

A decorative graphic in the top-left corner of the slide. It features a black crosshair (a vertical and a horizontal line) centered on a small blue square. To the right of the blue square is a yellow square, and below the blue square is a red square. The background of the slide is a light yellow color.

1

A Typical Memory Hierarchy

- By taking advantage of **the principle of locality** (局所性)
 - Present **much memory in the cheapest technology**
 - at **the speed of fastest technology**

The diagram illustrates a memory hierarchy with five levels, each represented by a box. A dashed line slopes upwards from left to right, indicating that as memory capacity increases, the speed of access also increases, while the cost per byte decreases. The levels are:

- On-Chip Components:** A large box containing:
 - Control:** A horizontal bar at the top.
 - Datapath:** A vertical rectangle on the left.
 - Register:** A small vertical rectangle next to the Datapath.
 - Cache:** A vertical rectangle to the right of the Register.
 - Cache:** A vertical rectangle to the right of the first Cache.
 - Cache:** A vertical rectangle to the right of the second Cache.
 - Cache:** A vertical rectangle to the right of the third Cache.
 - Cache:** A vertical rectangle to the right of the fourth Cache.
- Second-Level Cache (SRAM):** A box with a dotted pattern.
- Main Memory (DRAM):** A box with a light blue pattern.
- Secondary Memory (Disk):** A large, empty box on the far right.

Speed (%cycles):	1's	10's	100's	1,000's	
Size (bytes):	100's	K's	10K's	M's	G's to T's
Cost:	highest				lowest

2

- 2

MIPS Direct Mapped Cache Example

- One word/block, cache size = 1K words

The diagram illustrates the MIPS Direct Mapped Cache structure and operation:

- Address:** 32 bits, split into a 20-bit **Tag** and a 12-bit **Index**.
- Cache Structure:** A table with 1024 entries (Index 0 to 1023). Each entry contains a 20-bit **Tag** and a 32-bit **Data** field.
- Operation:**
 - The **Index** selects a cache entry.
 - The **Tag** field in the selected entry is compared with the 20-bit **Tag** from the address to generate a **Hit** signal.
 - The **Data** field in the selected entry provides the 32-bit **Data**.

- 3

Four-Way Set Associative Cache

$2^8 = 256$ sets each with four ways (each with one block)

The diagram illustrates the internal structure of a Four-Way Set Associative Cache. The address is 32 bits long, divided into a 22-bit Tag and an 8-bit Index. The Index selects one of four sets. Each set contains four ways (V, Tag, Data). The Data outputs are ORed together to produce the final 32-bit Data. A Hit signal is generated if any way in the selected set is valid (V=1).

Address fields:

- Tag: 31 30 ... 13 12 11 ... 2 1
- Index: 8
- Byte offset: 1

Cache structure (4 sets, each with 4 ways):

Index	V	Tag	Data
0			
1			
2			
...			
253			
254			
255			

Each set's Data output is connected to a 4x1 select multiplexer. The Hit signal is generated by ORing the valid (V) signals of all four ways in the selected set.


- 4

Costs of Set Associative Caches

- **N-way set associative** cache costs
 - N comparators (delay and area)
 - MUX delay (set selection) before data is available
 - Data available **after** set selection and Hit/Miss decision.
- **When a miss occurs**, which way's block do we pick **for replacement** ?
 - **Least Recently Used (LRU)**:
the block replaced is the one that has been unused for the longest time
 - Must have hardware to keep track of when each way's block was used
 - For **2-way set associative**, takes **one bit per set** → set the bit when a block is referenced (and reset the other way's bit)
 - **Random**

5

- 5



キャッシュに関する論文

- M.K. Qureshi, D. Thompson, Y.N. Patt.
The V-Way Cache: Demand Based Associativity via Global Replacement.
In Proc. of Int. Symp. Computer Architecture, ISCA 2005.
- Kaushik Rajan and R.Govindarajan.
Emulating optimal replacement with a shepherd cache,
In MICRO-40, pp. 445-454, 2007.

