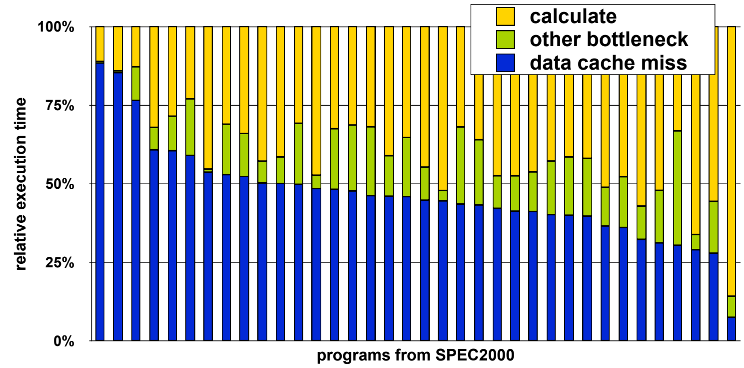
Program Interaction on Shared Cache

Theory and Applications

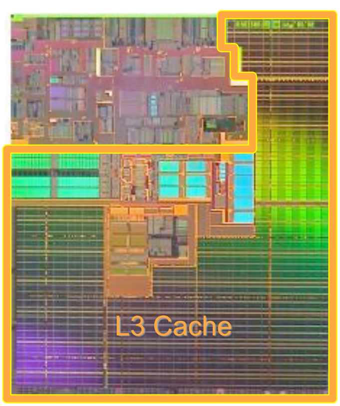
Chen Ding

Professor
Department of Computer Science
University of Rochester

Illustration: bottlenecks of SPEC2000 on Itanium1



Anant Aggarwal, MIT 6.975, 2007



Madison Itanium 2
2002

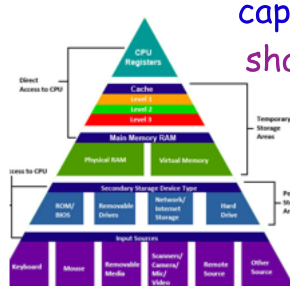
"Nothing travels faster than the speed of light ..." Douglas Adams

key problems:
latency/bandwidth
capacity
sharing

Matthew Hertz's beer

Trishul Chilimbi's cliff

Chen's Platform

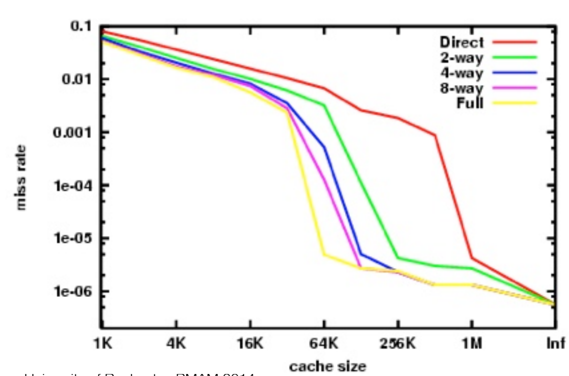


http://cse1.net/

Article Discussion Read Edit View history Search

CPU cache

From Wikipedia, the free encyclopedia



http://en.wikipedia.org/wiki/File:Cache.missrate.png

Cache Performance for SPEC CPU2000 Benchmarks

Version 3.0

May 2003

Jason F. Cantin
Department of Electrical and Computer Engineering
1415 Engineering Drive
University of Wisconsin-Madison
Madison, WI 53706-1691
jcantin@ece.wisc.edu
http://www.jfred.org

Mark D. Hill
Department of Computer Science
1210 West Dayton Street
University of Wisconsin-Madison
Madison, WI 53706-1685
markhill@cs.wisc.edu
http://www.cs.wisc.edu/~markhill

