ディスク・キャッシュ装置のシミュレーションによる効果測定†

菅 隆志†† 上田尚純†† 田中千代治††

計算機システムの主記憶と二次記憶（ディスク）のアクセス時間には大きな隔差があるが，CCD や磁気バブルのような，機械動作を伴わない記憶素子を使ったバッファ・メモリをディスク装置に組み込み，見かけ上のディスク・アクセスを高速化する，ディスク・キャッシュの手法が新しい計算機アーキテクチャのひとつとして最近注目されてきている．

本論文では，このディスク・キャッシュの構成に必要な機能を洗い出すと共に，実働計算機システムから収集したディスク・アクセス動作のトレース・データをもとにシミュレーションを行い，ディスク・キャッシュの効果を定量的に評価している．また，この効果に最も大きな影響を与えるファイル・アクセスの動的特性を利用形態が異なるファイル領域別に測定することにより，ディスク・スループットを最大にする最適なディスク・キャッシュの利用形態を考察している．

1. はじめに

計算機システムにおける主記憶とディスクのアクセス時間のギャップ（現在 10^5 程度の速度差がある）を緩和するため，CCD，磁気バブルあるいは低速RAMなどの大容量で安価な記憶素子の利用が考えられている[1]．アーキテクチャ上の実現方法としては，(i) 主記憶の階層化を行うためのバック・アップ・メモリ，(ii) ディスク・キャッシュ，(iii) 固体ディスクなどが提案され，実用化が進められている[2]．(i) の主記憶の階層化は単一仮想記憶手法と併用することにより，ソフトウェア作成の大幅な簡略化を達成できるため最も本格的なアーキテクチャと言える．(ii) のディスク・キャッシュは，CPU（中央処理装置）と主記憶間におけるキャッシュ・メモリの手法を主記憶とディスク間に適用したものであり，見かけ上，ディスク・アクセス時間を短縮することによりシステムの性能向上を達成できる．ディスク・キャッシュの存在は，ソフトウェアから見て透明であるように作れるため，既存システムのソフトウェアに変更を加えることなく採用できることがこの方式の利点である．(iii) の固体ディスクは従来のディスクやフロッピ・ディスクなどをCCD や磁気バブルなどの記憶素子で実現するものである．機械的動作がなく，コンパクト化と高信頼化が期待できるため，今後の普及が予想されるが，大容量の固体ディスクはコスト面から当分難しいであろう．

本論文では，(ii) のディスク・キャッシュのモデルを作成し，このモデルをもとにディスク・アクセス動作の実測データを使用してシミュレーションを行い，ディスク・キャッシュの効果について検討している．

なお，ディスク・キャッシュの効果は，そのシステムのファイル・アクセス動作の動的特性（特に，局在性）に最も依存する．このため，本シミュレーションでは，特にこの動的特性を LRU スタック[3]を利用して定量的に測定すると共に，利用形態が異なるファイル領域別にこの手法を適用することにより，各ファイル領域ごとにディスク・キャッシュの効果が大きく異なることを示し，これらの結果により，最適なディスク・キャッシュの構成方法を考察する．

以後では，まず，対象とする計算機システムの構成と利用形態とファイル管理方法について述べ，次にディスク・キャッシュのモデルを示し，その後，さまざまのシミュレーションの結果とその考察を順次述べる．

2. 対象システムとトレース・データ

2.1 対象システムの構成と利用形態

ディスク・アクセス動作のトレース・データ収集の対象として，3つの大型計算機システムを選んだ．各システムのソフトウェアの利用形態は**表1**に示すよう

† A Performance Evaluation of Disk Cache Device by Simulation by TAKASHI KAN, TAKASUMI UEDA and CHIYOJI TANAKA (Computer Laboratory Mitsubishi Electric Corporation).

†† 三菱電機（株）開発本部計算機研究部

表 1 ディスク・アクセス動作のトレース対象システムの特徴

Table 1 Specifications of traced systems.