- Adapted from *Computer Organization and Design*, Patterson & Hennessy, © 2005

The V-Way Cache: Demand Based Associativity via Global Replacement

Moinuddin K. Qureshi, The University of Texas at Austin
David Thompson, The University of Texas at Austin
Yale N. Patt, The University of Texas at Austin

ISCA-32 pp. 544-555

JEITA 2005-07-06

7

研究の目的と概要

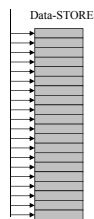
- メモリ参照のペナルティが大きくなり、L2キャッシュのヒット率が重要
- セットアソシアティブのキャッシュでは、それぞれのセットの利用頻度が異なる。
- セット単位で、連想度を変更するキャッシュ方式を提案評価する。
- Variable-Way set associative (V-Way) cache
- 提案方式は、キャッシュのミスの13%を削減する。
- プロセッサの性能を最大で44%、平均で8%改善する。

JEITA 2005-07-06

8

フルアソシアティブ方式

- あるキャッシュブロックを任意のエントリに格納できる。
- 利点
 - 競合ミスを排除できる。
 - もつとも必要のないエントリを探して置き換える(Global Replacement)ことができる。
- 欠点
 - 非現実的。
 - 必要となっているデータを探すために、莫大なタグ検索が必要。
 - 参照遅延が大きくなる。
 - 消費電力が大きくなる。



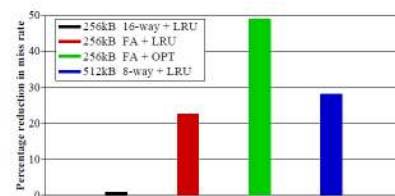
欠点を克服しながら、フルアソシアティブ方式の利点を使いたい。

JEITA 2005-07-06

9

フルアソシアティブ方式の可能性

- Need for efficient management of secondary caches.
- Ideal cache: fully associative with OPT replacement.

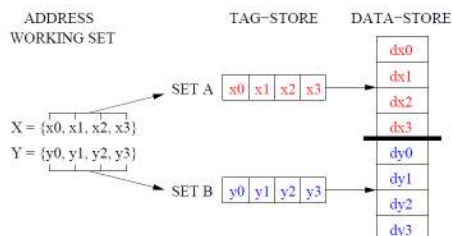


- 256KB 8-way + LRU の性能を基準とした削減率
- ハードウェア量を倍(512KB)にするよりも、フルアソシアティブ方式で、洗練された置き換え方式を利用の方が性能が高い。

JEITA 2005-07-06

10

Local Replacement の例 (1/2)

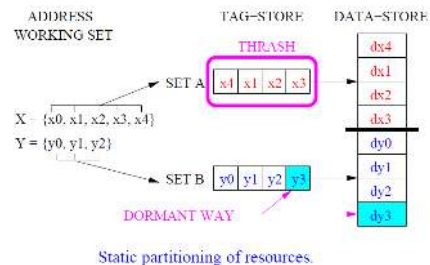


- 4ウェイセットアソシアティブ方式、簡略化するために2つのセットを想定
- セットAとセットBが同じ頻度で利用されている。
 - 深刻な競合は発生しない。

JEITA 2005-07-06

11

Local Replacement の例 (2/2)

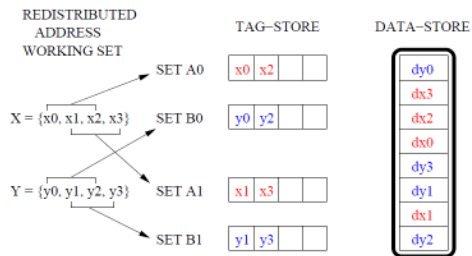


- セットAとセットBの利用頻度が異なる。
 - Dy3が利用されない。X0が追い出される。
 - 利用されていないウェイを活用することで、性能向上の可能性はある。

