セットアソシアティブメモリにおける最適ハッシュ関数の選択法

4D−8

郡　光則　David L. Heine　下間　芳樹
三菱電機　情報電子研究所

1．はじめに

セットアソシアティブメモリにおいては、アドレスとセット（カラム）の対応づけ（ハッシュ関数）の最適化によりヒット率を向上させることができる。しかし、実現可能なハッシュ関数の中から最適なものを効率的に選択する方法は、従来一般に知られていなかった。

本稿では、セットアソシアティブメモリにおける最適ハッシュ関数の効率的な選択法を提案する。この方法をTLBに適用し、その有効性を確認した。

2．最適ハッシュ関数の選択

セットアソシアティブメモリのヒット率の評価法としてTrace-driven simulation（TDS）が知られている。しかし、多数のハッシュ関数の候補に対してTDSを行なうのは、膨大な計算量を要するため現実的ではない。

セットアソシアティブメモリでは、一連の連続アドレスが同一のエントリに対応することが多い。本稿ではこの同一エントリに対応する単位をブロックと呼ぶ。本方式では、ブロック間の結合の強さを示す相関値に基づいてハッシュ関数の評価値を求める。

本方式は2パスより成る（図1）。Pass1では、アドレストレースを1回走査して相関値を求める。Pass2ではこの相関値から各ハッシュ関数の候補の評価値を計算し、評価値の最小となるものを選択する。

アドレストレース長をN、ハッシュ関数の候補の数をMとするとき、全てのハッシュ関数に対してTDSを行なった場合の計算量はO(M・N)である。一方、本方式ではO(N)+O(M)となり計算量が大幅に削減される。

3．ハッシュ関数の評価アルゴリズム

アドレストレース中のブロック参照列を$b_1,b_2,...,b_N$とする。Pass1では、ブロックB_i,B_j間の相関値$R[B_i,B_j]$をアルゴリズム(1)によって求める。アルゴリズム(1)で用いるコスト関数の一例を図2に示す。Pass2では、ブロック間の相関値$R[B_i,B_j]$から、ハッシュ関数hashの評価値P_{hash}をアルゴリズム(2)に従って求める。

本アルゴリズムによって求められる評価値P_{hash}は、以

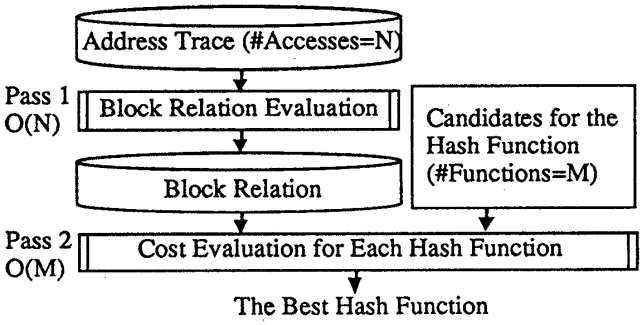

図1．最適ハッシュ関数の選択方式

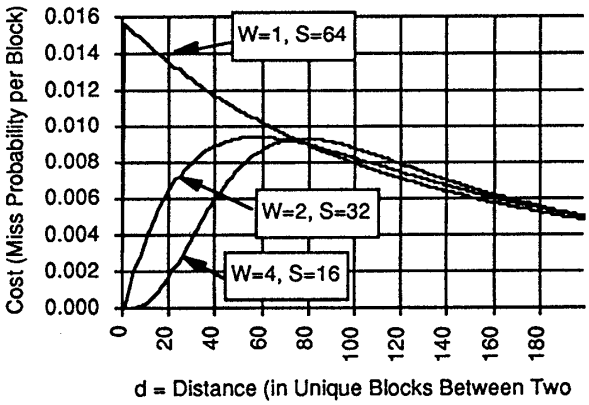

図2．コスト関数 cost(d,W,S)の一例

アルゴリズム(1) Block Relation Evaluation (Pass 1)

```
stack←empty; i←0
for n=1 to N do                          /* for each block access */
  find d such that stack[d]=bn
  if d not found then                                /* first access */
    i←i+1; Bi←bn
    push bn on the top of the stack        /* make it MRU */
  else
    if d>W then
      for t=0 to d-1 do        /* for each block in between */
        R[bn,stack[t]]←R[bn,stack[t]]+cost(d,W,S)
      end-for
    end-if
    remove stack[d]          /* remove the previous access */
    push bn on the top of the stack        /* make it MRU */
  end-if
end-for
L←i
```