	SYSTEM A	SYSTEM B	SYSTEM C
主な利用形態	・科学技術計算 ・TSS によるプログラム開発	・事務計算 ・在庫管理 ・トランザクション処理	・科学技術計算 ・TSS によるプログラム開発
スワッピング	YES	YES	YES
ディマンド・ページング	NO	NO	NO
平均ディスク・アクセス頻度	10.8 回/秒	30.6 回/秒	13.8 回/秒

に，それぞれ少しずつ異なっている．一方，ハードウェアの構成は各システムともほとんど等しく，主記憶容量＝0.5M バイト，ディスク容量＝100M バイト×7台であり，それぞれ十数台の端末が接続されている．

トレース・データの収集は，各計算機センターのクローズ業務の時間帯を対象として，ハードウェア・モニタとソフトウェア・モニタを使って行った．また，データ収集時にはシステムの稼動状況も合わせてトレースしており，これによるといずれのシステムとも常時バッチ・ジョブが 1〜3 本並列して処理され，オンライン端末は 7〜12 台が活動状態であった．

2.2 ファイル領域の管理

対象システムで採用しているファイル領域管理方法について記述しておく．

ディスクは固定長セクタ方式（1セクタ＝1k バイト）である．ディスク・ボリュームは，公用ボリュームと私用ボリュームとがあり，前者はシステム稼動中は常時マウントしておかなくてはならない．

公用ボリュームは次の4つの領域からなる．これらの領域の位置や大きさはシステム生成時に物理シリンダ単位で指定できる．

（i） **PSA** 領域 (Parmanent System Area)

先頭部はオペレーティング・システムが使用するシステム・プログラムとデータが，ファイルごとに物理的に連続して置かれ，残りはユーザのスワップ領域が物理シリンダ単位で割り付けられる．

（ii） **PPA** 領域 (Parmanent Paging Area)

ディマンド・ページング機能で使用するページング領域である．ページ単位（1ページ＝2k バイト）の割り付けがされる．

（iii） **PER** 領域 (Peripheral Area)

スプーリング領域である．カード読取装置からのジョブ・データ・ファイル，行印字装置への出力ファイルなどが置かれる．やはりページ単位で割り付けがなされる．

（iv） **PFA** 領域 (Permanent File Area)

ユーザ・ファイル領域であり，ファイル管理データとファイル・データが置かれる．ページ単位の割り付けがなされる．領域としては最も大きい．

私用ボリュームは，ユーザ・ファイル領域だけから構成される．領域の割当て方式は PFA 領域と同じである．

以上のように，PSA 領域の一部以外は，ディスク領域を固定長ブロック（ここでは，ページ長）に分割し，ディスク領域の要求があるごとに，このブロック単位で領域を割り当てるのがファイルシステムにおけるディスク領域管理の基本となっている．この手法は，ファイル作成時の初期領域割り付けが不要なことや常にディスク領域を目一杯に使用できる事などの利点があるため，TSS 指向のシステムでは特に有効であるが，さらに，固定長やセクタ方式のディスクと共に，単一仮想空間方式やディスク・キャッシュなどのアーキテクチャを実現する上でも便利な手法であるといえる．

2.3 ディスク・アクセス動作の実測データ

ディスク・アクセス動作のうち，ディスク・キャッシュのヒット率に直接的に影響を与える要素として，ファイル・データのワーキング・セット・サイズ*，転送データ長，READ と WRITE の比率などがある．

また，ディスク・キャッシュのシステム全体に対する効果に影響を与える要素として，単位時間当りの入出力回数，ディスク・アームのシーク距離，CPU とディスク入出力動作の並列実行の度合いなどがある．これらの要素の値は，やはりディスク・アクセス動作のトレース・データを使って統計をとった．

3. ディスク・キャッシュの基本モデル

3.1 基本モデル

シミュレーションで採用したディスク・キャッシュのモデルを**図1**に示す．キャッシュ・メモリ（以下，キャッシュと呼ぶ）をディスク装置ごとに持つ方式も考えられるが，1か所に集中して持つ方がコスト的にも効率的にも有利である．

ディスク領域とキャッシュ領域は共通の長さの固定長ブロックに分割され，このブロック単位でディスク・データのコピーがキャッシュ内に保持される．キャッシュの制御方式については，主記憶のキャッシュと比