JEITA 2005-07-06

12

Global Replacement の例 (1/2)

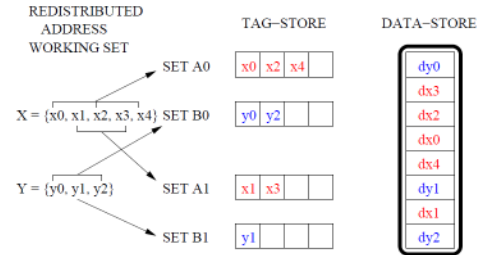


- タグ領域のエントリを倍に増やす。
- TDR (tag-to-data ratio) = 2
- タグ領域へのアドレッシングを変更する。

JEITA 2005-07-06

13

Global Replacement の例 (2/2)

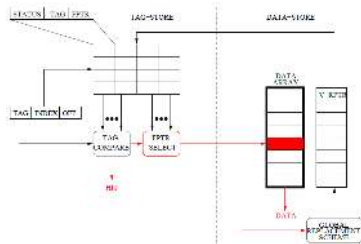


- X, Y の利用頻度が異なっても、有効にデータ領域を利用できる。
- X4 は、利用されていないDATA-STOREを検索して、そこを利用する。
- 動的にリソースを共有する。

JEITA 2005-07-06

14

V-Way Cache の提案

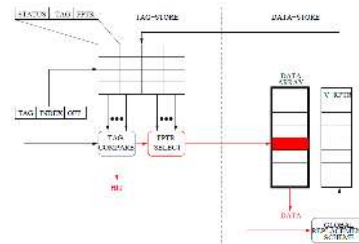


- TAG-STORE:
 - 通常のキャッシュのタグと有効ビットに加えて、DATA ARRAYのエントリを示す forward pointer (FPTR) を1要素とする。
 - この要素を複数用意して、1エントリとする。
- DATA-STORE:
 - データを格納する。加えて、有効ビットと、TAG-STORE への逆ポインタ RPTR を持つ。

JEITA 2005-07-06

15

V-Way Cache の動作



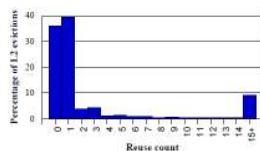
- 参照
 - TAG-STORE にヒットした場合には、そこに格納されているFPTRで DATA-STORE を参照して、データを供給
 - TAG-STORE にミスした場合には、TAG-STORE の情報を LRU で置き換える。この時、関連する DATA-STORE の情報も無効にする。その後、TAG-STORE, DATA-STORE に必要となるブロックを格納し、ポインタを作成する。

JEITA 2005-07-06

16

A Practical Global Replacement Algorithm

- LRU is impractical because there are thousands of lines
- Second level cache access stream is a filtered version of the program access stream
- Reuse frequency is skewed towards the low end

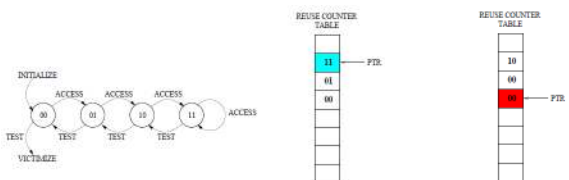


- L2 キャッシュの参照
 - L1 キャッシュのミスのみが、L2 キャッシュから観測される。
 - L1 キャッシュの様に、あるラインが非常に多数にわたって参照されることはない。
 - あるブロックがL2 キャッシュに格納されてから、追い出されるまでに参照された回数
 - 3回以内のブロックが非常に多い。15回以上のブロックは10%未満。

