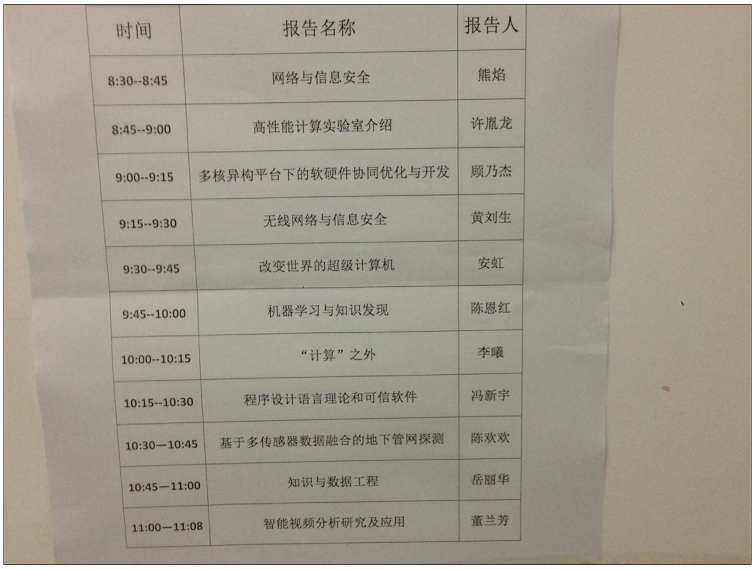
多核共享缓存下的编程优化和正确性
Program Behavior in Shared Cache: Performance and Correctness

软硬件协同管理和优化缓存
Collaborative Cache Management and Optimization

丁晨 Chen Ding
美国纽约州私立罗切斯特大学
计算机科学系教授
Professor
University of Rochester

2014 DragonStar Course at University of Science and Technology of China



时间	报告名称	报告人
8:30--8:45	网络与信息安全	熊焰
8:45--9:00	高性能计算实验室介绍	许胤龙
9:00--9:15	多核异构平台下的软硬件协同优化与开发	顾乃杰
9:15--9:30	无线网络与信息安全	黄刘生
9:30--9:45	改变世界的超级计算机	安虹
9:45--10:00	机器学习与知识发现	陈恩红
10:00--10:15	“计算”之外	李曦
10:15--10:30	程序设计语言理论和可信软件	冯新宇
10:30—10:45	基于多传感器数据融合的地下管网探测	陈欢欢
10:45—11:00	知识与数据工程	岳丽华
11:00—11:08	智能视频分析研究及应用	董兰芳

UNIVERSITEIT GENT
ELIS

Euro-Par 2002 and Ph.D. Thesis Defense 2004

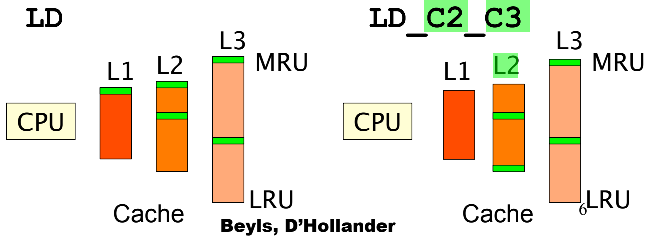
Reuse Distance-Based Cache Hint Selection

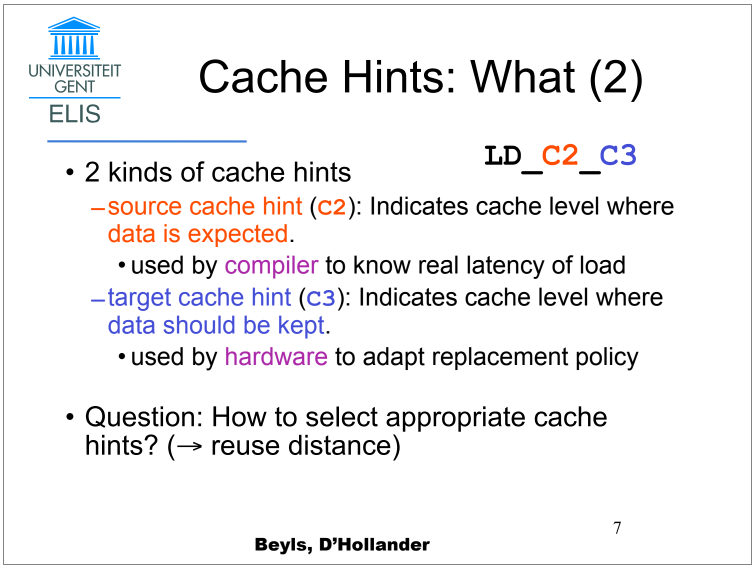
Kristof Beyls and Erik D'Hollander
Ghent University