* 単位期間当りにアクセスするディスク・データ領域の範囲．

較して，時間的に余裕があるため，ある程度複雑な制御をマイクロ・プログラムで行うことが可能である．このため，キャッシュとディスクのブロック・マッピング方式としてフル・アソシアティブ方式を行い，キャッシュ・メモリの内容の更新アルゴリズムとして LRU アルゴリズムを用いた．

また，主記憶，キャッシュ，ディスク間のデータ転送はすべてディスク・キャッシュ制御部（以下，制御部と呼ぶ）内のブロック長の転送バッファを経由して行う．

CPU からディスクに対して READ, WRITE コマンドが出されたときのディスク・キャッシュの動作を以下に示す．ただし，これらのコマンドに先立ち，必ず SEEK コマンドが出されており，制御部ではこれを受けて，SEEK アドレス（シリンダ番号，トラック番号，セクタ番号）をブロック番号に変換している．

3.2 READ 動作

要求するデータを含むブロックがキャッシュ上にあるかどうかを，キャッシュ制御部で保持しているキャッシュ・ブロック管理テーブルを参照することにより調べ，キャッシュ上にあればそこからデータを取り出して主記憶へ送る．また，キャッシュ上にない場合にはディスクからデータを取り出し主記憶へ送ると共にキャッシュにも書き込む．

3.3 WRITE 動作

WRITE 動作に対しては，次の2つのモードに対してシミュレーションを行った．

（a） Write Through モード

データは直接ディスクに書き込み，キャッシュには書かない．キャッシュ上に対応するブロックがあれば，そのブロックを無効にする．

（b） Write After モード

書き込みデータはキャッシュに書き込まれ，書き込み完了により WRITE コマンドは終了する．キャッシュからディスクへの書き出しは，(i)キャッシュへの書き込み終了後すぐに行う方法，(ii)ブロックがキャッシュから追い出されるときに行う方法などがあるが，シミュレーションでは，ヒット率に対する影響がないので，どの方法を採用するかは意識していない．

なお，以後の WRITE 動作において，特に記さない場合は，Write After モードを指す．

3.4 ヒット率の定義

ディスクの入出力データはさまざまな長さのものが存在するため，本論文ではヒット率として次の2つを定義しておく．

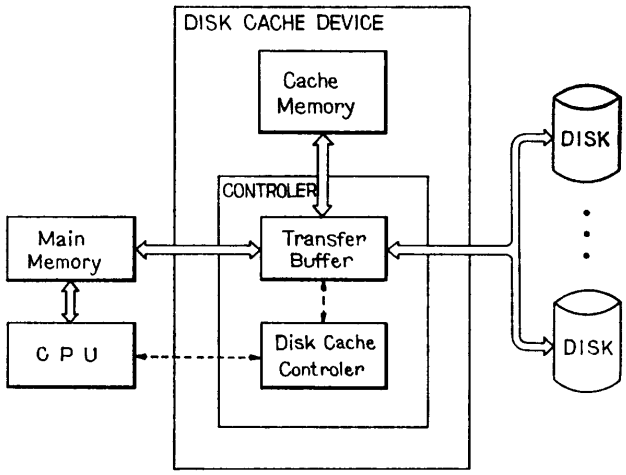

図 1 ディスク・キャッシュ装置のモデル
Fig. 1 Disk cache device model.

（i） コマンド・ヒット率

すべての入出力コマンドに対する，ヒットした入出力コマンドの割合．ただし，ここでヒットしたとは，ひとつのコマンドで要求されるデータのすべてがキャッシュ内にある場合のみを言う．以後，単にヒット率と記すときは，コマンド・ヒット率を指す．

（ii） ページ・ヒット率

各入出力コマンドにそのデータ長を重みづけして算出したヒット率．すなわち，要求されたすべてのページに対するヒットしたページの割合を示す．

4. シミュレーション結果と考察

4.1 キャッシュ容量—ヒット率

キャッシュ容量の増加に伴うヒット率の向上を**図2**に示す．ブロック・サイズにより多少異なるが，ページ・ヒット率は容量が 2M バイト程度で約 95% となり，ほとんど飽和する．また，コマンド・ヒット率は容量が 4M バイト程度で 80〜90% となって飽和する．