http://www.cs.wisc.edu/multifacet/misc/spec2000cache-data

```
--------------------------------------------------------------------------
| D-cache misses/inst: 1,197,717,058,456 data refs (0.34534--/inst);      |
|-------------------------------------------------------------------------|
| 782,173,506,477 D-cache 64-Byte block accesses (0.22949--/inst)         |
|-------------------------------------------------------------------------|
| Size  |   Direct    |  2-way LRU  |  4-way LRU  |  8-way LRU  |  Full LRU |
|-------+-------------+-------------+-------------+-------------+-----------|
|   1KB | 0.0890418-- | 0.0762018-- | 0.0699370-- | 0.0657938-- | 0.0652996-- |
|   2KB | 0.0651636-- | 0.0533596-- | 0.0486152-- | 0.0462573-- | 0.0453232-- |
|   4KB | 0.0480381-- | 0.0386862-- | 0.0353534-- | 0.0337222-- | 0.0325938-- |
|   8KB | 0.0362358-- | 0.0290652-- | 0.0264135-- | 0.0254564-- | 0.0245702-- |
|  16KB | 0.0277699-- | 0.0227735-- | 0.0211365-- | 0.0204821-- | 0.0196992-- |
|  32KB | 0.0223409-- | 0.0190920-- | 0.0181803-- | 0.0179048-- | 0.0175964-- |
|  64KB | 0.0189635-- | 0.0166430-- | 0.0161909-- | 0.0160494-- | 0.0159076-- |
| 128KB | 0.0158796-- | 0.0147737-- | 0.0144648-- | 0.0143748-- | 0.0142985-- |
| 256KB | 0.0138840-- | 0.0131826-- | 0.0130735-- | 0.0130274-- | 0.0130001-- |
| 512KB | 0.0119997-- | 0.0115157-- | 0.0114489-- | 0.0114018-- | 0.0113629-- |
|   1MB | 0.0096151-- | 0.0094354-- | 0.0092640-- | 0.0093510-- | 0.0093828-- |
--------------------------------------------------------------------------
Compulsory: 0.0000150365--
```

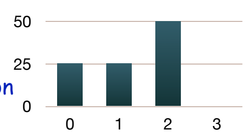
```
Benchmarks:   12
Sim Time: 1463.66 days, 4.007 years

File created 5/23/2003.
```


Program Locality

Reuse Distance

A Metric and A Tool Box

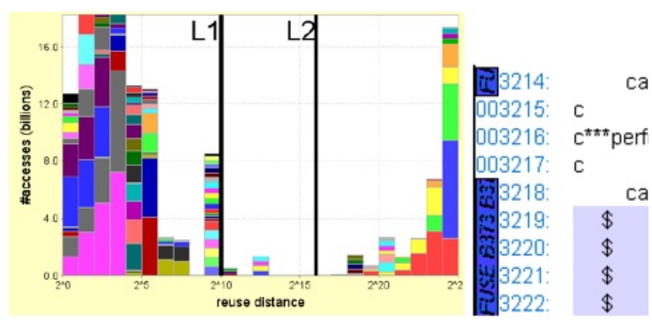
- Reuse distance
  - independent of coding styles, memory allocation, or hardware
  - possible to correlate between different runs
  - pattern analysis
    - aggregate or temporal
    - cross-program inputs
- Single basis for analysis/optimization
  - to analyze
    - to compose and decompose reuse distance
  - to optimize
    - to shorten long reuse distance



∞ ∞ ∞ 2 0 1 2
a b c a a c b

The SLO Tool by Beyls and D'Hollander

- SLO - Suggestions for Locality Optimizations: http://slo.sourceforge.net
- An example: 173.APPLU from SPEC 2K

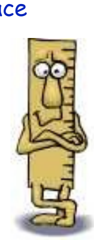


Measuring Reuse Distance

time: 1 2 3 4 5 6 7 8 9 10 11 12
access: d a c b c c g e f a f b
distance: |← 5 distinct accesses →|

(a) an example access sequence
the reuse distance between two b's is 5

- Naive counting, O(N) time per access, O(N) space
  - N is the number of memory accesses
  - M is the number of distinct data elements
- Too costly
  - N is up to 120 billion, M 25 million



Reuse Distance Measurement

Measurement algorithms since 1970	Time	Space
Naive counting	$O(N^2)$	O(N)
Trace as a stack [IBM'70]	O(NM)	O(M)
Trace as a vector [IBM'75, Illinois'02]	O(NlogN)	O(N)
Trace as a tree [LBNL'81], splay tree [Michigan'93], interval tree [Illinois'02]	O(NlogM)	O(M)
Fixed cache sizes [Winsconsin'91]	O(N)	O(C)
Approximation tree [Rochester'03]	O(NloglogM)	O(logM)
Approx. using time [Rochester'07]	O(N)	O(1)

N is the length of the trace. M is the size of data. C is the size of cache.