Where W: Number of ways (rows)
S: Number of sets(columns)
N: Number of accesses in the address trace
$b_1,b_2,..,b_N$: Block references in the address trace
stack[t]: Stack of blocks, stack[0] is the top.
L: Number of accessed blocks
$B_1,B_2,..B_L$: Accessed blocks
$R[B_i,B_j]$: Relation between block B_i and B_j

$$cost(d,W,S)=\frac{1}{d}\left(1-\sum_{i=0}^{W-1}\binom{d}{i}\frac{(S-1)^{d-i}}{S^d}\right)$$

An Optimization Method for Hash Functions in Set-associative Memories
Mitsunori Kori, David L. Heine and Yoshiki Shimotsuma
Computer and Information Systems Laboratory,
Mitsubishi Electric Corporation

下の仮定をおいた場合のミス率の期待値に等しい。

・あるブロックが参照されてから次に参照されるまでの間、各セットは一様に参照される。

・置換アルゴリズムはLRUである。

4．TLBハッシュ関数の評価

本方式に基づいてTLBの最適ハッシュ関数の選択を行ない、その有効性を確認した。

表1に対象としたTLB構成及びハッシュ関数の候補を示す。ページサイズは4KB、置換アルゴリズムはLRUとした。アドレストレースとして表2の３種を用いた。

各ハッシュ関数の評価値とTDSによるTLBミス率の比較を図3および図4に示す。評価値とミス率はきわめて高い相関を示しており、本方式の評価値の妥当性が示された。本方式により選択したハッシュ関数によるミス率の削減効果を図5及び図6に示す。

また、本方式に要した計算機時間は全ハッシュ関数に対してTDSを行なった場合の1/5000程度であり、大幅な計算量の削減が確認された。

このように、現実的な時間内で最適なハッシュ関数を選択することができ、本方式の有効性が示された。

5．今後の課題

今後は、命令やデータのアドレス割付けの最適化にも本方式を適用し、その効果を評価する予定である。

アルゴリズム(2) Cost Evaluation (Pass2)

```
for k=0 to S-1 do                          /* for each set k */
  Set[k]←empty
end-for
r←0
for i=1 to L do                  /* for each accessed block Bi */
  add Bi to Set[hash(Bi)]
end-for
for k=0 to S-1 do                          /* for each set k */
  for each block pair Bi and Bj  where i≠j in Set[k] do
    r←r+R[Bi,Bj]
  end-for
end-for
Phash←(r・S+L)/N                  /* The evaluation value */
```

Where S: Number of sets(columns)
hash(): Hash function
L: Number of accessed blocks
B1,B2,..BL: Accessed blocks in the address trace
R[Bi,Bj]: Relation between block Bi and Bj
Set[k]: List of blocks for set k
Phash: Evaluation value for the hash function

表1．TLBハッシュ関数の評価で対象とした TLB構成

番号	種別	ウェイ数 (W)	セット数 (S)	ハッシュ関数 (hash)	関数の総数 (M)
I 1	命令	1	64	hash6	2016
I 2	命令	2	32	hash5	1512
I 3	命令	4	16	hash4	504
D 1	データ	1	64	hash6	2016
D 2	データ	2	32	hash5	1512
D 3	データ	4	16	hash4	504

ただし、VAは仮想アドレス。アドレスは32ビット。ビット0がMSB。
[x:y]はビットxからビットyまでのy-x+1ビットを示す。
hash4[0:3]=VA[16:19] xor VA[b0,b1,...b3], b0...b3をVA[6:15]から選択。
hash5[0:4]=VA[15:19] xor VA[b0,b1,...b4], b0...b4をVA[6:14]から選択。
hash6[0:5]=VA[14:19] xor VA[b0,b1,...b5], b0...b5をVA[6:13]から選択。

