三階層記憶方式の仮想記憶制御への適用と分析・評価

新井利明† 吉澤康文†
細内昌明† 大房義隆††

属性の異なる複数の記憶媒体を階層的に配置して仮想記憶を構築することにより，性能対価格比の向上および負荷への柔軟な追従が可能となる．本研究では，仮想記憶の構成要素に拡張記憶 (ES) を導入した三階層記憶制御方式を提案し，その効果を解析モデルにより評価し，実測結果と比較した．まず，中間階層に配置した記憶媒体の効果を評価するため，スループットに着目した解析モデルを構築した．三階層記憶制御方式の効果はプログラムの参照特性に依存するため，ローカリティの大きいプログラムと小さいプログラムの2種類を想定して，主記憶と2次記憶で構成される従来システム（二階層システム）と性能を比較した．また，コスト一定の条件の下で，各階層の最適容量比を求めた．三階層記憶制御の開発後，拡張記憶を利用してプログラムの参照特性を求め，前記の解析モデルを適用して効果を予測し，実測結果と比較した．その結果，トータル CPU 利用率が 100% に近い場合には予測との誤差は 10% 程度であった．しかし，ディスク等のネックが発生し，CPU 利用率が低い場合には予測の誤差が大きいことが判明した．また，TSS 環境において，プログラム開発環境を想定した一連のトランザクションを用いた実測結果では，主記憶量の3倍の拡張記憶を用意しておくことで負荷の変動に対しても安定した性能を確保できた．

Three Level Hierarchal Storage Management for Virtual Storage and Its Evaluation

Toshiaki Arai,† Yasufumi Yoshizawa,† Masaaki Hosouchi† and Yoshitaka Ohfusa††

Virtual-storage system consisting of several different storage devices outperforms simple virtual-storage system which is constructed from main storage and disk device. We introduced Three Level Storage Hierarchy System for virtual storage management which has expand storage (ES) between main storage (MS) and external storage (Disk), and predicted the performance. We constructed an analytic performance model and showed that Three Level Storage Hierarchy System establishes high-performance for both programs with low reference locality and with high reference locality. Then we showed the optimum proportion of MS size and ES size under the given cost. Developing Three Level Storage Hierarchy Management, we evaluated the performance and compared the measured performance to the forecasted performance. The model estimates CPU utilization 10% lower than that of real environment. Under TSS environment, equipping ES three times as much as MS guaranties stable performance.

1. はじめに

記憶階層を利用した記憶管理方式は，データアクセス時間の削減，メモリコストの低減，データの保全性の確保，の点で有効である[1)~3)]．計算機システムでは属性の異なる多種の記憶媒体が利用可能であり，これらを階層的に配置して論理的な記憶装置をユーザに提供することにより，個々の記憶媒体の特徴を発揮できる．

仮想記憶方式では，従来より主記憶と2次記憶（ディスク）の二階層で論理的な記憶装置を構築していた．しかし，電子的な記憶媒体である主記憶と機械的動作を伴うディスクのアクセス速度の差，いわゆるアクセスギャップは，半導体技術の進歩に伴い広がる一方である．そのため，データアクセス時間がデータの配置媒体に大きく依存し，負荷の変動に対する性能の保証が困難となりつつあった．そこで，拡張記憶 (ES) を用いてギャップを埋め，三階層からなる記憶階層を構築することでデータアクセス時間を保証する方式が考えられる．これにより，負荷の変動に柔軟に対応で

† (株)日立製作所システム開発研究所
Systems Development Laboratory, Hitachi, Ltd.
†† (株)日立製作所ソフトウェア開発本部
Software Development Center, Hitachi, Ltd.

きる仮想記憶システムが構築可能となる．

階層記憶の管理方法は従来より研究されてきたが，対象はディスクキャッシュ等の外部記憶の管理方式に関するものであった[4]．階層記憶管理を仮想記憶管理に適用する場合には以下を考慮する必要がある．

（1）論理的に関連のないページ単位でリプレースメント方式を決定する．一方，ディスクキャッシュではファイル単位およびアクセス法という情報を使用できる．

（2）CPU 処理と同期して実施しなければならない．ディスクキャッシュの場合には，CPU 速度に比べてアクセス速度が遅いため，種々の情報が収集可能である．

（3）中間階層の記憶容量が上位の容量に比べて大容量であるとは限らない．

本研究では，上記の条件の下に仮想記憶制御における三階層制御の方式の効果を以下の手順で定式化して予測した．

（1）各階層の記憶容量とスループットの関係を定式化

（2）各記憶媒体の単価と性能およびプログラムの参照特性を入力として，与えられたコストに対して媒体の最適容量の決定方式を提示

（3）解析モデルによる効果の検討

（4）負荷変動に対する柔軟性の検討

本研究では，まず，LRU によるリプレースメントと記憶階層間におけるデータのマイグレーションをモデル化して三階層記憶制御の性能を評価する．開発した三階層記憶制御方式の実測結果をモデルによる予想結果と比較し，モデルを検証した後，モデルを用いた各種の検討を実施する．

2. 三階層記憶制御方式

階層記憶制御においては，階層を構成する記憶媒体の種類が多くなればなるほど，多様な負荷への対応が可能となるが，その反面，制御が複雑となりスラッシングの危険性も高くなる．本研究では，複数媒体からなる階層記憶の基本特性を明らかにするため，下記の特性を前提として制御方式を定式化する．