次にヒット率が飽和してしまう原因を別な観点から探ってみる．**図3**は入出力コマンドがヒットしたときに，それが LRU スタック[3]のどの深さでヒットしたか示す分布図とその累積値である．図3ではシステムCの例を示してあるが，いずれのシステムともヒットしたデータの 95% 以上はスタックの深さが 500（容量では 1M バイト）までプッシュされる以前に再び

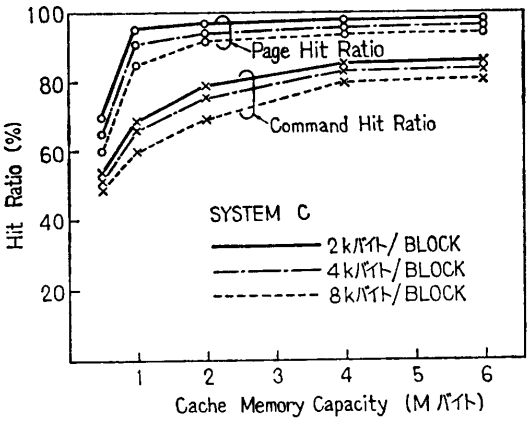

図 2 キャッシュ容量—ヒット率

Fig. 2 Cache memory capacity-hit ratio.

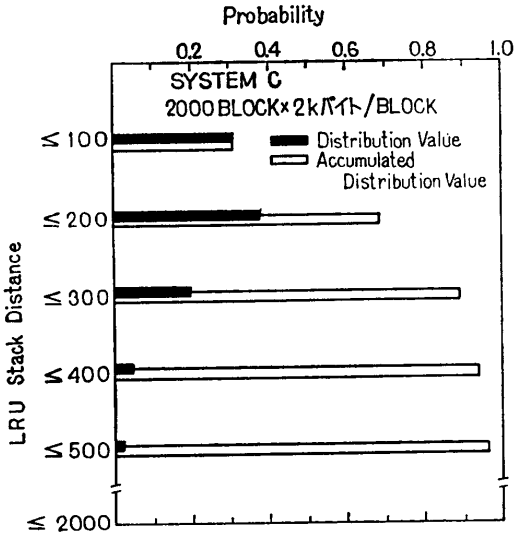

図 3 LRU スタックのヒット分布とその累積値

Fig. 3 Hit distribution in LRU stack.

アクセスされている．

4.2 ブロック・サイズ—ヒット率

キャッシュ・メモリとディスク間はブロック単位のデータ転送が行われるが，ブロック・サイズの決定にあたっては，ファイル・システムで採用しているファイル・データのディスク上での領域割り当て方法を考慮しておかねばならない．

本論文で対象としたシステムでは，**2.2** で述べたように，各ファイルの特性に適した割り付け方法をとっているが，その多くは要求があった時点でページ単位で割り付けを行っている．したがって，ディスク上では，一般的にファイルは物理的に不連続なページ単位で割り付けられている．このため，ブロック・サイズを数ページ長にすることは，むしろ余分なデータをキャッシュ内に置く結果となりヒット率は低下することが予想される．シミュレーションの結果は，図2に示す通り，ブロック長をページ長にとった場合がヒット率は最も良く，ブロック長を大きくするに伴い，ヒット率は低下している．

表 2 各システムのヒット率

Table 2 Hit-ratio for each system.

SYSTEM	A	B	C
Command Hit-rate	89.1%	83.5%	89.6%
Page Hit-rate	98.5%	95.1%	98.2%

2,000 BLOCK×2k バイト/BLOCK

表 3 ファイル領域別のヒット率

Table 3 Hit-ratio for each file area.