UNIVERSITEIT GENT
ELIS

Cache Hints: What

- Cache control instructions are emerging in a number of architectures.
- HP-PlayDoh EPIC architecture provides 2 kinds of cache hints:

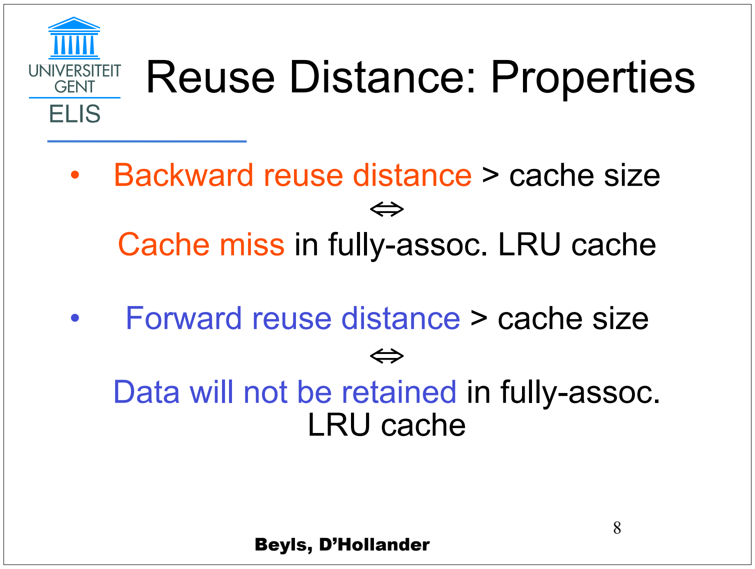




Cache Hints: What (2)

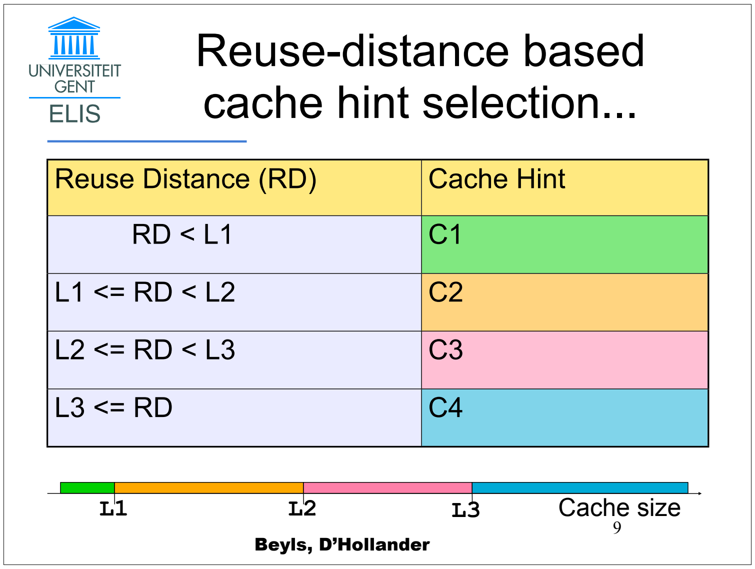
LD_C2_C3

- 2 kinds of cache hints
  - source cache hint (C2): Indicates cache level where data is expected.
    - used by compiler to know real latency of load
  - target cache hint (C3): Indicates cache level where data should be kept.
    - used by hardware to adapt replacement policy
- Question: How to select appropriate cache hints? (→ reuse distance)



Reuse Distance: Properties

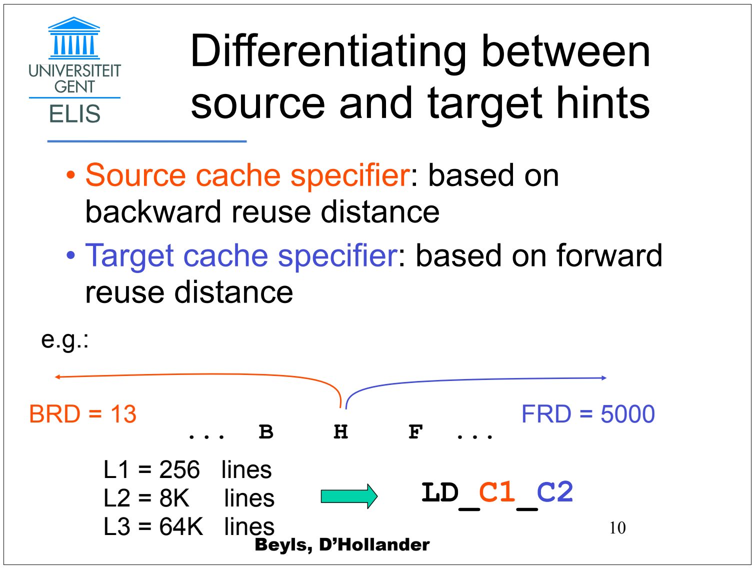
- Backward reuse distance > cache size
  ⇔
  Cache miss in fully-assoc. LRU cache
- Forward reuse distance > cache size
  ⇔
  Data will not be retained in fully-assoc. LRU cache



Reuse-distance based cache hint selection...