（1）最上位に位置する記憶媒体のみが処理装置からアクセス可能である（アクセス要求があったデータは，必ず一旦最上位の記憶媒体にロードされる）．

（2）任意の階層に配置されたデータは，最上位の記憶媒体との間で直接転送が可能である．

（3）アクセス要求があったときにのみ上位方向へのデータ転送が行われる（プリローディングは行わない）．

（4）下位方向へのデータ転送は任意の時点に行う．

上記の前提は，仮想記憶方式だけではなく，他の階層記憶にも該当する性質である[3]．従って，検討結果も一般的な記憶階層制御に応用可能である．

3. モデリングによる性能予測

3.1 解析モデル

本節では，解析モデルを構築して三階層記憶制御の効果を予測する[5]．モデル化の目的は，従来の三階層による仮想記憶方式と ES を用いた三階層記憶制御方式のスループットを比較すること，および，中間階層である ES の容量の最適値を決定することである．

定式化に際しての前提条件は以下のとおりである．

（1）CPU 利用率は常に 100% である．（ジョブの多重度が充分大きい）

（2）記憶制御（ページング，マイグレーション）処理に必要となる CPU オーバヘッド以外の CPU 時間はユーザが使用でき，それはスループットに比例する．

（3）主記憶（MS）から追い出されたページはまず拡張記憶（ES）に書き出される．さらに，必要に応じて二次記憶装置に書き出される．ES から二次記憶装置への書き出しは LRU アルゴリズムに従う．

（4）ES に空きが生じても二次記憶のデータを ES に取り込むことはしない．

上記の前提を基に，スループットに比例する値であるユーザの CPU 利用率 ρ_u を求める解析モデルを作成する．

まず，単位時間に発生するページフォールト率を f，ページフォールト処理に必要な CPU 利用率を ρ_p とすると，（1）式が成り立つ．

$$\rho_u + f \cdot \rho_p = 1 \qquad (1)$$

ページフォールト率 f は，プログラムの動作，メモリサイズ，および CPU 能力に依存する．f を定式化するため，まず，プログラムの動作を定義する．一般に，プログラムの動作は LRU スタックモデルを用いて記述できる．すなわち，スタック長 d のスタックと，一命令実行したときにスタック内に存在する確率 $p(d)$ とを用いてプログラム動作を定義できる．1−

$p(d)$は，データがスタック内に存在しなかった確率であり，この値と単位時間に実行可能な命令数 μ (CPU能力) の積が，ユーザ実行時のページフォールト率である．この値にプログラムの実行比率である ρ_u をかけた値が，システムのページフォールト率となる．MSサイズを m とすれば，ページフォールト率 f は以下の式で表される．

$$f=(1-p(m))\mu\rho_u \quad (2)$$

(1)と(2)から，(3)式が得られる．

$$\rho_u=1/(1+\rho_p\mu(1-p(m))) \quad (3)$$

次に，ページフォールト当りのOSのCPU利用率 ρ_p を定式化する．前提条件として，OSはページイン処理を行うと共に，次回のページフォールトに備えてMSから1ページ追い出して空きを用意するものとする．

従来の二階層の仮想記憶方式では(ESがない場合)，ρ_p は(4)式となる．

$$\rho_p=(C_i+C_o)/\mu \quad (4)$$

C_i と C_o はそれぞれ外部記憶からのページイン，ページアウト処理に必要な処理ステップ数である．

次に，三階層記憶制御の場合 (ESがあるとき) の ρ_u を求める．ρ_u は，ページフォールトが発生した際に，要求されるデータがESに存在するか否か，すなわちESにヒットしたか否かに依存する．MS容量が m，ES容量が e である場合，ページフォールトが発生した際にESからページインする確率 (ESヒット率) $h(m,e)$ は(5)式で表される．

$$h(m,e)=\{p(m+e)-p(m)\}/\{1-p(m)\} \quad (5)$$

ESがヒットした場合には，OSはESからのページインとページアウトを行う．ミスヒットの場合にはディスクからのページインに加え，MSに空を確保するためのESへのページアウト，およびES上のページをESから外部記憶へ書き出してES上に空きを確保するESマイグレーションを実施する．ESのページイン，ページアウトおよびESマイグレーションの処理ステップ数をそれぞれ，C_{Ei}, C_{Eo}, C_m とすれば，ρ_u は(6)式のようになる．

$$\rho_p=\{C_{Eo}+h(m,e)C_{Ei}+(1-h(m,e))(C_i+C_m)\}/\mu \quad (6)$$

次に，個々の処理ステップ数である C_{Ei}, C_{Eo}, C_m, C_i, C_o を決定する．これらの値はシステムによって異なるが，おおまかな傾向をつかむことは可能である．C_m, C_i, C_o はディスク装置への入出力が必要であるため，ページフォールトを解決するためのメモリ制御本来の処理のほかに，入出力起動，入出力完了待ち，完了割り込み，タスクスイッチ，等のオーバヘッドが必要となる．一方，C_{Ei}, C_{Eo} は高速な拡張記憶への同期命令を使用したデータ転送のみで良く，上記の処理は不要である．ここでは，ディスクへのページング処理に必要なCPUステップを5000ステップとし，また，拡張記憶へのページングオーバヘッドを1000ステップと仮定して解析を進める．これらの値は汎用大型計算機上におけるページングオーバヘッドの測定結果やハードウェア性能の調査結果ともほぼ一致している[6)~8)]．以上のことから，

$$C_m=C_i=C_o=5000\ \text{Step}$$
$$C_{Ei}=C_{Eo}=1000\ \text{Step} \quad (7)$$

として議論を進める．

(3)，(4)，(6)に(7)式を代入すると，

MSのみの場合には，

$$\rho_u=1/\{1+10000(1-p(m))\} \quad (8)$$

MS+ESの場合には，

$$\rho_u=1/\{1+1000(11-9h(m,e))(1-p(m))\} \quad (9)$$

さらに，(5)式を代入すると，

$$\rho_u=1/\{1+1000(11-2p(m)-9p(m+e))\} \quad (10)$$

となる．

ところで，(2)式は前提条件(1)を満足していない．前提条件(1)では多重度が十分大きく，CPU利用率が100%であることを仮定したが，(2)式はシステムで一つのプログラムのみが実行している場合に当てはまるからである．