FILE AREA	SYSTEM A	SYSTEM B	SYSTEM C
PSA	98.3%	96.6%	99.6%
PPA	—	—	95.3%
PER	82.7%	74.7%	84.9%
PFA	76.4%	91.1%	78.9%
Private	62.5%	60.0%	61.4%

1,000 BLOCK×2k バイト/BLOCK

4.3 各システムのヒット率

各システムのヒット率を**表2**に示す．システムBはデータベースを使った事務処理業務が主体であり，TSSによるプログラム開発や科学技術計算が主体のシステム A，C に比べてヒット率は低い．これは，システムBではデータベースとして使用する私用ボリュームへのアクセスの割合が多く，これらのヒット率が低いためである．

一方，システム A はスワッピング，システム C はスワッピングとページングの割合がそれぞれ多く，これらの処理に際してのディスグ・アクセス時は，次節で示すようにヒット率が非常に高いため全体としても高いヒット率となっている．

4.4 ファイル領域別のヒット率

各ファイル領域別の動的なアクセス特性を知るためにそれぞれ個別のファイル領域へのアクセスだけにキャッシュを使用した例を**表3**と**図4**に示す．以下，ファイル領域別に検討する．

PSA 領域——この領域には，アクセス頻度が高く局在性が高いシステム・プログラム領域とスワップ領域が存在するためにヒット率は非常に高い．また，図4からわかるように，ほかのファイル領域へのアクセスに比べ LRU スタックの深い位置でヒットする確率が高く，分布の様子もかなり異なる．これは，スワッピングにより比較的長く，しかもさまざまな長さの

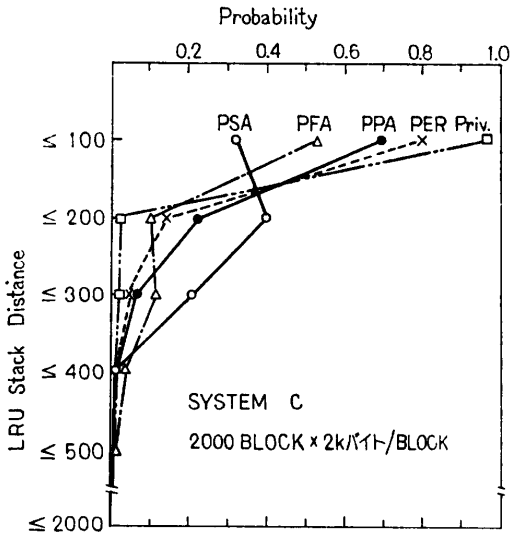

図 4 ファイル領域別の LRU スタックのヒット分布
Fig. 4 Hit distribution of LRU stack for each file area.

データ転送が行われるためである.

PPA 領域——スワッピングと同様，ページングのためのこの領域へのアクセスも局在性が高く，ヒット率は 95% 以上である．しかも，ページングによるデータ転送はすべてページ単位であり，データの転送時間に比べ，ディスクのアクセス時間の占める割合が大きい．これらのことからディスク・キャッシュの効果はページングに対して大きいことが予想される.

PER 領域——スプーリングのためのデータは基本的にはFIFOで処理され，場合によっては長時間大量のデータが留まることもある．そのため，ヒット率はそのときのシステムの状況によりかなり変化するが，一般的には，LRU アルゴリズムではヒット率の向上は期待できない.

PFA 領域——ヒット率は各ユーザの利用方法により大きく影響を受ける．一般に，多くのユーザがTSSによりプログラム開発を行うようなシステム A やシステム C よりは，ほぼ同種の事務処理業務が主体のシステムBの方がヒット率は高い.

私用ボリューム——データベースとして使用する場合が多く，ヒット率は低く約 60% である．また図4で示すように，この領域へのアクセスがヒットする場合にはそのほとんどが LRU スタックの浅い位置でヒットしており，キャッシュ容量を大きくしてもこの領域に対する効果はあまり変わらないことがわかる.

以上のように，ファイル領域ごとにそのアクセスの動的特性は異なっていることがわかる．前節で述べた各システム全体のヒット率は，これらの各ファイル領域へのアクセスの特性が，そのアクセス頻度に応じて複雑に重なり合ったものに対するヒット率である.

最適なディスク・キャッシュの構成法を考える際には，そのシステムの利用形態に応じて，各ファイル領域別に最適な制御を行うことが最も効果的となるといえる.

