論　　文

Database Management System における Buffer 管理方式*

宮　本　　勲**

Abstract

In this paper the performance problem which arises in the DBMS is discussed. The measures of the performance are derived from the characteristics of data references.

The data working set as a locality is measured and the interreference interval distribution is analyzed. Also the problem of a buffer management is discussed.

1. 序論

データベースシステムのパフォーマンスは主にデータレコードを参照する回数と時間，それに I/O の回数と時間で評価できるが，それらはデータベースの論理構造，検索方式，装置性能，物理装置上での格納形態，アプリケーションプログラムの処理手順，I/O バッファ領域とその管理方式などに影響される．

一般的にデータベースシステムにおけるアプリケーションからのデータ処理の所要時間は，次の算定式で表現される．つまり，DML コマンド当りの処理時間 t は平均的に，

$$t=t_1+n(t_2+\mu(s)T)$$

で表わされる．ここで t_1 はアプリケーションプログラムと DBMS とのインタフェース部での処理と，DML コマンド解釈ルーチンでの処理時間であり，t_2 はデータページ・ハンドラでのデータレコードの参照に要する時間で，検索条件との整合判定処理の時間も含む．t_1, t_2 はコマンドのタイプについてほぼ一定であり，残りの n, $\mu(s)$, T が可変である．

データレコードの参照回数 n は，DML コマンド処理のためにデータレコードを参照した回数で，データベースの論理構造，検索方式，検索条件付与法に影響される．missing-page 確率 $\mu(s)$ は s 個のページバッファを保有している時，ページバッファ領域内に存在しないデータレコードを参照する確率で，主にページバッファのサイズと数，そしてその管理方式に影響され，またアプリケーションプログラムの処理手順から生ずるデータ参照特性にも影響される．次に traverse time T は missing-page が発生した時点からページバッファに目的のレコードを含んだデータページをロードして参照可能になるまでの時間で，データベースの物理装置性能と，装置上での格納形態あるいは複数プログラムによる多重 I/O 操作の状態などに影響される．

本論文ではこれらのうちアプリケーションプログラムのデータ参照特性に着目し，参照特性とページバッファの管理方式との関連を主に解析している．

2. データベース・マネジメント・システム

データベース・システムの検索処理パフォーマンスを解析する前にデータベース・システムをまず知る必要がある．ここで CISS (Consolidated Information Storage System)[4] の例をとり，アドレススペース，データのハンドリング，検索モードについて概述する．

2.1 データベースのアドレススペース

物理ストレージ上では論理データレコードはデータページと呼ばれる等サイズのエリアに格納されている．データベースのアドレススペースがデータページに区切られ，それぞれのデータページ内は displacement アドレスで示される．データレコードのオカレンス間の論理関係はこのアドレスを使用して表現され，またリファレンス・ストリングもこのアドレスで得られる．

---

* Buffer Management Strategies for DBMS, by Isao Miyamoto (EDP System Eng. Division, NEC)

** 日本電気（株）コンピュータ方式技術本部

2.2 データ・ハンドリング

アプリケーションプログラムでデータを参照するには，そのデータを含んだデータレコードがメインメモリ上に存在することが必要である．メインメモリの一部をページバッファと呼ばれるエリアに分割し，I/Oバッファとして用いる．データベースに対するデータの読み書きはページ単位で行なわれ，DBMS は，アプリケーションプログラムが要求したデータレコードを含んだデータページの論理アドレスをアドレス変換ルーチンで物理アドレスに変換してリードオペレーションを行ない，データベースからページバッファを通してユーザ・ワーキング・エリアに与える．あるいは逆にユーザ・ワーキング・エリアからページバッファを経てデータベースに書き込む．

ページバッファとしてアロケートされたメモリ領域を管理する手法は，ページングと呼ばれるメカニズムと同様であり，DBMS ではソフトウェアによって達成されている．

2.3 検索モード

データを検索する場合必ずそのもとになるエンティティが必要である．その基準エンティティの指定手法として，直接エンティティを指定するものと，カレンシ・インディケータを利用するものとが考えられる．CISS で許されている検索モードの代表的なものは次の通りである．

