣＬＵ＝０となっている確率と定義すると，参照確率をこの P_i で代用できる．これにより

$$P_i = \{CLU=0 \mid CLU' = i\} \qquad (3.1)$$

$$1-P_i = \{CLU=i+1 \mid CLU' = i\} \qquad (3.2)$$

ここで ´ は前回のリプレース処理時のファイルの状態であることを示し，クラスとサイズは変化しないものとした．なおクラスＣ，サイズＳ，ＣＬＵの値ｉの時の参照確率を P_i（Ｓ，Ｃ）と書くと

①と②はＳ´＞Ｓならば

$$P_i\ (S', C) \geqq P_i\ (S, C) \qquad (3.3)$$

③と④はｋ＞ｉならば

$$S_i\ (S, C) \geqq P_k\ (S, C) \qquad (3.4)$$

⑤ は

$$P_i\ (S, \text{分割型順編成のクラス}) \geqq P_i\ (S, \text{順編成のクラス}) \qquad (3.5)$$

であることを示している．なお，九州大学大型計算機センターで１９８２年６月から１９８３年２月まで測定した参照確率を表１に示す．

３．２　ファイル・リコールのためのコスト

ファイルがリコールされることによるコスト（リコール・コスト）は一般的にはファイルの状態によって異なり，該当するファイルを転送に要する時間に関連する要素と転送を余儀なくされたため利用者の抱く不満足感に関連する要素によって決定されよう．前者についてはファイルのサイズによる所が強いものであり，後者については参照履歴と強い相関があるものと考えてよいから，参照履歴に関連する要素と置き換えることにする．従って，リコール・コストは，転送されるファイルのサイズと参照履歴の関数となるとして一般性を失わないであろう．参照履歴としてＣＬＵを採り，リコール・コストの性質について考察する．

ＣＬＵ＝ｉ，サイズ＝Ｓのリコール・コストをＵ（ｉ，Ｓ）とすると

①サイズの大なるファイルはリコール・コストも大である．ＵはＳに関して増加関数となる．すなわち，ａとｂは負でないコンスタントとして

$$U(i, S) \propto G(S) \fallingdotseq a + b \cdot S \qquad (3.6)$$

のような形で近似できる．ここでG（S）はSの増加関数．

②参照された後日の浅いファイルがリプレースされリコールされると利用者の不満足感は非常に大きいが，参照されて日の経たファイルについては逆に，利用者はやむをえないものと考えるはずである．

サイズを固定するとCLUの値に関し減少関数となる．これもCLUの値iが小なる場合極めて大なる値を持ちiの値が大きい場合ほとんど差がないような関数であろう（図2　実線）．これをF（i）とし，リコール・コストUはF（i）に比例するものとする．

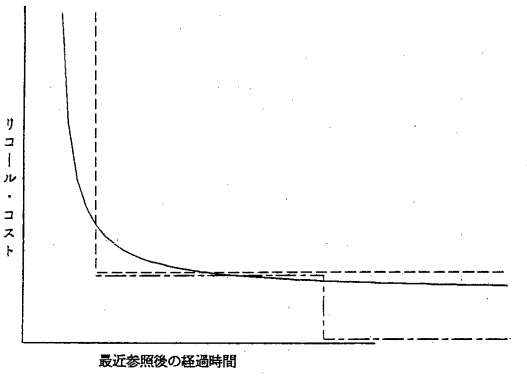

図　2　リコール・コスト

リコール・コストはサイズ項とCLR項の積の形に書け

$$U(i, S) = G(S) \cdot F(i) \fallingdotseq (a + b \cdot S) \cdot F(i) \quad (3.7)$$

となる．

実際のリコール・コストはシステム運用環境およびシステム管理者の方針によってシステム管理者によって決定される性格のものである．例として磁気ディスク装置とMSSを使用するシステムを考えよう．リコールの時間はMSSのカートリッジをセットアップする時間に支配され，サイズにはよらないものと見なしリコールの回数をシステム効率の重要な要素と考える場合には，ファイルのサイズに関する項を無視することができる．また，参照された後，時間（日）のあまり経過していないファイルはリプレースをするべきでないと考える場合は，F（i）の項を，iの値が小さい場合，特に0の場合にはF（i）を無限大とすることによってシステムに管理者の意思を反映できる（図2　点線）．逆に，参照後ある程度以上時間が経過したファイルはすべてリコールされるべきである場合も容易に処理できる（図2　鎖線）．

