2レベル階層化補助記憶システムにおける最適制御方式について†

景川耕宇†† 武富 敬†† 末永 正†††

大規模オペレーティングシステムでは，補助記憶システムの階層化が，すでに多くのシステムで，実現されている．しかし，このシステムの効率を左右する記憶階層間の転送データを選択する制御方式が，仮想記憶制御との類比で，LRU 法 (Last Recently Used) の域をでていないため，効率のよいシステムとなっていない．この論文では，ファイルの参照確率および参照時にファイルを，リコールすることによる利用者の不満足感を含めたコストを定義する．さらに，コストを最小にするマイグレーションすべきファイルの最適な選択方式を提案する．

1. はじめに

計算機システムの利用形態がバッチ処理から TSS に移行するに伴い，磁気ディスク装置の使用量が急速に増加した．磁気ディスク装置の進歩による記憶容量の増加も著しいが，何らかの使用の制限をしない限り，その需要増を満たすことができないのが現状である．しかし，このような，共用磁気ディスク装置上のファイル(以後単にファイルという)の中で，継続して使用されているものは，非常に少ないものと予想される[6]．この点の解明のため，九州大学大型計算機センターにおいて，8カ月間，ファイルの利用状況を調査した．この結果，ファイル利用にも，仮想記憶参照における現象に類似した局所性が見られ[4]，同一時期に利用されるファイルはごく一部に集中し，他の多くのファイルは長期間使用されていないことがわかった．ほとんど利用されないファイルを，MSS (Mass Storage System) などの下位レベルの記憶媒体（仮想ディスクという）に移し（マイグレーションという），その使用領域を空け，実際に使用する時に，高速アクセスの可能な磁気ディスク装置（実ディスクという）に移動させること（リコールという）ができれば，実ディスクを有効に利用することができ，利用者には，あたかも，実ディスク用の記憶媒体が大量に増加するように見えることとなる．このようなファイル移動を自動的に行うシステムを，階層化補助記憶システムと呼ぶ．階層化補助記憶システムの概念図を図1に示す．

† The Optimal File Selection Policy for Two-Level-Hierarchical Auxiliary Storage by KOU KAGEKAWA, HIROSI TAKETOMI (Computer Center, Kyushu University) and TADASI SUENAGA (Educational Center for Information Processing, Kyshu University).

†† 九州大学大型計算機センター
††† 九州大学情報処理教育センター

これまで，多くの計算機システムの上で，補助記憶装置の階層化が実現されている．仮想記憶方式と同様に，このシステムでも，利用者が同一時期に使用するファイル数，使用記憶容量の制限などの方法で，ディスクシステム使用の局所性を積極的に高めることは，システムの利用効率向上の一方法であるが，これを利用者に期待することはできない．ディスクシステム使用の局所性向上は，ファイル管理システムの改善による方法が最も有効と考えられるが，まだ実現は困難である．これまで，ファイル参照記録を調べ，参照間隔の分布関数を予測し，それにより，効率のよいシステムを構築する試み[1),2)]，あるいはマイグレーションすべきファイル選択方式のシミュレーションを種々行って，試行錯誤的に効率のよい方式を発見しようとする試み[3)]がある．確かに，不特定多数の多様な利用者にサービスをする計算センターにおいては，利用者のファイル利用動向を統計的に調べ，その情報をもとにマイグレーションするファイルを選択し，これにより，リコールするファイルを少なくすることが，効率のよいシステムを構築する最も現実的で，有効な方法である．しかし，実現されているシステムでは，最後に参照あるいは更新（以後まとめて参照という）された後の経過期間の大なるファイルを，優先的にマイグレーションする方式（LRU 法）を採用している．この方式は非常に単純で，適用が容易ではあるが，後に述べるように，階層化補助記憶システムのように，ファイルごとにデータ転送量が異なるシステムには，必ずしも適当な方式ではない．システムの効率が，マイグレーションすべきファイルを，選択する制御方式（選択方式という）によって大きく左右されることは理解できるとしても，LRU 法を採用しているという事実は，ファイル選択方式に関する理論が，確立されてい

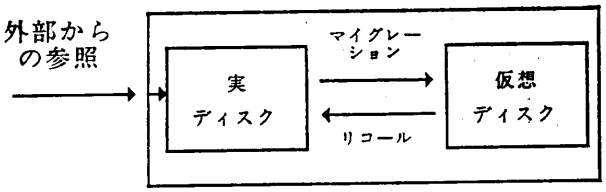

図 1 階層化記憶システム概念図
Fig. 1 System configuration of two-level memory hierarchy.