- **直接検索モード**：リファレンス・コードを与え，直接に目的のデータにアクセスする．
- **構造検索モード**：データセグメント内を物理的正方向にシリアル・アクセスする．
- **複合検索モード**：論理構造を階層的にたどって内容アクセスする．
  - **論理結合検索モード**：カレンシ・インディケータを使用しない検索法で，AND 的，OR的検索の2種ある．
  - **順序検索モード**：カレンシ・インディケータを使用し，フィリィアル・セット内の次のレコードをアクセスする．

3. Data Locality

アプリケーションプログラムが実行中データを参照するとき，データベースのアドレススペース全体にわたって一様な確率で参照するわけではなく，参照の確率分布にかたよりがある．このかたよりを Data Locality[6] と呼び，これを具体的に表わすために，Working Set と Interreference Interval の概念[1,3,5]がある．

3.1 Virtual Time

Virtual Time はデータページ参照のインターバルを1単位とした時間で，データページのリファレンス・ストリングはこの時間単位で得られ，window size[1,3,5] τ もこの単位で示され，参照時性を表現するときには非常に便利である．

3.2 Working Set

時刻 $t-\tau$ から t の期間に process P の参照しているデータページの集合 $W_P(t,\tau)$ を process P の Working Set と呼ぶとき，$W_P(t,\tau)$ は process P が動くために必要とするデータ容量になる．

$$W_K(\tau)=\frac{1}{K}\sum_{t=1}^{K} W_P(t,\tau)$$

を $t=1$ から $t=K$ までの期間の平均 Working Set Size とする．本文の WS Size の分布カーブは $W_K(\tau)$ を用いている．

3.3 Interreference Interval

process P のリファレンスストリングにおいて同じデータページに対する参照間の virtual time を X とした場合，Interreference Interval X の分布関数は

$$F_X(u)=P_r\{X\leq u\}$$

で表わされ，密度関数は

$$f_X(u)=\frac{d}{du}F_X(u)$$

Interreference Interval の平均は

$$\bar{X}=\int_0^\infty u f_X(u)du=\int_0^\infty (1-F_X(u))du$$

で表わされる．本文では，

$$f(\tau)=1-F_X(\tau)$$

ただし $\quad F_X(\tau)=\int_0^\tau f_X(u)du$

を Interreference Interval のカーブとして扱ってい

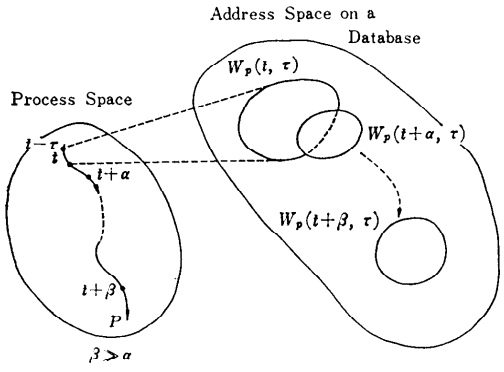


Fig. 1 Time Movement of a Working Set

る.

4. データ参照特性

4.1 参照パターン

ここでアプリケーション・プログラムのデータ参照特性を data locality を基準に分類すると次のようになる.

Hard-type……圧縮不能なページ・セットを参照する.

Soft-type……圧縮性のあるページ・セットを参照する.

Special-type……Hard-type の特殊なケース.

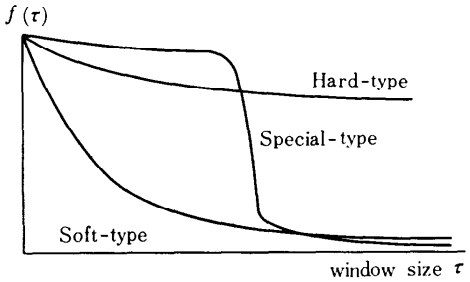
Hard-type というのは, window size τ を変数とした working set size の平均値 $W_K(\tau)$ のカーブの convexity の小さい, locality の低いものである. Soft-type はより convexity の大きな, locality の高い参照特性で, Soft/Hard の判別は Interreference Interval の分布形と, 平均値 $\bar{X}$ の値によって行なう.

Special-type は, Interreference Interval の関数 $f(\tau)$ のカーブで, $\tau=\bar{X}$ の値近くまで Hard-type の特徴を示しながらその点を境に急激に変化し減少するような参照特性である. Fig. 2 に上記3つの type を図示しているがこれらは相対的な判別であることに注意する必要がある.