4.5 バイパス・モード

今回測定対象としたシステムでは，ページ単位（＝2kバイト）での入出力が多いため，ディスク・キャッシュの効果は，データ転送時間の短縮化よりもアクセス時間の短縮化の方がより大きい．このため，データが長くなるほど転送時間の比重が大きくなるため，長いデータに対してはキャッシュの使用を抑止するバイパス方式が考えられる.

試みに，2kバイトより長いデータ（ほとんどはスワッピング・データである）をすべてバイパスさせた場合のシミュレーションを行ってみた．各システムともヒット率が約 10% 低下した．これは，スワッピング・データ（一般にデータは数十kワードのサイズとなる）のヒット率がそのほかのデータのヒット率に比べ非常に高かったためであるが，ディスク・キャッシュの効果を（アクセス時間＋転送時間）の短縮の割合として見た場合には，高負荷時ではバイパス・モードの方がやや優れていた.（図6参照）

4.6 Write-Through モード

次に，**3.3** で述べた Write Through モードによる結果を示す．たとえばシステム C で，2kバイト長の 1,000 ブロックで構成されるキャッシュの場合のヒット率は Write After モードでは 82% であるのに対し，Write Through モードでは Read だけでのヒット率は 84%，全体でのヒット率は 70% であった．また，LRU スタックのヒットの分布を**図5**に示す．この図より，Write-Through モードに比べ，Write-After モードの方が，LRU スタックの浅い位置でのヒットが多いことがわかる．これは同じ位置に対する WRITE と READの要求が時間的に接近して出される場合が多いためであると考えられる.

4.7 ディスク・スループットの向上度

ディスク・キャッシュの効果を考える場合，ヒット率と共にデータの転送時間も考慮に入れる必要がある.

本論文ではディスク・キャッシュを用いることによ

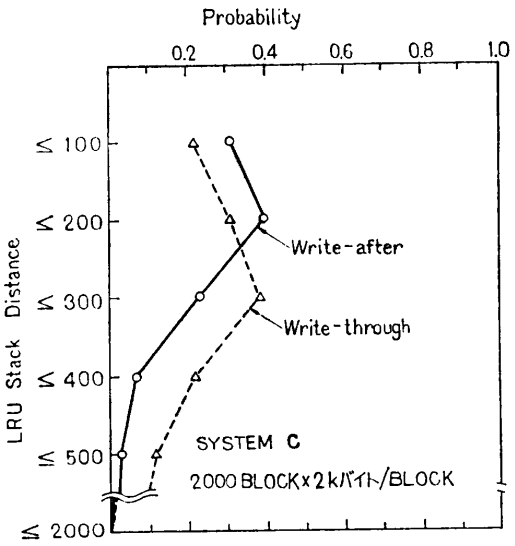

図 5 Write through と Write after
Fig. 5 Write through and Write after.

るディスク・スループットの向上度 Δ を次の式で求める.

$$\Delta = T/T^* \qquad (1)$$

ここで, T と T^* は, それぞれ, ディスク・キャッシュを用いない場合と用いた場合の（アクセス時間＋データ転送時間）の期待値を表わし,

$$T = Sd + Wd + L/Td \qquad (2)$$

$$T^* = H\times(Ac + Lh/Tc + \alpha) + (1-H)\times(Sd + Wd + Lf/Td + \beta) \qquad (3)$$

ここで

Sd：ディスクの平均シーク時間
Wd：ディスクの平均回転待ち時間
Td：ディスクの転送速度
L：転送データの平均長
H：キャッシュのヒット率
Ac：キャッシュの平均アクセス時間
Tc：キャッシュの転送速度
Lh：ヒットしたデータの平均長
Lf：ヒットしなかったデータの平均長
α：ヒットしたときの平均オーバヘッド時間
β：ヒットしないときの平均オーバヘッド時間

一例として

Ac=2 ms, Tc=1,200 kバイト/S
Sd=30 ms, Wd=8 ms
Td=800 kバイト/S

の場合の Δ を**図 6** に示す. ただし, ここでは, α と β が無視できるほど小さく, Lh と Lf は近似的に等しく L としている. そのほかの値はシミュレーションより得られた値を使用している.

