Estimating the Performance of Cache Replacement Policies

Daniel Grund, Jan Reineke

Department of Computer Science
Saarland University

MeMoCode, June 2008

Outline

Outline

Locality & Caches

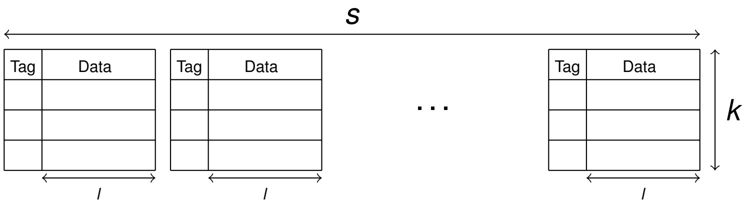
- Typical programs do not behave randomly, they exhibit
  - temporal locality
  - spatial locality

  in their memory access behavior

- Caches try to exploit locality
- Have great influence on system performance

Cache Parameters

Line size l	Size of blocks stored in cache
Associativity k	Number of lines one block may be stored in
Number of sets s	Each set comprises k lines
Capacity	Determined by $s \cdot k \cdot l$
Replacement policy	*next slide*



Replacement Policies

- The replacement policy determines the cache line whose contents are replaced upon a cache miss
- Examples:

LRU	Least recently used
FIFO	First-in first-out
PLRU	Approximation on LRU
MRU	Most recently used
...	

Problem Statement

- Cache performance depends on (locality of) executed program
- There are lots of design possibilities

Problem

Given a program, determine the best cache configuration.

Stack Histograms (Mattson et al. 1970)

- Concisely capture locality in memory access behavior
- Age of cacheable element *e*:
  number of accesses on different elements since last access on *e*
- Stack distance of a memory access:
  age of the accessed element just before the access
- Stack histogram:
  frequency distribution of stack distances

Example

Reference string	a	b	a	c	b	b	c	a
Stack distance	∞	∞	1	∞	2	0	1	2

Stack distance d	0	1	2	3	≥ 4
Rel. frequency f_d	$\frac{1}{8}$	$\frac{2}{8}$	$\frac{2}{8}$	0	$\frac{3}{8}$

Estimation for LRU Is Easy

Mattson et al. (1970)

- LRU always retains the k most recently used elements
- Exactly the ages $0, \dots, k-1$ are cached

Example

Associativity $k = 8$

d	0	1	2	3	4	5	6	7	8	9	10	11	12	$\geq$
f_d	always cached								never cached					

$\Rightarrow$ Miss ratio $= 1 - \sum_{i=0}^{7} f_i$

For Other Policies It Is Not

Problem:

- In general, the ages of cached elements vary during execution
- Number and shape of combinations of ages depend on policy

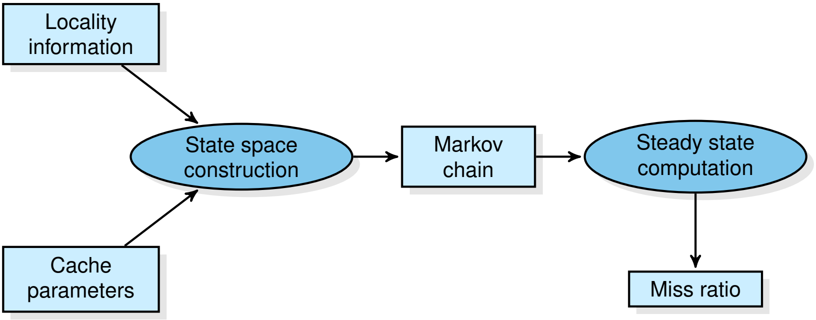
Idea: determine

- all possible combinations (state space)
- probability of each state
- probability of a cache miss in each state

Outline

Our Approach

- Static method
- Given . . .
  - a concise characterization of the program's locality
  - parameters of the cache architecture
- . . . we construct a Markov chain to estimate the miss ratio



Contributions

- Replacement policy is taken into account
- Policy tables
  - uniformly represent replacement policies
  - ease automatic model construction
- History stack histograms
  - extend well-known stack histograms
  - increase precision of estimates

Policy Tables

- Uniform representation for a class of replacement policies
- Class contains LRU, PLRU, FIFO, MRU
- Eases automatic model construction

$$\begin{array}{c} \pi_0 \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_m \end{array} \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \end{bmatrix}$$