ACM DIGITAL LIBRARY

Program locality analysis using reuse distance

Full Text: Pdf Buy this Article

2009 Article
- Research
- Refereed

Authors: Yutao Zhong George Mason University, Fairfax, VA
Xipeng Shen The College of William and Mary, Williamsburg, VA
Chen Ding University of Rochester, Rochester, NY

Published in:
· Journal
ACM Transactions on Programming Languages and Systems (TOPLAS) TOPLAS Homepage archive
Volume 31 Issue 6, August 2009
ACM New York, NY, USA
table of contents doi>10.1145/1552309.1552310

Analysis Speed

benchmarks	length	data size (64B lines)	unmodifed time (sec)	FP alg time	FP alg cost (X)	RD alg time	RD alg cost (X)	LF alg time	LF alg cost (X)
176.gcc	1.10E+10	3.99E+06	85.1	345	4.1	2,392	28.1	5,489	65
181.mcf	1.88E+10	2.52E+06	398	1,126	2.8	10,523	26.4	121,818	306
164.gzip	2.00E+10	1.41E+06	150	501	3.3	5,823	38.8	44,379	296
252.eon	2.51E+10	1.54E+04	77.4	503	6.5	5,950	76.9		
256.bzip2	3.20E+10	1.47E+06	173	726	4.2	7,795	45.1	36,428	211
175.vpr	3.56E+10	5.08E+04	210	964	4.6	13,654	65.0	51,867	247
186.crafty	5.31E+10	3.20E+04	75.5	1,653	21.9	18,841	249.5	117,473	1,556
300.twolf	1.08E+11	9.47E+04	368	2,979	8.1	27,765	75.4	155,793	423
197.parser	1.22E+11	6.52E+05	230	3,122	13.6	35,562	154.6	106,198	462
11 2K INT avg	**4.73E+10**	**1.14E+06**	**196**	**1,324**	**8**	**14,256**	**84**	**79,931**	**446**
179.art	1.20E+10	5.93E+04	591	734	1.2	4,032	6.8	36,926	62

47 billion accesses

3m16s

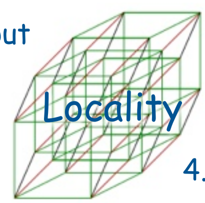
3h57m

reference affinity [PLDI'04, ICS'05, POPL'06]

2. Data

whole-program locality [PLDI'03, PACT'03, LACSI'03, TOC'07, TOPLAS'09]

1. Input



3. Code

program opt and tuning [JPDC'04, ISMM'09, ISMM'11, ISMM'12]

4. Time

locality phases, dynamic opt [PLDI'99, ASPLOS'04, ExpCS'07, JPDC'07]

5. Environment

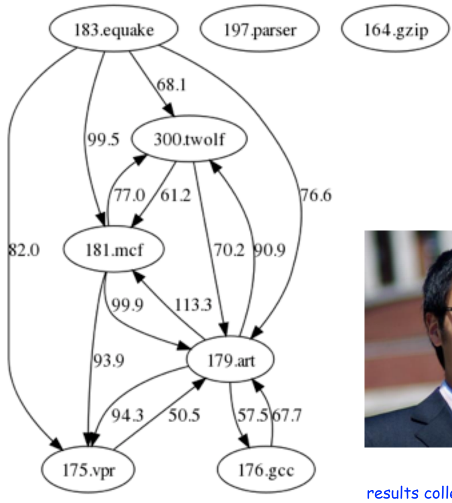
data, cache, and memory sharing [ISMM'06, PPOPP'11, PACT'11, CCGrid'12, CGO'13, ASPLOS'13]

Active Sharing (now)

The End of Cache Monopoly

- Multicore
  - desktop, cloud, and handheld
- Multicore cache
  - a mixture of private/shared caches
    - Intel Nehalem 256KB private L2, 4MB to 8MB shared L3
    - IBM Power 7 256KB private L2, 32MB shared ERAM L3
    - ERAM to appear on Intel processors
- New problems
  - available cache resource is variable
    - not the full size, not constant size
  - not just performance but also stability
  - not just parallel program but also sequential program