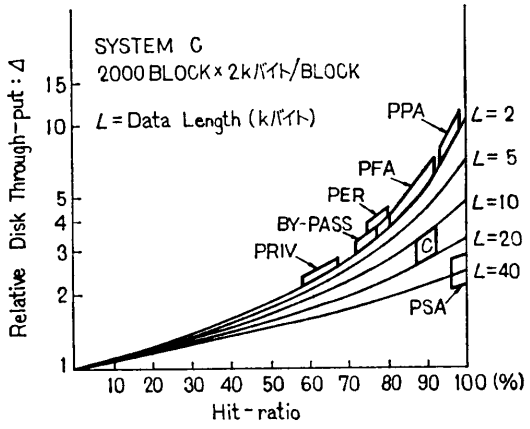

図 6 ディスク・スループットの向上度
Fig. 6 Relative disk through-put.

図 6 において, システム C の全ファイル領域に対する Δ を C で示し, バイパス・モードの場合も合わせて示している. この例では高負荷時には 2.5～3.5 倍のディスク・スループットの向上を得られることがわかる. また, ファイル領域名が記されている部分は各ファイル領域へのアクセスだけがキャッシュの全容量を利用したときの Δ を示している. 各ファイル領域に対する効果はかなり異なっており, 特にページングの領域に対する効果が非常に大きい. ディスク・キャッシュを最適に構成するためには, 各ファイル領域へのアクセス頻度とその効果を十分に考慮することが必要となる.

さらに, データ収集時のシステム稼動状況が知れれば, 以上の結果を用いてシステム全体に対するディスク・キャッシュの効果を知ることができる.

5. ま と め

本論文では階層記憶を構成する上で重要視されているディスク・キャッシュについて, 使用用途の異なる 3 つの実働計算機システムから収集したディスク・アクセス動作のトレース・データを使ってシミュレーションを行い, その効果と構成法について検討した.

シミュレーションの結果, Write-After モードではページ・ヒット率として 95～98%, コマンド・ヒット率として 83～89% のヒット率を達成できること, ヒット率は 4M バイト前後で飽和することなどが明らかになり, この結果を用いて, 測定対象システムでは

総ディスク容量の 0.5%～1% 程度の容量のディスク・キャッシュを利用することによりディスクのスループットを実質的に 2.5～3.5 倍程度向上できる可能性のあることを示した．また，ディスク・アクセス動作はファイル領域ごとに異なるためディスク・キャッシュの効果に差が出てくること，それに対応して，ディスク・データに応じてディスク・キャッシュでの扱いを工夫することにより，同じキャッシュ容量でも実質的な効果を向上できることがわかった．

ディスク・キャッシュの効果に影響するディスク・アクセス動作はファイル管理システムの作り方に大きく依存すると思われ，またディスク・キャッシュによるディスク・スループットの向上が計算機システムのスループット向上に寄与する度合いは，更に多種多様のシステム要素と複雑にからんで影響するものと思われる．これらソフトウェアでのディスク使用方法やシステムの諸要素とディスク・キャッシュの効果をより関連づけて定量的に把握するのが今後の課題である．

最後に，本研究の機会を与えていただいた，当社計算機研究部首藤勝部長，ならびに，熱心にご討論，ご助言していただいた，計算機製作所飯川昭一氏および計算機研究部溝口徹夫研究員に深謝いたします．

参考文献

1) Toombs, D.: CCD and Bubble Memories System Implications, IE³ Spectrum (May 1978).
2) Juliussen, J. E.: Bubbles and CCD Memories-Solid State Mass Storage, AFIPS Conf. Proc., vol. 47 (1978).
3) Coffman, E. G. and Denning, P.J.: Operating Systems Theory, Prentice-Hall, INC. (1973).
4) 菅，上田： ディスク・キャッシュ装置のシミュレーション，情報処理学会研資，計算機アーキテクチャ 35-1 (1979).

(昭和 55 年 6 月 16 日受付)
(昭和 55 年 9 月 18 日採録)