計算機アーキテクチャ 35－1
（1979. 9. 12）

ディスク・キャッシュ装置のシミュレーション

菅　隆志・上田尚純
（三菱電機K.K.計算機研究部）

1．まえがき

計算機システムの高速化をはかるうえで，大きな障害となる要因のひとつがディスク・アクセス動作の機械的な遅さである。　即ち，最近の計算機の1次記憶（主記憶）のアクセス時間（数百ns）と2次記憶（磁気ディスク）のアクセス時間（数十ms）の間には約10^5のアクセス・ギャップがある。　特に，大容量データを高速に処理することが要求されるシステムでは，このギャップをいかにしてカバーするかが大きな課題となっている。

この問題の解決策として，CPU－主記憶間に置かれる高速バッファ（メモリキャッシュ）と同様に，主記憶－ディスク間にも　アクセス速度が速いバッファ（例えばCCD）を設ける方式が考えられている。　これは，アクセス頻度が高い情報をこのバッファ上に置くことにより，結果的に，ディスクに対する見かけ上のアクセス速度を向上させようというものである。　このための装置をディスク・キャッシュ装置と呼んでいる。

記憶の階層化の一般的な例を図1に示す。　ここで，メモリ・キャッシュ　及び　ディスク・キャッシュは，それぞれ，主記憶とディスクに対する高速バッファであり，いずれの場合も　そのアクセス動作に　局在性（Locality）がある事を利用している。　即ち，ある時間幅で見ると　アクセスの大部分が記憶領域のある部分に集中する傾向がある。

しかし，メモリ・キャッシュの効果が　主記憶上のプログラムやデータの局在性に依存するのに対して，ディスク・キャッシュの効果は，様々な構成とアクセス法を持ったファイルへのアクセス動作の特性により決定される。

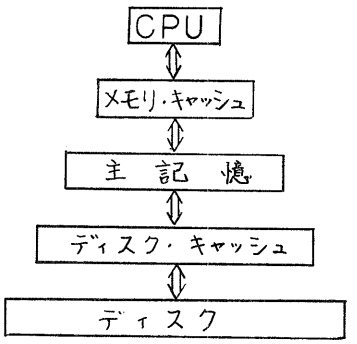


図1　記憶の階層化

本報告では，MELCOM－COSMO　700システムにおけるディスク・アクセス動作の実測データを使って，これらの動的特性を明らかにし，さらに，ディスク・キャッシュ装置のシミュレーションを行い，その効果と構成方法について検討を行なう。

2．基本モデル

2.1　構成

ディスク・キャッシュ装置は，キャッシュ・メモリとキャッシュ制御部から構

成される。（図2 参照）

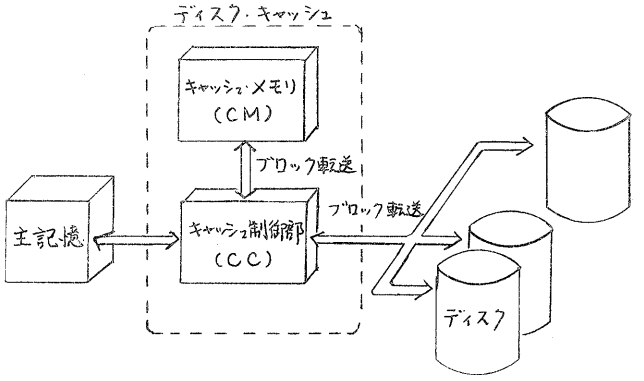


図2 ディスク・キャッシュ・モデル

キャッシュ・メモリ（以後CMと略ぶ）は 固定長のブロック単位に分割されていてこのブロック単位にディスク・データのコピーを保存する。

キャッシュ制御部（以後CCと略ぶ）は、ディスク・キャッシュ装置の動作を制御する。 CCでは、ディスクのブロックをキャッシュ・メモリ上のブロックとの対応関係を記した キャッシュ・テーブルを持ち、これを管理しており、適当なアルゴリズム（LRU，FIFOなど）によってテーブルの内容を更新する。

ディスクに対するすべてのデータは、CC内にある転送バッファを経由する。また、CCは、ブロック／デブロック機能を持ち、CC↔CM，CC↔ディスク間のデータ転送はすべてブロック単位で行われる。