そこで，さらに以下のような前提を付け加える．ここで，想定するシステムは十分大きな多重度multで運用され，すべて同一のプログラムが実行されている．そしてそれらのプログラムに均一にCPU資源が割り当てられ，それらの合計が ρ_u であるとする．この時，m/multを新たに m とし，e/multを新たな e とすれば(2)式以降は変更なく適用することができる．

3.2 解析モデルの検証

3.2.1 三階層記憶制御の実現

三階層記憶制御機能はすでに利用可能である．

評価の際に中間階層として使用したESのハードウェア構成を**図1**に示す．ES上のデータは，プログラムから通常の命令のオペランドとしては直接アクセスすることはできない．しかし，ページ (4KB) 単位でMSとの間で10マイクロ秒オーダの高速なデータ転送が可能である．転送の方法には，同期転送 (転送

命令の実行中 CPU は次の命令を実行せずに待つ）と非同期転送（データ転送はチャネルが行い CPU は他の命令を実行可能）がある．非同期転送は大容量データ転送時に CPU オーバヘッドを軽減する．また，ES から外部記憶へのページ移動処理の効率向上を目的として，ES, MS, ディスク間の大量データ転送を 1 回の I/O 起動と I/O 割込み処理で完了する三媒体間転送機能が用意されている．

表 1 ユーザ CPU 利用率の比較
Table 1 Comparison of user-used CPU utilization.

構成		MS システム		MS+ES システム			
ケース		1	2	3	4	5	6
使用メモリ量 (MB)	MS	9.8	14.8	6.8	4.5	14.8	9.8
	ES	—	—	16.0	21.8	16.0	21.8
ユーザ CPU 利用率 (%)	実測結果	1.3	2.0	57.6	57.2	68.6	73.5
	予測	45.9	46.9	45.2	44.8	61.6	67.2

3.2.2 バッチジョブを用いたモデルの検証

本節では，プログラムを高多重で実行させた場合の実測結果を解析モデルに基づく予測値と比較する．

解析モデルを用いて効果を予測するためには，まず実行するプログラムの参照特性を知る必要がある．ここでは，以下のように十分な容量の拡張記憶を用いて参照特性を求めた．

参照特性は，プログラムに与える主記憶量を変化させてその時々のページフォールト率を測定することで求めることができる．しかし，ページングデバイスとしてディスク装置を使用した場合には，ページングが多発した際に，入出力操作等の CPU オーバヘッドや入出力完了待ち時間等の外乱が測定対象プログラムの動作を乱すため，測定の誤差が大きく，測定できるメモリ量の範囲も限定されていた．そこで，ここでは，ページングデバイスとして ES を用いることにより，入出力割り込みやタスクスイッチを起こすことなくプログラムの参照特性を測定した．

具体的には，十分な容量（プログラムが使用する総仮想記憶容量を超える容量）の ES を用意し，主記憶容量と測定対象となるプログラムの多重度を変化させて実行させ，定常状態となったところで，ページング率とユーザ CPU 利用率を測定した．測定対象のジョブとしては，32 MB のデータをクイックソートするジョブを使用した．

この時，多重度を mult，MS 容量を M，その時のページング率を P，ユーザ CPU 利用率を ρ_u とすれば，プログラムのメモリ使用量が M/mult の時のページフォールト発生間隔は，

$\rho_u \times$ MIPS 値/P（ステップ）

となる．

上記の式を測定結果に適用してクイックソートのプログラム特性を求めた結果が**図 2** である．

次に，MS, ES 容量および多重度を変化させ，ディスクへのページングが発生する環境でクイックソートプログラムを実行させて実測結果を予測値と比較した．**表 1** に結果を示す．

ケース 1 およびケース 2 の主記憶のみの構成の場合には予測と実測結果はまったく一致しない．予測ではユーザ CPU 利用率は 40% 台となるはずであるが，実測結果は数 % 程度であった．これは，ディスクへのページングが多発してディスクがネックとなり，いわゆるスラッシングが発生したためである．ここでは，ジョブを最大 12 多重で起動しており，トータル CPU 利用率は 50% 台であった．このことは，OS のページング処理にのみ CPU が用いられ，ユーザジョブはほとんど進捗していないことを示している．

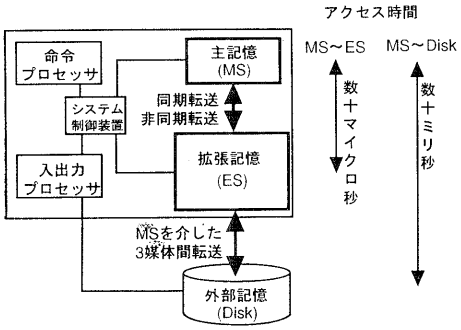


図 1 拡張記憶装置のハード構成
Fig. 1 Organization of extended storage.

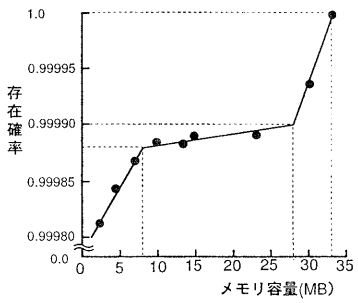


図 2 クイックソートジョブの参照特性
Fig. 2 Program behavior of quick sort job.