参照確率（%）

サイズ / CLU	40KB以下	40～160KB	160～640KB	640～2560KB	2560KB以上
1日	41.0	41.5	44.5	48.3	53.5
2日	18.2	18.7	20.0	23.1	25.6
3日	12.4	12.0	13.2	14.5	17.6
4～6日	7.9	8.0	8.8	10.4	12.1
7～12日	4.4	4.8	5.1	6.0	6.9
13～18	2.8	3.1	3.2	3.6	4.5
19～24日	2.3	2.4	2.5	2.8	3.1
25～48日	1.6	1.6	1.8	2.1	2.2
49～72日	1.0	1.2	1.1	1.3	1.2
73日以上	0.5	0.5	0.6	0.7	0.6

表　1

3．3　最適リプレース・アルゴリズム

定義

D　ファイルの集合．

D_1　ML_1 に含まれるファイルの集合

D_2　ML_2 に含まれるファイルの集合

d　Dの要素

P　ファイルの参照確率

S　ファイルのサイズ

U　D_2 に含まれている該当するファイルをML_1 に転送するコスト（リコール・コスト）

TS　D_1 に含まれるファイルのサイズの合計

C　D_2 に含まれるファイルが参照される時のコストの期待値

以上の定義より

$TS=\Sigma\ S \mid\ d \varepsilon D_1$ (3.8)

$C=\Sigma P \cdot U \mid\ d \varepsilon D_2$ (3.9)

リプレース・アルゴリズムは以上の定義を使用するとファイルの集合Dを集合D_1と集合D_2に分割する方式のことといってよい．2．で述べたモデルで効率のよいシステムを構築するために，ファイル参照時に該当ファイルを1個のみリコールし，ファイルをまとめてリプレースする場合のシステムでのコストの期待値を定義し，コストの期待値を最小ににするリプレースアルゴリズムを調べる．ここで

TS（A）　　アルゴリズムAによって定まるD_1に含まれるデータセットのサイズの合計

C（A）　　アルゴリズムAによって定まるD_2に含まれるデータセットが参照される時のコストの期待値

と定義して，次の2種類の分割アルゴリズムについて考えてみる．

<u>アルゴリズムOPT</u>

域値Lを定め，H＜LなるファイルをすべてD_2に含め，H≧LなるファイルをD_1に含める．ここで

$H=P \cdot U / S$ (3.10)

<u>アルゴリズムOPT′</u>

H＜LなるファイルをすべてD_2に含め，H＞LのファイルをD_1に含める．H＝Lなるファイルについては他の条件によってD_2，D_1のいずれかに含める．

補題）

$x>0$，$y>0$　である2つ組（x，y）の有限集合Dを部分集合D_1，D_2に分割し，S，Cを以下のように定義する．

$D=D_1+D_2$ (3.11)

D_2の要素（x，y）のx，yそれぞれの和を，

$S=\Sigma\ x$ ，$C=\Sigma\ y$ (3.12)

とする．ここで

$(x_1, y_1) \varepsilon D_1$，$(x_2, y_2) \varepsilon D_2$ (3.13)

ならば，

$y_1 / x_1 > y_2 / x_2$ (3.14)

となるように分割を定義し，この分割により定まるS，CをそれぞれS_0，C_0とすると　$S \geqq S_0$　かつ　$C_0 > C$　とする分割は不可能である．