なかったことを示すことにほかならない．参照間隔の分布関数を推定する方式[1),2)]などのように，長期間のデータ収集を必要とする方式は，参照形態の期間変動に迅速に対応することが容易でないため，実際のシステムへの適用が非常に難しい．一方，ここで提案する方式は，LRU 法と同程度に適用が容易であり，利用形態の変動に追従しやすいという特徴を持っている．

以下，2章で階層化補助記憶システムの効率について考察し，3章でファイルの参照確率と，リコール・コストの概念を導入し，マイグレーションすべきファイルの最適な選択方式（以後最適選択方式）を提案する．さらに，マイグレーションファイル選択方式をシステムに適用する方法を考察し，実際のファイル参照記録を使用したシミュレーションの結果を示す．

2. 階層化補助記憶システムの効率評価

実ディスク装置のみからなる補助記憶システムを階層化することによって，効率評価上追加される尺度は，リコール，マイグレーションに関連する要素，および階層化することによる見掛上の実ディスク記憶容量の増加である．シーク時間，回転待ちデータ転送速度など，磁気ディスク装置固有の他の評価要素の階層化による差は無視でき，以下の効率の議論では省略する．

2.1 リコール，マイグレーション

仮想ディスクと実ディスク間のファイル移動が少ないほどシステムに対する負荷は少ない．しかし，リコールによる転送と，マイグレーションによる転送は，明確に区別しなければならない．後者は，緊急を要する処理ではないため，利用者に対する影響はないものと見なすことができるが，前者については，ファイル利用時に行われるため，優先度の高い処理をする必要があり，これに要する時間は，システムの負荷あるいは利用者への影響を考慮すると，最も重要な評価測度であることがわかる．

2.2 記憶容量

階層化補助記憶システムの記憶容量は，実ディスクと仮想ディスクの記憶容量の和と考えてよい．階層化の目的は，システム全体の単位記憶容量当たりのコストを，低くすることにある．したがって，システムに要求される記憶容量を固定して考えれば，より高コストの実ディスク用の記憶媒体の容量が少ないシステムほど，階層化の効率が高いと見なすことができるが，実ディスクの記憶容量の減少は，階層間のデータ転送量の増加を伴い，システム負荷の増大あるいは操作性の悪化を招く．すなわち，実ディスクの記憶容量も，もう一つの重要な評価測度であることがわかる．

3. マイグレーションすべきファイルの最適選択方式

3.1 ファイル参照モデル

システム内の時刻 t におけるファイルに，番号を付け，$D(t)=\{1,2,\cdots,n\}$ を時刻 t におけるファイルの集合とし，ファイルの参照系列を $r_1, r_2, \cdots, r_i, \cdots$ とする．r_i はそれぞれ時刻 t_i に参照されたファイル番号とする．時刻 t と時刻 $t+\Delta t$ の間に参照されたファイルの集合を参照集合 $R(t, t+\Delta t)$ とする．

$$R(t, t+\Delta t)=\{r_i \mid t<t_i \leqq t+\Delta t\}$$

ファイル i が $R(t, t+\Delta t)$ の要素となる確率が，既知であるものとし，これをファイル i の区間 $t, t+\Delta t$ 間における参照確率 $P_i(t, t+\Delta t)$ と定義しよう．参照確率は，$\lim_{\Delta t\to\infty} P_i(t, t+\Delta t)=1$ を満たし，区間長 Δt に関する増加関数となる．時刻 t において実ディスクおよび仮想ディスク上に存在するファイルの集合を，それぞれ $D_r(t), D_v(t)$ とし，区間 $(t, t+\Delta t)$ において，$i\in R(t, t+\Delta t)$ かつ $i\in D_v(t)$ なるファイルのリコールに要するコスト（以後リコール・コスト）を $U_i(t, t+\Delta t)$ とする．$i\in R(t, t+\Delta t)$ かつ $i\in D_r(t)$ なるファイルはリコールする必要はないので，参照時のリコール・コストは 0 と考える．したがって，区間 $t, t+\Delta t$ におけるリコール・コストの和の期待値は，

$$C(t, t+\Delta t)=\sum_{i\in D(t)} U_i(t, t+\Delta t)\cdot P_i(t, t+\Delta t)$$
$$=\sum_{i\in D_v(t)} U_i(t, t+\Delta t)\cdot P_i(t, t+\Delta t)$$

となり，ファイル i の時刻 t に占める記憶領域の大きさを $S_i(t)$ とすると，時刻 t に実ディスクを占める記憶領域の総和は，

$$TS(t)=\sum_{i\in D_r(t)} S_i(t)$$

となる．さらに，TS_{max} をシステムで使用可能な実ディスクの記憶容量の上限とすると，Δt 間隔でマイグレーションを行うシステムの，時刻 t における最適な選択方式は，ファイルの集合 $D(t)$ を $D_r(t)$, $D_v(t)$

へ分割する方式の中で，$TS(t) \leqq TS_{max}$ の条件下で，$C(t, t+\Delta t)$ を最小にするものと考えることができる．

ここで，次の2種類の分割を考える．

・選択方式 OPT

閾値 $L>0$ を定め，ファイルの集合を次のように分割する．

$$H_i(t, t+\Delta t)<L \rightarrow i \in D_r(t)$$
$$H_i(t, t+\Delta t) \geqq L \rightarrow i \in D_s(t)$$

ここで，ファイル i の選択パラメータ $H_i(t, t+\Delta t)$ は，次式で定義されるものとする．

$$H_i(t, t+\Delta t)=P_i(t, t+\Delta t) \cdot U_i(t, t+\Delta t)/S_i(t) \quad (3.1)$$