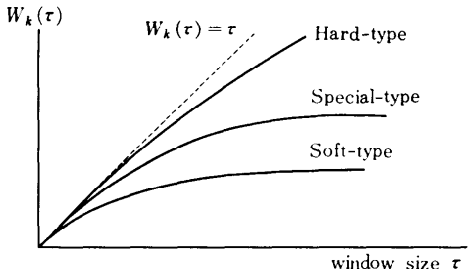
Table 1 Benchmark Programs

CHARACTERISTICS		A	B	C
COMMANDS	Retrieve	6	1	40
	Move	43	8303	40
	Get Ref. Code	0	0	0
	Insert	40	2	0
	Delete	20	40	0
	Change Master	0	0	0
	Replace	0	3124	0
Number of References		1013	25585	440
No. of Sequential References		406	612	360
Subfile Size (CYL.)		5	200	200
Retrieval Modes[1]		(b)(c)(d)	(b)	(a)
Reference Pattern		Hard	Soft	Special
data page size (ch)		1500	2300	500
Traverse Time		AV 16 ms	AV 24 ms	AV 12 ms

(注) 1) Retrieval Mode は次のようである.
(a) 複合検索 mode で currency indicator を使用しない AND 的論理結合検索.
(b) 複合検索 mode で currency indicator を使用する AND 的論理結合検索.
(c) 複合検索 mode で currency indicator を使用する OR 的論理結合検索.
(d) 構造検索 mode で serial search.



(a) Average Working Set Size Curves



(b) Interreference Interval Distribution

Fig. 2 Reference Patterns

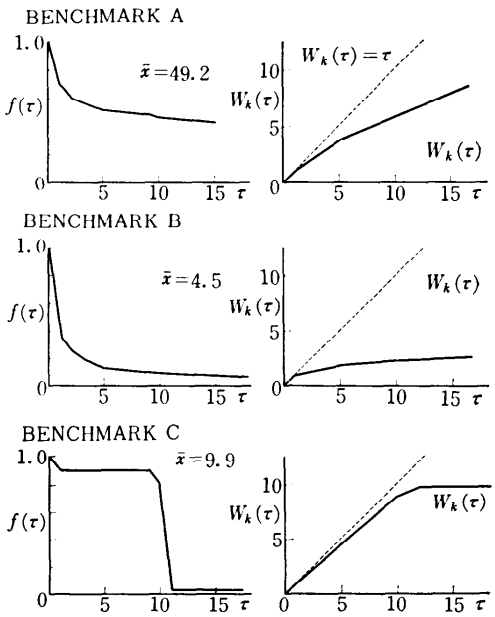


Fig. 3 Reference Chracteristics

4.2 検索モード vs. 参照パターン

Table 1 のベンチマークの参照特性を示すとそれぞれ, Fig. 3 のようである. ベンチマークAは DBMS

機能テスト用のプログラムで複雑に検索モードを使用している．対象トリー構造も非常に複雑である．ベンチマークBは人事ファイルの更新プログラムであり，多量のデータ件数を処理する．検索モードはカレンシ・インディケータを使用したAND検索である．ベンチマークCは検索条件がインタラクションごとに変わり，カレンシ・インディケータを利用しない Search Length Path[7] をたどるような処理内容である．種々の実験[7]から次のようにいえる．すなわち，直接検索モードの場合は参照特性はほとんど Hard-type である．構造検索モードの場合も，Hard-type が多いが，データページ内にレコードがどの位格納されているかによって Soft-type にもなる．複合検索モードでは，Table 1 の Note に示した(a)，(b)，(c)の3つの検索モードが代表的であるが，(a)の場合には参照特性は Special-type である．検索モードが(b)，(c)の場合は参照特性は Hard もしくは Soft-type のどちらかに属し，その判別は $f(\tau)$ の分布形と $\bar{X}$ の値によって行なう．

5. バッファ管理方式とパフォーマンス

5.1 バッファ管理方式

ページバッファの管理方式として LRU, Biased LRU, WS Strategy を選んだ．

LRU (Least Recently Used)[2]

LRU はページ参照に対して容量 C を満たすまで空ページ・バッファを埋めていき一杯になったあと，least recently used のページがリプレースされていくアルゴリズムである．