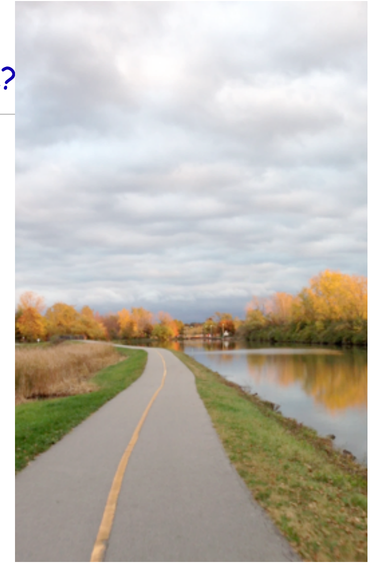
The End of Cache Monopoly (by Henry Kautz)



results collected by Bin Bao

Old Wine in New Bottle?

- Time sharing systems (Multics)
  - memory sharing
  - well studied and solved
    - routine by modern OS
- Cache sharing is more complex
  - hardware managed
    - coffee cup analogy
  - levels, private/shared
  - more frequent access
    - content wiped out in 1ms
  - can't buy more cache
  - asymmetry/circular feedback



rd = 5

program 1 a b c d e f a

ft = 4

program 2 k m m m n o n

rd' = rd+ft = 9

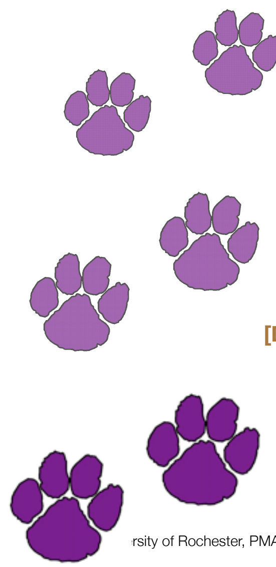
program 1&2 a k b c m d m e m f n o n a

- Private cache locality

  P(capacity miss by me) =
  P(my reuse distance >= cache size)

- Shared cache locality

  P(capacity miss by me) =
  P(my reuse distance + peer footprint >= cache size)

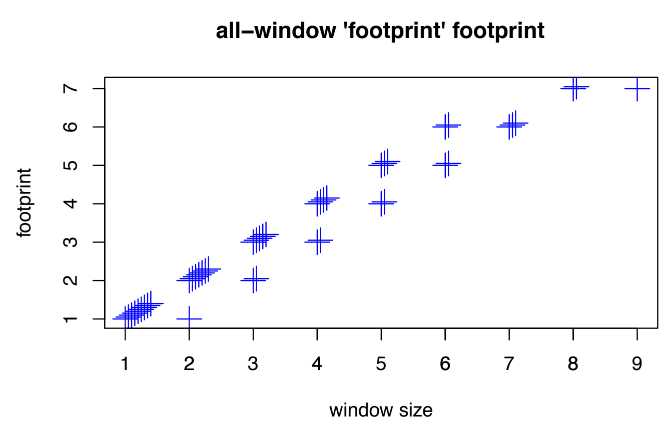


Footprint Locality

[Ding, Xiang, et al. PPOPP 2008/11, PACT 11, ASPLOS 13]

Footprint

- Example: "abbb"
  - 3 length-2 windows: "ab", "bb", "bb"
    - footprints 2, 1, 1
  - the average fp(2) = (2 + 1 + 1)/3 = 4/3



Footprint Measurement 1972 - 2007

- Working set
  - limit value in an infinitely long trace [Denning & Schwartz 1972]
- Direct counting
  - single window size [Thiebaut & Stone TOCS'87]
    - seminal paper on footprints in shared cache
  - same starting point [Agarwal & Hennessy TOCS'88]
- Statistical approximation
  - [Denning & Schwartz 1972; Suh et al. ICS'01; Berg & Hagersten PASS'04; Chandra et al. HPCA'05; Shen et al. POPL'07]
  - level of precision couldn't be directly checked
- No precise definition/solution for all windows
  - can't be measured for real
  - can't know the accuracy of an estimate