解析モデルとの乖離の原因は，ページングオーバヘットの設定にあると考える．モデルでは，ページングオーバヘッドの多くの部分は入出力処理であると考え，ページング一回当たり 5000 Step として解析した．しかし，スラッシング状態においてはページリプレースメント処理のオーバヘッドが多く必要となる．すなわち，スラッシング状態では，ページングフォールトが多発するため主記憶の空きが不足し，それを補うためページリプレースメント処理が常に実施されている．ここでは，システム内のすべてのページに対して UIC の更新とページアウト対象の検索を行っており多くのオーバヘッドを必要とする．この値を厳密に想定することはできないが，一秒間の間に装備されているすべての主記憶がリプレースメントの対象として検索されているとすると，それだけで CPU 数十%のオーバヘッドとなる．これらのことから解析モデルとの間に誤差が生じたものと考える．しかし，実システムではスラッシング状態でシステムを稼働させることは考えられないため，この状態における誤差は問題ない．また，ディスクへのページング頻度に着目することで，モデルを用いた解析結果からスラッシングの発生が予測できるため，モデルの有効範囲も特定することが可能である．

一方，MS+ES システムでは，予想値が実測値より約 10% 程度少な目な値となっているが，傾向は一致している．特に，ケース 5 とケース 6 では MS 容量と ES 容量の合計がほぼ同容量であるが，ユーザ CPU 利用率の傾向は一致している．すなわち，図 2 らわかるように，使用メモリ量が 8 MB から 28 MB までの間はメモリ量を増加させても存在確率はほとんど変化しない．しかし，28 MB 以上 33 MB までの間では大きく変化する．従って，主記憶を約 5 MB 削減しても ES を増加させて，MS+ES 容量を約 1 MB 増設した方が効果的である．

予測に比べて実測のユーザ CPU 利用率が高い原因は OS のスケジューリングの荒さに問題があると思われる．解析では，同時に実行されるすべてのプログラムが同等の CPU サービスを受け，その結果等しい量の MS と ES を割り付けられていると仮定した．しかし，実際には CPU サービスの基準となる優先順位の設定は数十ミリ秒周期で行われる[9]．その間，何らかの原因でいったん ES を多く割り付けられたプログラムは，中断されることなく ES へのページングを多発させて処理を続行させる．一方，ディスクを多く割り付けられたジョブはディスクへのページングのために処理を中断されてしまう．その結果，ES に配置されていたページもマイグレーションによってディスクへ追い出される可能性が高くなり，再びディスクへのページングが発生しやすくなってしまう．一方，ES を多く使用しているプログラムは連続して実行できるため，ES に配置されたページがマイグレーションによってディスクへ書き出される可能性が少なくなる．そのことが再びそのジョブを連続して実行させることとなり，全体としての CPU 利用率は高くなる．

以上の検討から，3.1 節で構築した解析モデルは，ES と MS が装備されスラッシングが発生しない環境，かつ，OS のスケジュールが均等である場合には，実環境のユーザ CPU 利用率の変化の傾向を良く反映している．計算機システムがスラッシング状態にある場合には，性能劣化が激しく実用的ではないため，本モデルは実用的な範囲で利用可能である．

4. モデルを利用した三階層記憶制御の効果予測

本章では，モデルを利用して三階層記憶制御の効果を予測し，ES 容量の最適値を決定する．

4.1 負荷モデルの設定

効果予測に使用するプログラムの特性を設定する．プログラム動作には一般にローカリティがあると言われているが，実際にはローカリティのないプログラムの例も多く報告されている[10),11)]．ここでは，ローカリティの小さなプログラムと大きなものの 2 種類のモデルを用いて評価する．

（1） ローカリティ小のプログラム

ローカリティが小さくシステム全体の性能に悪影響を与えるプログラムである．例えば，ソートプログラムは対象となるデータの全体を繰り返し参照するため，総仮想記憶使用量 V と頻繁に参照される領域量（ワーキングセット）はほぼ等しくなる．**図 3** に本モデルのプログラムの動作特性を示す．

（2） ローカリティ大のプログラム

ローカリティが大きい典型的なプログラムである．総仮想記憶使用量 V に比べ，ワーキングセット w が小さい．**図 4** に本モデルで使用するプログラムの動作特性を示す．

4.2 拡張記憶を用いた三階層記憶制御の効果

上記の特性を持つプログラムに対して，(3)，(4)(6)式を用いて MS および ES 容量の最適値を決定

する．ここでは，メモリに投資できるコスト c が一定であるとして，スループットが最大となる MS と ES の容量を決定する．そのためには MS と ES の単位容量あたりの価格比を知る必要がある．MS と ES は共に半導体記憶装置であるが，ES はページ単位にのみアクセス可能でありアクセス速度も MS に比べて低速である．従って，一般的に ES は MS に比べて安価である．ここでは，まず，MS と ES の単位容量当たりのコストの比が 1/3 であるとする[12]．

MS と ES の単価をそれぞれ a, b とすれば，$am+be=c$ かつ $a:b=3:1$ の条件の下で(10)式を最大とする m と e を求めればよい．(10)式が最大となるのは，分母の変動部分である $2p(m)+9p(m+e)$ が最大となる場合である．

図 5 および **図 6** は，先に定義した 2 つのプログラム特性モデルにおいて，コスト c を変化させたときにユーザ CPU 利用率を最大にするための MS と ES の容量を示したものである．図の横軸はコストを示している．プログラムが使用する仮想記憶総容量 v と MS 単価の積がコストの最大となる．図より，少ないコストを有効に使用するためには，なるべく多くの ES を使用して，MS, ES の合計容量をプログラムが使用する総仮想記憶容量に近付けることが重要であることがわかる．ただし，MS 増設の効果が顕著な場合にはこの限りではない．その判定は，曲線 $p(x)$ の傾きで判断できる．同じコストで ES は MS の 3 倍の容量を装備できるのであるから，ある時点における MS 容量が m, ES 容量が e であるとした場合，点 $p(m)$ の傾きとの点 $p(m+e)$ の傾きを比較し，

$$p'(m)/2p'(m+e)>9/2$$

であれば ES を増やさずに MS を増加させねばならない．ここで，分母の係数が 3 ではなく 2 であるのは，MS 容量を増加すれば MS+ES 容量も増えるからである．