8-way FIFO

$$\begin{array}{c} \pi_0 \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_m \end{array} \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \\ 0 & 2 & 3 & 4 & 5 & 6 & 7 & 1 \\ 0 & 1 & 3 & 4 & 5 & 6 & 7 & 2 \\ 0 & 1 & 2 & 4 & 5 & 6 & 7 & 3 \\ 0 & 1 & 2 & 3 & 5 & 6 & 7 & 4 \\ 0 & 1 & 2 & 3 & 4 & 6 & 7 & 5 \\ 0 & 1 & 2 & 3 & 4 & 5 & 7 & 6 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \end{bmatrix}$$

8-way LRU

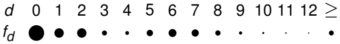
- *Policy state* is virtual order on cache lines

  $[3, 2, 1, 0, 4, 5, 6, 7]$
- *Updates* are specified by permutations
  - π_i for cache-hit on position i
  - π_m for cache-miss

$$[3, 2, 1, 0, 4, 5, 6, 7] \xrightarrow{\pi_4} [3, 2, 1, 0, 5, 6, 7, 4]$$

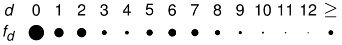
History Stack Histograms

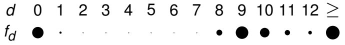
Circle plots of stack histograms



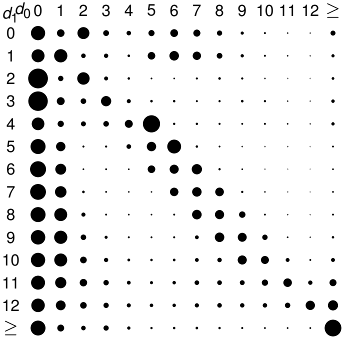
History Stack Histograms

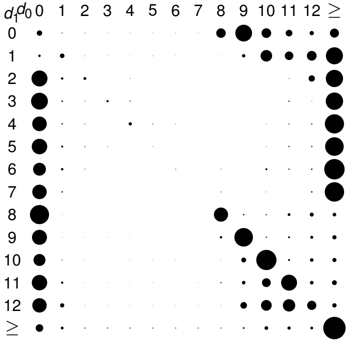
Circle plots of stack histograms





History stack histograms consider context of occurring stack distances





$\Rightarrow$ increases precision of estimates

Model Construction: States (1)

- Concrete cache set state: tuple of elements and status bits
- Two FIFO states: $[a, \underline{b}, c, d]$ and $[d, a, \underline{b}, c]$

- Only relative order w.r.t. replacment matters
- Abstract from physical positions

$\Rightarrow$ Normalized tuple: $[b, c, d, a]$

- Only ages of elements matter
- Abstract from names

$\Rightarrow$ Normalized tuple of ages, e.g. $[4, 1, 5, 0]$

Model Construction: States (2)

- Problem: arbitrarily high ages $\Rightarrow$ infinite model
- Introduce cutoff-age, abstract from high ages

$\Rightarrow$ $\alpha([4,1,5,0],[6,1,4,0]) = [\mathbf{4},1,\mathbf{4},0]$

- Disambiguate states by history to fit history stack histograms

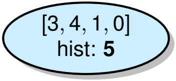
$\Rightarrow$ Abstract cache set state:

$[\mathbf{4},1,\mathbf{4},0]$
hist: $0,2$

Model Construction: Transitions

State space construction:

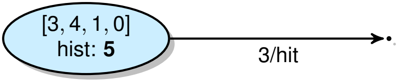
- Pick unexplored abstract state *s*



Model Construction: Transitions

State space construction:

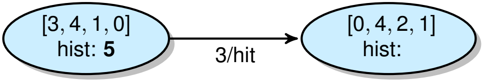
- Pick unexplored abstract state *s*
- Enumerate all possible accesses (on ages)



Model Construction: Transitions

State space construction:

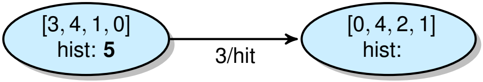
- Pick unexplored abstract state s
- Enumerate all possible accesses (on ages)
- For each successor state s'
  - Update ages



Model Construction: Transitions

State space construction:

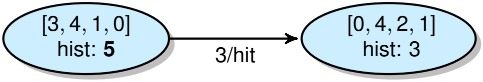
- Pick unexplored abstract state s
- Enumerate all possible accesses (on ages)
- For each successor state s'
  - Update ages
  - Update order (apply permutation from policy table)