2.2 動作

ディスク・キャッシュ装置の動作を SEEK，READ，WRITE について簡単に述べておく。

SEEK — READ，WRITE命令の前には必ずSEEKコマンドが出され、SEEK情報（ユニット，シリンダ，ヘッダ，セクタの番号）がディスク側へ送られてくるが、ディスク・キャッシュでは、ブロック単位でデータを扱うため、この情報をもとに、要求されたデータを含むブロック番号を計算する。

READ — READコマンドが出されると、CCは、要求されたデータが キャッシュ・テーブル上にエントリされているかどうかを探す。
テーブル内にブロック番号がエントリされている場合（ヒットと呼ぶ）には、そのデータを含んだブロックが CC内のバッファにとり出され、これを主記憶へ送る。 この時、もし更新アルゴリズムとしてLRU（Least Reacently Used）アルゴリズムを使用した場合（今後は これを仮定する。）要求されたデータを含むブロックはLRUスタックの最上位にくる。（図4参照）

WRITE — WRITE動作については、アクセスの高速化やデータの信頼性などにより、いくつかの方法が考えられるが、ここでは Store through モードと Store after モードの2つのモードを設定する。
Store through モードでは、書きこみデータは、直接ディスクへ書きこまれ、もしキャッシュ内に対応するブロックがあれば、これを更新する。 こ

のモードでは WRITE動作に関しては、ディスク・キャッシュによる効果は得られない。

Store after モードでは、書きこみデータは まずキャッシュ・メモリに書きこまれる。 この場合、キャッシュ・メモリ内に対応するブロックがあればその内容が更新され、なければ新しくエントリされる。 これらのデータは、キャッシュの更新アルゴリズムにより、キャッシュ・メモリから追い出される時にディスクへ書きこまれる。（本報告内では、特に断らない限りこのモードを仮定する。）

3. シミュレーション

3.1 概要

シミュレーション全体の概要を図3に示す。 シミュレーションにあたっては、システム全体に対する評価が行えるように考慮してあり、また、大量の実測データを入力して処理するため、高速化、汎用化をはかっている。

シミュレーションは、大きく5つの作業からなり、各々の概要を以下に示す。

(a) データ収集部

実働システムのディスク・アクセス動作の詳細なデータを収集する。 ハードウェア・モニタによる方法とソフトウェア・モニタによる方法の2通りで行った。

(b) 前処理部

上記2通りのデータ収集法では、フォーマットが異なり、これを標準のフォーマットに変換する。 さらに次のシミュレータ部でくり返される処理を前処理しておき、高速化をはかる。

(c) シミュレータ部

ディスク・キャッシュの属性をパラメータとして与えてやることにより、様々なタイプのキャッシュのシミュレーションを行う。 また同時に、ディスク・アクセス動作の分析、統計も行える。

シミュレーションでは、LRUスタックの動作を忠実に行う（図4参照）LRUスタックモードと 高速に実行を行うハッシングモードとがある。

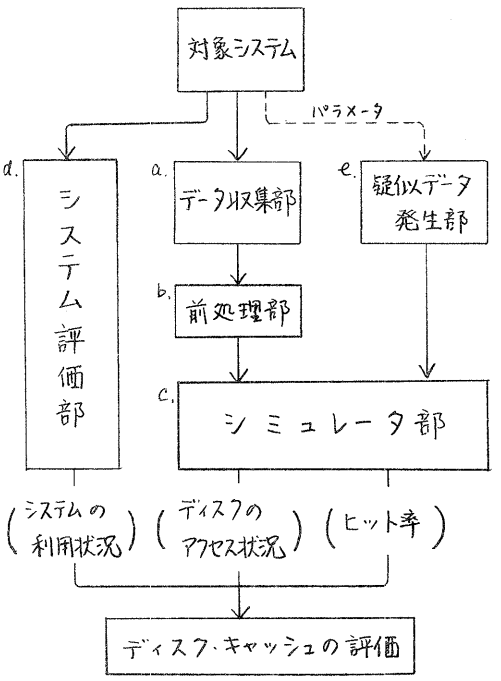
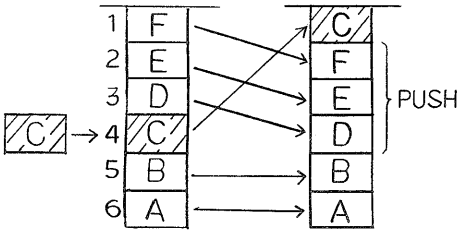


