

Practical Cache Performance Modeling for Computer Architects

Yan Solihin, NCSU, solihin@ncsu.edu
Fei Guo, NCSU, fguo@ncsu.edu
Thomas Puzak, IBM, tpuzak@us.ibm.com
Phil Emma, IBM, pemma@us.ibm.com

Computer Performance Modeling

- Goal: Estimating and understanding the performance of computer systems
- **Low-Level Models**
 - Various levels of details: functional, trace, cycle-accurate, etc.
 - **Pros**
 - Versatile: adaptable to new architectures and workload
 - Details: can embed very detailed statistics
 - Popular: SimpleScalar, SIMICS widely used
 - **Cons**
 - $O(n)$ overhead, scales with workload size: each instruction/event is simulated to capture its effect
 - Slow: realistic workload has many billions of instructions

2

Computer Performance Modeling

- **High-Level Models**
 - Purpose: Evaluating gross trade-offs of designs
 - **Pros**
 - Short execution time, and sometimes $O(1)$
 - Requires little coding
 - Reveal basic relationships of variables
 - May reveal non-obvious trends and insights
 - **Cons**
 - Less versatile
 - Requires performance modeling expertise
 - Uses:
 - Early design cycle: for pruning design search space
 - Entire design cycle: to re-evaluate design search space if requirements change

3

Modeling Methods

- **White Box**
 - Model incorporates knowledge about the system (e.g. relationships of parameters are known a priori)
 - Analytical or heuristics-based
 - Pros: models reveal insights & explain “why”, no training required
 - Cons: problem-specific solution
- **Black Box**
 - Model learns knowledge about the system
 - AI-based: neural networks, decision tree, curve fitting, etc.
 - Pros: can model complex problem/system
 - Cons: prediction without insights, requires training

4

Focus of this tutorial

- Types:
 - High Level, Hybrid High/Low level
- A priori knowledge:
 - White Box
- Scope:
 - Miss count & rate
 - Miss cost
 - Bandwidth usage

5

Program

- 8:30 – 8:45: Introduction
- 8:45 – 9:00: Capturing temporal locality behavior
- 9:00 – 9:30: Modeling cache sharing
- 9:30 – 10:00: Modeling cache replacement policy
- 10:00 – 10:30: Coffee Break
- 10:30 – 11:30: Analysis of the effects of miss clustering on the cost of a cache miss
- 11:30 – 12:30: Interaction of Caching and Bandwidth Pressure

6

Capturing Temporal Locality Behavior

Temporal Locality Behavior

- Programs exhibit locality of references
- **Spatial locality:** the neighbor of recently-accessed data tends to be accessed in the near future
- **Temporal locality:** recently-accessed data tends to be accessed again (or *reused*) in the near future
- **Significance:** *temporal reuse & cache parameters determine all non-cold misses*
 - If each memory block is accessed exactly once, we only have cold misses
 - Cold misses affected by block size
- How can we capture temporal locality?

8

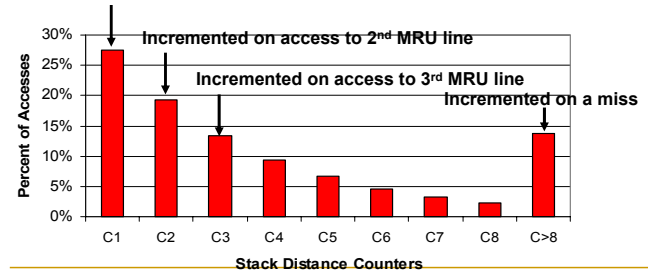
Stack Distance Profiling [Mattson'70]

- Early attempt to capture temporal reuse behavior
- Models LRU stack with a counter for each stack position
- Example: fully associative cache with 8-entry stack
 - C1: incremented whenever the MRU block is accessed
 - C2: incremented whenever the 2nd MRU block is accessed
 - C3: incremented whenever the 3rd MRU block is accessed
 - ...
 - C8: incremented whenever the 8th MRU (or LRU) block is accessed
 - C>8: incremented whenever the 9th, 10th, ... block is accessed

9

Typical Shape

- Empirical observation \Rightarrow Geometric or exponential sequence
 - Due to temporal locality
 - $C_{i+1} = C_i \times r$, where $0 < r < 1$ is the common ratio
- Incremented on access to MRU line



10

Stack Distance Properties

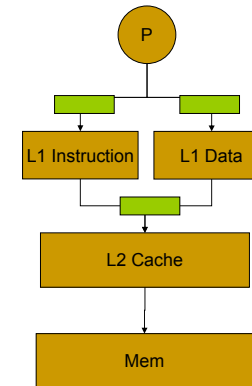
- For fully-associative LRU cache with A blocks, the number of misses of the cache is
- For A-way set associative LRU cache, we can collect set-specific stack distance profile, and the number of misses of the set is:
- Alternatively, keep per-set stacks, but use global set of counters

$$Misses = \sum_{i=A+1}^{\infty} C_A$$

$$Misses = \sum_{i=A+1}^{\infty} C_A$$

11

Where to Profile



For capturing temporal reuse patterns at the L1 cache levels
 \Rightarrow Predict cache misses for various L1 cache configurations

For capturing temporal reuse patterns at the L2 cache levels
 \Rightarrow Predict cache misses for various L2 cache configurations

12

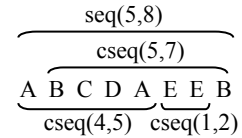
Limitations of Stack Distance Profile

- Useful only for predicting cache misses across different cache associativities
- For other purposes, we need to capture temporal reuse patterns in greater details
- So, use **Circular Sequence Profile** [Chandra'05]
 - Extends stack distance profiling
 - Counts the occurrence of $cseq(d,n)$

13

Definitions

- $seq(d,n)$ = sequence of n accesses to d distinct addresses (in a cache set)
- $cseq(d,n)$ (circular sequence) = a sequence in which the first and the last accesses are to the same address



14

Relationship with Stack Distance Profile

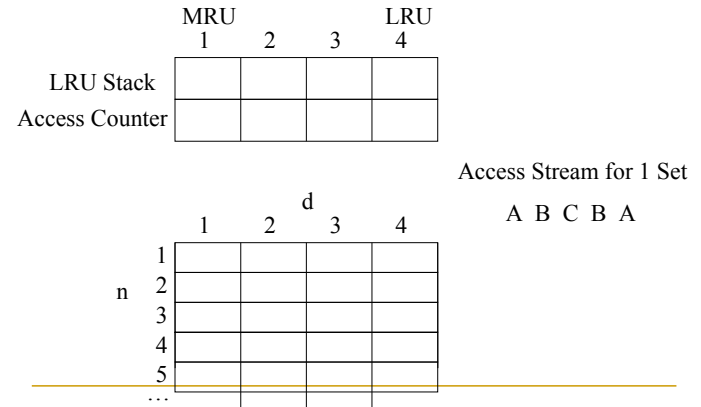
- C_x = number of circular sequences $cseq(d=x, n=\text{any value})$, or

$$C_x = \sum_{n=x}^{\infty} cseq(x, n)$$

- Hence, stack distance profile is a subset of circular sequence profile

15

Collecting Circular Sequence Profile



16

Collecting Circular Sequence Profile

	MRU			LRU
	1	2	3	4
LRU Stack	A			
Access Counter	1			

Access Stream for 1 Set

	1	2	3	4
n				
1				
2				
3				
4				
5				
...				

A B C B A

17

Collecting Circular Sequence Profile

	MRU			LRU
	1	2	3	4
LRU Stack	B	A		
Access Counter	1	2		

Access Stream for 1 Set

	1	2	3	4
n				
1				
2				
3				
4				
5				
...				

A B C B A

18

Collecting Circular Sequence Profile

	MRU			LRU
	1	2	3	4
LRU Stack	C	B	A	
Access Counter	1	2	3	

Access Stream for 1 Set

	1	2	3	4
n				
1				
2				
3				
4				
5				
...				

A B C B A

19

Collecting Circular Sequence Profile

	MRU			LRU
	1	2	3	4
LRU Stack	C	B	A	
Access Counter	2	3	4	

Found a Circular Sequence!

Access Stream for 1 Set

	1	2	3	4
n				
1				
2				
3		1		
4				
5				
...				