・選択方式 OPT_1

閾値 $L>0$ を定め，ファイルの集合を，次のように分割する．

$$H_i(t, t+\Delta t)<L \rightarrow i \in D_r(t)$$
$$H_i(t, t+\Delta t)>L \rightarrow i \in D_s(t)$$
$$H_i(t, t+\Delta t)=L \rightarrow i \in D_r(t) \vee i \in D_v(t)$$

次の最適選択方式定理を導くことができる．

最適選択方式定理

選択方式 OPT の，閾域 L による，時点 t におけるファイルの分割によって，実ディスクを占める記憶容量 $TS_{OPT(L)}(t)$，および参照によるコストの期待値 $C_{OPT(L)}(t, t+\Delta t)$ が定まる．他の選択方式Aによって定まる，実ディスクを占める記憶容量，参照によるコストの期待値を，それぞれ $TS_A(t), C_A(t, t+\Delta t)$ とすると，$TS_A(t)<TS_{OPT(L)}(t)$ ならば，$C_A(t, t+\Delta t)>C_{OPT(L)}(t, t+\Delta t)$ となる．

証明

付録に与えた補題における (x, y) の集合を，それぞれ，ファイル i に対する二つ組 $(S_i(t), P_i(t, t+\Delta t) \cdot U_i(t, t+\Delta t))$ の集合に対応させることにより，補題を適用できる．

証明終了

最適選択方式定理は，いかなる選択方式による分割であっても，実ディスクに占める記憶容量が，選択方式 OPT の，閾域 L によって定まった，実ディスクを占めるファイル記憶容量より少ないならば，コストの改善は不可能であることを示している．閾値 L と等しい選択パラメータ値を持つファイル群を実ディスク，仮想ディスクのいずれかに含めることを許すように条件を緩めた OPT_1 についても成立することは容易にわかる．

ここで，ファイル集合 D が与えられ選択方式 OPT と閾値 L が与えられると，時点 t における分割により，$TS_{OPT(L)}(t)$ が決定される．しかし，逆は必ずしも定義されるとは限らない．すなわち，ある TS_{max} を決めても，一般的には，閾値 L を決定することは，できないことに注意しなければならない．

L_{max} を，$TS_{OPT(L)}(t) \leqq TS_{max}$ を満たす最大の L の値とする．ここで，$TS_{OPT(L_{max})}(t)<TS_{max}$ とすると，$TS_{OPT(L_{max})}(t)<TS_A(t) \leqq TS_{max}$ を満たし，$C_A(t, t+\Delta t)<C_{OPT(L_{max})}(t, t+\Delta t)$ とする選択方式Aが，存在する可能性がある．したがって，方式 OPT(OPT_1) は，$TS \leqq TS_{max}$ の制約条件の下で，コストの和の期待値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_{max} はファイルの個々の大きさ S_i に比較し大きく，実ディスク，仮想ディスクのいずれにも，極めて大量のファイルが格納されているので，$TS_{max}-TS_{OPT(L_{max})}(t) \ll TS_{max}$ が成立し，$C_{OPT(L_{max})}(t, t+\Delta t)$, $C_A(t, t+\Delta t)$ の差は無視できる．したがって，大量のファイルを対象とするこのシステムでは，方式 OPT(OPT_1) は $TS \leqq TS_{max}$ の制約条件の下で，C を最小にするファイルの最適選択方式と見なすことができる．

3.2 ファイル参照確率

参照確率とリコール・コストを既知とすれば，マイグレーションすべきファイルを選択する最適な方式が存在することを前節で示した．スケジュールされたファイル利用を行っているシステムでは，参照確率を確度をもって予測できるが，通常は，個々のファイルの参照確率を知ることができない．したがって，それまでファイルが参照されてきた状況（参照履歴という），ファイル占有領域（サイズという），ファイルの用途，編成（以後参照履歴およびサイズを除くすべてのファイルの性格をファイルのクラスという）によって，システム内のファイルを分類し，参照確率を推定する．すなわち，時点 t で，階層化記憶システムに存在するファイルの集合 $D(t)$ の部分集合 A に含まれ，時間 Δt を経過するまでに，少なくとも1度は参照されるファイルの数を $RF_A(t, t+\Delta t)$，部分集合 A に含まれるファイルの総数を $NF_A(t)$ と定義することにより，

$$P_A(t, t+\Delta t)=RF_A(t, t+\Delta t)/NF_A(t) \quad (3.2)$$

を，その部分集合 A に含まれるファイルの参照確率の推定値とする．

参照履歴を表現する方法は各種考えられるが，ここで参照履歴として，CLU (Count Since Last Recently Used) を導入し，参照確率を容易に推定できるこ

とを示そう．

CLU はファイルが参照された時に0にリセットされ，一日の定められた時点で，1が加算されるものと定義する（この処理を以後加算処理という）．この場合，加算処理が時刻 t に行われると，次回の加算処理は，時刻 $t+\Delta t$ に行われることとしよう．前回の加算処理後に，参照されたファイルの CLU は，次の加算処理時には，0となっている．毎日加算処理を行う場合（$\Delta t=1$）は，最近参照日以後の経過日数（Day Since Last Recently Used）となる．参照履歴を CLU で代表させ，