図3 シミュレーションの概要

LRUスタックモードは シミュレーションの初期パラメータを見つける上で重要である。 このモードでシミュレーションを行うと 実行速度はかなり遅くなるが、ディスク・アクセスの動的特性（特に アクセスの局在性）を明確に表現することができる。 これにより、キャッシュの最適構成をある程度

予測でき，従って シミュレーションの試行回数をかなり軽減することができる。

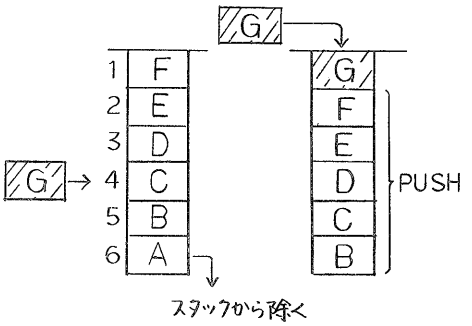
高速ハッシュモードは，上記のモードによって得た結果をもとに，種々のパラメータにより試行をくり返し，その度にヒット率を算出し，最適なキャッシュの構成を見出すためのモードである。 このため，特殊なオープンハッシュと2方向リンクを使い高速な実行が行えるようになっている。



(a) ヒットした場合
(スタックの4番目がヒット)

(d) システム評価部

シミュレーション結果から ディスク・キャッシュがシステム全体に与える効果を定量的に評価するためには，データ収集時のシステムの状況を十分にとらえておくことが重要である。 このために 即存のシステム管理プロセッサにより ユーザの状況とシステムの状況をとらえておく。



(b) フォルトの場合

図4 LRUスタックの動作

(e) 疑似データ発生部

ここでは 2通りの使用方法がある。 ひとつは，シミュレータのテスト用のデータ生成に使い，もうひとつは 任意の特性を持たせた疑似データの発生に使用する。 特に 最悪状況や特殊状況でのシミュレーションに使用できる。

3.2 対象システム

本報告では 利用形態が少しずつ異なる3つのシステムについてデータを収集し，シミュレーションを行っている。 それぞれの主な特殊を表1に示す。

いずれもオンラインとバッチユーザのサービスを行っているMELOM-COSMO 700 システム(モニタはUTS/VS)である。 データの収集は センターのクローズ業務の時間帯を選び どのシステムも 平均して オンラインユーザ 7～12 バッチジョブ 1～3 が常時並列して実行されている。 いずれもディスクの総容量は 680 メガ・バイト，主記憶は 128 KW のシステムである。

	システム A	システム B	システム C
主な特徴	・TSSによるプログラム開発のユーザ主体 ・スワップの回数が比較的多い。	・システムAとほぼ同じであるが，モニタにディマンドページング機能を追加	・データベース(EDMS)による業務が主体

表1 対象システムの主な特徴

3.3 対象システムのファイル構成

UTS/VSでは、ディスク・ファイルは次のように分類されている。

- パブリック・ボリューム　システム常駐ボリューム
  - PSA (Permanent System Area)
    OSとスワッピングのための領域
  - PER (Peripheral Area)
    シンビオント(スプール)のための領域
  - PPA (Permanent Paging Area)
    ページングのための領域
  - PFA (Permanent File Area)
    利用者のファイルが作成される領域

  （PSA・PER・PPA・PFA）いずれもページ単位で領域の割り当てがなされる。(1ページ=2KB)

- プライベート・ボリューム　利用者ファイルの領域

4. シミュレーション結果

4.1 ヒット率の定義

まず、キャッシュのヒット率として、次の2通りを定義しておく。

オーダ・ヒット率 ── 全I/Oオーダに対して、ヒットしたオーダの割合を示す。
ただし、1つのオーダで複数ページ(1ページ=2Kバイト)の転送要求がある場合に、1ページでもキャッシュ内になければフォルトとみなす。