Reuse Distance (RD)	Cache Hint
RD < L1	C1
L1 <= RD < L2	C2
L2 <= RD < L3	C3
L3 <= RD	C4

L1
L2
L3
Cache size



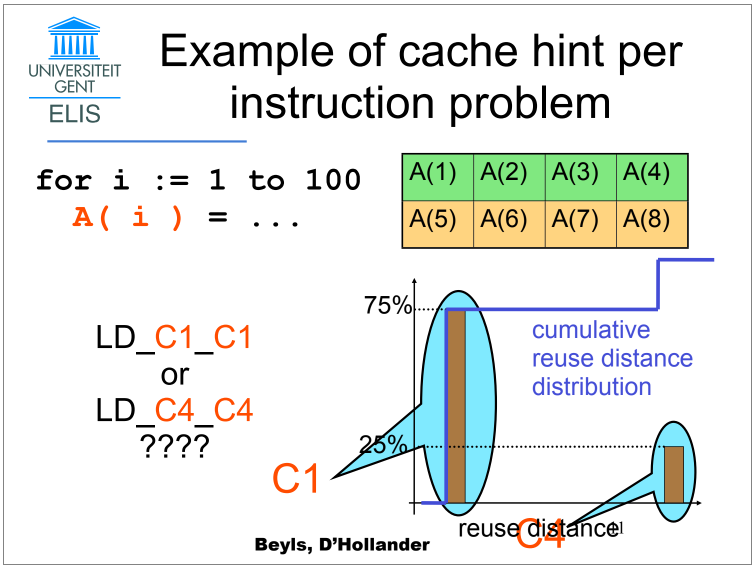
Differentiating between source and target hints

- Source cache specifier: based on backward reuse distance
- Target cache specifier: based on forward reuse distance

e.g.:

L1 = 256 lines
L2 = 8K lines
L3 = 64K lines

LD_C1_C2

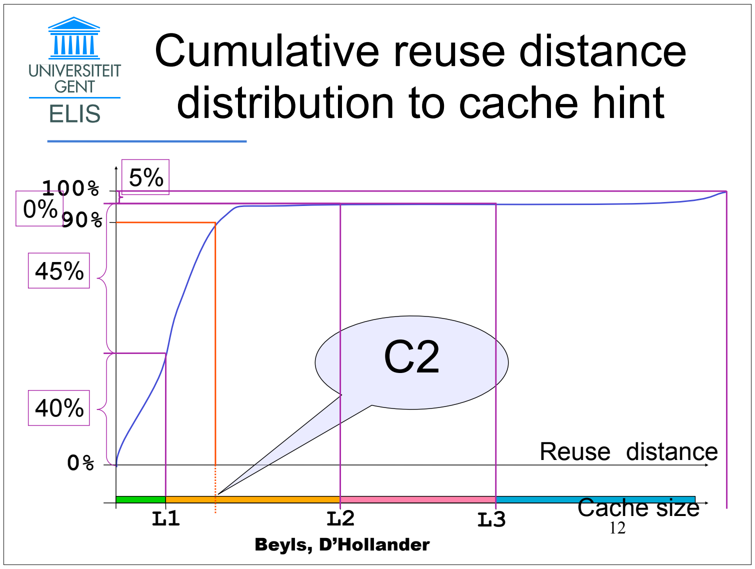


Example of cache hint per instruction problem

```
for i := 1 to 100
  A( i ) = ...
```

A(1)	A(2)	A(3)	A(4)
A(5)	A(6)	A(7)	A(8)

LD_C1_C1
or
LD_C4_C4
????



Cumulative reuse distance distribution to cache hint

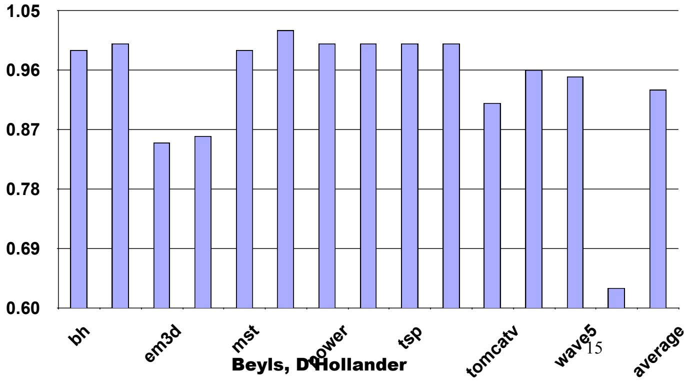
Strategy

- Instrument the program to obtain memory access stream.
- Profile to obtain reuse distance distribution.
- Generate cache hints.
- Execute optimized program.