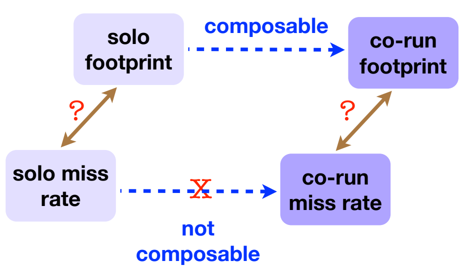
Footprint Measurement 2008 - 2013

- Footprint distribution
  - all-window enumeration [Ding/Chilimbi PPOPP 2008]
    - max/min/median/percentiles
  - trace compression [Xiang+ PPOPP 11]
    - 70X speedup
    - 4 hours per program
- Average footprint [Xiang+ PACT 11]
  - Xiang formula
    - 22 minutes per program
- Footprint Sampling [Xiang+ ASPLOS 13]
  - shadow profiling
  - 0.5%

Xiaoya Xiang

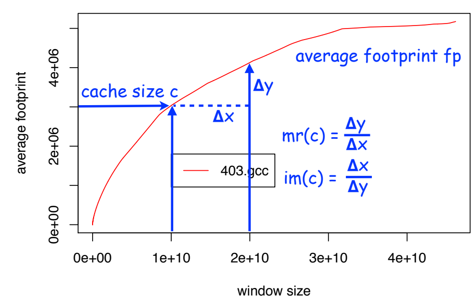
- HUST BS 2005
- ICT MS 2008
- Rochester PhD (expected)
- Twitter 2013

组合性



Footprint to Miss Rate Conversion

- Δx
  - average time for Δy misses
- $\frac{\Delta y}{\Delta x}$
  - miss rate at size c



Conversion Formulas

The Xiang formula for average footprint [PACT'11]

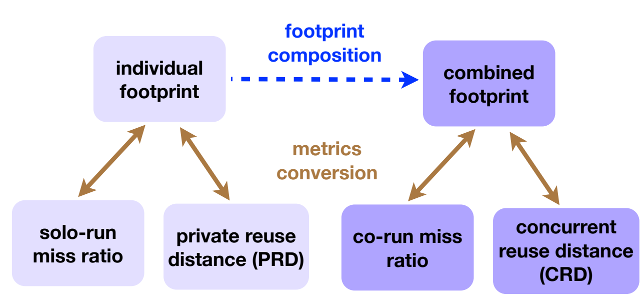
- **rt: reuse time**
- **m: data size**
- **n: trace length**