JEITA 2005-07-06

17

Practical Global Replacement: Reuse Replacement

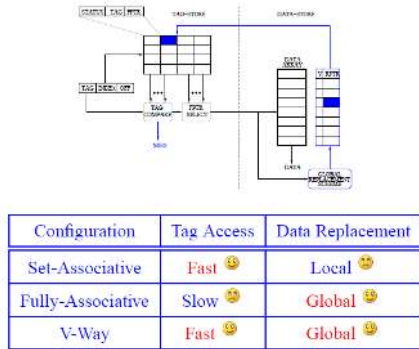


- DATA-STORE のそれぞれのエントリに2ビットのカウンタを追加
 - ブロックをロードした時にカウンタをゼロで初期化する。
 - そのブロックが参照された時に、カウンタの値をインクリメント
- レジスタ PTR を一つ用意
 - 置き換えるエントリを検索するために、PTR の内容を増やしながら当該エントリのカウンタをデクリメントする。カウンタがゼロのエントリを置き換える。

JEITA 2005-07-06

18

V-Way Cache の特徴

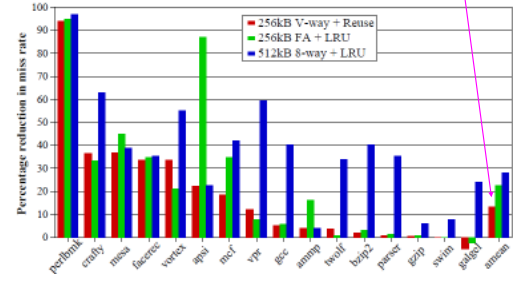


JEITA 2005-07-06

19

V-Way Cache のミスの削減率, 256KB 8-way をベースとして

- Primary upper bound: Fully associative cache
- Secondary upper bound: Double sized cache



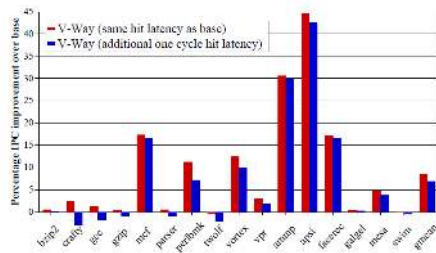
セットアソシアティブ方式と比較して、13%の予測ミスを削減する。

JEITA 2005-07-06

20

V-Way Cache による性能向上

- Pipeline: 12 stage, 8 wide with 128 entry reservation station
- L1 hit latency of 2 cycles and L2 hit latency of 10 cycles
- L3/Main memory: access-latency of 80 cycles



セットアソシアティブ方式と比較して、平均で8%の性能を改善する。

JEITA 2005-07-06

21

Session 8: Cache Replacement Policies

Emulating Optimal Replacement with a Shepherd Cache

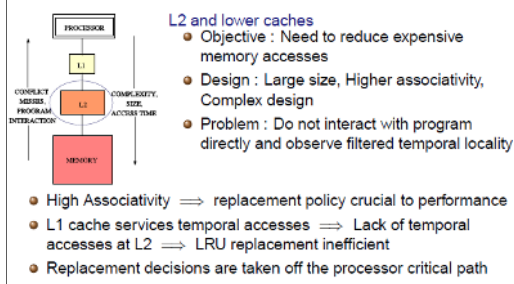
Kaushik Rajan, *Indian Institute of Science*
Govindarajan Ramaswamy, *Indian Institute of Science*
MICRO-40 pp. 445-454

JEITA 2008-02-07

22

はじめに、メモリ階層

Memory Hierarchy



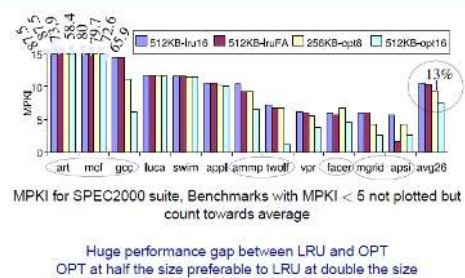
JEITA 2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

23

LRUには改善の余地がある。

LRU vs OPT



JEITA 2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

24

OPT: Optimal Replacement Policy

The Optimal Replacement Policy

- **Replacement Candidates** : On a miss any replacement policy could either choose to replace any of the lines in the cache or choose not to place the miss causing line in the cache at all.
- **Self Replacement** : The latter choice is referred to as a self-replacement or a cache bypass