図 7, **図 8** は各コストにおいて最適な容量の MS と ES を装備した場合のユーザ CPU 利用率を，MS のみのシステムと比較したものである．各図の左上がりの線は，図では直線で近似しているが，わずかに下に凸な曲線である．図より，ローカリティが大きい場合

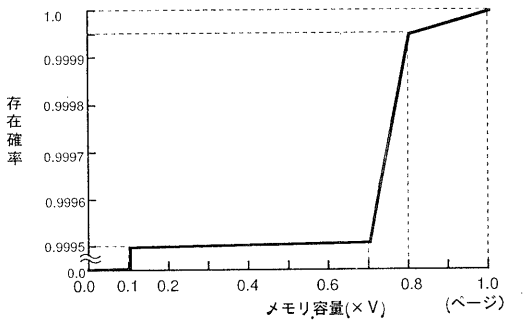


図 3 ローカリティの小さいプログラムの負荷モデル
Fig. 3 Program behavior for a program without locality.

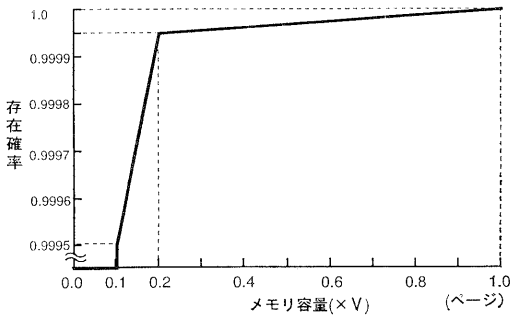


図 4 ローカリティの大きいプログラムの負荷モデル
Fig. 4 Program behavior for a program with large locality.

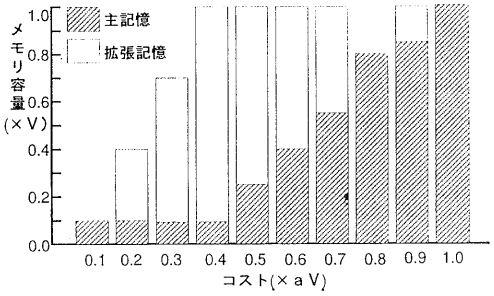


図 5 ローカリティの小さいプログラムに対する主記憶，拡張記憶の最適容量
Fig. 5 Optimal MS and ES size for a program without locality.

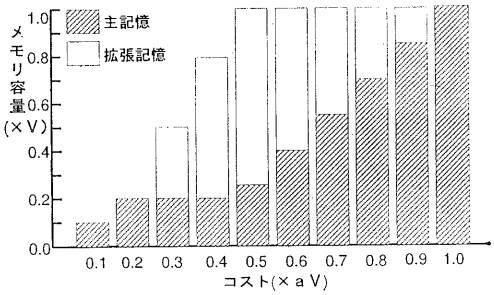


図 6 ローカリティの大きいプログラムに対する主記憶，拡張記憶の最適容量
Fig. 6 Optimal MS and ES size for a program with large locality.

にも小さい場合にも ES を用いたシステムがスループットの面からは優れていることがわかる．特にローカリティの小さいプログラムに対する優位性が顕著である．ただし，MS+ES システムにおいて ES 容量が 0 である場合には MS のみのシステムが若干性能が良い．これは，MS+ES システムでは必ず一旦 ES を介してデータをディスクへ書き出すモデルとなっているため，ES 容量が 0 である場合にも ES へ書き出すためのオーバヘッドが必要となっているからである．しかし，その値は無視できる程度である．

次に，ある性能を発揮するために必要なメモリコストを求める．どの程度の性能を目標とするかはシステムによって異なるため，一般的な指標を決定することは困難であるが，ここでは以下の 2 種類を想定する．

（1）ページングを発生させない理想的な環境で実行した場合に近い性能
（ユーザ CPU 利用率＝90%）

（2）スラッシングを回避してなんとかジョブを実行できる性能
（ユーザ CPU 利用率＝50%）

これらの性能を発揮できるコストを図 7，図 8 から求める．

（a） ローカリティ小のプログラム

90% の性能を得るために必要なコストは，総仮想記憶容量を MS で実装した場合を 1 とすると，MS+ES システムで 0.87，MS システムで 0.96 である．ES を使用することで 1 割以上のコスト削減が可能である．

50% の性能を得るためには，MS+ES システムで 0.4，MS システムで 0.77 のコストである．この場合には，コストを約半減できる．

（b） ローカリティ大のプログラム

90% の性能を得るために必要なコストは MS+ES システムで 0.45，MS システムで 0.83 である．

50% の性能を得るためには，どちらも 0.18 のコストで良い．

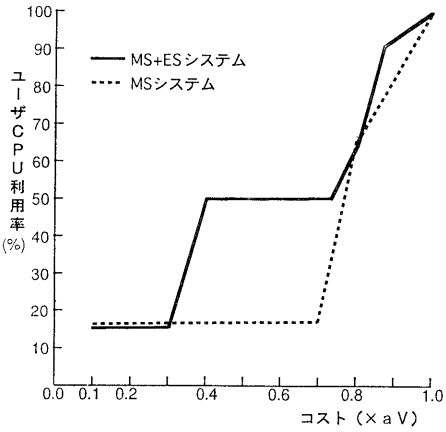


図 7 ローカリティ小のプログラムに対する同一コストにおけるユーザ CPU 利用率の比較
Fig. 7 CPU utilization for a program without locality.