$$fp(x) \approx m - \sum_{k=x+1}^{n-1}(k-x)P(rt=k)$$

$$mr(c) = mr(fp(x)) = \boxed{\frac{fp(x+\Delta x)-fp(x)}{\Delta x}}$$

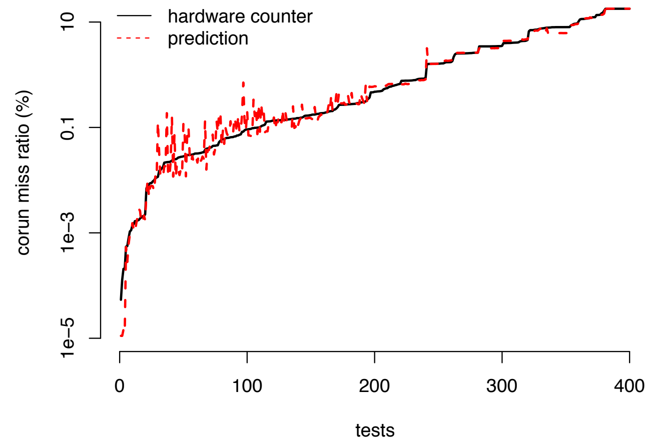
$$P(rd=c) = mr(c-1) - mr(c)$$

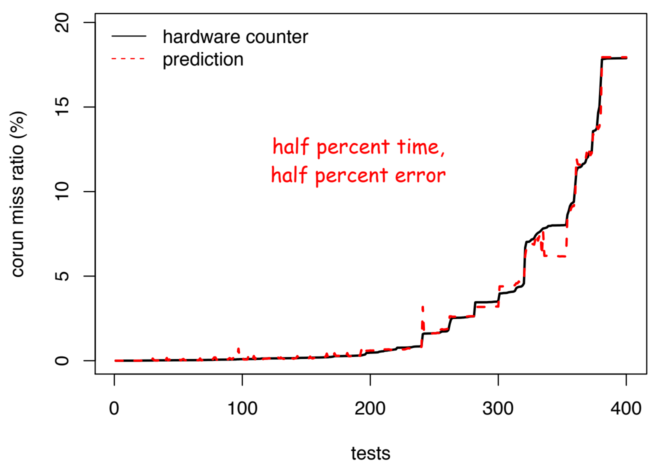
Composition + Conversion



Reality Check

- 20 SPEC 2006 programs
  - 190 different pair runs
- Modeling
  - per program footprint
  - composition
  - a few hours
    - prediction for all cache sizes
- Exhaustive parallel testing
  - 190 pair runs
  - 380 hw counter reads (OFFCORE.DATA_IN, 8MB 16-way L3)
  - ~9 days total CPU time



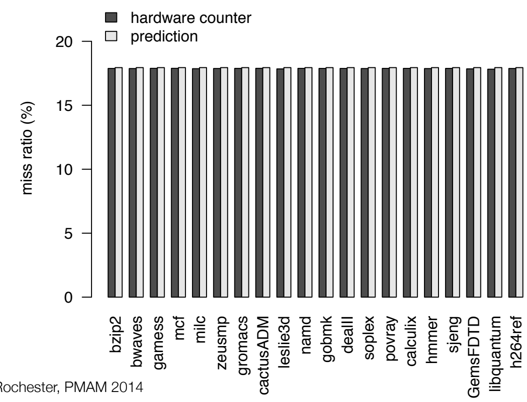


Co-run interference of libquantum;

high miss ratio, zero sensitivity;

measured miss ratio 17.82% to 17.89%,

predicted 17.94% to 17.94%

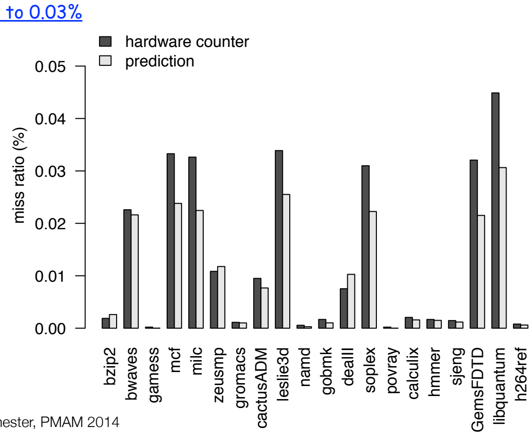


Co-run interference of gamess;

low miss ratio, high sensitivity

measured miss ratio 0.0002% to 0.04%,

predicted 0.000013% to 0.03%



Denning's Law of Locality

What's the relation between reuse frequency and footprint?

abc ... abc ...

aaa ... bbb ...

Limit value [Denning and Schwartz, CACM 1972]

Time space [Denning and Slutz, CACM 1978]

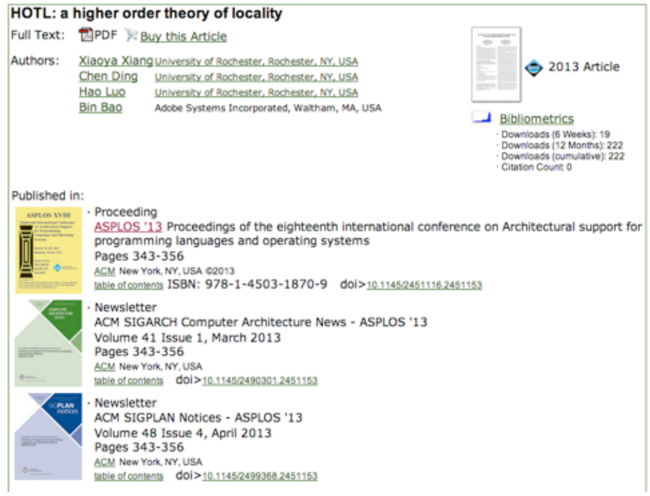
All program traces [Rochester, ASPLOS 2013]

An Old Open Question

How quickly can we measure the miss rate for all cache size?

