ショート・ノート

可変閾値型入れ換えアルゴリズムとその性能シミュレーション*

飯塚　肇**　坂村　健***

Abstract

In this paper the new replacement algorithm, called variable threshold recently used (VTRU) algorithm, and its performance simulation are described. The algorithm has one parameter which indicates how far past of access sequence is taken account for the selection of replacement block, and has characteristics of easy implementation and good performance for many cases. Some of the simulation results for microcache are given.

1. はしがき

キャッシュメモリ[1]や仮想記憶システムにおいて，高速記憶からの追出しブロックを決定するアルゴリズムは重要な設計パラメータの一つである．

われわれは現在開発中のモジュール型複合計算機システムACEのマイクロプログラム記憶にキャッシュ方式を採用することにしたが，その入れ換えアルゴリズムとして，マイクロプログラムの特性を考慮し，通常用いられている LRU**** アルゴリズムを変形した可変閾値型 (Variable Threshold Recently Used: VTRU) アルゴリズムを考案して用いることとした．本稿では VTRU アルゴリズムの特徴と 2, 3 のシミュレーション結果について報告する．

2. VTRU アルゴリズム

2.1 概　要

各種の入れ換えアルゴリズムの内，LRU が平均的に高性能をあげることが，これまでに何回か報告されているが，このアルゴリズムはハードウェア的実現がやや複雑になる欠点がある．

今，入れ換えアルゴリズムを実現という観点からとらえてみると，それには次の2つの部分があることがわかる．

① 各アクセスごとに追出しに必要な情報を更新する．

② 追出すブロックを選択する．

対象がマイクロプログラムの場合はキャッシュ中で書き換えは発生しないから，スワップアウトは不必要で，②は最初のスワップイン・データが入ってくるまでに行なえばよく，若干の時間的余裕があるが，アクセスはマシンサクイルごとに必ず発生するから①は極めて短時間に行なわねばならない．他の場合でも②より①に高速性が求められることが多いだろう．

さて，LRU にはいくつかの実現法があるが，今，古典的なアクセスされた順序にブロックに番号をつけておく方法を考える．例えば，**Table. 1** に示したようにブロック数が8の場合，カラム(2)のような状況にあった時に，ブロック1がアクセスされたとすると，LRU ではカラム(3)のように更新しなければならない．この更新のやっかいなのはアクセスされたブロックのカウント値より小さい値を持つブロックのカウントのみ＋1しなければならず，カウント内容が更新法に影響

Table. 1 An exmaple of count value

(1)	(2)	(3)	(4)	(5)
ブロック番号	最初のカウント	LRU	VTRU(7)	VTRU(n) $n \geq 8$
(1)	5	0	0	0
(2)	3	4	4	4
(3)	4	5	5	5
(4)	7	7	7	8
(5)	1	2	2	2
(6)	0	1	1	1
(7)	6	6	7	7
(8)	2	3	3	3

* "Variable threshold recently used" replacement algorithm and its performance simulation by Hajime IIZUKA (Computer Division, Electrotedmical Laboratory) Ken SAKAMURA (Graduate school of Keio University)

** 電子技術総合研究所　電子計算機部

*** 慶応義塾大学大学院工学研究科

**** Least Recently Used: 最後に使用されてから，最も時間が経過したブロックを追出す．

を与える点にある．

そこで，カウントアップをカウント値によらず行なうことにし，そのかわりカウンタの上限を‘ブロック数−1’に固定せず，任意の値nまで取り得るように変形する．これをカウント上限値がnの可変閾値型アルゴリズム (VTRU(n)) と呼ぶことにするが，先の例ではVTRU(7)の場合はカラム4のように，$n\geq 8$ならカラム(5)のようになる．また，VTRU(n)をALGOL風に記述すれば次のようになる．

```
tineger          m,n, pointer:=0;
integer array    count [0: (m−1)]:=n;
procedure        update (blk);
  begin if   conut(blk)≠0
          then begin for i:=0 step 1 until (m−1) do
                 if i≠blk ∧ count(i)≠n
                   then count(i) :=count(i)+1;
               count(blk):=0;
               end
  end
procedure        access (hit, ablk, tblk);
  boolean        hit;
  begin if       hit=true
        then     update (ablk);
        else     find a block of maximum count, searching
                 in the increasing order (mod m) from the
                 block pointed by 'pointer'; set its block
                 number into 'tblk' and 'pointer' to its number
                 plus one (mod m).
                 update (tblk);
end
```

mはブロック数，ablkはアクセスされたブロック番号，tblkは入れ換えブロック番号，最大カウントのブロックが複数個の時はpointerを用いたFIFO方式である．