A B C B A

20

Collecting Circular Sequence Profile

	MRU 1	2	3	LRU 4
LRU Stack	B	C	A	
Access Counter	1	2	4	

Access Stream for 1 Set

	1	2	d	3	4
1					
2					
3					
4		1			
5					
...					

A B C B A

21

Collecting Circular Sequence Profile

	MRU 1	2	3	LRU 4
LRU Stack	B	C	A	
Access Counter	2	3	5	

**Found a
Circular Sequence!**

Access Stream for 1 Set

	1	2	d	3	4
1					
2					
3					
4		1			
5				1	
...					

A B C B A

22

Collecting Circular Sequence Profile

	MRU 1	2	3	LRU 4
LRU Stack	A	B	C	
Access Counter	1	2	3	

Access Stream for 1 Set

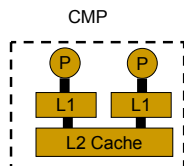
	1	2	d	3	4
1					
2					
3					
4		1			
5				1	
...					

A B C B A

23

Predicting Contention Across Cache

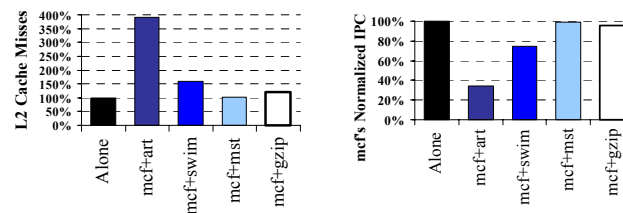
Shared Cache Challenge



- In today's CMP, L2 cache is shared by multiple cores
- Applications on different core compete for L2 cache space

25

Impact of Cache Space Contention



- **Application-specific**
- **Coschedule-specific**
- **Significant:** Up to 4X cache misses, 65% IPC reduction

How to model the impact of cache sharing?

26

Modeling Goal

- Given n applications, predict the miss rates of any pair of applications
- Input:
 - Behavior of each application
 - Cache parameter
 - Relative speed when the pair runs together
- Output:
 - Number of cache misses for each application in the pair

27

Assumptions

- LRU Replacement Algorithm
- Applications share nothing
 - Mostly true for sequential apps (except for library and OS code)
- Applications not similar
 - Parallel apps: threads likely to show uniform behavior, so predicting their miss rates is trivial

28

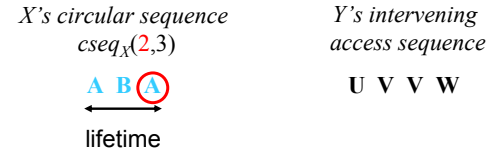
Circular Sequence Properties

- Thread X runs alone in the system:
 - Given a circular sequence $cseq_X(d_X, n_X)$, the last access is a cache miss iff $d_X > Assoc$
- Thread X shares the cache with thread Y:
 - During $cseq_X(d_X, n_X)$'s lifetime if there is a sequence of intervening accesses $seq_Y(d_Y, n_Y)$, the last access of thread X is a miss iff $d_X + d_Y > Assoc$

29

Example

- Assume a 4-way associative cache:

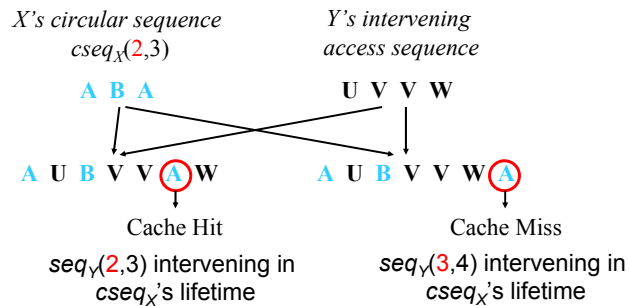


No cache sharing: A is a cache hit
Cache sharing: is A a cache hit or miss?

30

Example

- Assume a 4-way associative cache:



31

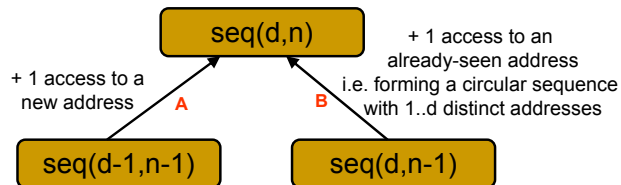
Inductive Probability Model

- Define $P_{miss}(cseq_X)$ = probability of the last access is a cache miss
- For each $cseq_X(d_X, n_X)$ of thread X
 - Compute the number of intervening accesses from thread Y during $cseq_X$'s lifetime \Rightarrow denote as n_Y
 - It is possible to have $d_Y = 1, 2, \dots, n_Y \Rightarrow$ compute the probability of each d_Y , denoted as $P(seq(d_Y, n_Y))$.
 - For each $d_Y = 1, 2, \dots, n_Y$
 - If $d_Y + d_X > Assoc$, $P_{miss}(cseq_X) = P_{miss}(cseq_X) + P(seq(d_Y, n_Y))$
 - If $d_Y + d_X \leq Assoc$, $P_{miss}(cseq_X)$ is kept the same
 - Misses = old_misses + $\sum P_{miss}(cseq_X) \times F(cseq_X)$

32

Computing $P(seq(d_Y, n_Y))$

- Basic Idea:



- This is a Markov process with 3 states, and 2 edges
- $P(seq(d, n)) = A * P(seq(d-1, n-1)) + B * P(seq(d-1, n))$

$$B = \frac{\sum_{i=1}^d C_i}{\sum_{i=1}^{\infty} C_i} \quad \text{and} \quad A = 1 - B$$

33

Overall Formula

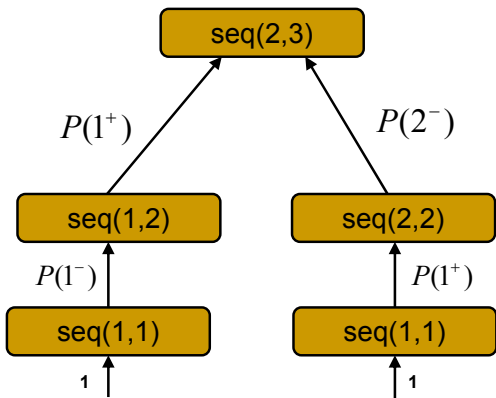
- Define:
$$P(d^-) = \frac{\sum_{i=1}^d C_i}{\sum_{i=1}^{\infty} C_i} \quad \text{and} \quad P(d^+) = 1 - P(d^-)$$

- $P(seq(d, n))$ is computed by:

$$P(seq(d, n)) = \begin{cases} 1 & d = n = 1 \\ P((d-1)^+) \times P(seq(d-1, n-1)) & d = n > 1 \\ P(1^-) \times P(seq(1, n-1)) & n > d = 1 \\ P(d^-) \times P(seq(d, n-1)) & n > d > 1 \\ + P((d-1)^+) \times P(seq(d-1, n-1)) & \end{cases}$$

34

Example



35

Final prediction

- After we obtain $P_{miss}(cseq_X(d_X, n_X))$ for all $cseq_X(d_X, n_X)$,
- Predict the total misses for thread X:

$$miss_X = oldmiss_X + \sum_{d_X=1}^A P_{miss}(cseq_X(d_X, \bar{n}_X)) \times C_{d_X}$$

36

Observations

- Based on how vulnerable to cache sharing impact:
 - Some are highly vulnerable
 - Some are not vulnerable
 - Many are somewhat / sometimes vulnerable
- Insights:
 - Traditional characterizations: not indicative of impact of sharing
 - Low vs. High IPC
 - Int vs. Floating-Point
 - High Miss Rate vs. Low Miss Rate
 - Rather, interaction of temporal reuse behavior determine impact of cache sharing

37

Modeling Replacement Policy Performance

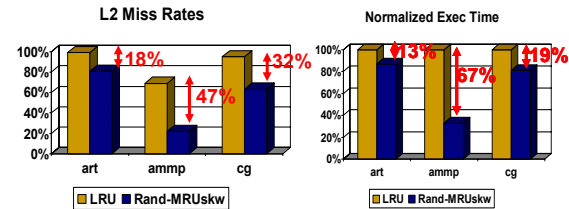
Motivation

- Cache design critical to performance
 - Memory wall: cache miss cost hundreds of processor cycles
 - Capacity pressure: Multi-core design, Virtual Machine
- Important parameters: size, associativity, block size, and **replacement policy**

39

Motivation

- Performance variation due to replacement policy is significant



- No agreement on the “best implementation”
 - Intel Pentium: LRU
 - Intel XScale: FIFO
 - IBM Power 4: tree-based pseudo-LRU
 - Others: round robin, random, replacement hints, etc.