$NF_{i,s,k}(t)$，$RF_{i,s,k}(t, t+\Delta t)$

を，それぞれ時点 t で CLU$=i$，サイズ s，クラス k のファイルの数および区間 $(t, t+\Delta t)$ で参照されたファイルの数と定義する．経過時間の間に，参照されたファイルのサイズ，クラスが変化しないものとすると，

$$RF_{i,s,k}(t, t+\Delta t)=NF_{i,s,k}(t)-NF_{i+1,s,k}(t+\Delta t)$$

すなわち，

$$P_{i,s,k}(t, t+\Delta t)=1-NF_{i+1,s,k}(t+\Delta t)/NF_{i,s,k}(t) \tag{3.3}$$

が成り立つ．したがって，参照履歴を CLU で代表させると，参照確率は，ある時点におけるファイルの存在数のみによって，推定できる．

参照集合が定常状態にあることを仮定すれば，参照確率は，t には依存しない．時間帯によって，ファイル参照の頻度が著しく異なっていたとしても，毎日同じ程度の利用者がファイルを参照しているという仮定の下では，t および Δt の単位として，日（day）を採れば，参照確率は t には依存しないことがわかる．さらに，

$\Delta t=1$ 日

とすると，t および Δt を省略でき，参照確率を参照履歴 h，サイズ s，クラス k により，$P_{h,s,k}$ と表すことができる．

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（$\Delta t=1$ 日）を図2に示す．これによると，ほぼ次の関係が成立していると考えてよいだろう．

$$s_1>s_2 \quad \text{ならば} \quad P_{i,s_1,k} \geqq P_{i,s_2,k} \tag{3.4}$$

$$i_1>i_2 \quad \text{ならば} \quad P_{i_1,s,k} \leqq P_{i_2,s,k} \tag{3.5}$$

ここでは，参照履歴 h を CLU 値 i で代表させている．

また，ここでは示していないが，

po＝分割型順編成のクラス，

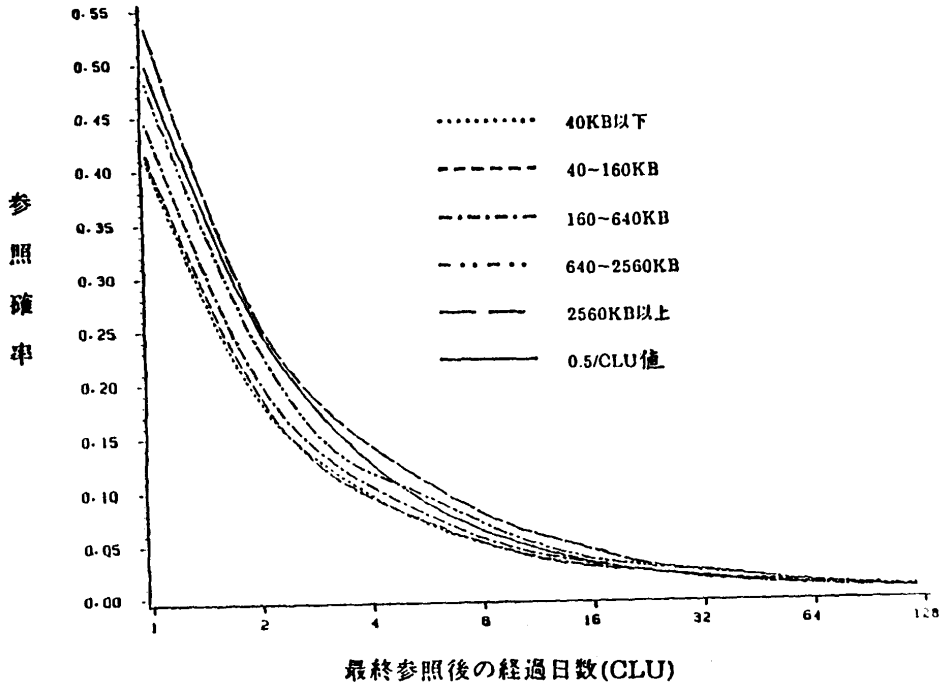

図2　ファイルサイズ別の参照確率
Fig. 2　File size dependent reference probability.

ps＝順編成のクラス

とすると，

$$P_{i,s,po} \geqq P_{i,s,ps} \tag{3.6}$$

であった．

(3.4)はファイルのサイズが，大なるものの方が，小なるものより，参照されやすいことを示している．ファイルの，サイズに応じたファイル利用料金を課している場合は，このような傾向があるのは当然であろう．(3.5)は前回参照されてからの経過時間の長いファイルほど，参照されにくいことを示し，CLU 値以外のファイルサイズ，クラスなどの属性が同一のファイル集合に対して，LRU 法を採用する根拠ともなっている[5]．(3.6)は多くのプログラムあるいはデータ単位を含む PO ファイルのほうが，通常，単独のプログラムあるいはデータファイルを含む PS ファイルより，参照されやすいことを示している．

図2によると，CLU 値を i とし，$0.5/i$ は参照確率を比較的よく近似していることがわかる．

3.3 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システムへの負荷，レスポンスの遅れによる利用者の抱く不満足感を総合し，リコール・コストと定義する．リコール・コストは，該当するファイルの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とによって定まると考えることができる．前者については，ファイルのサイズ，データ転送速度に依存し，後者については，参照履歴とも強い相関がある．したがって，リコール・コストは，転送されるファイルのサイズと，参照履歴に依存すると考えることができる．参照履歴，サイズ以外のファイルの特性については，以下の議論を複雑にすることを避けるため，無視することにしよう．