2.2 VTRUの性質

VTRUは要するにアクセスの順番づけをnまでに制限し，それ以上は順番をつけず，同一に扱うことにしたアルゴリズムであって，次のような性質がある．

① $n=0$ はランダムまたはFIFOアルゴリズムに相当

② $n=\infty$ はLRUアルゴリズム

③ $n\leq$ (ブロック数−1) の時はカウント値$=n$のブロックが常に少くも1個存在する．

④ $n\geq 2$ではカウント値が0または1のものは1ブロックずつ存在する．

証明は容易であるから省略するが，nを変化させることによって，任意量の過去のアクセス状況を利用できること，それに更新作業が他のブロックのカウントに影響されず並列に行なえ*，高速で実現が容易なのがこのアルゴリズムの特徴である．なお，最大値を持つカウンタが複数個ある場合にその中の一つを追出しブロックに選ぶ方法によって細部の状況は変ってくる．ランダムでも，ポインタを用意し，サイクリックにサーチしてもよいだろう．$n=0$ の場合，前者はランダムアルゴリズムに，後者はFIFOになる．

また，LRUにするために必要なnの値は通常はブロック数に比し，あまり大きくする必要はなく，ブロック数の数倍，すなわち，カウンタのビット数を$\log_2$〔ブロック数〕に1〜3ビットを加えたものでよい．この方法によるLRU実現の問題点はnがブロック数を越えると各時点での最大値が一定でなくなるために，それを発見するのに時間と費用がかかる点にある．

3. シミュレーション

VTRUを実際にに適用するために，nを変化させてミス率**の変化をシミュレーションによって調べた．

3.1 シミュレーションの条件と方法

シミュレーションはマイクロプログラム計算機HP-2100Aのマイクロプログラム記憶にキャッシュを備えた場合を想定し，各種のプログラムに対しトレースされたデータを用いて，nを変化させながら($n=0$〜15は1きざみ，それ以上は2倍ずつ，LRUまで) ミス率を測定した．キャッシュの各種パラメータは **Table. 2** に示す通りであるが，これはこのアルゴリズムを実際に適用しようとしているACEのマイクロキャッシュのパラメータとの関連で選択した．また，トレーサは現在研究中のマイクロキャッシュ・シミュレーション・システム[3]の一部を利用した．ACEシステムのマイクロキャッシュの詳細やこのシミュレーションシステムについては別の機会に報告するとして，今回はVTRUのシミュレーションに話を限ることにする．

Table. 2 Parameters of microcache

パラメータ種類	パラメータ値
原籍容量	256語(HP-2100A マイクロプログラム)
キャッシュ容量	64 語
マッピング方式	セットアソシアティブ
セット数	2
ブロック数／セット	8
語数／ブロック	4
入れ換えアルゴリズム	VTRU(n) (ただし最大カウント選択法はFIFO)

* ビッドマトリックスを用いる新しいLRU論理でも，ブロックの状況に影響しない更新ができる．ブロック数＝8の時ビットマトリックスは28ビット．VTRU(7) では3ビットカウンタ8個でほぼ同規模であるが，ブロック数がこれより増すとVTRU (ブロック数−1) の方がハード量がずっと少なくなり，また，一般性もあると考えられる．

** 必要なデータがキャッシュ中に存在しない割合

シミュレーションに取りあげたプログラムは，①アセンブラのパス1(主に記号表の作成)，②ローダ，③FORTRAN で記述した数値積分の実行，④同じく行列計算，⑤同じく素数計算 (30000 から 30500 の間) 等システムプログラムと技術計算プログラムである．