Model Construction: Transitions

State space construction:

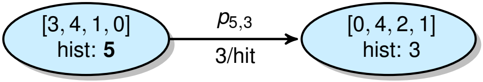
- Pick unexplored abstract state s
- Enumerate all possible accesses (on ages)
- For each successor state s'
  - Update ages
  - Update order (apply permutation from policy table)
  - Adjust history



Model Construction: Transitions

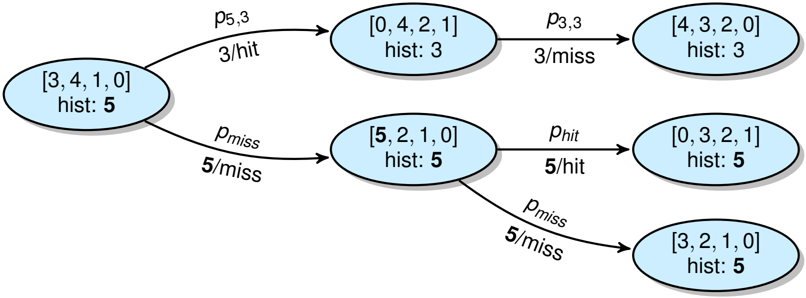
State space construction:

- Pick unexplored abstract state s
- Enumerate all possible accesses (on ages)
- For each successor state s'
  - Update ages
  - Update order (apply permutation from policy table)
  - Adjust history
  - Compute probability for transition $s \longrightarrow s'$ from history stack histogram



Model Construction: Example

4-way FIFO, history length 1, cutoff age 5



Computing the Miss-Ratio Estimate

1. Compute steady state probabilities $p(s_i)$ of the Markov chain
   - e.g. with the Gauss-Seidel algorithm
2. Compute miss probabilities $p(m_i)$
   - simple sum of values in history stack histogram

$\Rightarrow$ Miss ratio $= \sum_i p(s_i) \cdot p(m_i)$

Outline

Evaluation: Precision

- Shared 2nd level caches with different parameter combinations
- C programs of SPEC CPU2000
- Evaluated against full simulation runs (SimpleScalar)

- Average absolute error between 0.18% and 2.92%
- Better to spend resources on history than on higher cutoff age

- Example: 8-way associative, 1024 sets, 32 bytes line size

	LRU	PLRU	FIFO	MRU	RAND
Average error	0	0.23	0.58	2.92	1.61

Evaluation: Runtime

- k=4 is handled easily
- k=8 often requires cheaper model parameters
  i.e. stronger abstraction / less precision

- Example: Most expensive model parameters
  for $k = 8$ and 2 GB main memory

	PLRU	FIFO	MRU	RAND
#states	4,166,048	8,216,745	4,129,362	2,687,856
∅runtime [s]	386	440	263	234

- Comparison: ∅simulation time was 18 hours

Limitations / Future Work

- Sometimes precision is limited by resources/model parameters, e.g. 8-FIFO or 8-RAND with history
- History length is currently limited to 1.
  ≥ 2 requires $>$ 2GB main memory or stronger abstractions
- Implementation limited to policies in policy table class.
  Others (e.g. 2Q, EE-LRU, ARC) conceivable
  but most likely require different abstractions

Related Work

- Mattson, Gecsei, Slutz, and Traiger
  Evaluation techniques for storage hierarchies
  IBM Systems Journal, 9(2), 1970.

- Guo and Solihin
  An analytical model for cache replacement policy performance
  SIGMETRICS Perf. Eval. Rev., 34(1), 2006.

- Mattson et al. limited to LRU, but seminal work and still useful
- Guo and Solihin
  - policies directly specified in terms of probability functions
  - hence method more efficient
  - but real-world policies (PLRU, FIFO, MRU) impossible

Outline

1. Introduction
   - Locality & Caches
   - Problem & Prior Work
2. Stochastic Model
   - Contributions
   - Model Construction
3. Evaluation
   - Precision & Runtime
   - Related Work
4. Conclusions

Conclusions

- Static method for miss-ratio estimation
  - stochastic model yields precise estimates
  - allows for considering real-world policies
- History stack histograms
  - extend well-known stack histograms
  - increase precision of estimates
- Policy tables
  - provide uniform representation of policies