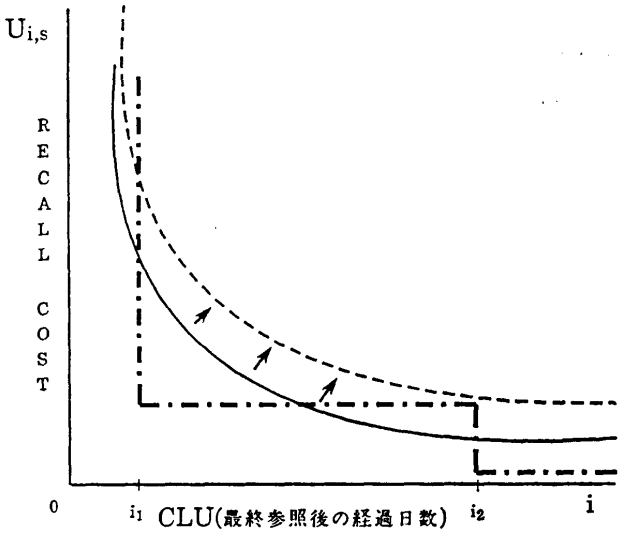

図 3 リコール・コスト
Fig. 3 Recall cost.

参照履歴として CLU を採用し，リコール・コストの性質について考察する．CLU＝i，サイズ s のファイルのリコール・コストを $U_{i,s}$ とすると，

$$s_1 < s_2 \rightarrow U_{i,s_1} \leqq U_{i,s_2} \tag{3.7}$$

と考えるのが自然である．したがって，$U_{i,s}$ は s に関し，非減少な s の関数 $G(s)$ に比例するものとしよう．データ転送路のビジー率が低い場合は，$G(s)$ は s に関する1次式 $G(s) \fallingdotseq a+b \cdot s$ で近似できよう．

また，参照後の経過時間の短いファイルがマイグレーションされ，参照時にリコールされることになると，ファイルのサイズには関係なく，利用者の不満足感は非常に大きい．参照後の経過時間の長いファイルについては，逆に，参照時にリコールを伴うとしても，利用者は，やむをえないと考えるであろう．すなわち，$U_{i,s}$ は i に関する非増加関数となり

$$i_1 < i_2 \rightarrow U_{i_1,s} \geqq U_{i_2,s} \tag{3.8}$$

が成立する．さらに，$U_{i,s}$ は i が小なる場合，極めて大なる値を持ち，i が大なる場合は，ほとんど変化がなく，s には依存しないような i の関数 $F(i)$ に比例するとするのが自然である．すなわち，リコール・コストはサイズ項と CLU 項の積の形に書け，

$$U_{i,s} = G(s) \cdot F(i) \tag{3.9}$$

と表現できる．以上より，$U_{i,s}$ は**図3**における実線のような形をとり，s が大になると→の向きに動き，点線のような形をとる．実際のリコール・コストは，システム運用環境によって，強く影響を受ける性格のものである．簡単な例を磁気ディスク装置と MSS を使用するシステムで考えよう．リコールに要する時間が 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すことができれば，ファイルのサイズに関する項を無視し，$G(s)=1$ とすることが可能である．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が，参照時にリコールされると，利用者の抱く不満足感は，極めて大きいはずである．すなわち，

$$i < i_1 \rightarrow F(i) \gg 1 \tag{3.10}$$

とすることにより，リコール・コストを大にし，マイグレーションを抑止することができ，逆に，

$$i>i_2 \rightarrow F(i)\ll 1 \qquad (3.11)$$

とし，参照後ある程度以上時間が経過したファイルのリコール・コストを極めて小にし，そのようなファイルを強制的にマイグレーションすることも可能となる．このような場合は $U_{i,s}=G(s)\cdot F(i)$ は図3鎖線で示すような形をとる．

3.4 最適選択方式の適用方法について

マイグレーションの可否を判定するために使用するファイルパラメータを選択パラメータと呼ぶこととする．最適選択方式では，ファイルのマイグレーションの可否を判定するために，(3.1)式で定義される $H_i(t,t+\Delta t)$ を使用する（以後 H_i あるいは H と略記）．これを最適選択パラメータと呼ぶことにしよう．

実際にマイグレーションすべきファイルの選択方法として，次の2方法が考えられる．

① 閾値 L を決め，$H_i\leqq L$ を満足するファイル i のマイグレーションを行う．

この適用方法では，常時他の処理と共存させ，個々のファイルのマイグレーション処理が可能であるが，マイグレーション処理後の実ディスク使用領域を，ある範囲内に納めるためには，閾値 L をマイグレーション時に，変化させることが必要である．