Biased LRU

通常は LRU のアルゴリズムで置換えが行なわれ，同じデータ・セグメント内の連続したデータ・ページを参照する場合，最も最新に参照されたページを置換えるアルゴリズムで，DBMS 特有の処理特性を反映している．これは CISS 独自のために考えられたものである．

Working Set Strategy[1]

Working Set の定義に従って実行 process が最近の τ_0 期間内に参照したデータページのセット $W(t, \tau_0)$ をメインメモリに持つ．データページが参照されたとき常にそのページに window size τ_0 の値を与え，1 Virtual time ごとにその値を1ずつカウントダウンしてゆき，その値が0になったら，すなわち参照された時点から τ_0 Virtual time 経過したなら，そのページを Working Set からはずすというアルゴリズムである．

Table 2 Replacemeut Algorithms

LRU $\mu(s=4)=\frac{7}{11}$ 平均 page buffer=4.0

time	1	2	3	4	5	6	7	8	9	10	11
Page Trace	a_0	b	b	c	b	a_1	d	c	a_1	a_2	b
LRU Stack ($s=4$)	a_0	b	b	c	b	a_1	d	c	a_1	a_2	b
		a_0	a_0	b	c	b	a_1	d	c	a_1	a_2
				a_0	a_0	c	b	a_1	d	c	a_1
						a_0	c	b	b	d	c
Stack Distance	∞	∞	1	∞	2	∞	∞	4	3	∞	∞

WS Strategy $\lambda(\tau_0=4)=\frac{7}{11}$ AV. Working Set Size=2.91

time	1	2	3	4	5	6	7	8	9	10	11
Page Trace	a_0	b	b	c	b	a_1	d	c	a_1	a_2	b
WS ($\tau_0=4$)	$a_0(4)$	$b(4)$	$b(4)$	$c(4)$	$b(4)$	$a_1(4)$	$d(4)$	$c(4)$	$a_1(4)$	$a_2(4)$	$b(4)$
		$a_0(3)$	$a_0(2)$	$b(3)$	$c(3)$	$b(3)$	$a_1(3)$	$d(3)$	$c(3)$	$a_1(3)$	$a_2(3)$
				$a_0(1)$		$c(2)$	$b(2)$	$a_1(2)$	$d(2)$	$c(2)$	$a_1(2)$
							$c(1)$	$b(1)$		$d(1)$	$c(1)$
missing	∞	∞		∞		∞	∞			∞	∞

Biased LRU $\mu(s=4)=\frac{6}{11}$ 平均 page buffer=4.0

time	1	2	3	4	5	6	7	8	9	10	11
Page Trace	a_0	b	b	c	b	a_1	d	c	a_1	a_2	b
Biased LRU Stack ($s=4$)	a_0	b	b	c	b	a_1	d	c	a_1	a_2	b
		a_0	a_0	b	c	b	a_1	d	c	c	a_2
				a_0	a_0	c	b	a_1	d	d	c
						a_0	c	b	b	b	d
Stack Distance	∞	∞	1	∞	2	∞	∞	4	3	∞	4

Table 2 に3種のアルゴリズムでのオペレーションを示している．この例でページ・トレース中の a_0, a_1, a_2 は同一データセグメント内の連続したデータ・ページである．Stack distance[2] あるいは missing の欄に∞印が入っているのは missing-page を起こしている事を示している．

Biased LRU で time 9 から 10 に移るとき，a_1, a_2 は同一データ・セグメント内の連続したデータページなので，最も最近に参照されたページである a_1 を，time 10 において a_2 で置換している．これによって time 11 の参照で missing-page を起こさないで済んでいる．使用ページ・バッファ数は，LRU と Biased LRU は持っているページ・バッファが未使用であってもそれはそのプロセスにアロケートされているのであるから全期間中，平均4.0個使用していることになる．一方，WS Strategy では Working Set Size しかメモリ領域はアロケートされないので全期間平均2.91個のページバッファを使用したことになり，

同じ missing-page 確率をもつ LRU よりコスト的にははるかに良好である.

LRU, Biased LRU, WS Strategy の相互関係

Biased LRU で同一データセグメント内の連続したデータページを参照する状況がリファレンス・ストリング内で起こる確率は，総参照回数と，シーケンシャル参照数の比 S_r で示される.