ページ・ヒット率 ── オーダ毎にそのページ長を蓄積して出したヒット率。
即ち、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。

4.2 容量 ── ヒット率

キャッシュの容量の増加に伴うヒット率の向上を図5に示す。ブロックサイズにもよるが、ページ・ヒット率は容量が2Mバイトの時に約95%となりほとんど飽和する。また、オーダ・ヒット率は4Mバイトで約80～90%となり飽和する。

ページ・ヒット率がオーダ・ヒット率に比べ高い理由は、後述するようにPSA領域のスワップのヒ

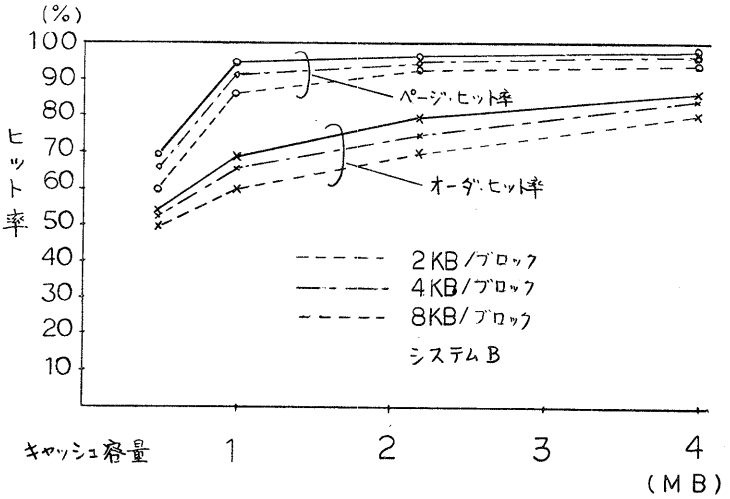


図5　キャッシュ容量 ── ヒット率

ット率が非常に高いことによる。

ヒット率が１Mバイトから４Mバイトの間で飽和してしまう理由は、図６から明らかになる。　このグラフは、ＬＲＵスタック・モードでシミュレーションを行った結果である。　Ｉ／Ｏ要求がヒットした場合、それがＬＲＵスタックのどの深さでヒットしたのかをカウントして、その確率分布と分布関数を示している。

このグラフはシステムＢの例であるが、いずれのシステムを見ても、ヒットしたデータの９５％以上は、スタックの深さが５００（容量としては１MB）までプッシュされる以前に再びアクセスされている。

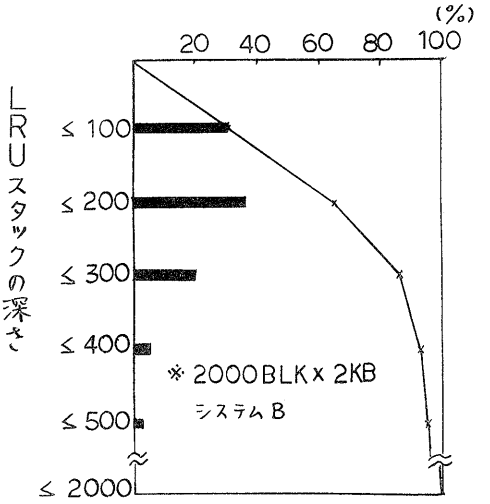


図６　LRUスタックにおけるヒット位置の確率分布と分布関数

４．３　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なされるが、ブロックサイズの決定にあたっては、ディスク上に、ファイルがどのように配置されるかが問題となる。

本報告の対象システムのモニタUTS／VSでは、ファイル管理により、様々なファイルの割り付けがなされるが、その多くはページ単位に分割され、その時点で最適な割り付けを行う。従って、ディスク上では、ファイル領域の多くは、不連続な、ページ単位の配置となっている。

このため、図５に一例を示してあるように、ブロック・サイズを数ページにすることは、（キャッシュの容量が一定の場合）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。

４．４　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Ｃは、データベース主体の業務を行っており、Ａ，Ｂに比べて、ヒット率がやや低い。　これは、Ａ，Ｂのシステムでは、スワッピングやページングの割合が多いのに対し、Ｃでは、データベースで使用するプライベート・ボリュームへのアクセスが多いためである。（表３参照）