② 実ディスク使用領域が，予定した範囲に納まるように，選択パラメータ値 H_i の小なる順にマイグレーションを行う．

この方法では選択パラメータにより，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．

いずれの適用方法も，実ディスクに存在するすべてのファイルの参照履歴ならびにサイズに関する情報を，調査する必要性がある．マイグレーション処理は，計算機システムの負荷の比較的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，大きな差異はないように見える．しかし，②では①と異なり，ファイルを順序付ける必要があり，マイグレーションすべきファイルの選択処理を，一括して行う必要がある．したがって，その間他の処理と共存させ難いため，運用をかなり制限するという欠点を持つことに注意しなければならない．この欠点のため，②以外の適用が不可能なファイル選択方式は，非常に利用し難く，むしろ実現が困難な方式といえよう．

最適選択方式は，ファイル集合を，実ディスクと仮想ディスクに分割する方式の一方式である．このことは，実ディスクから仮想ディスクへの転送ばかりでなく，仮想ディスクから実ディスクへの転送を必要とする場合もありうることを示している．仮想ディスクに存在しているファイルは，最後のマイグレーションの後，参照されていないので，(3.5)式により，選択パラメータ値 H_i が，閾値 L を越えることはない．したがって，適用方法①を採用する場合，仮想ディスクに存在するファイルの情報を調べる必要がない．

方式②では，マイグレーション処理を開始する時に，すでに実ディスク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，大量のファイルが一時的に消去された場合が，これに当たる．この場合には，新たなマイグレーションの必要がない．このようなことが発生することは，通常，まれではあるが，この場合に，仮想ディスク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．

3.5 実際のファイル参照記録を使用した選択方式のシミュレーション

九州大学大型計算機センターにおけるファイル参照記録を使用し，階層化補助記憶システムについて，次のようなシミュレーションを行った．CLU 値を i，次に参照されるまでの期間（日）を j とすると，各シミュレーションにおける選択パラメータ H は，

シミュレーション 1

$H=\infty$　　$i=1$

$H=1/i$　　$i\geqq 1$

シミュレーション 2

$H=\infty$　　$i=1$

$H=1/(i\cdot s)$　　$i\geqq 1$

シミュレーション 3

$H=\infty$　　$j=1$

$H=1/j$　　$j\geqq 1$

である．シミュレーション3では，将来の参照が完全にわかっていることを仮定しているので，現実には，有りえないモデルではあるが，マイグレーションファイル選択方式の効率を比較するモデルとして採用した．

なお，このシミュレーションではリコール・コスト U として，

$U_{i,s}=\infty$　　$i\leqq 1$

$U_{i,s}=1$　　$i>1$

を使用したことになる．このリコール・コストの設定は，リコール・コストがファイルのサイズにより変化

しないことと，リコール・コストを無限大にすることにより，少なくとも参照後一定期間（ここでは1日），マイグレーションを行わないこととを意味する．なぜならば，ファイルが一度参照されると，その後一定期間内に再度参照された場合，そのファイルがマイグレーションされていると，非常に不満が大きいからである．すなわち，図3で $i_1=1$, $i_2=\infty$ としたことと等しい．

シミュレーションの結果を図4に示す．

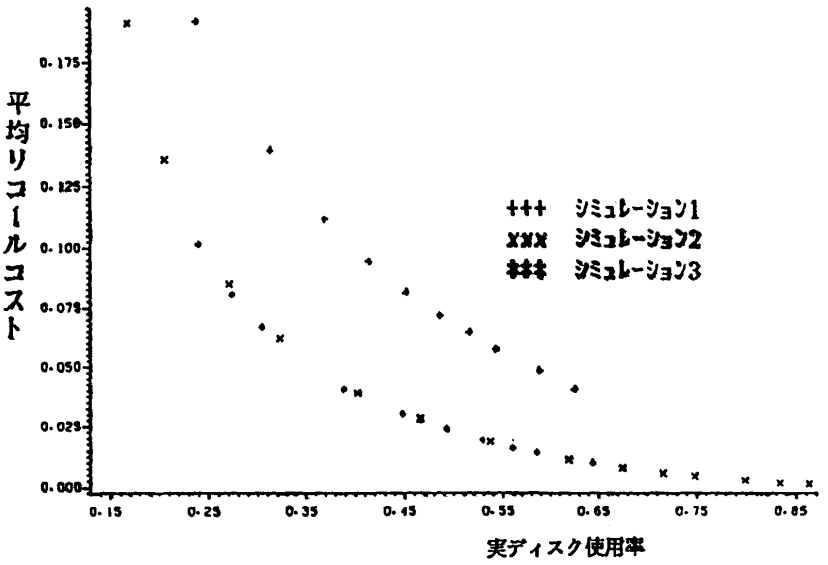

図4 実際のファイル参照記録を使用したシミュレーション結果
Fig. 4 Result of simulation by the real file reference data.