任意の時刻 t において window size を τ_0 と定めた場合の working set は $W(t,\tau_0)$ であり，この $W(t,\tau_0)$ に含まれるページ・セットが working set Strategy で保持される. LRU では更に $W(t,\tau_1)=\tau_0$ つまり τ_0 個だけの個有のページをスタック内に含む. Biased LRU でも τ_0 だけの個有のページを含むことができるが平均 S_r の比率分だけページセットからデータページがはずされているので，$W'(t,\tau_2)=\tau_0$ なる範囲のページが保持されることになる.

$W'(t,\tau_2)$ は S_r の比率分でオーバレイされたページ（これらのページは，データベースの検索処理特性により参照確率が非常に低い）を除くページセットであり，

$$W'(t,\tau_2)=W(t,\tau_2)-S_{r_t}\cdot\tau_2$$

で示される時刻 $(t-\tau_2)$ から t の範囲，すなわち Inter-reference Interval $X\leq\tau_2$ のデータページ参照が missing しなくて済む. ただし，S_{r_t} は時刻 $(t-\tau_2)$ から t までの期間でのシーケンシャル参照比率である.

5.2 評価尺度

ページバッファの管理方式のパフォーマンスへの影響度を示すために尺度として missing-page 確率，duty factor, paging rate 更にコストを表わすために Space-time product を採用する.

Missing-page Probability

ある期間内にページバッファ内にないデータページを参照する確率であり，S 個のページバッファを持つときの missing-page 確率を $\mu(s)$ と示す. ただし，WS Strategy の場合には τ を変数とした $\lambda(\tau)$ として得られる. この $\lambda(\tau)$ を $\mu(s)$ に変換することによってアルゴリズム間の効果の比較が可能になる.

Duty Factor

duty factor $\eta(s)$ は process で CPU time を使用できる可能性を表わす尺度であり，

$$\eta(s)=\frac{1}{1+\mu(s)\cdot T}$$

で示され，T は Traverse time, $\mu(s)$ は missing-page 確率である. これにより CP bound の程度が示される.

Paging Rate

missing したデータページをメインメモリへローディングする際のトラフィック密度を表わす尺度を paging rate $\rho(s)$ という.

$$\rho(s)=\frac{\mu(s)}{1+\mu(s)\cdot T}.$$

N 個のマルチジョブである場合には，

$$\rho=\sum_{i=1}^{N}\rho_i(s_i)$$

がデータベースへの全トラフィックを示す. ただし，書き戻し，シェアリング，排他コントロールあるいは，I/O エラー発生などにより変わり得る.

Space-time Product

時刻 t において $h(t)$ 個のページバッファを占有していたとき，実時間インターバル I のメイン・メモリに関する使用コスト $C(I)$ は Space-time product として

$$C(I)=\int_{t-\tau}^{t}h(t)dt$$

で求められる. 単位時間当りのコスト／ユニットタイムを次のように定義する.

$$G=\frac{C(I)}{V(I)}.$$

ここで $V(I)$ はインターバル I をユニットタイムで表わしたものである.

総所要実時間は $t_r=V+V\cdot\mu(s)\cdot T$ となり，この間メモリ容量 s は一定であるから，コストは

$$C(I)=s\cdot t_r=sV\{1+\mu(s)\cdot T\}$$

となり，1回の参照当りのコストは次式で示される.

$$G(S)=\frac{s\cdot t_r}{V}=s\{1+\mu(s)\cdot T\}.$$

WS Strategy の場合には，変換を考慮すると，

$$G(W_K(\tau))=W_K(\tau)\{1+\mu(W_K(\tau))\cdot T\}$$

となる.

5.3 実験

実際に，3つのアルゴリズムの効果を3つの参照パターンにおいて比較する. 比較のための評価尺度として missing-page 確率と，Space-time Product を主に用いる. 実験では，実際のプログラムをランしてデータ参照特性のデータとともにそのリファレンス・ストリングを収集し，3つのバッファ管理手法のシミュレータへ入力し，評価尺度に関するデータを計算し，出力可能にしたもの[7] を用いた.