システム	A	B	C
ページヒット率	98.5％	98.2％	95.1％
オーダヒット率	89.1％	89.6％	83.5％

※ 2000BLK x 2KB/BLK

表２　各システムのヒット率

４．５　ファイル領域別のヒット率

各ファイル領域別のアクセス特性を知るため、それぞれ個別のファイル領域へのアクセスだけをとり出してシミュレーションを行った。　その結果例を表３と図７に示す。

以下、各ファイル領域別に検討する。

PSA領域

ヒット率は 96～99％となる。これは、この領域内に、非常に頻ぱんにアクセスが行われるOSの領域と、スワッピングのための領域が存在するためである。さらに図7を見ると他の領域が指数分布に近い分布を持つのに対しPSAについては、アーラン分布に近いものとなっている。これは、スワップ領域が様々なデータ長のスワッピングに使われるためと考えられる。

(%)

	システムB	システムC
PSA	99.6	96.6
PFA	78.9	91.1
PPA	95.3	——
PER	84.9	74.7
プライベート	61.4	60.0

※ 2000BLK x 2KB/BLK

表3 ファイル領域別ヒット率

PFA領域

各ユーザの利用方法に大きく影響を受ける。多くのユーザがプログラムの開発を行っているようなシステムBよりは、データベース業務が主体であるシステムCの方がヒット率は高くなっている。

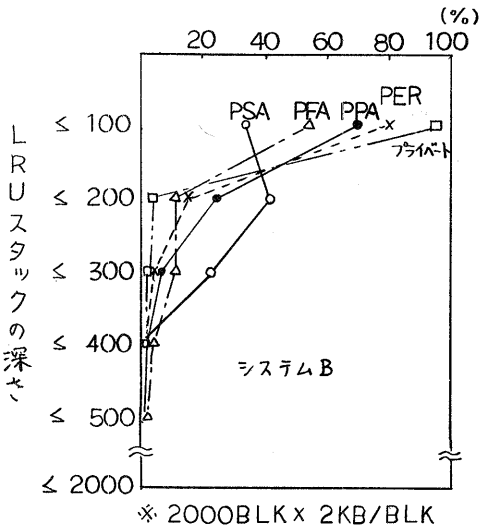


図7 LRUスタックにおけるヒット数の確率分布と分布関数

PPA領域

ヒット率は95％以上であり、しかもすべてページ単位の転送であるため、ディスクキャッシュによる効果が非常に大きい。ページングのためのシステムのオーバーヘッドは、これにより大幅に減少すると思われる。

PER領域

スプール・データは基本的にはFIFOで処理され、また、場合によっては長時間シンビオント領域に大量のデータがとどまることもある。このため、ヒット率はその時の状況によりかなり変化するが、LRUアルゴリズムを使用している限り、ヒット率はそれ程上がらない。

プライベート領域

データベースの処理で使用するデータが大部分を占めており、約60％のヒット率しか得られない。また図7のように、この領域へのアクセスがヒットする時には、その大部分はかなり短い周期でアクセスが行われている。

以上のように、ファイル領域別にそのアクセスの動的特性はかなり異なっていることがわかる。前述したシステムのヒット率は、これらのファイルアクセスが複雑に重なり合ったものと考えることができる。

また、これらのファイル領域別にディスク・キャッシュの制御を変えることにより、いっそう効果的な利用法が考えられる。

4.6 バイパス・モード

ディスク・キャッシュの利点は，アクセス速度がディスクのそれに比べて速い事であり，転送速度には大差がない。 このためスワップなどの長いデータに対してはキャッシュをバイパスさせ，その分だけ他の短いデータをキャッシュにエントリしておく方法が考えられる。

例えば スワッピングの対象となるデータをすべてバイパスさせた場合，シミュレーションでは ヒット率は約10%低下した。 これはスワッピングのデータのヒット率がその他のデータのヒット率より高かったためと思われる。