3000+ Cache Sizes In the analysis, the footprint and reuse distance numbers are bin-ed using logarithmic ranges as follows. For each power-of-two range, we sub-divide it into 256 equal-size increments. As a result, we can predict the miss ratio not just for power-of-two cache sizes, but 3073 cache sizes between 16KB and 64MB.

Xiang et al. ASPLOS 13
(Tongxin Bai's tool)


HOTL: a higher order theory of locality

Full Text: PDF Buy this Article

Authors: Xiaoya Xiang University of Rochester, Rochester, NY, USA
Chen Ding University of Rochester, Rochester, NY, USA
Hao Luo University of Rochester, Rochester, NY, USA
Bin Bao Adobe Systems Incorporated, Waltham, MA, USA

2013 Article

Bibliometrics
- Downloads (6 Weeks): 19
- Downloads (12 Months): 222
- Downloads (cumulative): 222
- Citation Count: 0

Published in:

- Proceeding
ASPLOS '13 Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems
Pages 343-356
ACM New York, NY, USA ©2013
table of contents ISBN: 978-1-4503-1870-9 doi>10.1145/2451116.2451153

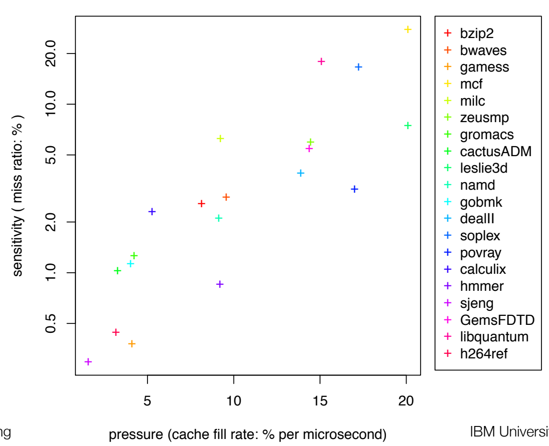
- Newsletter
ACM SIGARCH Computer Architecture News - ASPLOS '13
Volume 41 Issue 1, March 2013
Pages 343-356
ACM New York, NY, USA
table of contents doi>10.1145/2490301.2451153

- Newsletter
ACM SIGPLAN Notices - ASPLOS '13
Volume 48 Issue 4, April 2013
Pages 343-356
ACM New York, NY, USA
table of contents doi>10.1145/2499368.2451153

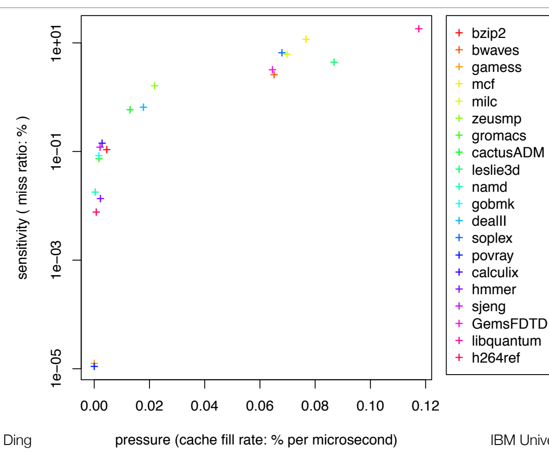
An Old Open Question

What's the relation between miss rate and cache pressure? Does a higher miss rate mean higher pressure?

Miss Ratio vs Pressure, 32KB Cache



Miss Ratio vs Pressure, 4MB Cache



An Old Open Question

Is there a machine independent way to compare program behavior in shared cache? How do programs in different domains differ?

An Old Open Question

Does LRU cache produce optimal partition? [Thiebuat and Stone, 1992]

The second type of sharing happens between the instruction and the data of a program. Stone et al. [1992] investigated whether LRU produces the optimal allocation. Assuming that the miss rate functions for instruction and data are continuous and differentiable, the optimal allocation happens at the points "when miss-rate derivatives are equal" [Thiébaut and Stone, 1992]. The miss rate functions, one for instruction and one for data, were modeled instead of measured. The authors showed that LRU is not optimal, but left open a question whether there is a bound on how close LRU allocation is to opti... model in Chapter 4 can be used to compute the ... answer the open question for any group of program...

