Notes

The V-way Cache : Demand based Associativity via Global Replacement

Introduction

↳ In Set associative cache, No. of entries visible to replacement policy is limited by no. of ways in each set.

↳ Cache Replacement Policy } critical for miss rate
Associativity } analysis

↳ Increases Associativity results in —
① less Conflict misses (+ve)
② Global Replacement (+ve)
③ more hit latency
④ More power Consumption } -ves

Motivation

Victim Caches : Small fully associative buffers for heavily utilized entries in DM caches

Hash-rehash Cache : Primary cache itself is secondary cache after miss.

Seq. Associative Cache : trades variable hit latency for increased associativity

↳ Effectiveness of above caches degrades on increasing associativity

↳ Static associativity results in local replacement

Soln

↳ doubling the number tag bits by doubling the number of sets.

↳ No. of data lines & valid tags remains same

↳ No. of sets doubled instead of associativity is due to power consumption & maintain cache hit latency.

Effects of doubling sets —

↳ No. of bits used to index tagstore (or set) inc. by 1

↳ No longer static one to one mapping b/w tag store & data lines.

↳ This implies tag comparison & data look up must be performed serially.

V-way Cache

- TDR = Tag to data ratio
  (normally taken 2 in paper)

- Entry of tag store contain —
  - valid bit, dirty bit, forward PTR
    ↓ use to identify data line.

- Data store entry contain dataline, a valid bit & Reverse PTR
  - point to tag

- Tag store uses traditional LRU replacement & data store uses global replacement scheme.

Operation

- If it is a cache hit, using FPTR data will be accessed & Replacement info will be updated

- If it is a miss —

  Case 1: Exist atleast one invalid store entry in target set —

  Using global Replacement, data victim will be identified & evicted &
  RPTR of data victim will be invalidated
  ↑ tag

- Tag victim is updated with new tag bits, RPTR of new data point to Tag victim & FPTR is updated to point to data victim. Valid bit set.

Case 2 : All the tag store entries are valid
We do this like normal replacement, choosing tag victim using LRU.

- A V-way Cache can achieve miss rate comparable to a traditional cache of twice its size ~~of~~ or FA of same size.

Practical Global Replacement Algo —

- Replacement Algo is very imp as random Replacement or FIFO increases the miss rate. Perfect LRU has ~~sp~~ SC $\rightarrow O(n^2)$

Reuse frequency

- L2 has much lower measure of temp. locality than recorded by L1.
- ~~Four~~ 2bits can be used to track 4 diff. reuse count states 0, 1, 2, 3+
- We only need 4 levels as 80% lines have reuse count less than or equal to 3.

Reuse Replacement

- Every data line has 2 bit for Reuse Counter
- RCT (Reuse Counter Table) structure is physically separate to not cause any delay in read/write

to cache line when RCT are updating.

Algo for replacement —

① PTR reg point to start of search of data victim
② for newly installed cache line, reuse counter ~~associated w~~ is initalised to zero,
③ On hit reuse counter is incremented by 1.
④ ~~We~~ Start from PTR, check if reuse counter is zero, if yes then it is data victim ~~others~~ otherwise decrement reuse counter by 1.
⑤ Continue till data victim is found. (wraparound)
⑥ After Data Victim found, increase PTR to point to next data line. (Helps in reaching to representative value) before being tested.

→ Maximum Victim distance = $(2^N - 1) \times$ No. of counters.
where N = no. of bits for storing reuse counter

→ Usually Victim distance will be less because of 2 cases —
① Majority cache line exhibit little to reuse
② decoupling of tag & data store has effect of randomising cache line in data storage reducing stride based access patter to generate long victim distance.

→ Prob(~~for~~ Victim distance ≤ 5 cycle) = 99.2
We can also do early termination with negligible impact on miss rate

Experiments & Benchmarks

L1 I	= 16 kb	2 way	64 B line	LRU Replacement
L1 D	= 16 kb	2 way	64 B line	"
L2	= 256 kb	8 way	~~64~~ 128 B line	"

↳ ~~to~~ Benchmark used : SPEC CPU 2000

↳ Benchmark with low miss rate on Baseline excluded also benchmark with less improvement (< 4%) on doubling size are also excluded.

Results

- V-way Cache provide an avg. Miss rate reduction of 13.2% compared to baseline
- V-way Cache has 2 upper bounds -
  1. Primary (Fully Associative)
  2. Secondary (Double size Cache)
- Usually

  gain by V cache < min (gain by FA, gain by double sized cache)

  Only outperform when Reuse Replacement perform really good.
- Reuse Replacement performs almost similar (marginally better) than Perfect LRU on very less hardware, cost and complexity

↳ On 128 bit cache line, V-way uses 5.8% more hardware than baseline

↳ Added Latency cause 1 ~~more~~ extra cycle to be added in hit latency.

↳ Energy Consumption goes higher due to additional tag store

Analyses

↳ V-way cache provide 8.5% for 10 cycle latency & 6.8% for 11 cycle latency, improvement over IPC

↳ for some benchmark 1 extra cycle result in degrade of performance due to large working set. L1 Data misses hide by OOO execution but Instruction misses can reduce IPC improvement.

↳ Tag to Data ratio = 2 will suffice for most of the benchmarks

↳ V-way reduces miss rate compared to traditional 8-way cache for both 512kB & 1Mb cache. Results are not as good as of 256 kb as some of benchmarks starts fitting into cache.

↳ The V-way Cache support variable cache demand by increasing associativity of high demand sets while reducing the associativity of ~~high~~ low demand sets.