Implementation

- Open64 compiler for Itanium (IA-64)
  - –source cache hints: in instruction scheduler
    - only visible in internal compiler representation of instructions.
  - –target cache hints: in assembly output
    - shows up in assembly code e.g. ld.**nta** r34 = [r47]
- Programs from Olden and Spec95fp.

Execution times


Normalized execution time after cache hint selection

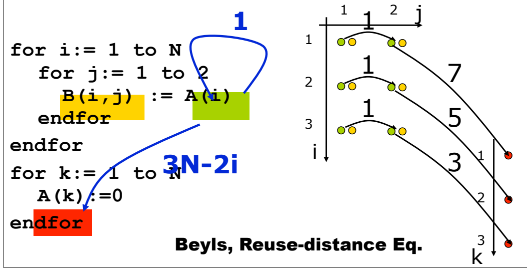
Results

- In Olden (pointer chasing), speedup comes mainly from target cache hints (better replacement policy) (4% on average)
- In Specfp (numerical loops), speedup results mainly from source cache hints (better latency hiding through instruction scheduling). (10% on average)

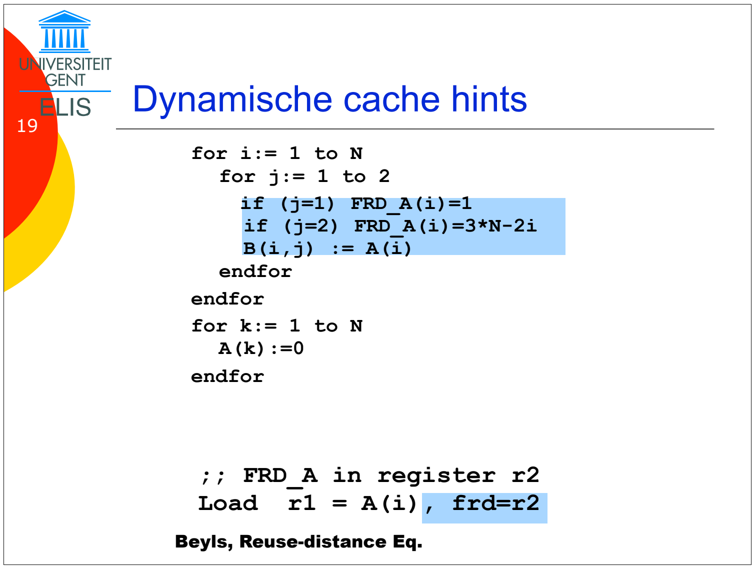
Conclusion

- Reuse distance is independent of cache parameters such as size or associativity.
- ⇒ good metric for optimizations targeting multiple cache levels.
- As such, it is an appropriate measure to base cache hint selection on.
- The implementation in an EPIC compiler resulted in 7% speedup on average with a maximum of 36%.

Stap 3: Tel aantal aangesproken elementen



```
for i:= 1 to N
  for j:= 1 to 2
    B(i,j) := A(i)
  endfor
endfor
for k:= 1 to N
  A(k):=0
endfor
```

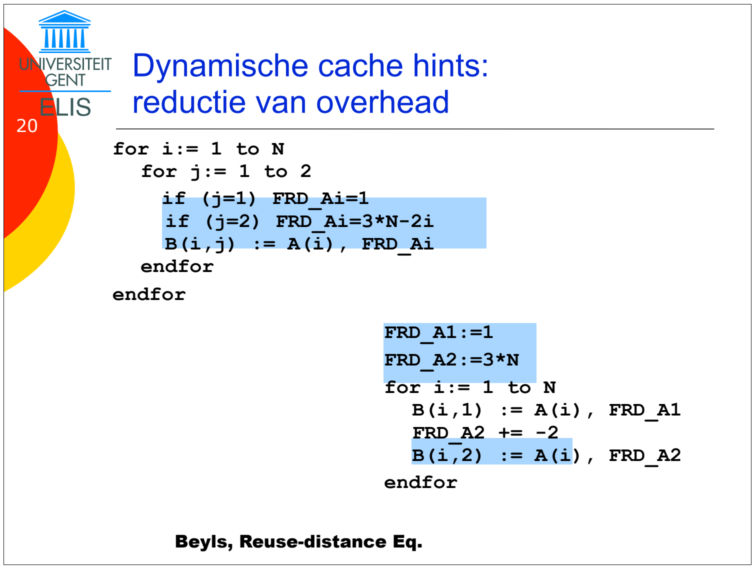


Dynamische cache hints

```
for i:= 1 to N
   for j:= 1 to 2
     if (j=1) FRD_A(i)=1
     if (j=2) FRD_A(i)=3*N-2i
     B(i,j) := A(i)
   endfor
endfor
for k:= 1 to N
   A(k):=0
endfor
```

```
;; FRD_A in register r2
Load  r1 = A(i), frd=r2
```

Beyls, Reuse-distance Eq.