Optimal Replacement Policy

On a miss replace the candidate to which an access is least imminent [Belady1966,Mattson1970,McFarling-thesis]

- **Lookahead Window** : Window of accesses between miss causing access and the access to the least imminent replacement candidate. Single pass simulation of OPT make use of lookahead windows to identify replacement candidates and modify current cache state [Sugumar-SIGMETRICS1993]

OPT: あまり切迫していないものを置き換える。

JEITA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

25

Optimal Replacement Policy の例

Understanding OPT

Access Sequence	A ₅	A ₁	A ₆	A ₃	A ₁	A ₄	A ₅	A ₂	A ₅	A ₇	A ₆	A ₈
OPT order for A ₅	0	1	2	3	4							
OPT order for A ₆			0	1	2	3					4	

- Consider 4 way associative cache with one set initially containing lines (A₁, A₂, A₃, A₄), consider the access stream shown in table
- Access A₅ misses, replacement decision proceeds as follows
 - 1 Identify replacement candidates : (A₁, A₂, A₃, A₄)
 - 2 Lookahead and gather imminence order : shown in table, lookahead window circled
 - 3 Make replacement decision : A₅ replaces A₂
- A₆ self-replaces, lookahead window and imminence order in table

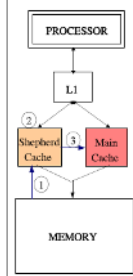
JEITA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

26

Shepherdキャッシュを利用してOPTを実現

Emulating OPT with a Shepherd Cache



- Split the cache into two logical parts
 - Main Cache (MC) for which optimal replacement is emulated
 - Shepherd Cache (SC) used to provide a lookahead and guide replacements from MC towards OPT
- Operation
 - 1 Buffer lines temporarily in SC before moving them to MC, SC acts as a FIFO buffer
 - 2 While in SC, gather imminence information and emulate lookahead
 - 3 When forced out of SC, make an MC replacement based on the gathered imminence order

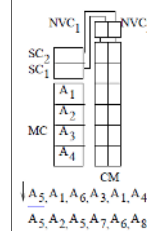
JEITA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

27

Shepherdキャッシュの概要

Overview of Shepherd Caching

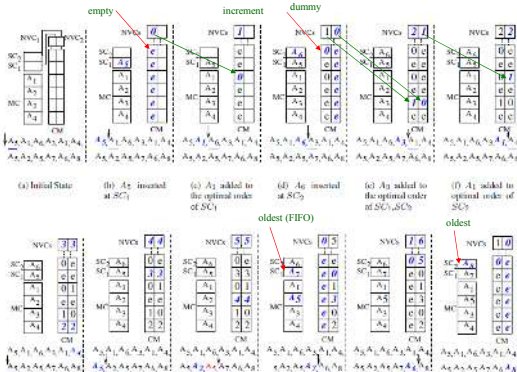


- To emulate MC with 4 ways per set and 2 SC ways per set
- To gather imminence order add a counter matrix (CM)
- CM has one column per SC way to track imminence order w.r.t to it
- CM has one row per SC and MC line as any of them can be a replacement candidate
- Each column has one Next Value Counter (NVC) to track the next value to assign along column

JEITA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

28



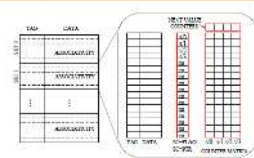
JEITA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

29

提案手法の構成

Cache Organization



- Divide cache into SC and MC with some ways of each set are SC ways
- Use SC flag to identify SC ways, use SC-ptr to link SC line to its column
- No need to physically move lines from SC to MC, just adjust SC flag/ptr
- Larger SC-assoc → more lookahead → OPT in smaller MC