図4の横軸は，実ディスクを占めるファイルの記憶容量と，システムに存在するすべてのファイルの記憶容量の比をとり，縦軸には，参照されたファイル1個当たりのリコール・コストの平均をとった．

リコール・コストを上のように設定すると，縦軸は，リコールされたファイルと参照されたファイルの比(リコール率)とも等しくなる．これによると，シミュレーション1（図4では＋＋＋）は，以下に示すように，LRU 法と等価であるが，非常に効率の悪い方式であること，シミュレーション2（図4では×××）は，参照確率を k/i（k: 比例定数，i: LRU 値）で近似し，最適選択方式を適用したことと等価になり，シミュレーション3（図4では＊＊＊）と同等の有効性を示していることがわかる．

ここで，シミュレーション1，2の選択パラメータを最適選択方式の立場から考察する．

① シミュレーション1（LRU 法）

リコール・コストがファイルのサイズに比例するかあるいは，すべてのファイルのサイズ，リコール・コストが同一であるという条件の下では，最適選択パラメータは参照確率に比例することになる．図1にも示したように，参照確率は CLU 値 i に関して，単調に減少する．したがって，最適選択パラメータとして，i に関する単調減少関数，例えば，$H=1/i$ を使用する方式は LRU 法と等価な方式となる．しかし，図5に示すように，ファイルのサイズは極めて広い範囲に分布し，また，仮想ディスクとして MSS を使用する場合は，リコール・コストのサイズに比例する項は必ずしも支配的でない．したがって，上の条件は成立しないので，図4（＋＋＋）で示したように，LRU 法は効率の良い方式とはならない．

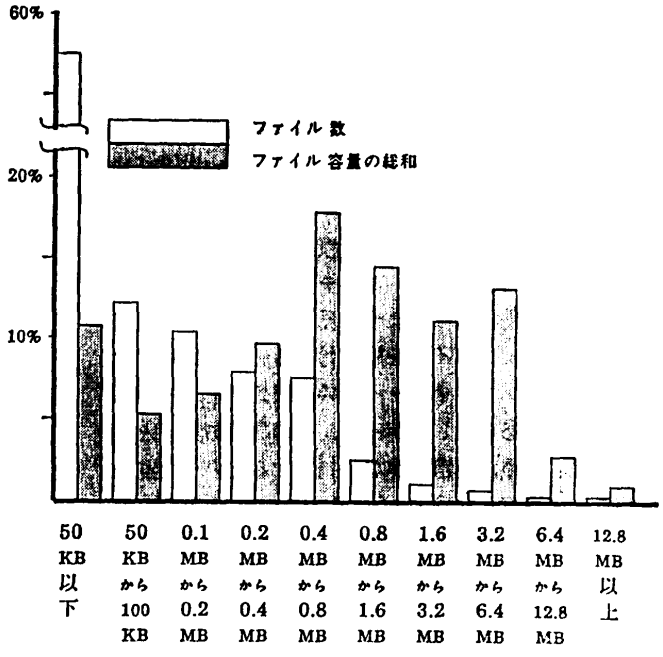

図5 ファイルサイズ分布
Fig. 5 Distribution of file size.

② シミュレーション2 (LRU 法にファイルサイズを考慮する方式)

$H=k/(i^{\alpha}\cdot s)$ (k は比例定数, i は CLU 値) の形式の選択パラメータを使用したシミュレーションで, よい結果が得られると報告されている[2),3)]. ファイルのサイズによる差異を無視し, α に適当な値を設定すれば, 参照確率を k/i^{α} の形式で近似できるような利用環境では, 最適選択パラメータを $H=k/(i^{\alpha}\cdot s)$ とすることができる. 各々[2),3)]の利用環境でも, k/i^{α} が参照確率を比較的良く近似できるため, この方式が, 最適選択方式にたまたま近く, 効率の良い方式になっているのであろう.

図2に示したように, 九州大学大型計算機センターでも, 調査期間中の参照確率の近似式として $0.5/i$ を利用できる. したがって, これに利用環境によるリコール・コストを設定し, 最適選択パラメータを構成すれば, 容易に最適選択方式を実現できる.

4. むすび

現実の階層化記憶システムでは, 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. 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, 優先的にマイグレーションすることが, リコール率を低くするための有効な方法であるが, 現実のシステムでは, これが採用されていないのは, 大きなファイルは, 極端に短期間にマイグレーションされ, マイグレーションとリコールが, 繰り返し行われることを, 恐れているからにほかならない. 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.

現実のシステムでは, 従来のように, 単にリコール率のみを考慮するだけではなく, 利用者の不満足度などを含め, 総合的にシステムの効率を考えねばならない. 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, そのシステムの負荷, 利用者の不満足度を総合的に配慮して, リコール・コストを決定すれば, 最適選択方式により, 最適なシステムを実現できるという明確な指針を示した. しかし, ここでは, マイグレーション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. また, 最適選択方式の証明では, 参照確率の推定方法を特別に定めていない. このため, ファイルのクラスに応じて, 異なる参照履歴などを使用した参照確率の推定を行い, 参照確率の推定精度を向上させることができれば, システムの効率が, さらに向上する可能性がある. これらは今後の問題として残されている.

なお, 九州大学大型計算機センターでは, ここで述べた方式を適用しているが, 適用結果については, ある程度の期間運用した後に報告したい.

階層化の試みとして, 文献6) で提案された各階層へ利用者が直接参照を可能とすることにより, 参照時のリコールを回避する方式も, 非常に興味のある方式である. しかし, この方式では, すべての階層の記憶媒体の参照方式を同一にしなければ, 操作性の面で問題があるため, 各レベルの記憶媒体を選択するときの制約, 記憶媒体の利用方式の制約が避けられないと思われる. ここで述べた方式は, ファイルを参照時にリコールする場合があるとしても, システムの参照方式の統一化を容易にし, 実現されやすいと考えられる.