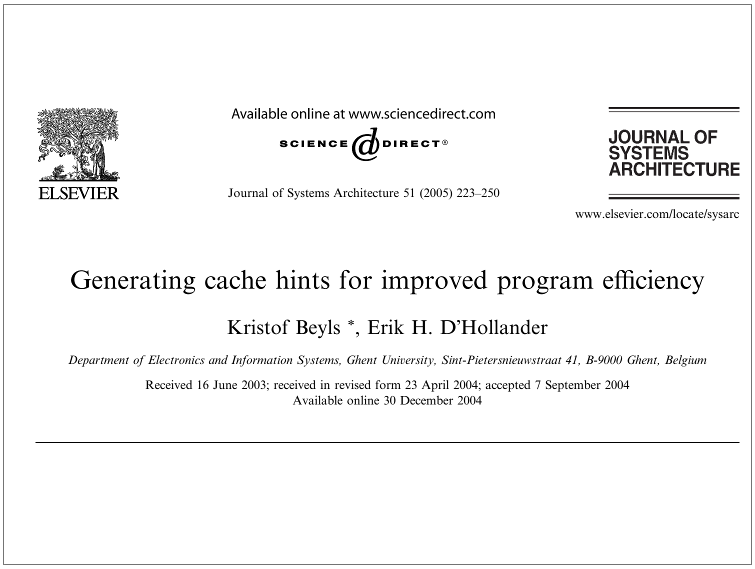


Dynamische cache hints: reductie van overhead

```
for i:= 1 to N
   for j:= 1 to 2
     if (j=1) FRD_Ai=1
     if (j=2) FRD_Ai=3*N-2i
     B(i,j) := A(i), FRD_Ai
   endfor
endfor
```

```
FRD_A1:=1
FRD_A2:=3*N
for i:= 1 to N
   B(i,1) := A(i), FRD_A1
   FRD_A2 += -2
   B(i,2) := A(i), FRD_A2
endfor
```

Beyls, Reuse-distance Eq.



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Systems Architecture 51 (2005) 223–250