<u>証明</u>

上で定義された分割集合D_1，D_2の部分集合をそれぞれE_1，E_2とし，

分割　$F_1=D_1-E_1+E_2$，$F_2=D_2-E_2+E_1$

を考えよう．この分割によって定義されるS，CをそれぞれS'，C'とする．

$(e_1, f_1) \varepsilon E_1$，$(e_2, f_2) \varepsilon E_2$ とし

$m_1=min(f_1/e_1)$，$m_2=max(f_2/e_2)$

とすると，定義より

$m_1 > m_2$　また

$\Sigma f_1 \geqq m_1 \cdot \Sigma e_1$　，　$\Sigma f_2 \leqq m_2 \cdot \Sigma e_2 \leqq m_2 \cdot \Sigma e_1$

∵　$S'-S_0=\Sigma e_1-\Sigma e_2 \geqq 0$

$C'-C_0=\Sigma f_1-\Sigma f_2 \geqq m_1 \cdot \Sigma e_1-m_2 \cdot \Sigma e_1$

$=(m_1-m_2) \cdot \Sigma e_1 > 0$

従って空でない部分集合E_1，E_2をどのように選択しても，$S' \geqq S_0$　かつ　$C_0 > C'$　とすることはできない．

証明終了

補題により次の定理を導くことができる．

最適アルゴリズム定理

アルゴリズムＯＰＴによるファイルの分割によってML_1を占める記憶容量，参照によるコストの期待値が定まる．これをＴＳ（ＯＰＴ（Ｌ））．Ｃ（ＯＰＴ（Ｌ））とする．ＴＳ（Ａ）＜ＴＳ（ＯＰＴ（Ｌ））でかつ，Ｃ（Ａ）＜Ｃ（ＯＰＴ（Ｌ））とするアルゴリズムＡはない．

証明

補題におけるｘ，ｙをそれぞれファイルのＳ，Ｈとする．ここでＨ＝Ｐ・Ｕ／Ｓ

ML_1，ML_2に存在するファイルのＨの値をそれぞれH_1，H_2とすると，

$$H_2 \leq H_1 \qquad (3.15)$$

を満足するように分割を定義し，ＳをML_2を占める記憶容量とすることにより補題を適用できる．

証明終了

ここでファイル集合Ｄが与えられアルゴリズムＯＰＴによりＬが与えられると，ＴＳ（ＯＰＴ（Ｌ））が決定する．すなわち，ＬからＴＳへの写像は必ず定義されるが，逆は必ずしも定義されるとは限らない．すなわち，あるＴS_0を決めてもＬを決定することは一般的にはできないことに注意しなければならない．なおアルゴリズムＯＰＴ′を使用し，域値Ｌと等しいパラメター値を持つファイル群をD_1，D_2のいずれかに含めることを許すように条件を緩めるとしても最適アルゴリズム定理は成立することは容易にわかる．この場合，与えられたＴS_0に対し，Ｔ$S \leq TS_0$を満足させ，Ｈの値が域値Ｌに等しい他のどのファイルを移動しても，Ｔ$S > TS_0$となるようにできる．このようにして得たＳに対して域値Ｌよりパラメター値の大きなファイルであっても，Ｕのより小なる値を持つファイルと入れ替えることにより，Ｔ$S < TS' \leq TS_0$を満たしかつ$C > C'$とすることができる場合がある．従って，アルゴリズムＯＰＴ（ＯＰＴ′）は，Ｔ$S \leq TS_0$の制約条件の下でＣを最小にするという意味では，最適ではない．しかし実際ＴS_0はファイルの大きさに比較しきわめて大でありML_1にもML_2にも極めて多数のファイルが格納されているので境界で，少々のファイルの入れ替えを行ったとしても，Ｃの値の差はたかだかファイル１個分である．従って，多数のファイルを対象とする場合はＴ$S \leq TS_0$の制約条件の下でＣを最小にするファイル選択アルゴリズムとしてアルゴリズムＯＰＴ（ＯＰＴ′）を採用するのが適当である．