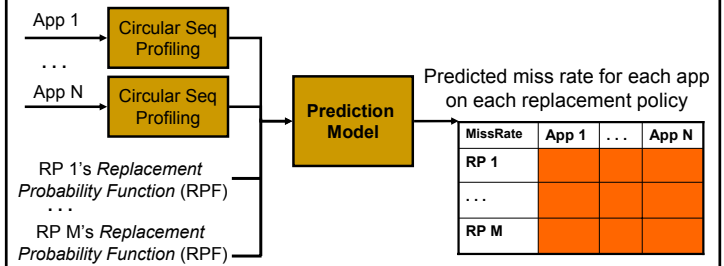
40

Motivation

- No analytical model, past models assume
 - LRU [Cascaval03, Chandra05, Ghosh97, Quong94, Sen02, Singh92, and Suh01]
 - or Random [Agarwal89, Berg04, Ladner99]
- LRU/Random simplifies modeling, but
 - Ignores performance variation due to replacement policy
 - Inaccurate for highly associative caches

41

Would be useful to model replacement policies



42

Outline

- Input of the Model
- Replacement Policy Model
- Case Study
- Conclusions

43

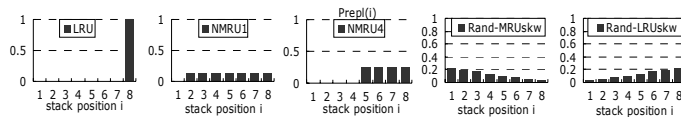
Outline

- **Input of the Model**
 - Replacement Probability Function (RPF)
 - Circular Sequence Profiles
- Replacement Policy Model
- Validation
- Conclusions

44

Replacement Prob Function (RPF)

- **RPF**, denoted as $P_{\text{repl}}(.)$ = a probability function, where $P_{\text{repl}}(i)$ is the probability that a cache block on the i^{th} stack position is replaced on a cache miss.



$P_{\text{repl}}(.)$ on 8-way assoc cache

- Stack only needed for modeling, not necessarily for hardware implementation

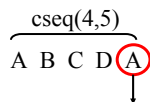
45

Outline

- Input of the Model
- Replacement Policy Model
 - Markov states
 - Markov state transitions
- Case Study
- Conclusions

46

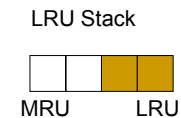
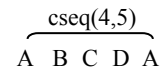
Tracking Cache Miss Probability



- Basic idea:
 - Reconstruct each circular sequence by adding each access
 - In the mean time, track if the target block is replaced
- Markov State = (d, n, p) , where
 - d = number of distinct addresses **yet to appear**
 - n = number of accesses **yet to appear**
 - p = current stack position of the target block

47

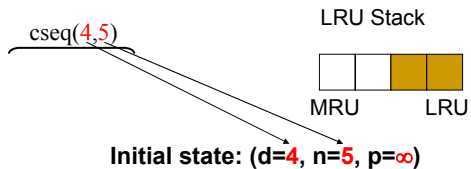
Illustration



- Assume
 - 4-way associative cache \Rightarrow 4-entry stack
 - NMRU-2 replacement policy $\Rightarrow P_{\text{repl}}(3) = P_{\text{repl}}(4) = 0.5$
- Goal
 - Compute the probability that the target access misses

48

Illustration



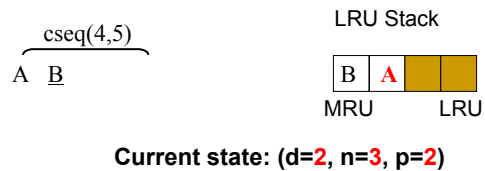
49

Illustration



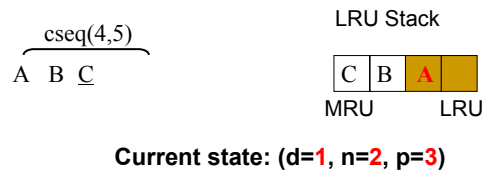
50

Illustration



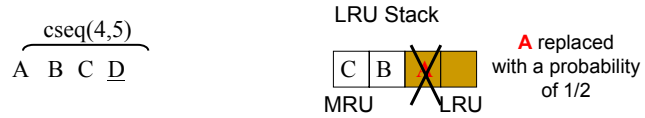
51

Illustration



52

Illustration



Current state: (d=0, n=1, p=?)

53

Illustration



Final state: (d=0, n=1, p=∞)

54

Illustration



Final state: (d=0, n=1, p=∞)

- So the probability cseq(4,5)'s target access is a cache miss is 0.5

55

Modeling Overview

- Track current state and transition probabilities into new states
- Final states:
 - Target block replaced \Rightarrow cache miss
 - $p >$ cache associativity \Rightarrow cache miss
 - End of circular sequence reached
- Accumulate probabilities of cache miss

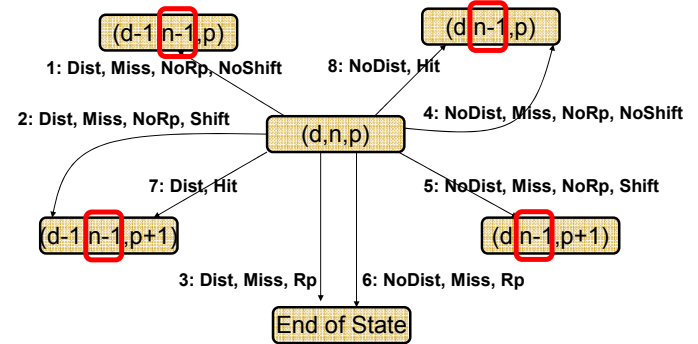
56

State Transitions

- New state depends on 8 events:
 - **Dist** and **NoDist**
 - **Dist**: the new access is to "distinct" address (not seen before in this circular sequence)
 - **Miss** and **Hit**
 - **Miss**: the new access is a cache miss
 - **Rp** and **NoRp**
 - **Rp**: the new access causes the target block to be replaced
 - **Shift** and **NoShift**
 - **Shift**: the new access causes the target block to be shifted down in the LRU stack
- Note:
 - P_{Dist} , P_{Rp} , P_{Shift} are directly computable (see [SIGMETRICS'06])
 - P_{Shift} dependent on RPF
 - P_{Miss} is the object of prediction