JOURNAL OF SYSTEMS ARCHITECTURE

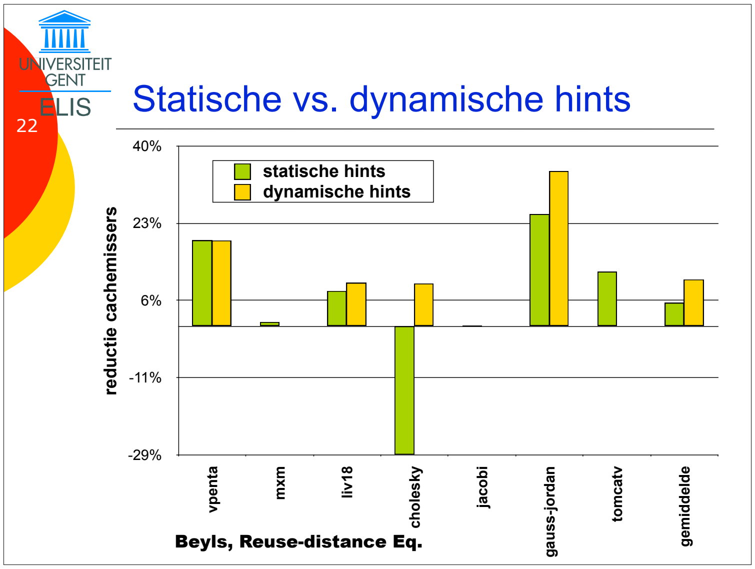
www.elsevier.com/locate/sysarc

Generating cache hints for improved program efficiency

Kristof Beyls *, Erik H. D'Hollander

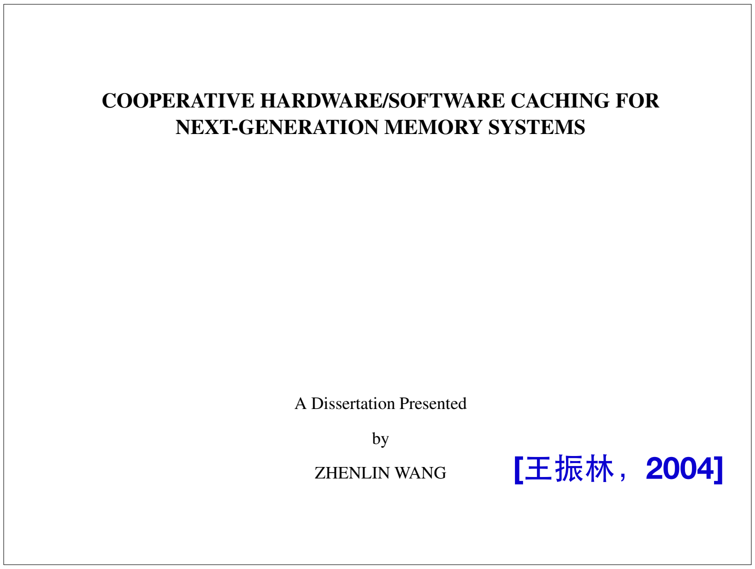
Department of Electronics and Information Systems, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium

Received 16 June 2003; received in revised form 23 April 2004; accepted 7 September 2004
Available online 30 December 2004



Statische vs. dynamische hints

Beyls, Reuse-distance Eq.



COOPERATIVE HARDWARE/SOFTWARE CACHING FOR NEXT-GENERATION MEMORY SYSTEMS

A Dissertation Presented

by

ZHENLIN WANG

[王振林，2004]

Can Collaborative Caching Be Optimal?

Optimal Caching

- Optimal cache management
  - MIN by Belady, CACM 1966
  - OPT by Mattson et al., IBM SJ 1970
  - Improvements over LRU by large factors
    - proportional to cache size in theory [Sleater&Tarjan, CACM 1985]
    - up to a hundred times in simulation [Burger et al., ISCA 1996]
- Assuming no program rerodering
  - Optimal ordering is NP-hard
    - Computation fusion, Ding & Kennedy, JPDC 2004
    - Data layout, Petrank & Rawitz, POPL 2002

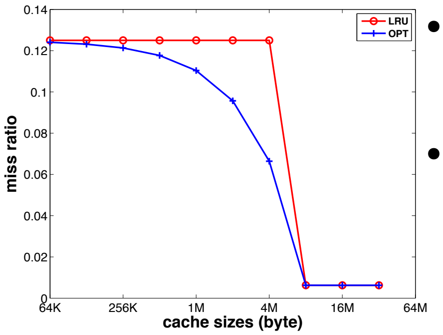
Optimal Collaborative Caching: Theory and Applications

Xiaoming Gu
08.15.2013

OPT Cache Replacement Policy

- Optimal
- The furthest reused data element is the victim when an eviction is needed
- Impossible for real hardware
- Previously only useful for studying the performance upper bound

Streaming Program



- LRU: no cache reuse until data fits in cache
- OPT: miss ratio drops proportionally with cache size increase

Trace w/ Mixed Locality

trace	xyaxybxycxydxyexyfxygxyaxybxycxydxye ...
LRU distance	---33-33-33-33-33-33-1191191191191l9 ...
MRU distance	---23-34-45-56-67-78-89292434545656 7 ...
OPT distance	---23-23-23-23-23-23-234235236237238 ...

capacity misses when cache size = 5

- Stack distance
  - Requires the inclusion property
  - A cache miss iff distance > cache size
- LRU: bad for streaming (low locality) data
- MRU: bad for high locality data
- OPT: store all high locality data and as much as low locality data as possible

Collaborative Caching

- Software provides cache hints to influence hardware cache management
- The term coined by Wang et al [PACT'02]
- Available cache hint interfaces in real hardware
  - Intel X86 & Itanium, IBM Power
- Previous works not aimed to be optimal

Outline

- Introduction
- Theory
- Applications

New Cache Types and Formal Properties

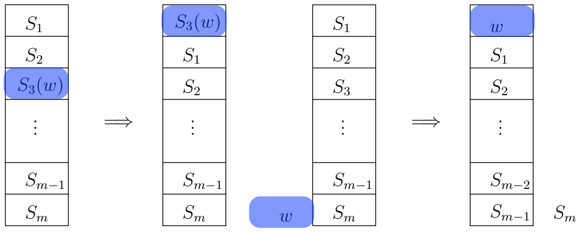
- Three collaborative cache types
  - Trespass LRU cache [LCPC'08]
  - LRU-MRU cache [ISMM'11] ★
  - Priority LRU cache [ISMM'12] ★
- Two formal properties
  - Optimality
  - Inclusion

Inclusion Property

- A larger cache always contains the content of a smaller cache
  - Miss curve is non-increasing
  - Miss ratio can be simulated as a stack in one pass for all cache sizes
  - Stack distance exists