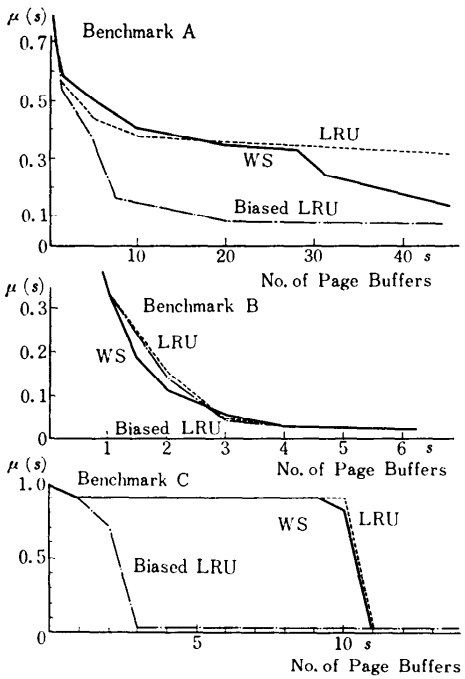


Fig. 4 Paging Performance

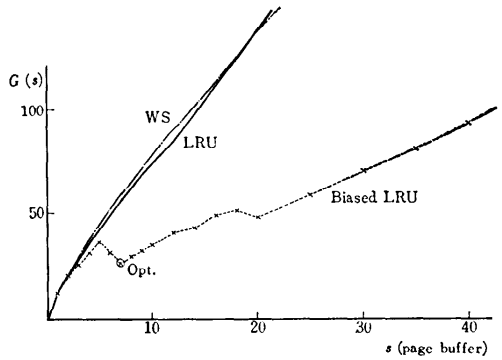


Fig. 5 A. Space-Time Products for Benchmark A.

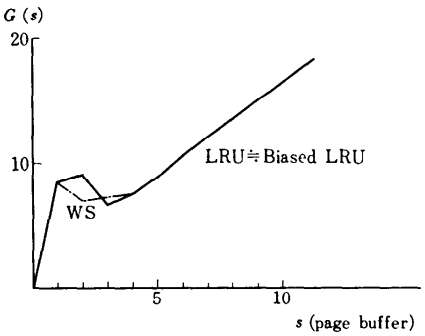


Fig. 5 B. Space-Time Products for Bechmark B.

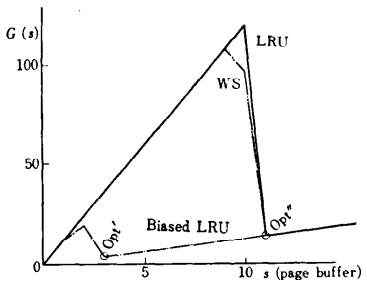


Fig. 5 C. Space-Time Products for Benchmark C

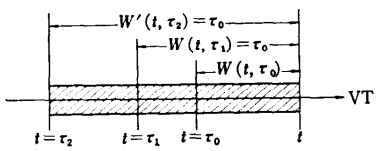
Fig. 4 に Table 1 のベンチマークでの3種のアルゴリズムについて $\mu(s)$ のカーブを描いている．また，Fig. 5 には Space-time product カーブを示している．これらから，ベンチマーク A では WS, LRU の2つと Biased LRU とではかなりの差異があり，$s<40$ の範囲では Biased LRU の場合最適なるページバッファ数 $s=7$ が求まる．B ではアルゴリズム間の差異はほとんどなく $s=3$ が最適値であり，C では Biased LRU の場合，$s=3$，LRU では $s=11$ が最適値であり，WS のときは $W_K(\tau_0)\fallingdotseq 11$ なる window size τ_0 が最適値である．

また Fig. 4 ベンチマーク A, B で，LRU と WS の関係が逆転している部分があるが，これは WS Strategy の場合，$\lambda(\tau)$ として window size τ に対して得られる missing-page 確率を，$W_K(\tau)$ の値に対して $\mu(W_K(\tau))$ を求めるために変換する際の誤差から生じたものである．変換の際，平均値で扱うので，実際とは少し誤差が出る．また Fig. 7, 8 にベンチマーク A の paging rate と duty factor のカーブを描いている．

5.4 ページングパフォーマンスの評価

参照特性とページングパフォーマンスの関係を，実験結果（Ref. 7 も含めて）から整理する．

まず，Fig. 3 と 4 を見較べてみると，LRU, WS



Working set strategy: $W(t,\tau_0)$
Least recently used: $W(t,\tau_1)$
Biased LRU: $W'(t,\tau_2)$