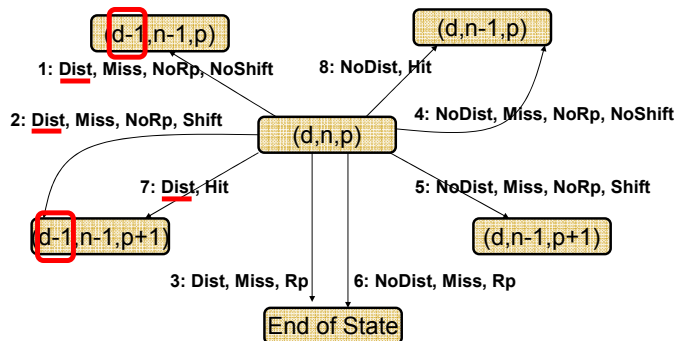
57

State Transitions Diagram



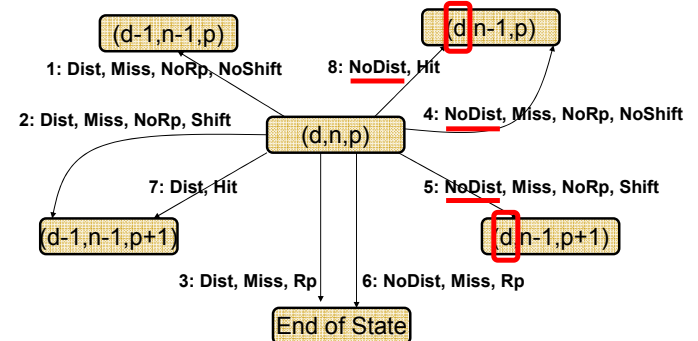
58

State Transitions Diagram



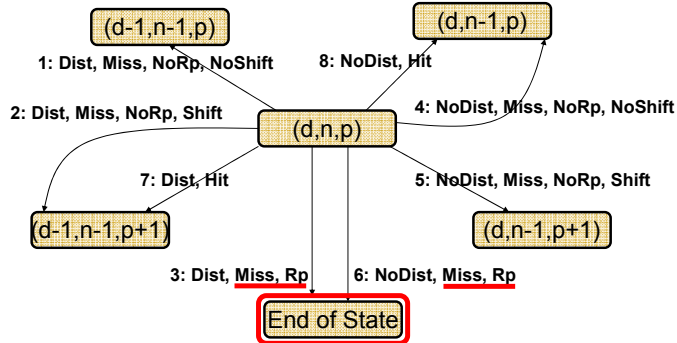
59

State Transitions Diagram



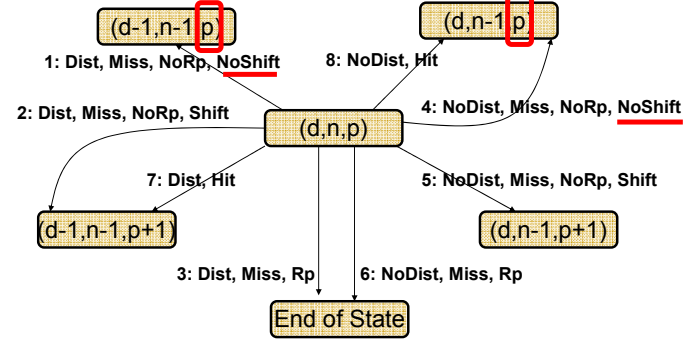
60

State Transitions Diagram



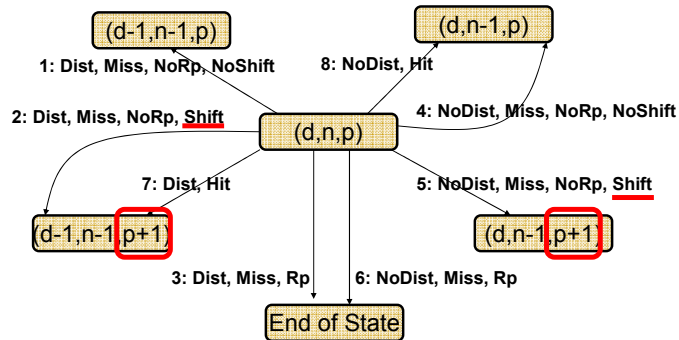
61

State Transitions Diagram



62

State Transitions Diagram



63

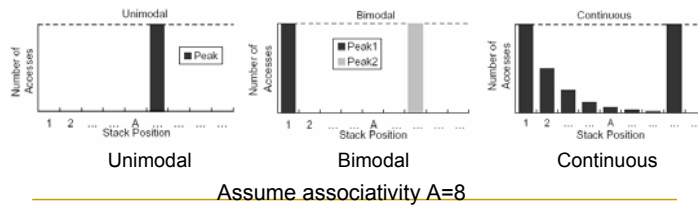
Outline

- Input of the Model
- Replacement Policy Model
- Case Study
- Conclusions

64

Case Study: Using Model Only

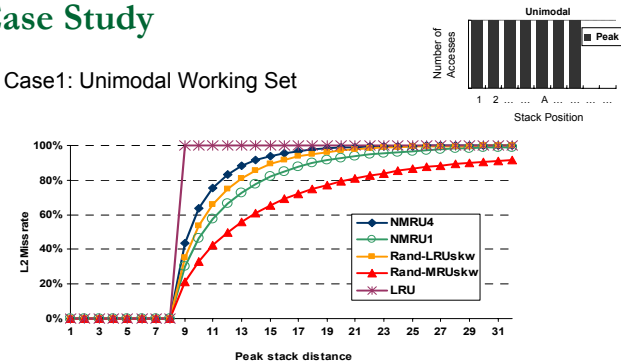
- Goal : When is LRU pathological?
 - Hard to pinpoint with simulations because many possible contributing factors
 - Isolate the impact of temporal reuse pattern
- Synthetic Stack Distance Profiles:



65

Case Study

- Case1: Unimodal Working Set

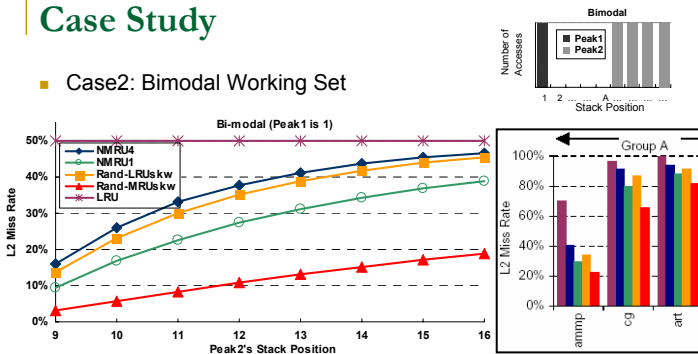


- LRU exhibits pathological performance
- Miss Rates: Rand-MRUskw < NMRU1 < Rand-LRUskw < NMRU4

66

Case Study

- Case2: Bimodal Working Set



- LRU exhibits pathological performance
- Miss Rates: Rand-MRUskw < NMRU1 < Rand-LRUskw < NMRU4
- Performance order the same as in art, ammp, cg
- In fact, those apps have approx. bimodal stack distance profiles

67

Conclusions

- Two useful modeling tools:
 - Circular sequence profiling
 - For capturing temporal reuse patterns
 - Markov processes
 - For capturing probabilities of certain cache states
 - Cache miss is an event associated with certain cache state
- Modeling can reveal non-obvious insights:
 - Cache miss rates due to shared cache space contention
 - Not capturable by simple metrics low IPC vs. high IPC, low miss rates vs. high miss rates, int vs. floating-point
 - Interaction of temporal reuse patterns of several applications
 - Choosing replacement policy
 - LRU has quite a few pathological cases
 - For apps with working set size 1-4X cache size, others policies outperform LRU

68

How to Use ACAPP

Introduction

■ Analytical Cache Performance Prediction (ACAPP) tool suite

- Prediction for different cache associativities
- Prediction for different cache replacement policies
- Prediction for cache contention when two threads share the cache
- Adding new replacement policies with user-specified RPFs

■ Input

- Circular Sequence profile of each application
 - Generated by any simulator follow certain format
 - Providing extension code for SimpleScalar

■ Released

- Available for download in <http://www.ece.ncsu.edu/arpers>

70

Prepare Input Files

```
$HOME/acapp/addOnSimpleScalar/  
acappProfiler.c  
acappProfiler.h  
cache.c  
sim-outorder.c  
Makefile
```

```
$HOME/Simplesim-3.0/
```

```
cache.c  
sim-outorder.c  
Makefile
```

```
# cd $HOME/Simplesim-3.0
```

```
# make
```

```
# ./sim-outorder swim.ref.eio -max:200  
... <benchmark output>
```

```
# ls swim_train.eio.*
```

```
-rw-r--r-- swim.ref.eio.csq
```

```
#sets 1024  
#assoc 4  
#scaling_factor 4  
#block_size 64  
#cseq  
1 2 3 4 5 6 7 8 9 10 11 23 34 29 35 35  
#stackDist  
6877989 4199671 2083871 2653828 2327123  
3051944 5588497 5034757 2996018 794764  
39391 520 387 350 383 465 32532308
```

71

Startup

./acapp -h

```
***** ACAPP TOOL HELP MENU *****  
General Usage:  
-h -- HELP MENU  
Prediction under varying cache associativity:  
-a <assoc> [<min assoc> <max assoc>] -f1 <profile1>  
Prediction under varying cache replacement policies:  
-p <rpindex> -f1 <profile1>  
-pA -f1 <profile1>  
Prediction under cache sharing:  
-c -f1 <profile1> -f2 <profile2>  
Adding new replacement policy:(require 'usr_rp.in')  
-n (default) or  
-n <dx> <nxmin> <nxmax>  
Print supported replacement policies  
-log
```

72

Prediction under varying cache associativity

acapp -a <assoc> [<min assoc>** **<max assoc>**] -f1 <profile1>**

EXAMPLE

```
#!/acapp -a 4 7 -f1 ./csq/benchmark1.csq
```

OUTPUT

The CSQ file is generated using the following cache parameters:
 Sets: 1024
 Associativity: 4
 Block size: 64
 Original miss rate: 0.768043
 Miss rate for A = 4: 0.768043
 Miss rate for A = 5: 0.733912
 Miss rate for A = 6: 0.689150
 Miss rate for A = 7: 0.607186

Note: benchmark1.csq represents the L2 profile of *swim* (ref input set).
 benchmark2.csq represents the L2 profile of *apsi* (ref input set).
 benchmark3.csq represents the L2 profile of *ammp* (ref input set).