3.2 シミュレーションの結果

Fig. 1 にシミュレーション結果の代表例を示した．縦軸はミス率，横軸はカウント上限値(n)，T.M.I. はシミュレートされたマイクロ命令の総数を表わしている．右にいくほど LRU に近づき，一度 LRU に達する (点線は LRU) とそれ以上変化しない．各プログラムごとの曲線は程度の差こそあれ，右さがりになることを予想していたが，結果の中にはそのようにならず，興味深いカーブを示すものもある．すなわち，多種の命令が比較的均等に発生する FORTRAN のプログラムでは，nを大にすることでだいたいミス率も低下するが，半数以上が記憶参照命令であるローダ等では LRU に近づくほど，逆にミス率が増大する奇妙な現象が発生している．これはロードされるプログラムを変えてもやはり似たようなことになる．

以上の事実は限られたプログラム，限られた条件のもとでの結果ではあるが，LRU が必ずしも最適ではなく，適当なランダム性が入った方がかえって性能がよくなるケースがかなりあることを如実に示すものであろう．つまり，費用をかけて，LRU にすることはあまり意味がないといわねばならない．

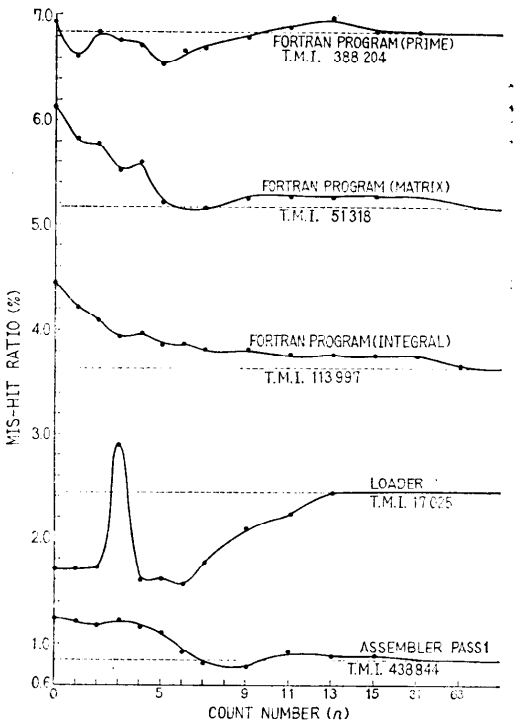

Fig. 1 Some Simulation results

4. むすび

VTRU は過去のアクセス状況の記憶量を1つのパラメータで表示でき，実現が非常にやさしいという点でこれまでないアルゴリズムであると思われる．わずかのシミュレーションではあるが，順序づけを行なうことが必ずしも性能的に有利とは限らないこともわかった．したがって，動的にnを変化させることも考えられるがそれは少しやりすぎというものであろう．

われわれの場合は，8ブロックなので，VTRU(7)を採用することにしたが，今度の結果を見る限り，コスト・パフォーマンスからして，これは正しい選択と考えられる．

なお，nがブロック数と等しく，カウントアップ法が VTRU とは若干異なるアルゴリズム (MLRU) によって，仮想記憶の場合に LRU に近い性能を得られることが報告されているが[5]，VTRU の方がより一般性があると思われる．

このシミュレーションに用いた HP—2100 のトレースプログラムは慶大大学院山田光一氏の製作によるものである．ここに記して謝意を表する．また，本研究の機会を与えられた電子技術総合研究所，西野，黒川，石井各部長，御討論いただいたマイクロプロセッサ研究グループの諸氏，ならびに日頃御指導いただく慶大工学部相磯秀夫教授に感謝します．

参考文献

1) 飯塚 肇：キャッシュメモリ・システム(1, 2)，情報処理，Vol. 13, No. 7 & 8, pp. 467～473 & pp. 540～547 (1972)．
2) 吉広，黒沢：バーチャルメモリについて(1, 2)，情報処理，Vol. 14, No. 9 & 10, pp. 701～706 & pp. 778～785 (1973)．
3) 相磯，高橋，山田，飯塚：マイクロ・キャッシュに関する一考察，ダイナミック・マイクロプログラミング シンポジウム報告集，pp. 160～169 (1973)．
4) 飯塚 肇：コンピュータ・モジュール，電気学会全国大会予稿，S 12—2(1974)．
5) 由良，桑原，山下：試作ミニコン仮想記憶システムシミュレーション 電子通信学会，計算機専門委，EC 73—74(1974)．

(昭和49年5月31日受付)