Fig. 6 Relationship among the algorithms

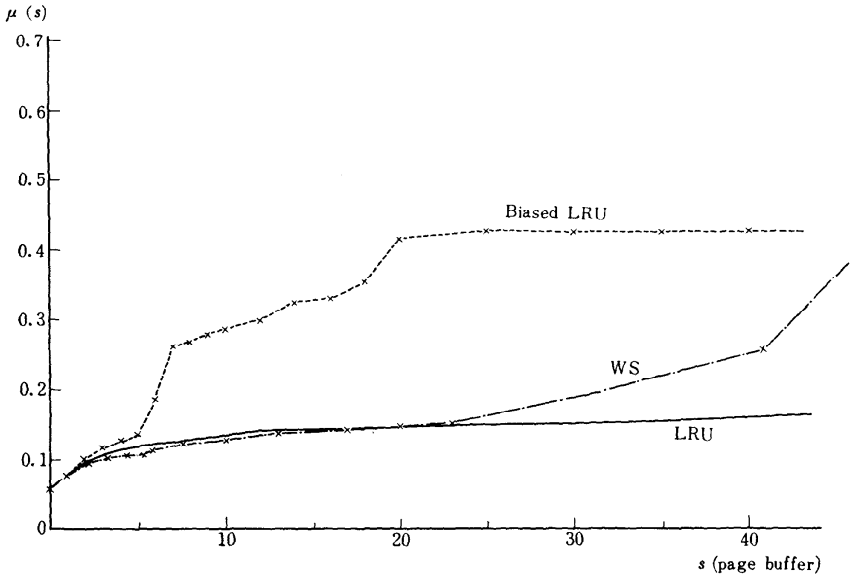


Fig. 7 Duty Factors for Benchmark A

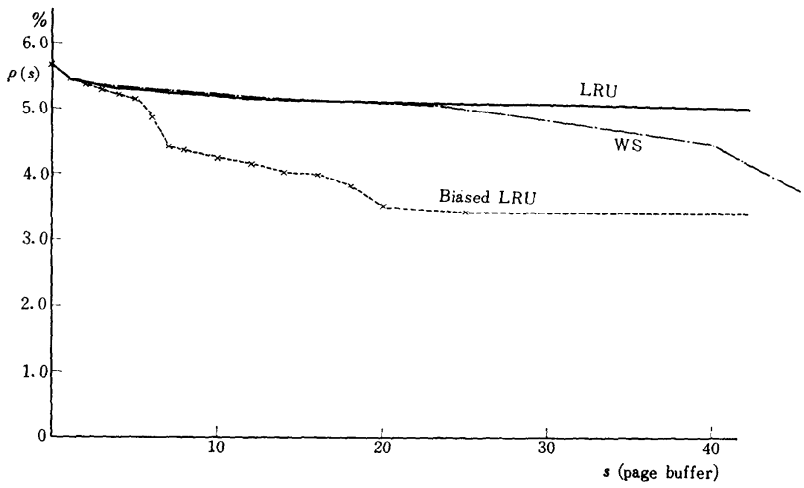


Fig. 8 Paging Rates for Benckmark A

strategy の $\mu(s)$ のカーブと $f(\tau)$ のカーブは非常によく似た形状をしていることが判かる．これは $F_X(\tau)$ が基礎となって missnng page 確率 $\mu(s)$ が定まることによる．このことは参照特性がページングパフォーマンスに与える影響は非常に大きく，そして参照特性がわかれば missing-page 確率が予測され得ることを示唆していて，参照特性が Special-type で LRU, WS Strategyの場合は missing-page 確率が X の値付近で急激に変化する点が存在し，ページバッファ数はこの点を含むだけの数を採ればよい．Soft-type は，LRU, WS Strategy の場合に限らずパフォーマンス的に良好なるパターンでありページバッファ数を増加すると効果は上がるが比較的小さな値で saturateする．Hard-type の場合はページバッファ数を増してもそれほど効果が大きくない．またこの場合にはパフォーマンス的にアクセプタブルでない場合が多くそのためにはデータベースの論理・物理面で改善するかもしくはアプリケーションプログラムの処理手順を変更することによって参照特性を改善する必要がある．