表2．TLBハッシュ関数の評価に使用したアドレストレース

アドレストレース	処理内容
RDB	あるリレーショナルデータベース処理
OLTP	あるオンライントランザクション処理
BATCH	あるバッチ処理

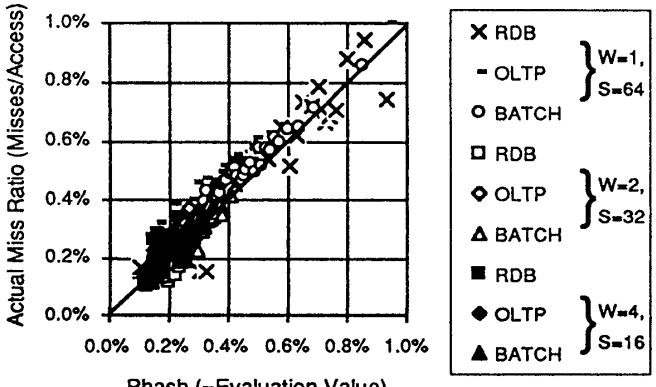

図3．命令TLBにおける評価値とミス率の関係

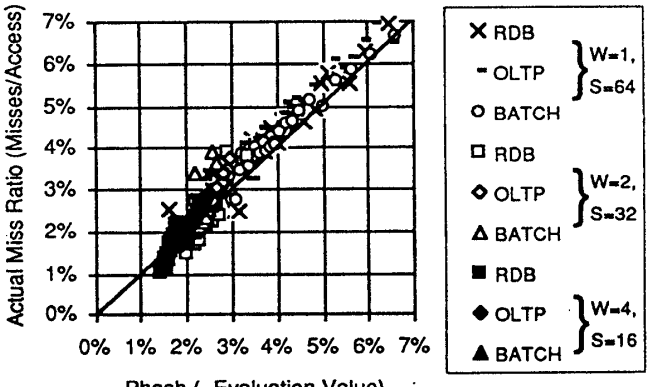

図4．データTLBにおける評価値とミス率の関係

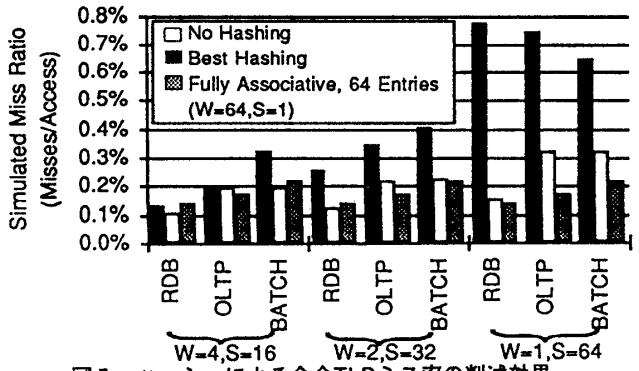

図5．ハッシュによる命令TLBミス率の削減効果

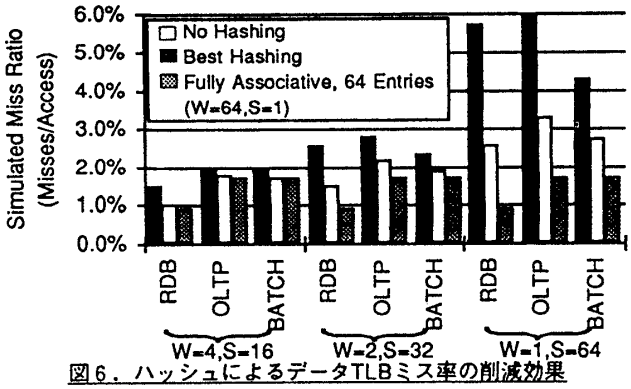

図6．ハッシュによるデータTLBミス率の削減効果