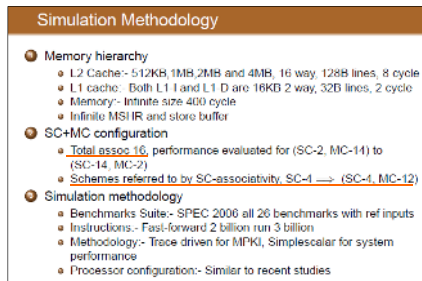
SC-N : Nを増やすと
先見能力が増加

JEITA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

30

シミュレーション環境, SC + MC 16として評価

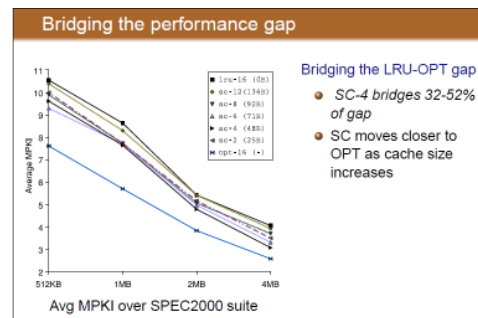


JETTA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

31

評価結果 LRU, OPT, 提案手法SCのミス率



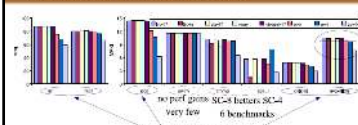
JETTA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

32

関連研究との比較, 提案手法SCが良い結果

Comparison with related schemes



MPKI for 512KB 16 way cache, select benchmarks, all 26 in average

- On average both SC-4 and SC-8 out-performs LRU, DIP, v-way, fa and victim by 4-10%
- For most benchmarks SC-4 performs better than SC-8
- Different SC associativities best for different benchmarks

Scheme	Parameters
DIP (Pruvost-HACAO)	16 32KB dedicated sets, apstion 1/32
v-way cache (Dauwal-HACAO)	32 tags per set, set 2
global reuse distance replacement	
Victim Buffer (Kumar-HACAO)	MC=16, 16 same basic as SC
fa cache	fully associative with LRU
OPT-16	16 way associative OPT replacement

Performance compared with base LRU, DIP, victim buffer with one additional way per set to compensate for size overhead

JETTA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

33

提案手法Shepherdキャッシュによる性能向上

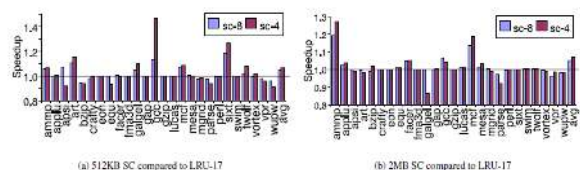


Figure 13: Speedup over LRU-17

SC-4では、512KBの構成で平均7%、2MBでは平均7.1%の性能向上

JETTA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

34

発表スライドのまとめ

Conclusions and Future Work

Conclusion

- Using part of cache for lookahead and emulating OPT in remaining cache effective at improving L2 performance
- Quarter to Half of cache space should be dedicated to SC
- Performs better than related proposals and successfully bridges 32-52% of the LRU-OPT gap

Future Work

- Different SC associativity works best with different benchmarks, can the SC-associativity be adapted at runtime?
- Study the benefits of dissociating SC and MC?
- Tuning the organization to a shared CMP cache?

JETTA
2008-02-07

MICRO-40 Emulating Optimal Replacement with a Shepherd Cache

35

講義計画

<http://www.arch.cs.titech.ac.jp/sub5.html>

- 導入: マイクロプロセッサ
- スーパースカラプロセッサの基礎と命令レベル並列性
- キャッシュ
- 分岐予測
- 動的命令スケジューリングと投機処理
- メモリデータフローとデータキャッシュ
- 組込技術, 低消費電力技術
- チップマルチプロセッサ
- オンチップネットワーク, メモリーコアアーキテクチャ
- 【成績評価】レポートおよび、期末レポートにより評価する。

Adapted from Computer Organization and Design, Patterson & Hennessy, © 2005

36