４．　最適リプレース・アルゴリズムの適用

４．１　参照確率の推定

前回の観測で同一クラス同一サイズのファイルの全体の中でＣＬＵ＝ｉであったファイルの数の比率がn_i（ｔ－１）であったとすると今回の観測では

$$n_0(t) = \sum_{i=1} P_i \cdot n_i(t-1) \qquad (4.1)$$

$$n_{i+1}(t) = (1-P_i) \cdot n_i(t-1) \quad i > 0 \qquad (4.2)$$

となる．このシステムが平衡状態にあるものとすると

$$n_i = \lim_{t \to \infty} n_i(t) \qquad (4.3)$$

とし

$$n_0 = \sum_{i=1} P_i \cdot n_i \qquad (4.4)$$

$$n_{i+1} = (1-P_i) \cdot n_i \qquad i > 0 \qquad (4.5)$$

以上より

$$n_0 = 1/C \qquad (4.6)$$

$$n_i = \prod_{k=1} (1-P_k) / C \qquad i > 0 \qquad (4.7)$$

ここで

$$C = 1 + \sum \prod (1-P_i) \qquad (4.8)$$

このように，参照確率から平衡状態におけるファイルの数の比を求めることができる．逆に，同一クラス，同一サイズのファイルの平衡状態にある数を測定し，（４．５）式により，参照確率を推定できることがわかる．ここでは，例として，ＣＬＵをパラメターとした参照確率を推定する１方法を述べたが，最適リプレース・アルゴリズムの証明では，参照確率の推定方法を特定していない．このためクラス，サイズ別に異なる方式で参照確率

を推定しアルゴリズムを適用できる．

4．2　リプレースの方式

実際のリプレース，すなわちML_1からML_2への転送すべきファイルの選択には，次の2方式が考えられる．

なお，最適リプレース・アルゴリズムではファイルのリプレースをの可否を判定するために，

$H = P \cdot U / S$

を使用する．これを最適判定関数とし，このような目的に使用するファイルの関数を判定関数と定義する．

① 域値Lを決め，判定関数$H \leq L$を満足するデータセットをML_1から2次ディスクに転送する．

② 安全率を見越し，リプレース処理直前に1次ディスク記憶容量が予定した範囲に納まるようにリプレース処理後，ML_1に存在するデータセットのサイズの合計の最大値ＬＳを決め，
判定関数の値の小なる順にML_2に転送する．

①では，リプレース処理後の1次ファイルのサイズは大きく変化する可能性があること，このためリプレース処理後のML_1に存在するファイルの記憶容量をある範囲内に納めるためにはLの値をリプレースする時に，変化させることが必要である．一方，②では判定関数によりファイルを順序付ける手順が入るので，①と比較し処理時間がより必要となるなどそれぞれ長短がある．いずれの方式を採用するとしても，ML_1に存在するファイルすべてについて，参照履歴，サイズに関する情報を調べる必要があること，さらにこの処理は，比較的，計算機システムの負荷の少ない時間帯に行われることを考慮すると，システムに与える負荷の面では，差異はないようであるが，方式②は方式①と異なりファイルを順序付ける必要があるので，集中的に処理しなければならないという欠点を持ち運用をかなり制限することに注意しなければならない．最適リプレース・アルゴリズムは，ファイルをML_1とML_2に分割する方式であることを述べた．このことはリプレース処理を行う場合，ML_1からML_2への転送ばかりでなく，ML_2からML_1への転送を必要とする場合がありうることを示している．しかし，ML_2にすでにリプレースされているファイルは，リプレースされた後参照されていないので，3．1　④を仮定すれば参照確率がより大となることはなく，リプレース処理の直前でも，（3.11）式を満足してる．従って，方式①を採る場合，ML_2の情報を調べる必要がないが，方式②ではリプレース処理を開始する時に，すでにML_1に格納されているファイルの全記憶容量が条件を満たしていることがありうる．たとえば，大量のファイルが消去された場合がこれに当たる．この場合，リプレースをする必要がない．このようなことが発生することは通常極めて稀であるが，ML_2にあるファイルの情報を調べ逆の転送をするかどうかは，システム管理者の方針によるであろう．