73

Print supported replacement policies

acapp -log

EXAMPLE

```
#!/acapp -log
```

OUTPUT

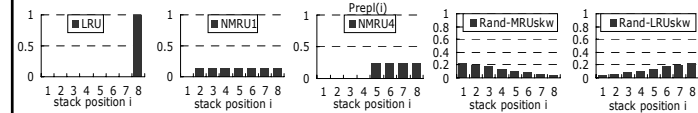
```
***** SUPPORTED REPLACEMENT POLICIES LOGFILE *****  

1 - NMRU4  

2 - NMRU1  

3 - LRUskw  

4 - MRUskw
```



74

Prediction under varying cache replacement policies

acapp -p <rpindex> -f1 <profile1>

EXAMPLE

```
#!/acapp -p 2 -f1 ./csq/benchmark1.csq
```

OUTPUT

The CSQ file is generated using the following cache parameters:
 Sets: 1024
 Associativity: 4
 Block size: 64
 Prediction Result for NMRU1
 LRU: 0.768043
 Pred: 0.739310

75

Prediction for all supported replacement policies

acapp -pA -f1 <profile1>

EXAMPLE

```
#!/acapp -pA -f1 ./csq/benchmark3.csq
```

OUTPUT

The CSQ file is generated using the following cache parameters:
 Sets: 1024
 Associativity: 8
 Block size: 64

1 - NMRU4 Prediction Result for NMRU4 LRU: 0.702653 Pred: 0.406974 *****	3 - LRUskw Prediction Result for LRUskw LRU: 0.702653 Pred: 0.376921 *****
2 - NMRU1 Prediction Result for NMRU1 LRU: 0.702653 Pred: 0.331667 *****	4 - MRUskw Prediction Result for MRUskw LRU: 0.702653 Pred: 0.201158 *****

76

Prediction under cache contention

accap -c -f1 <profile1> -f2 <profile2>

EXAMPLE

```
#!/accap -c -f1 ./csq/benchmark1.csq -f2 ./csq/benchmark2.csq
```

OUTPUT

The CSQ file is generated using the following cache parameters:

```
Sets: 1024
Associativity: 4
Block size: 64
***** RESULTS *****
```

	./csq/benchmark1.csq	./csq/benchmark2.csq
Accesses:	68182266	11805186
Predicted miss rate:	0.868560	0.338920
Original miss rate:	0.768043	0.244366

77

Adding new replacement policy

require "usr_rp.in" file:

```
#This is the configuration file of user specified replacement policy.
#Please do no change the format of this file and the NAME, ASSOC or PROB keywords. Only
#the values of each of line can be changed by the user.
NAME newRp
ASSOC 8
PROB
0.2 0.1 0.05 0.25 0.15 0.07 0.03 0.15
```

accap -n (default)

EXAMPLE

```
#!/accap -n
```

OUTPUT

```
Creating new replacement policy...
Coefficient file(s) added to: ./fine/newRp/d_0
Coefficient file(s) added to: ./fine/newRp/d_1
...
Coefficient file(s) added to: ./fine/newRp/d_11
Replacement policy: '5 - newRp' Added Successfully!
```

78

Adding new replacement policy (continue)

By using ./accap -n (default), a certain number of coefficient files that cover the most popular combinations of d and n are generated. In case some benchmarks' cseq profiles have special combinations of d and n, (reported as "Missing Coefficient files" by the tool if any). User can also generate coefficient files for certain d and n.

accap -n <dx> <nxmin> <nxmax>

EXAMPLE

```
#!/accap 5 7 9
```

OUTPUT

Adding coefficient file(s) to rp:newRp

```
Coefficient file(s) added to directory: ./fine/newRp/d_5
Coefficient files for Replacement policy: '5 - newRp' Added Successfully!
```

79

Acknowledgement

■ Researchers

- Fei Guo
- Dhruva Chandra
- Shaunak Joshi
- Seongbeom Kim

■ Funding Agency

- NSF
- Intel
- IBM

80

The Effects of Miss Clustering on the Cost of a Cache Miss

Phil Emma
Allan Hartstein
Thomas R. Puzak
Viji Srinivasan

Dept: Systems Technology and Microarchitecture
IBM T. J. Watson Research Center

Acknowledgments
Arthur Nadas
Jim Mitchell
Jane Bartik
Dan Prener
Peter Oden
Doug Logan
John Griswell
C R Attanasio
Danny Lynch
Moin Qureshi

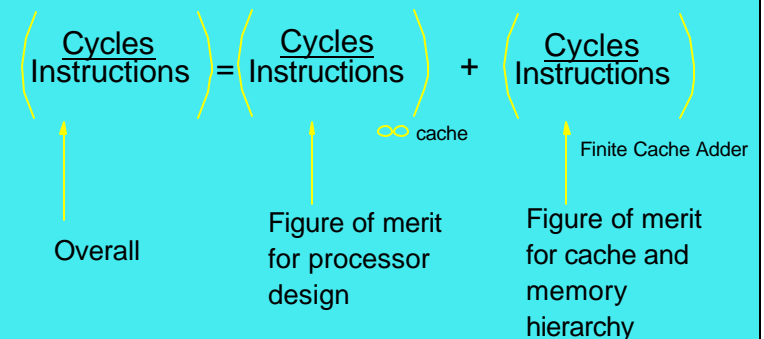
How do You Measure The Cost of a Cache Miss?

Pipeline Spectroscopy is a new technique that allows classification (analysis) of single events in a Processor's Pipeline.

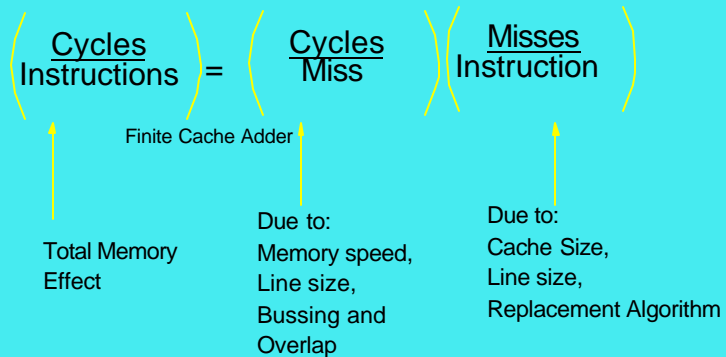
Initial focus of project was to develop a means to measure the cost of a cache miss but spectroscopy leads to much greater insight in pipeline dynamics, including effects due to cache miss behavior, prefetching, pipeline recycle, branch prediction errors, and trailing edge effects.

Cost of each miss is displayed as a Histogram. The graphs are called spectrograms because they reveal certain signature features of the processor's memory hierarchy, the pipeline, and the miss pattern itself (amount of overlap between misses in the miss cluster).

Processor Performance (cycles/instruction) has two components



Memory Hierarchy (cycles/instruction) has two components



Substituting

$$\left(\frac{\text{Cycles}}{\text{Instructions}} \right)_{\text{Overall}} = \left(\frac{\text{Cycles}}{\text{Instructions}} \right)_{\text{cache}} + \left(\frac{\text{Cycles}}{\text{Miss}} \right) \left(\frac{\text{Misses}}{\text{Instructions}} \right)_{\text{Finite Cache Adder}}$$

To Calculate Miss Penalty (Cycles/Miss)

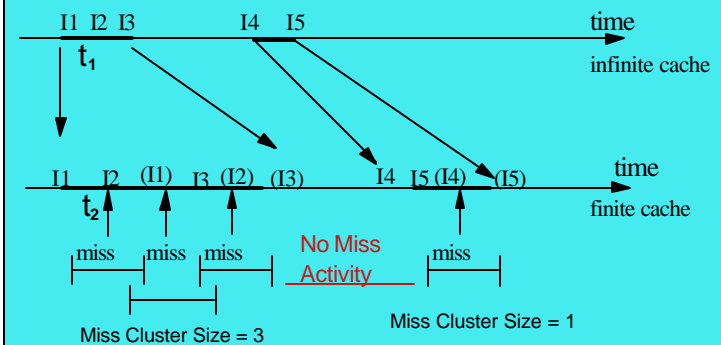
$$\frac{\left(\frac{\text{Cycles}}{\text{Instructions}} \right)_{\text{Overall}} - \left(\frac{\text{Cycles}}{\text{Instructions}} \right)_{\text{cache}}}{\left(\frac{\text{Misses}}{\text{Instructions}} \right)} = \left(\frac{\text{Cycles}}{\text{Miss}} \right)$$

Constructing Miss Spectrogram

Misses are grouped into clusters and the run time associated with the instruction sequence that 'surrounds' the miss cluster is compared to the infinite cache run time of the same instruction sequence.