Collaborative Rationing

```
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.

Jacob Brock and Raj Parihar

Optimal Collaborative Caching: Theory and Applications

by

Xiaoming Gu

Submitted in Partial Fulfillment

of the

Maximal cache performance?
Answer:

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

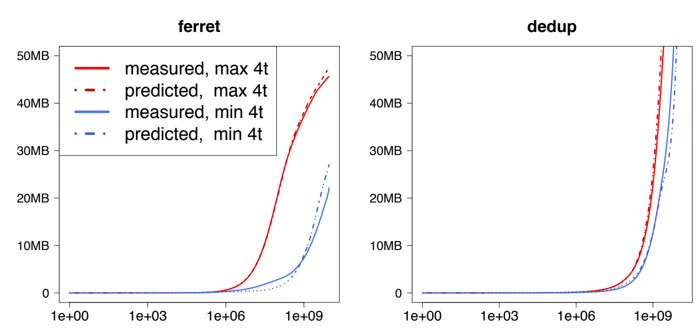
Rochester, New York

2013

On-going Studies

Shared Footprint Analysis

with Hao Luo and Pengcheng Li



All thread-group locality prediction. Min/max locality in all 70 four-thread groups for two PARSEC programs with 8 asymmetric threads.

DEPARTMENT OF COMPUTER SCIENCE

Peer-Aware Program Optimization

Bin Bao

Advisor: Chen Ding

Recent Developments

- Competitiveness, politeness, sensitivity
  - Jiang et al. [TPDS'11, HiPEAC'10]
- Intensity and sensitivity
  - Zhuravlev et al. [ASPLOS'10]
- Niceness, pressure and sensitivity
  - Mars et al. [CGO'12, Micro'12]
- Interference of cache
  - composable models [Stone+ TOCS'87/TOC'92; Suh+ ICS'01; Chandra+ HPCA'05; Xiang+ PPOPP'11/PACT'11/ASPLOS'13]
  - threaded code [Ding/Chilimbi MSR'09, Jiang+ CC'10/TPDS'12, Schuff+ PACT'10, Wu/Yeung PACT'11/ISCA'13]
- Interference model of execution time/speed
  - bubble-up [Mars+ Micro'12, ISCA'13]
  - QoS-aware scheduling [Delimitrou/Kozyrakis ASPLOS'13]

Recent Developments [cont'd]

- Parallel reuse distance measurement
  - cluster [OSU, IPDPS 2012]
  - GPU [ICT and NCSU, IPDPS 2012]
  - sampling
    - footprint shadow sampling [Rochester, ASPLOS 2013]
    - multicore reuse distance [Purdue, PACT 2010]
    - reuse distance sampling [Chang & Zhong, PACT 2008]
- Reuse distance in threaded code
  - multicore reuse distance [Purdue, PACT 2010]
  - CRD/PRD scaling [Maryland, ISCA 2013, to appear]

Recent Developments (cont'd)

- Asymptotic locality effect in parallel algorithms
  - Leslie Valiant, PACT 2011 keynote
  - Guy Blelloch et al. CMU, MIT, Intel Labs Pittsburgh [MSPC 2013]
  - Morris Herlihy and student, [PPOPP 2014]
- Shared footprint [Rochester, WODA 2013]
- Static reuse distance analysis in Matlab [Indiana, ICS 2010]
- Static footprint analysis [Rochester, CGO 2013]
  - peer-aware program optimization [Bao, dissertation'13]
- Collaborative caching
  - practical uses [UT, Ghent, Google etc]
  - optimal collaborative LRU cache [Gu, ISMM'11/12/13, dissertation'13]

Summary

- Program interaction in multicore
  - data sharing in threaded code
  - cache and memory bandwidth sharing by all programs
- Locality theory
  - working set, footprint, shared footprint
  - metrics composition and conversion
  - higher order theory of cache locality (HOTL)
- Recent research
  - locality in parallel algorithms
  - peer-aware program optimization
  - sharing conscious task scheduling
  - collaborative caching