ページングパフォーマンスとしては Biased LRU

が良い結果を示している．LRU と WS Strategy では Locality があるかぎり，平均的かつコスト的に観ると，WS Strategy の方が良い．しかし参照特性が Hard-type に近ずくに従って LRU と WS Strategy はほとんど差がなくなる．LPU と WS Strategy の場合には，Locality だけがページングパフォーマンスに効いてくるが，Biased LRU では Locality とシーケンシャルリファレンス確率が効いてくる．このシーケンシャルリファレンス確率の大きさは Locality とは無関係に定まるものである．したがって Soft-type でなくとも Hard-type, Special-type いずれでも S_r の値の高いものが Biased LRU では良い結果を示すことになる．WS Strategy と Biased LRU とは同列にはできないが，データベースの処理特性を反映している分だけ Biased LRU が良いといえる．ただ注意を要するのは WS Strategy は通常のスタックアルゴリズムと違ってダイナミックアロケーションを前提としていることである．

全体的にいって，バッファ管理方式としては **5.1** のアルゴリズム間の関係のところで示したとおり，実験でも Biased LRU が良い結果を示した．参照特性は Soft-type が好ましく，しかもシーケンシャルリファレンス確率の高いものが望ましいといえる．

またアルゴリズム間の比較評価の際には，評価尺度として $\mu(s)$ だけでなく Space-time Product も効果的であることが実証された．$\mu(s)$ だけでは比較できない場合でも，スタック・アルゴリズム，ダイナミックアロケーション方式の場合でも，1回の参照当りのコストとして Space-time Product を採れば，プログラムの大小に関係なく公平に比較評価できる．Space-time Product のカーブでは $s=0$ 以外で極小値を示す点が最適であるといえる．本文での実験では検索処理が stationary でないため，極値が複数個存在するものもある．

paging rate, duty factor は評価尺度というより，プログラムの I/O あるいは CPU の使用の特性を表わすものとして便利である．

6. 結論と今後のアプローチ

われわれの考えたバッファマネジメント手法である Biased LRU は比較評価の結果3つのアルゴリズムの範囲ではインプリメントの平易さ，効果の高さで最も良いと結論づけられる．また，アルゴリズムの評価に用いた missing-page 確率，duty factor, paging rate, Space-time Product などは尺度として非常に有効であった．

さらにアプリケーションプログラムのデータ参照特性を表わすには Working Set あるいは Interreference Interval で表わされる Data Locality に着目するのが効果的である．つまり参照特性がパフォーマンス特に missing-page 確率 $\mu(s)$ に与える影響は非常に大きく高いページングパフォーマンスを達成するにはデータ参照特性を考慮する必要がある．それには次の事柄を検討せねばならない．まずアプリケーションプログラムでのデータ検索モードと参照レコード数の関係であり，これにはデータベースの論理構造あるいは検索における条件付与法がかかわってくる．次に参照レコード数とデータページの参照数との関係であるが，この時にはデータレコードの物理データページへのパッキング問題を取扱わねばならない．さらにデータページの参照数と，参照特性との関係が分析される必要がある．またアプリケーションプログラムにおけるデータ参照のパラレリズムの考慮によって data locality を向上させることも考えねばならない．データベースの場合は，missing-page 確率も重要なポイントであるが，何よりもデータレコード参照数を少なくすることが第1であることが算定式からもわかり，残された問題については別の機会に議論したい．

7. 謝 辞

諸論文と情報を送っていただいた Purdue 大学の P. J. Denning 教授に感謝の意を表したい．

参考文献

1) P. J. Denning, The working set model for program behavior, CACM, May 1968.
2) J. Gecsei, D. R. Slutz, I. L. Traiger, Evaluation techniques for storage hierarchies, IBM System Journal, No. 2, 1970.
3) P. J. Denning, Resource allocation in multiprocess computer systems, MAC-TR-50, MIT, May 1968.
4) CISS Programming and Operation Manual, EOI-39937, NEC.
5) Jeffrey R. Spirn, P. J. Denning, Experiments with program locality, Proc. of FJCC, 1972.
6) 宮本勲, Database System における Data 参照特性と Performance の解析, 13回大会予稿集 1972 年 12 月.
7) 宮本勲, データベースマネジメントシステムのパフォーマンス解析, Tech. Rep. コンピュータ技術本部, 1972 年 2 月.

(昭和48年 9 月14日受 付)
(昭和48年10月29日再受付)