The difference between these two times is used to construct the miss spectrogram

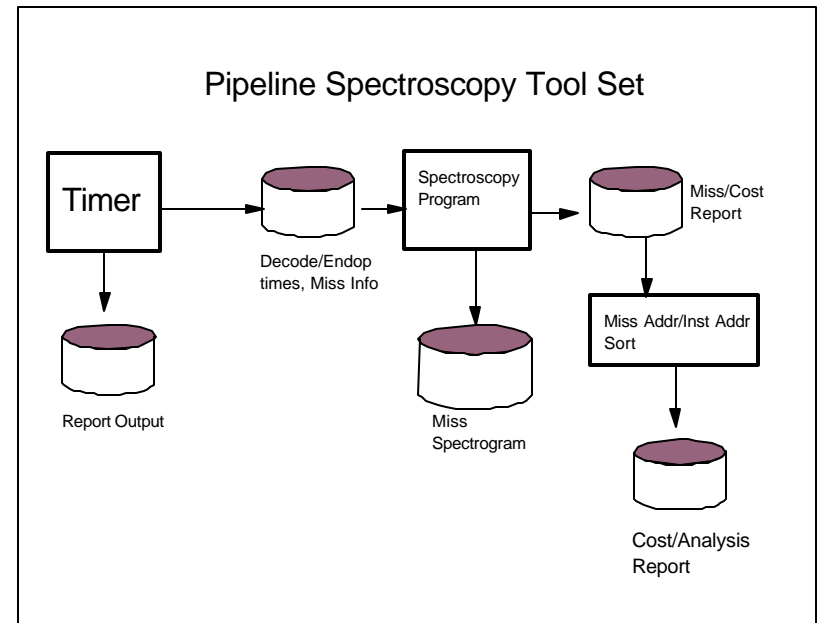
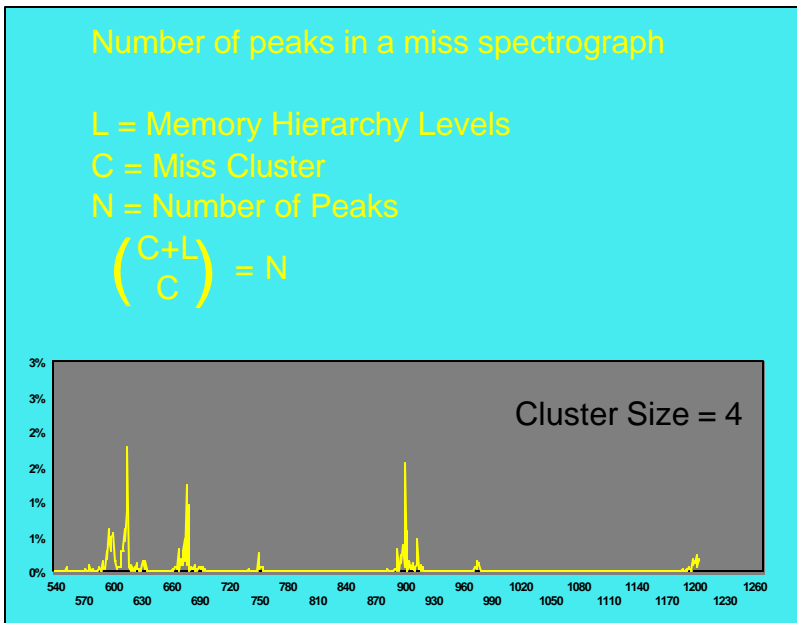
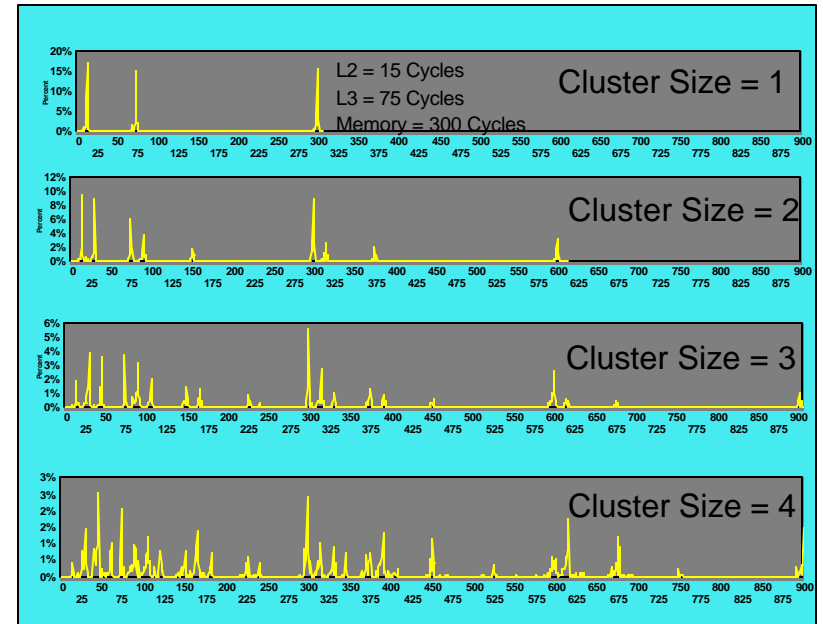
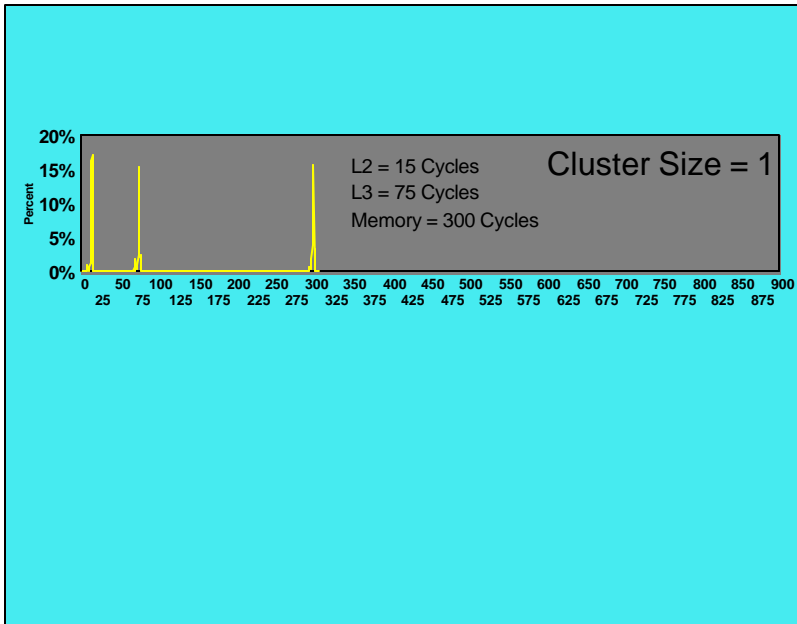
Constructing Miss Spectrogram



Miss Cluster Size = Number of Misses During Miss Facility Busy Interval

$$[(I3-I1)_{FC} - (I3-I1)_{IC}] + [(I5-I4)_{FC} - (I5-I4)_{IC}] + [\dots] = \text{Finite Cache Adder}$$

$$\Delta_1 = (t_2 - t_1) \text{ for the } i\text{th cluster} \quad \sum \Delta_i = (\text{CYC}_{FC} - \text{CYC}_{IC})$$