しかし，データの転送時間も考慮に入れた場合のスループット（アクセス時間＋転送時間）の向上率は，バイパス・モードの方がやや優る。（図9参照）

4.7 Store through モード

本報告ではこれまでStore afterモードを仮定してきたが，ヒットしたオーダを調べてみると，その75～85%がREADであった。 これは，もともとI/O要求全体の70～80%がREADであるためもあるが，この値と比較しても，READの方がヒット率が高いと考えられる。

これらの点をはっきりさせるために2.2で定義したStore throughモードでのシミュレーションを行った。 この場合，WRITE時にヒットしても，必ずディスクに書きこむため，速度的には利点はなく 従ってこのモードでは，READ要求に関してだけのヒット率を考える。

シミュレーションによると システムBにおいて，キャッシュが1000ブロック×2KBの場合，オーダ・ヒット率が84.1%，ページ・ヒット率が97.5%であった。これは，Store afterモードの結果（前者が82.2%，後者が98.4%）と比べると若干良くはなるが，ほとんど変わらない。

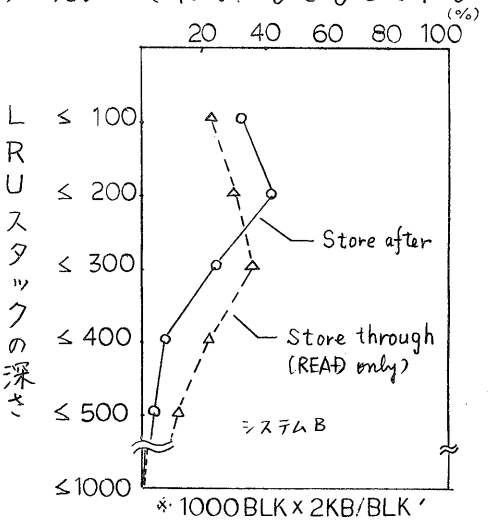


図8 Store throghモードと Store afterモード

しかし，LRUスタックのヒット位置の分布を見ると 図8に示すように，かなり分布が異なり，Store afterの方が より浅い位置でアクセスがヒットしていることがわかる。 これは，同じ場所に対してのWRITEとREADが，続いて起こる場合が多いためである。

4.8 スループット

ディスク・キャッシュ装置の効果を考える場合，ヒット率と共に重要な事は，ディスク・キャッシュ装置をつけることによって，（アクセス時間＋転送時間）がどれ位向上するかと言うことである。

ここで，ディスク・キャッシュ装置を使用した場合の（アクセス時間＋転送時間）の期待値T^*は，次式によってきまる。

$$T^* = H \times (CA + HL/CT + \alpha) + (1-H) \times (DS + DL + FL/DT + \beta) \quad (式1)$$

ここで

H ； キャッシュのヒット率
CA ； キャッシュの平均アクセス時間†
CT ； キャッシュの転送速度†
HL ； ヒットしたデータの平均データ長
α ； ヒットした場合のオーバーヘッド#
DS ； ディスクの平均シーク時間†
DL ； ディスクの平均回転待ち時間†
DT ； ディスクの転送速度†
FL ； フォルトのデータの平均データ長
β ； フォルトの場合のオーバーヘッド#

(注) † これらの値は、キャッシュ又は、ディスクの装置により定まる定数
α, β はキャッシュテーブルのサーチ、更新などによるオーバーヘッドであるが、ハッシングなどの手法を用いることにより、近似的には無視できる。

上式を使って、ディスク・キャッシュを用いることによる、ディスクのI/Oスループットの向上率を知ることができる。

例として、CA=2ms, CT=1200 KB/S のキャッシュと、DS+DL=38ms, DT=806 KB/S の磁気ディスクを使用した場合を図9に示す。ここで、縦軸は、キャッシュを使用しない時の(アクセス時間+転送時間)の期待値Tと上記のT^*との比 T/T^* を示している。この値は、高負荷時にディスク・キャッシュを使用することによる、I/Oスループットの向上の度合を表している。