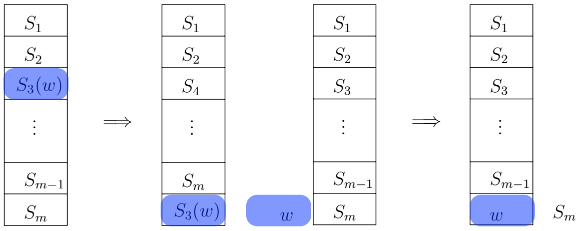
LRU-MRU Cache

- Two access types: LRU or MRU
  - 1-bit hint interface



LRU hit LRU miss

LRU-MRU Cache



MRU hit MRU miss

- MRU data have lower priority
  - Evicted before LRU data

LRU-MRU Cache

- Can be made optimal
- Inclusion property holds
  - Two-page proof (pp. 41 -- 42)
  - Stack distance exists

Optimal Hint Insertion

- Use forward OPT distance
  - If distance > cache size => MRU
  - Otherwise => LRU

trace	xyaxybxycxydxyexyfxygxyaxybxycxydxye ...
OPT distance	---23-23-23-23-23-23-234235236237238 ...
Fwd OPT distance	234235236237238239234235236237238239 ...
Optimal hint (c=5)	LLLLLLLLMLLMLLMLLMLLLLLLLLMLLMLLMLLM ...
LRU-MRU distance	---33-33-32-32-32-32-325334336327328 ...

MRU accesses fwd OPT dis > cache size

same misses as OPT (for c=5)

Theorem 1 Bypass LRU Is Optimal

- Proof by contradiction
  - first z’ that is hit in OPT but miss in BLRU
  - let z be the previous access to the same data d
- Two cases
  - z is a bypass access
    - z’ cannot be cache hit in OPT
  - z is a normal access
    - y exists that evicts d after z
    - if OPT cache is not full, trivial
    - if cache is full
      - d is at the bottom -> all elems are brought in by normal accesses
      - OPT and BLRU have same content before y
        - if d’ in BLRU not OPT, d’ has to be brought in by a bypass
      - OPT must evict some data x, then x has to be from a bypass

Non-optimal Uses

- LRU/MRU may mix in arbitrary ways
- Maybe not optimal
- Inclusion property still holds
  - Same two-page proof (pp. 41 -- 42)
  - Stack distance exists

LRU-MRU Stack Distance

- Inclusion property traditionally requires a single priority list
- Two types of priorities
  - LRU and MRU data managed differently
- Solution --- combine two priorities
  - LRU priority: the last access time
  - MRU priority: the negation of the last access time

An Example of LRU-MRU

access time		1	2	3	4	5	6	7	8	9	10	11
accessed data		a	b	c	d	d	c	e	b	e	c	d
LRU or MRU		M	L	L	L	M	M	L	L	L	M	L
priority list (data-priority)	1	a:-1	b:2	c:3	d:4	d:-5	c:-6	e:7	b:8	e:9	c:-10	d:11
	2			b:2	c:3	c:3	d:-5	d:-5	e:7	b:8	e:9	e:9
	3				b:2	b:2	b:2	b:2	d:-5	d:-5	b:8	b:8
	4											
LRU-MRU stack distance		∞	∞	∞	∞	1	2	∞	3	2	∞	∞

- LRU priority: the last access time
- MRU priority: the negation of the last access time
- MRU data evicted before LRU data

Optimal Collaborative Caching:
Theory and Applications

by

Xiaoming Gu

Submitted in Partial Fulfillment

of the

Optimal cache performance?
Answer: Yes

Miss rate in all cache sizes?
Answer: LRU-MRU (Gu) distance
[Gu et al. LCPC 2008, ISMM 2011/2012/2013, Rochester Dissertation 2013]

Rochester, New York

2013

Priority Hint

- 1-bit hint in LRU-MRU cache
  - Only two choices: LRU or MRU
- Priority hint
  - A number encoding a priority
  - Unlimited choices
  - Priority LRU cache

Priority LRU

- Inclusion property holds
  - Six-page proof (pp. 55 -- 60)
  - Non-uniform
    - The priority list does not exist
      - Traditional (uniform) inclusion requires a single priority list
    - A brand new algorithm for stack distance

Example of Non-uniform Inclusion

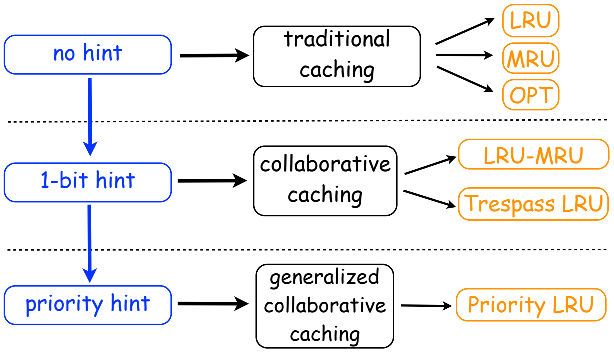
access time	1	2	3	4	5	6	7	8	9
data & hint	A:2	B:2	C:5	D:1	B:6	D:6	A:4	C:1	A:4
cache: 1				D	D			C	C
2	A	B	B			A			
3		A	A	B	A				
4				A			A		A
5			C					Ⓐ	