4．3　従来のアルゴリズムとの比較

①　ＬＲＵ法

この方式では，参照されてから判定関数として，ＣＬＵに関して減少する関数を使用する．例えば，

$H = 1 / CLU$

を判定関数とする場合と等価である．最適判定関数とと対応させると，この方式が最適となるファイル利用環境では，

$F(CLU) = P \cdot U / S$

すなわち，

$P = F(CLU) \cdot S / U$

とならなければならない．（ここでＦはＣＬＵのについて減少関数とする）すなわち，

・リコール・コストＵがサイズＳに比例する

・ファイルのサイズがほとんど同じであってＵが定数である

ような利用環境では，最適な方式となる．しかし，ファイルのサイズは極めて広い範囲に分布していること，ML_2としてＭＳＳを使用する場合は，すでに3．2で述べたようにリコール・コストのサイズに比例する項は支配的でないので，効率の良い方式ではない．

②　ＬＲＵ法にサイズを考慮する方法

判定関数として

$H = 1 / (S \cdot CLU^{\alpha})$　　$\alpha > 0$

を使用する方法である．この方式では

$P = 1 / (U \cdot CLU^{\alpha})$

の場合に最適判定関数となる．ここでUを定数とすると，

$P \propto 1 / CLU^{\alpha}$

となり，（3．4）式を満足しているので，参照確率のファイルのサイズによる差異が大きくない場合には，αを適当に設定することにより参照確率の有効な近似となり，判定関数として十分に使用可能なものとなる．文献2）では$\alpha = 1.4$，　3）では$\alpha = 1.0$とした場合が，この方式での最も良いαの値であると報告されている．

③　次に参照されるまでの期間の期待値を使用する方式

文献2）では，CLU＝iの値を持つファイルが次に使用されるまでの期間の期待値をE（i，S）とし，判定関数を

$H = 1 / E(i, S) \cdot U / S$

とすれば，非常に効率の良い方式であると報告している（Uが定数の場合）．この方式では，参照確率を

$P \propto 1 / E(i, S)$

と近似することになる．この近似は（3．3），（3．4）式を満足する．

5　むすび

現実の階層化記憶システムでも，仮想記憶制御方式との類似性により，安易にLRU法を採用していることが多い．ファイルのように領域確保量が広い範囲に分布している場合は，サイズの大きなファイルを優先的にリプレースすることがリコール率（参照されたファイルとリコールされたファイルの比率）を低くするための有効な方法であることを直感的に理解できるが，このような方式では大きなファイルは極端に短期間のうちにリプレースされ，リプレースとリコールが繰り返し行われることを恐れているからに他ならない．このことはまた，階層化記憶システムの制御方式に対する明確な指針がなかったことを示している．この報告では，ファイルの利用状況から参照確率を推定し，そのシステムの総合的な負荷を配慮してリコール・コストを決定すれば，デマンド・リコール方式を採るシステムでは，最適リプレース・アルゴリズムを利用して，最適判定関数を選択できるという明確な指針を示した．しかし，この方式では，リプレース処理によるシステムへの負荷，ファイル利用者の情報を利用して関連するファイルをあらかじめリコールするプリ・リコール方式等について考慮されていないが，これらは今後の問題としてまだ残されている．また，この報告でML_1として象徴的に名付けた記憶媒体の技術の進歩による記憶容量の増加ですら利用者の補助記憶の利用量の増加には追い付けないのが現状である．従って，階層化記憶システムの重要性は今後さらに増すものと考えられる．このためには，消極的に参照情報を利用してシステムを制御するだけでなく，ファイルの参照構造を変えることによって，積極的に参照の局所性を上げ，システム効率の向上を検討しなければならないであろう．

参考文献

1）Smith A. J. Long term file reference patterns and their application to file migration algorithms. IEEE Se－7，4（Jul. 1981），403－416，

2）Smith A. J.：Long term file migration：Developement and evaluation of algorithms. Comm. ACM 24，8（Aug. 1981），521－532

3）Lawrie D. H.，Randal J. M.，Barton R. R.：Experiments with automatic file migration. Computer 15，7（July. 1982），45－55

4）景川耕宇　保存データセット利用統計について　全国共同利用大型計算機センター研究論文集　3，（Dec. 1981），93－97