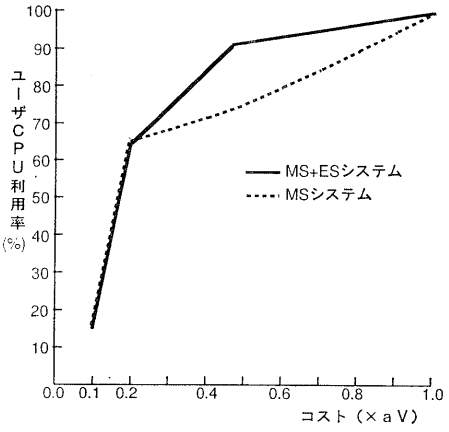


図 8 ローカリティ大のプログラムに対する同一コストにおけるユーザ CPU 利用率の比較
Fig. 8 CPU utilization for a program with large locality.

拡張記憶を用いた三階層記憶制御は，低コストで，ローカリティ小のプログラムに対してはスラッシングのような最悪の状態を回避でき，ローカリティ大のプログラムに対しては高性能化を達成できる．

4.3 ES 容量の最適値

次に，上記の性能を満足するために必要な MS，ES の容量比を求める．

一般的に実行するプログラムの参照特性をユーザが知ることは困難である．さらに，特性に合わせて MS と ES の容量を変更させて実行させることはほとんど不可能である．しかし，計算センタ等の運用においては具体的な MS，ES 容量を決定する必要がある．一般的にはなるべく多くの ES を用意してローカリティの悪いプログラムに備える必要がある．しかし，逆に ES を増量しすぎると MS が不足して性能が低下してしまう危険性もある．

そこで最低限の MS 容量の一つの目安として，MS+ES 容量を総仮想記憶容量と等しい値に保ったままコストを変化させて，MS のみのシステムとの性能を

比較した．**表2**に結果を示す．ローカリティ小のプログラムに対する性能比において，コスト0.8の場合に瞬間的に性能が低下するが，おおむね，コストが0.5以上の場合にはMSシステムより高性能である．

この結果からは，MSとESの容量比を1：3とすれば，低コストでMSシステム以上の性能を発揮できる，と言える．すなわち，あるコストが与えられた場合に，その半分のコストをMSに使用し，残り半分をESに使用すれば，元のコストをすべてMSに使用した場合より高性能を達成できる．さらに，元のコストの2倍のコストをすべてMSで実装した場合と比較した場合でも，ローカリティ大のプログラムに対しては約90%，ローカリティ小のプログラムに対しても約50%の性能を保証できる．

5. 結果の検討

5.1 プログラムのローカリティ

プログラムの参照特性は多様である．解析に使用した2種類のプログラムは極端な例であり，すべてを代表しているわけではない．しかし，通常のプログラムはこの2つのプログラムの中間に存在すると思われるため，この2例に対する効果を明らかにすれば多くのケースにおいても効果があると考える[1),6),7),13)]．

例えば，第3章でモデルの検証に使用したクイックソートジョブの特性の最適なMS，ES容量比とそのときのユーザCPU利用率を，図2を元に求めた結果が**図9，図10**である．

図10では，想定した参照特性では見られなかった特徴がある．それは，コストが10MB（主記憶換算）を超えたところで，一旦主記憶容量が少なくなり，ESが増加した点である．これは，参照特性が2段階に変化したことによる．いわゆるニーポイントが2か所に存在するためである[1)]．このような参照特性においてコストを0から増加させた場合，まず第一番目のニーポイントまでESを増やそうとする．その次にMS+ES容量を第一のニーポイントに保ったままMS容量を増やす．その後，第二のニーポイントに向かってESを増やそうとするが，ある時点までコストを増やした際に，MS容量を削減してもES容量を増加させてMS+ES容量を第二のニーポイントに近づけようとする．この傾向は，第一のニーポイントへの勾配に比べて第二のニーポイントへの勾配が大きいほど，傾向が大きくなる．

以上のことが原因となって，図10は，形状的には，参照特性を二つ合わせたような形となっている．しかしマクロな参照特性は，本解析で使用した2種類の特性の中間に属する．

図3で示したプログラム特性は，ローカリティの低いと言うよりはローカリティの無い，あるいは負のローカリティとも言うべき特性のあるプログラムであ

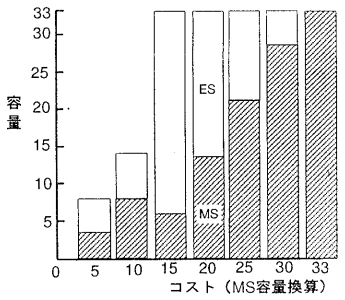


図9 クイックソートジョブの最適容量比
Fig. 9 Optimal MS and ES size of quick sort job.

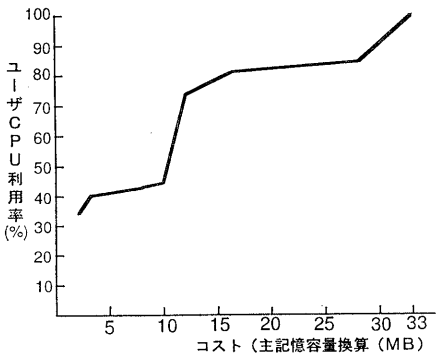


図10 クイックソートプログラムに対する同一コストにおけるユーザCPU利用率の変化
Fig. 10 CPU utilization for quick-sort program.

表2 和を一定（総仮想容量）としてMS, ES容量を変化した場合の性能の変化
Table 2 Performance comparison with fixed sum of MS and ES capacity.