Miss/Cost Report

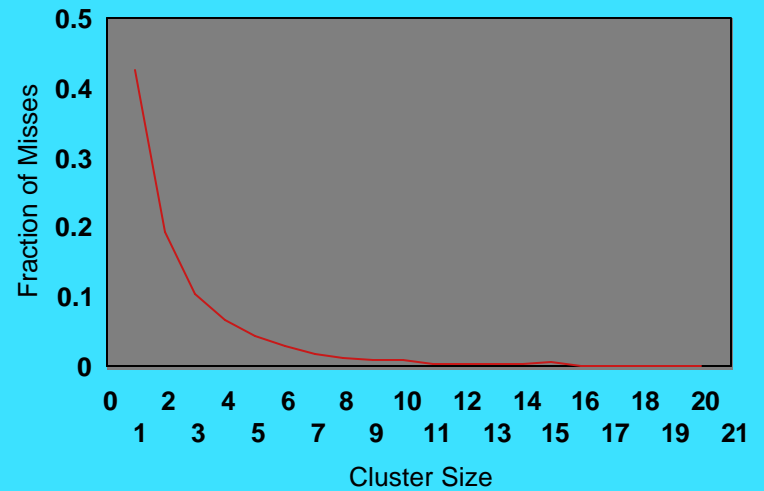
Cluster Size	Cost	Infimum Inst	Supremum Inst	Miss Address	Inst Address	Inst Number
2797	3	45	348389	348417	20B1FA20	000668EC 348389
2797	3	45	348389	348417	20B1FB40	000668EC 348401
2797	3	45	348389	348417	20B1FC00	000668EC 348409
2798	3	45	348421	348449	20B1FD20	000668EC 348421
2798	3	45	348421	348449	20B1FE40	000668EC 348433
2798	3	45	348421	348449	20B1FF00	000668EC 348441

Cost Analysis Report

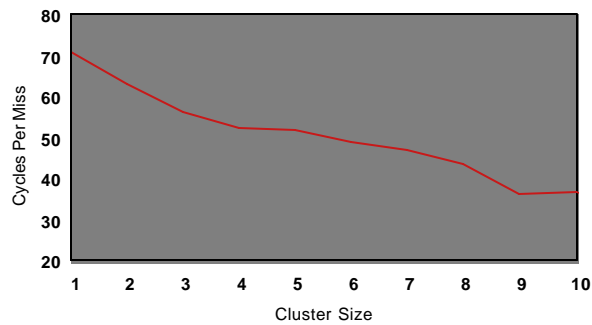
Highest count items:				Highest cost items:			
ASID	Inst Addr	Count	% of Total	ASID	Inst Addr	Total Cost	% of Total
012E	93DAA	4410	0.49%	0146	9724EA6	144045	0.33%
0134	9CDAA	3854	0.43%	0146	3BE10A3E	142423	0.32%
0146	3B46E	3023	0.34%	0146	3BE10A02	139804	0.32%
0146	9724EA6	2596	0.29%	0146	96F2A68	136345	0.31%
012E	3CF3DD62	2444	0.27%	012E	93DAA	125253	0.28%

Cycle Per Miss and Cluster Size Analysis

Misses By Cluster size Base Case



Cycles Per Miss Versus Cluster Size



How do you use a spectrogram?

Analyze Prefetching algorithm
Hardware or Software

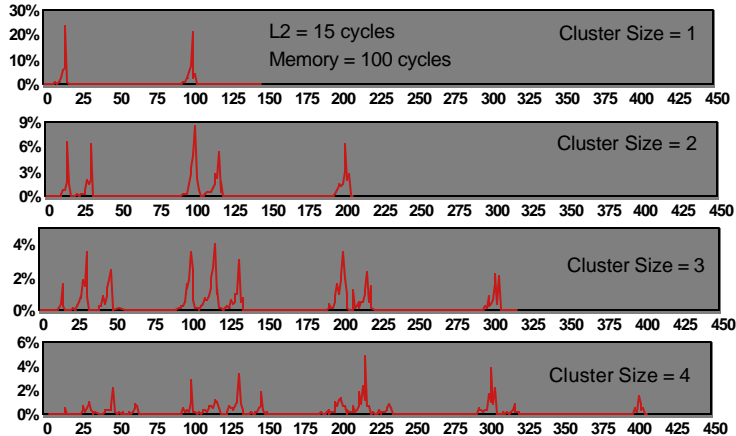
Analyze a hardware Design

Analyze Cluster Patterns and Cost of a Miss
Science or Theory of Misses

Theory:

Can we Predict the shape of a spectrogram (cluster size = 3, 4, or 5)
From analyzing smaller miss clusters?

Observations:



Theory:

Can we Predict the shape of a spectrogram (cluster size = 3, 4, or 5)
From analyzing smaller miss clusters?

Observations:

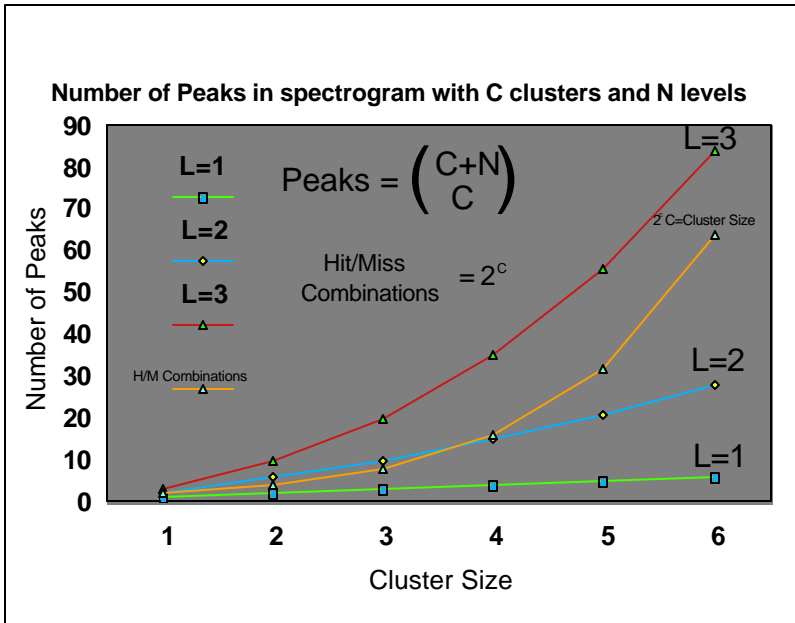
For cluster size = 1, we can have a Hit or Miss in the L2
H or M

For cluster size = 2, we can have $2^2 = 4$ possible outcomes.
HH, HM, MH, MM

For cluster size = 3, we can have $2^3 = 8$ possible outcomes.
HHH, HHM, HMH, HMM, MHH, MHM, MMH, MMM,

For cluster size = 4, we have $2^4 = 16$ possible outcomes
HHHH, MMMM

For cluster of size C there are 2^C possible outcomes.



Determine unique sums for N items choosing C

$$\sum_{i=0}^C \binom{N}{i} = \binom{N}{0} + \binom{N}{1} + \binom{N}{2} + \binom{N}{3} + \dots + \binom{N}{C}$$

Using Relation $\binom{N}{k} = \binom{N+K-1}{K}$

$$= \binom{N-1}{0} + \binom{N}{1} + \binom{N+1}{2} + \binom{N+2}{3} + \dots + \binom{N+C-1}{C}$$

We can combine first two term using $\binom{N}{K} = \binom{N-1}{K-1} + \binom{N-1}{K}$ (1a)

$$= \binom{N+1}{1} + \binom{N+1}{2} + \binom{N+2}{3} + \dots + \binom{N+C-1}{C}$$

Applying 1a repeatedly, the series collapses to $\binom{N+C}{C}$

Theory: Predicting the Shape of a Spectrogram

Observations From a Smaller Cluster Spectrogram Probabilities

If Hit and Misses in the L2 were independent then we could use the binomial distribution to describe H/M clusters