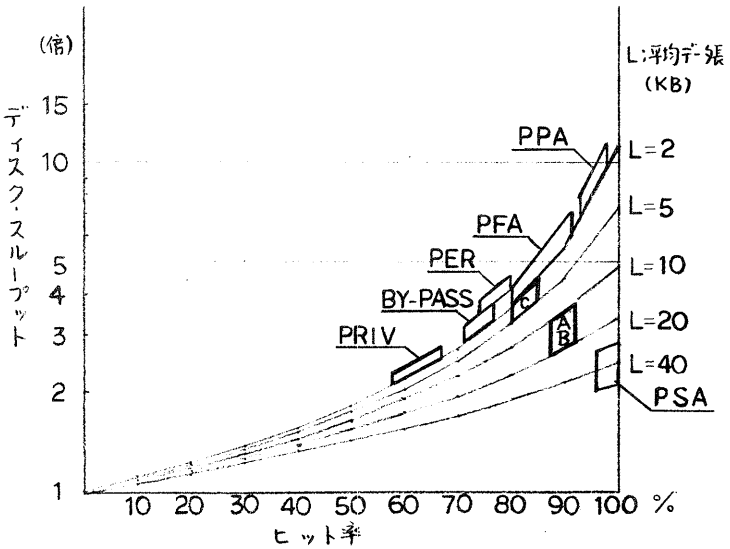
この例では、ブロック・サイズは2KBとし、簡略化のため、HLとFLを近似的に等しいとみなし、これをLとしている。このLとヒット率Hは、シミュレーションの結果を使用する。(2000 ブロック × 2 KB/ブロック 構成)

図9において ファイル領域名が記されている部分は、そのファイル領域へのアクセスだけにキャッシュが使用され、しかもそれが高負荷である場合のディスクのスループットの向上率を示している。

また、対象とした3つのシステムにディスク、キャッシュを使用し、しかもディスクに対する負荷が高い場合の、ディスク・スループットの向上度をA,B,Cで示している。

これによると、システムA,Bで2.5~3倍、システムCでは3.5~4倍となる。

さらに、この結果と、データ収集時のシステムの稼動状況から、システ



※ 2000BLK x 2KB/BLK

図9 ディスク・スループットの向上(高負荷時)

ム全体に対する効果を知ることができる。　図9の値を用いると　システム全体のスループットは、高負荷時には、いずれのシステムも 1.24～1.32 倍になることが期待できるという結果が出た。

5．　ま　と　め

本報告では、階層記憶の構成上重要視されている　ディスク・キャッシュ装置のシミュレーションを行い、その効果と構成法を検討した。

主な特徴は

(i) 利用形態が異なる3つのシステムに対して、ディスク・アクセス動作のデータを収集し、この実測データによりシミュレーションを行った。

(ii) LRUスタックを利用することにより、単にヒット率だけにとどまらず、ディスク・アクセス動作の動的特性を明確にすることができた。

(iii) ファイル領域別にシミュレーションを行ったことにより、各ファイル領域へのアクセスの動的特性の相違を明らかにした。

主な結論としては、

(i) キャッシュ容量が　全ディスク容量の 0.5 %程度で、ヒット率はほとんど飽和し、ページ・ヒット率は 94～97 %、オーダ・ヒット率は 78～83 % であった。

(ii) 各ファイル領域へのアクセス動作は、それぞれ異なった特性を持っており、領域別にキャッシュの効果が異なる。　そこで、これらの特性をふまえたディスク・キャッシュの構成法が重要となる。

(iii) 対象とした3つのシステムにおいて、十分な容量のキャッシュを持たせることにより、高負荷時に、ディスクのI/Oスループットは 2.5～4 倍に、またシステム全体では、1.25～1.3 倍の向上が望める。

なお、本報告では、MELCOM-COSMO 700 システムの3つのシステムのみを対象としたが、ディスク・アクセスの特性は、OSのファイル管理方法、システムの構成、さらに　ユーザの利用状況などによって、かなり異なるものと思われる。　従って、今後さらに、他のシステムに対して、同様のシミュレーションを行うことが必要であると思われる。

（謝　辞）

本研究の機会を与えていただき、熱心な議論をしていただいた、計算機研究部飯川主任研究員と、データ収集などで協力していただいた計算機製作所内の方々に深く感謝の意を表します。

（参考文献）

1. Memorex 3770 ディスク・キャッシュ資料

2. "Operating Systems Theory"　by EG.COFFMAN, P.J.DENNING　1973