コスト		0.4	0.5	0.6	0.7	0.8	0.9	1.0
MS, ES 容量比		1：9	1：3	1：2.5	1：0.82	1：0.43	1：0.18	1：0
性能比 (MSシステム=1)	ローカリティ小	3.0	3.0	3.0	3.0	0.75	1.16	1.0
	ローカリティ大	0.69	1.20	1.16	1.12	1.08	1.04	1.0

り，この種のプログラムに対しては三階層記憶制御の効果は大きい．通常のシステムではこのようなプログラムがシステム全体の性能を低下させることが多い．

5.2 MS, ES の価格および性能

解析では，MS と ES の単価比を 3：1 であるとした．この値が変化した場合には，解析結果は以下のように変化する．

（a） ES が高価である場合

MS と ES の価格が近づくのであるから，一般的に ES の効果は小さくなる．ローカリティの低いプログラムに対しては ES を用いて低コストでワーキングセットを満たすだけの MS, ES 容量を実現しなければならないが，ES が高価であればこのために必要なコストが増加する．

例えば，単価比が 2：1 と仮定した場合に MS, ES を 1：3 の容量比で装備としようとすると，単価比が 3：1 の場合に比べて 25% のコスト増となる．

（b） MS が安価である場合

上記の逆であり，より少ないコストで目標の性能を達成できる．MS, ES の最適容量比においても，より MS 容量を削減することができる．

また，これまでの検討は，ES の性能も固定して検討してきた．すなわち，ES へのページングはディスクに対するそれの 1/5 程度のオーバヘッドを仮定してきたが，この値が大きくなれば大きくなるほど三階層記憶制御の効果は薄れる．しかし，ES が半導体メモリを用いて実現され，CPU からの同期的命令でアクセスされる限りはこの値は大きくは変化せず，ES を使用することの効果は大きい．さらに，後述するように，ES とディスクのアクセス時間は数百倍から千倍程度 ES が早いため，応答時間の短縮にはさらに効果が大きい．

5.3 ディスク転送時間および応答時間

解析はスループットに着目し，ユーザが使用する CPU 利用率を最大とすることを目的として定式化してきた．しかし，例えば TSS やオンライントランザクション処理などのように応答時間が重要なファクタとなるケースも多く存在する．

ディスク転送時間は応答時間だけでなく，これまで前提としてきた CPU 利用率が 100% であるという仮定にも大きく影響する．これまでの解析では，ページング処理に必要なオーバヘッドを求め，その残りの CPU 資源をユーザがすべて利用できるものとして定式化してきた．しかし，ディスクへのページングが発生した場合などでは CPU 利用率が 100% とはならない状況も発生する．例えば，主記憶の存在確率が 0.9995 であるようなプログラムを 10 MIPS のマシンで実行させた場合，ディスク入出力時間を 20 ミリ秒とすれば，単純には各プログラムが 0.2 ミリ秒実行して 20 ミリ秒ウェイトするのであるから，10 多重で実行してもシステムの CPU 利用率は 10% である．従って，これまでの解析結果はディスクへのページングがあまり発生しない場合にのみ適用でき，そうでない場合には CPU 利用率は高めに評価される．

次に，ディスクアクセス時間が応答時間に与える影響を検討する．ページングによる応答時間の伸びはページング時間の合計に等しい．ここで，ディスクに対するアクセス時間は 20 ミリ秒程度であり，ES は数十マイクロ秒[8]であることを考慮すると，ほとんどの場合，ディスクに対するページングを発生させないことが，応答時間を短縮するための必要条件となる．すなわち，応答時間を重視する場合には，MS と ES を用いてプログラムが使用する総仮想記憶容量のメモリを用意することが必須となる．この結論は，スループットを重視した場合の解析結果と同一である．

また，そのことが実質的な多重とを減らし，一つとトランザクションが使用するメモリを増やして，再び応答時間の短縮に貢献する．

5.4 TSS 環境における効果の比較

三階層記憶制御の効果を TSS 環境で実施した．ここでは，数百台の TSS 端末がプログラム開発とプログラム実行を実施している環境を想定した．TSS 環境の構築には仮想端末を実現する多端末シミュレータを使用した．

図 11 に，TSS コマンドの参照特性を示す．図よりローカリティの高い参照特性であることがわかる．こ

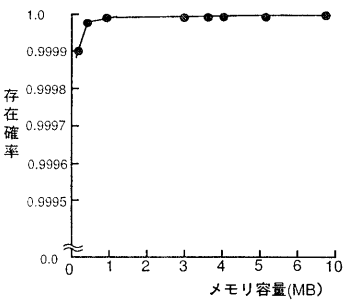


図 11 TSS ジョブの参照特性
Fig. 11 Program behavior of TSS.

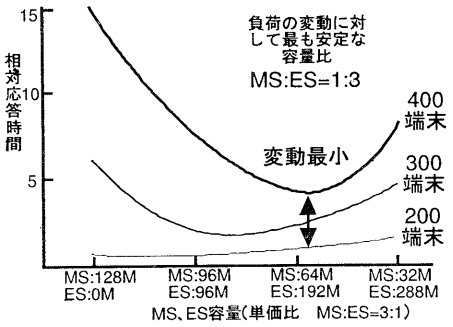


図 12 コスト一定とした場合の TSS 応答時間の推移
Fig. 12 TSS response time with fixed cost.

のような場合には，ES を用いて総仮想記憶容量まで MS+ES を増やすことで低コストで高性能が達成できる．解析モデルでは，プログラムの負荷特性にも依存するが，MS : ES の容量比を 1 : 3 程度に設定した場合が性能価格比が優れているという結論であった．

実システムの負荷を用いて同一コスト時に，負荷，容量比をそれぞれ変化させ，負荷の変動に対して最も安定した容量比を求めたものが**図 12** である．図より，負荷の変動に対して最も性能の変化が少なく安定した性能が得られるのは，MS, ES の容量比が 1 : 3 とするときであることが分かる．解析モデルとは異なる前提ではあるが，共に容量比が 1 : 3 であるときに性能価格比が優れている，という結果を得た．