cache size = 5

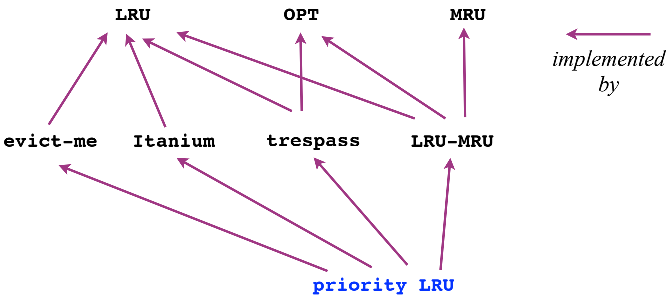
access time	1	2	3	4	5	6	7	8	9
data & hint	A:2	B:2	C:5	D:1	B:6	D:6	A:4	C:1	A:4
cache: 1				D	D			C	C
2	A	B	B			A			
3		A	A	B	A		C		
4				A		C	A	Ⓐ	A
5			C		C	B	B	B	B
6				C	B	D	D	D	D

cache size = 6

Cache Policy Hierarchy



Cache Policy Hierarchy



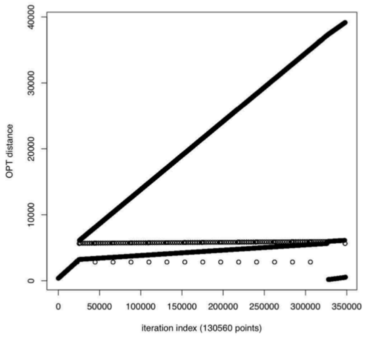
Outline

- Introduction
- Theory
- Applications

PACMAN

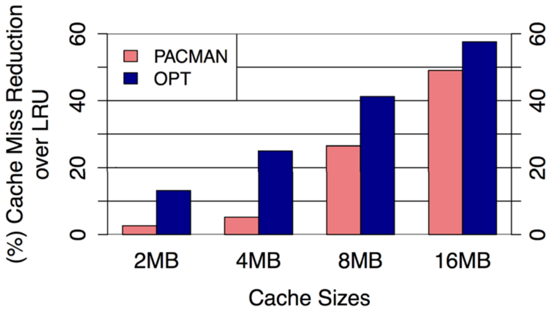
- Program-assisted Cache Management
  - A feedback-based cache-hint optimization
  - Reference-based PACMAN [ISMM'11]
  - Loop-based PACMAN [ISMM'13]

Cross-input Prediction



- Grid regression for pattern recognition
- Prediction based on input sizes

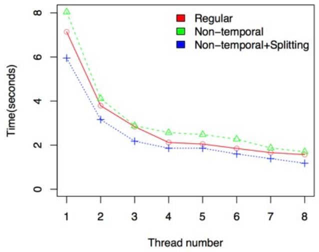
Improv. on Swim



- Training on input 256 by 256 and 384 by 384
- Testing on 512 by 512

Improv. on Real Hardware

- An OpenMP streaming example



A 12MB array is traversed many times

- The inner parallel loop is split into two
  - 1st => 8MB & normal access
  - 2nd => 4MB & non-temporal

Thesis Statement

- Optimal cache management can be done efficiently through software-hardware collaboration
- Collaborative hardware can be as simple and efficient as existing cache and robust against Belady anomaly
- Collaborative software can be general and optimized for all cache sizes

On-going Studies

Cache Rationing

with Jacob Brock and Raj Parihar (ECE)

Collaborative Rationing [PACT 2014 poster]

```
Thread 1      | a b c a b c a b c
Hint Bit      | 0 1 0 1 0 1 0 1 0
Access Bit    | 1 0 1 0 1 0 1 0 1
Misses        | M M M   M   M   M
--------------|------------------
Thread 2      | x y z x y z x y z
Hint Bit      | 0 1 0 1 0 1 0 1 0
Access Bit    | 1 0 1 0 1 0 1 0 1
Misses        | M M M   M   M   M
```

Two threads, each accessing three elements and using two-element cache. Best per thread and overall cache utilization --- 50% miss rate for each program.

Summary

- Many techniques of non-LRU management
  - most are LRU-MRU variations
  - collaborative caching coined by Wang et al. PACT 2002
- LRU-MRU cache properties
  - optimality
  - inclusion
  - Gu distance
- Compiler cache hint insertion
  - Beyls and D'Hollander, JSA 2004