記憶媒体の技術の進歩により, それらの記憶容量が急速に増加しているにもかかわらず, その増加も利用者の記憶容量の需要増には追い付けない. したがって, 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などによって, 積極的に参照の局所性を上げ, システム効率を向上させる方策を検討しなければならないであろう.

謝辞 本研究に, 種々有益なご助言をいただいた九州大学工学部情報工学科牛島和夫教授, および九州工業大学情報工学部大槻説乎教授に感謝します.

参考文献

1) Smith, A. J.: Long Term File Reference Patterns and Their Application to File Migration Algorithms, *IEEE Trans. Softw. Eng.*, Vol. SE-7, No. 4, pp. 403-416 (1981).
2) Smith, A. J.: Long Term File Migration: Development and Evaluation of Algorithms, *Comm. ACM*, Vol. 24, No. 8, pp. 521-532 (1981).
3) Lawrie, D. H., Randal, J. M. and Barton, R. R.: Experiments with Automatic File Migration, *Computer*, Vol. 15, No. 7, pp. 45-55 (July 1982).
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開発論文集, No. 3, pp. 93-97 (1981).

5) Coffman, Jr., E. G. and Denning, P. J.: *Operating Systems Theory*, p. 331, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1973).
6) 藤井 実, 浅井 清: 階層的ファイル自動管理システムの設計, 情報処理学会論文誌, Vol. 21, No. 6, pp. 442-453 (1980).

付 録

補題

$x>0$, $y>0$ である2つ組 (x, y) の有限集合 D を2個の部分集合 A, B に分割する. D の要素は A, B いずれかの部分集合に含まれるものとする.

ここで, $S_{A,B}$ および $C_{A,B}$ を

$$S_{A,B}=\sum_{(x,y)\in A} x \qquad (A.1)$$

$$C_{A,B}=\sum_{(x,y)\in B} y \qquad (A.2)$$

と定義する.

上で定義した分割の中で, 関係(A.3)

$$\min_{(x,y)\in A} y/x > \max_{(x,y)\in B} y/x \qquad (A.3)$$

を満足する分割を $A=D_1$, $B=D_2$ とすると, 他のいかなる排他的かつ網羅的な D の分割 F_1, F_2 であっても, $S_{D_1,D_2}>S_{F_1,F_2}$ ならば $C_{D_1,D_2}<C_{F_1,F_2}$ である.

証明

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 とすると, 排他的で, 網羅的なすべての分割は, 分割 $F_1=D_1-E_1+E_2$, $F_2=D_2-E_2+E_1$ で表現できる. ここで $\min_1=\min_{(x,y)\in E_1} y/x$, $\max_2=\max_{(x,y)\in E_2} y/x$ とすると, 前提より $\min_1>\max_2$

$$S_{D_1,D_2}-S_{F_1,F_2}=\sum_{(x,y)\in E_1} x-\sum_{(x,y)\in E_2} x>0$$

であるから

$$\sum_{(x,y)\in E_1} y\geqq \min_1 \sum_{(x,y)\in E_2} x,$$

$$\sum_{(x,y)\in E_2} y\leqq \max_2 \cdot \sum_{(x,y)\in E_1} x$$

$$C_{F_1,F_2}-C_{D_1,D_2}=\sum_{(x,y)\in E_1} y-\sum_{(x,y)\in E_2} y$$

$$>\min_1 \cdot \sum_{(x,y)\in E_2} x-\max_2 \cdot \sum_{(x,y)\in E_2} x$$

$$\geqq \min_1 \cdot \sum_{(x,y)\in E_1} x-\max_2 \cdot \sum_{(x,y)\in E_2} x$$

$$=(\min_1-\max_2)\cdot \sum_{(x,y)\in E_2} x\geqq 0$$

したが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, $S_{D_1,D_2}>S_{F_1,F_2}$ ならば $C_{D_1,D_2}<C_{F_1,F_2}$ である.

証明終了

(昭和61年5月22日受付)
(昭和62年3月25日採録)

景川 耕宇 (正会員)

昭和15年生. 昭和37年東京大学工学部応用物理学科(数理工学専攻)卒業. 昭和39年同大学院修士課程修了. (株)三永通信, (株)新日本パイプを経て昭和42年より九州大学勤務. 現在九州大学大型計算機センター講師. 主な研究分野: 計算機システムの性能評価.

武富 敬 (正会員)

1947年生. 1970年九州大学理学部物理学科卒業. 1975年同大学院理学部研究科博士課程単位修得退学. 同年九州大学大型計算機センター助手, 現在に至る. センター運用・管理のための情報システムの構築, プログラムライブラリ管理システム, 計算機システムの評価等に興味を持つ. 生物物理学会会員.

末永 正 (正会員)

昭和24年生. 昭和47年九州大学工学部電子工学科卒業. 九州大学大型計算機センターを経て, 現在, 九州大学情報処理教育センター助手. オペレーティング・システムの効率, およびマンマシン・インタフェースに関心がある. 人工知能学会会員.