ここでは，400 端末の場合の応答時間がキーポイントとなる．すなわちこの場合には，MS+ES 容量が 256 MB となる点が 400 端末時の最小応答時間を発揮する点となり，結果的に安定な応答時間を得られるポイントとなる．しかし，この時の容量比の 1 : 3 は MS, ES の価格比が 3 : 1 である場合にのみ成立する．例えば，価格比が 2 : 1 であると仮定し，同一コストで合計容量を 256 MB としようとすると ES をより多く装備しなければならない．そして，そのことは MS 容量を削減し，400 端末時の応答時間を悪化させる．その結果，端末台数が変化した場合の応答時間の変動が大きくなり，三階層記憶制御の効果は小さくなる．すなわち，ES が安価であればあるほど三階層記憶制御の効果は大きく，安定した性能を達成することができる．

6. おわりに

主記憶と外部記憶とのアクセス速度の差であるいわゆるアクセスギャップの存在は，計算機システムの性能向上の大きな問題点である．

本研究では，主記憶と外部記憶の間に拡張記憶を設ける三階層記憶制御を提案し，その効果をモデル化によって明らかにした．

現実の計算機システムでは，負荷の予測は困難であるため，負荷の変動に対しても安定した性能の確保できる三階層記憶制御方式は有効である．

今後もハードウェア技術の進歩と共に異なる性質を持つ記憶媒体が数多く実現されるであろう．複数の記憶媒体で構成される記憶階層の管理方式は，高性能計算機システムを構築するための重要な技術となる．

謝辞 本研究の推進にあたり(株)日立製作所システム開発研究所堂免信義元所長ならびに久保隆重元副所長の御指導に深謝する．また，ソフトウェア開発本部ならびに日立ソフトウェアエンジニアリングの方々からは，実用化における多くの御意見と御協力をいただいた．特に，ソフトウェア開発本部小平光彦元部長には，本研究を VOS 3/AS に適用する機会をいただき，多大の御支援をいただいた．

参考文献

1) Denning, P. J.: Working Sets Past and Present, *IEEE Trans. Softw. Eng.*, Vol. ES-6, No. 1, pp. 64-84 (1980).
2) Tanenbaum, A. S.: *Modern Operating Systems*, pp. 74-124, Prentice-Hall International (1992).
3) Gelb, J. P.: The Storage Hierarchy: Past, Present, and Future Considerations, *CMG '90 Proceedings*, pp. 922-931 (Dec. 1990).
4) Smith, A. J.: Disk Cache—Miss Ratio Analysis and Design Considerations, *ACM TOCS*, Vol. 3, No. 3, pp. 161-203 (1985).
5) 片田ほか：拡張記憶を利用した仮想記憶制御方式ならびに性能評価，情報処理学会オペレーティングシステム研究会資料，91-OS-50, pp. 1-8 (1991. 3. 15).
6) Lewars, W. D.: Use of an SSD for Service Level Improvement, *CMG '90 Proceedings*, pp. 516-525 (Dec. 1990).
7)「90 年代前半を担う新世代の大型汎用機」，日経コンピュータ，1990. 10. 15, pp. 46-58.
8)「大型機アーキテクチャはどこへ向かうか」，日経コンピュータ，1989. 9. 11, pp. 56-77.
9) HITAC マニュアル：VOS 3/AS センタ運営—JSS 3 編—，6180-3-101-10.
10) Franklin, M. A., Graham, G. S. and Gupta, R. K.: Anomalies with Variable Partition Paging Algorithm, *CACM*, Vol. 21, No. 3, pp. 232-236 (1978).

11) Denning, P. J. and Slutz, D. R.: Generalized Working Sets for Segment Reference String, *CACM*, Vol. 21, No. 9, pp. 750-759 (1978).
12) 日経ウォッチャー IBM版, 1993. 11. 8, 日経BP社.
13) Smith, A. J.: A Modified Working Set Paging Algorithm, *IEEE Trans. Computer*, Vol. C-25, No. 9, pp. 907-914 (1976).

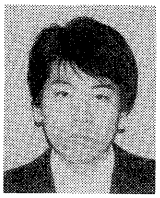
(平成5年4月28日受付)
(平成6年6月20日採録)

新井 利明（正会員）

昭和29年生．昭和53年早稲田大学大学院理工学研究科修了．同年(株)日立製作所入社．システム開発研究所に勤務し，オペレーティングシステムの性能評価および性能向上方式の研究に従事．現在は，分散システムの基本ソフトウェアの研究開発を担当．

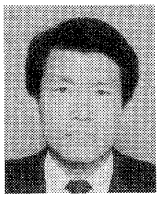
吉澤 康文（正会員）

1944年生．1967年東京工業大学理工学部応用物理学科卒業．同年(株)日立製作所入社．中央研究所に勤務し，HITAC 5020/TSS の研究開発に従事．1973年発足したシステム開発研究所に勤務．以来，仮想記憶，大規模 TSS，オンラインシステム，など大型計算機のオペレーティングシステムの性能評価ならびに記憶管理方式の研究開発に従事．これらの研究により，1981年3月東京工業大学より工学博士を授与．また，オペレーティングシステムのテスト・デバッギングシステムの開発に従事．現在はハイエンドサーバ，超並列計算機，リアルタイムシステムなどの研究開発を推進している．情報処理学会論文賞（昭和47年度）授与．IEEE Computer Society 会員．同研究所主管研究員．著書「オペレーティングシステムの実際」(昭晃堂)．東京農工大学，電気通信大学，東京工業大学，非常勤講師．



細内 昌明（正会員）

昭和40年生．昭和59年東北大学工学部情報工学科卒業．同年(株)日立製作所入社．現在，同社システム開発研究所に勤務．汎用大型計算機のオペレーティングシステムの主にメモリ管理の研究開発に従事．



大房 義隆（正会員）

昭和25年生．昭和44年富山工業高校機械科卒業．同年(株)日立製作所入社．現在，同社ソフトウェア開発本部主任技師，汎用大型コンピュータのオペレーティングシステムの研究・開発に従事．