For example, if $\Pr[M] = p$ then $\Pr[H] = (1-p)$ then probability of k misses in a cluster of size N is

$$\binom{N}{k} p^k (1-p)^{N-k}$$

So in a cluster of size 1 $\Pr[M] = \Pr[H] = .50$

Then in cluster of size 2

$\Pr[HH] = \Pr[HM] = \Pr[MH] = \Pr[MM] = .25$

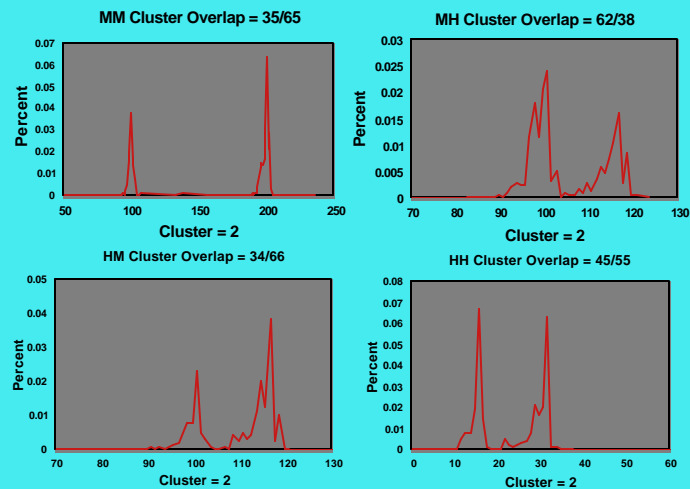
Theory: Predicting the Shape of a Spectrogram

From a Smaller Cluster Spectrogram
Observations: Configure L2 to have 50/50 % Hit/Miss Probabilities

Hit/Miss Probabilities for Cluster Sizes = 2 and 3

Cluster = 2	HH	HM	MH	MM
Prob's	.306	.175	.188	.331

CL 3	HHH	HHM	HMH	HMM	MHH	MHM	MMH	MMM
Prob	.208	.105	.090	.098	.103	.094	.083	.219



Individual Miss Spectrogram for Cluster Size = 2, L1=64KB, L2=256KB 15 Cycle Latency, L3 = 100 Cycles Latency, Data for OLTP

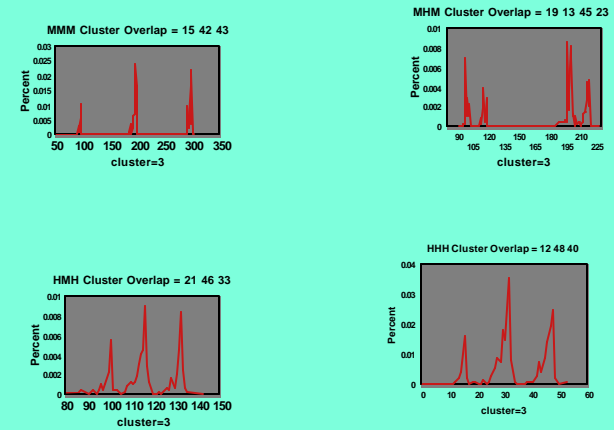


Figure 7. Individual Miss Spectrogram for Cluster Size = 3, L1=64KB, L2=256KB 15 Cycle Latency, L3 = 100 Cycles Latency, Data for OLTP

Let $\text{Prob}[M] = P$ Then $\text{Prob}[\text{Hit}] = (1-P)$

If Hit and Miss Probabilities are independent then
 $\text{Prob}[HH] = \text{Prob}[HM] = \text{Prob}[MH] = \text{Prob}[MM] = .25$

$\text{Prob}[MM] = P \times P^+$

In Order to increase Prob of MM we define a
 and form Convex Combination between P and 1

$$aP + (1-a) = 1-a+aP$$

so $\text{Prob}[MM] = P \times (1-a+aP)$

What is a ?

Let X_i represent i th hit or miss and let 0 represent a hit and 1 represent a miss, then $X_i=M$ or $X_i=H$ is equivalent to $X_i=0$ or $X_i=1$.
 By definition the correlation coefficient

$$\rho = \frac{\text{COV}(X_1, X_2)}{\sqrt{\text{Var}(X_1)\text{Var}(X_2)}}$$

$$\text{COV}(X_1, X_2) = E[X_1, X_2] - E[X_1]E[X_2]$$

$$E[X_1, X_2] = \Pr[X_1=1, X_2=1] = \Pr[X_1=1, X_2=1] = p(1-a+ap)$$

$$E[X_1] = E[X_2] = p$$

$$\text{COV}(X_1, X_2) = p(1-a+ap) - p^2 = (1-a)[p(1-p)]$$

$$\text{Var}(X_1) = \text{Var}(X_2) = E[(X_1 - \mu)^2] = E[X_1^2] - p^2 = p(1-p)$$

$$\rho = \frac{(1-a)[p(1-p)]}{p(1-p)}$$

$$\rho = (1-a) \quad \text{or} \quad a = (1-\rho)$$

Let $\Pr[M] = P$ and $\Pr[H] = (1-P)$. We seek Function that increases $\Pr[MM] > P \times P$.
 To Increase $\Pr[X=M | X_i=M]$, define $a, 0 \leq a \leq 1$ and form convex combination between values P and 1 such that $\Pr[X=M | X_i=M] = ap + (1-a) = (1-a+ap)$

Note the function $(1-a+ap)$ has the properties we desire. For $0 \leq a \leq 1$ and $P \leq 1$, $(1-a+ap) \geq P$.

So $\Pr[MM] = P(1-a+ap)$

Also for $a = (1-\rho)$, as $\rho \rightarrow 1$, (the correlation between X_i, X_{i+1} becomes stronger) $a \rightarrow 0$ and $\Pr[X=M | X_i=M] = (1-a+ap) \rightarrow 1$. Perfect Correlation

For $\rho \rightarrow 0$, (the correlation between X_i, X_{i+1} becomes weaker) $a \rightarrow 1$ and $\Pr[X=M | X_i=M] = (1-a+ap) \rightarrow P$. As if they are independent.

Same technique is used to increase $\Pr[HH] > (1-P) \times (1-P)^*$.

To Increase $\Pr[X=H | X_i=H]$, define $a, 0 \leq a \leq 1$ and form convex combination between values $(1-P)$ and 1 such that $\Pr[X=H | X_i=H] = (1-P)a + (1-a) = (1-aP)$

Note the function $(1-aP)$ has the properties we desire. For $0 \leq a \leq 1$ and $P \leq 1$, $(1-aP) \geq (1-P)$.

So $\Pr[HH] = (1-P)(1-aP)$

Also for $a = (1-\rho)$, as $\rho \rightarrow 1$, (the correlation between X_i, X_{i+1} becomes stronger) $a \rightarrow 0$ and $\Pr[X=H | X_i=H] = (1-aP) \rightarrow 1$. Perfect Correlation

For $\rho \rightarrow 0$, (the correlation between X_i, X_{i+1} becomes weaker) $a \rightarrow 1$ and $\Pr[X=H | X_i=H] = (1-aP) \rightarrow P$. As if they are independent.

In Order to increase Prob of HH we define form Convex Combination between $(1-P)$ and

$$\frac{1}{a} = \frac{(1-P) + (1-a)}{1-aP}$$

Similarly $\text{Prob}[HH] = (1-P) \times (1-aP)$

$$X_i \begin{matrix} & \begin{matrix} \text{Hit} & \text{Miss} \\ \text{Hit} & \begin{pmatrix} 1-aP & aP \\ a-aP & 1-a+ap \end{pmatrix} \\ \text{Miss} & \end{matrix} \\ \text{Hit} & \\ \text{Miss} & \end{matrix} X_{i+1}$$

Probability State Transition Matrix For HH, MM, MH, and HM Events

For example, H/M sequences HMM, MHMM, and HMMM then have the prob

$$(1-p)(ap)(1-a+ap) \text{ and } (p)(a-aP)(1-aP)(ap) \text{ and } (1-p)(ap)(a-aP)(1-a+ap)^2$$

H/M probabilities of any length can be calculated using the generating function

$$(1-p)(1-aP)^W (ap)^X (a-aP)^Y (1-a+ap)^Z \text{ if it starts with a hit}$$

$$\text{and } (p)(1-aP)^W (ap)^X (a-aP)^Y (1-a+ap)^Z \text{ if it starts with a miss}$$

$$X_i \begin{matrix} & \begin{matrix} \text{Hit} & \text{Miss} \\ \text{Hit} & \begin{pmatrix} 1-aP & aP \\ a-aP & 1-a+ap \end{pmatrix} \\ \text{Miss} & \end{matrix} \\ \text{Hit} & \\ \text{Miss} & \end{matrix} X_{i+1}$$

Using The Transition Matrix we can Generate a Hit/Miss Probability for any size cluster using

$$(1-ap)^w (ap)^x (a-ap)^y (1-a+ap)^z$$

where $w + x + y + z = C - 1$ where C =Cluster Size

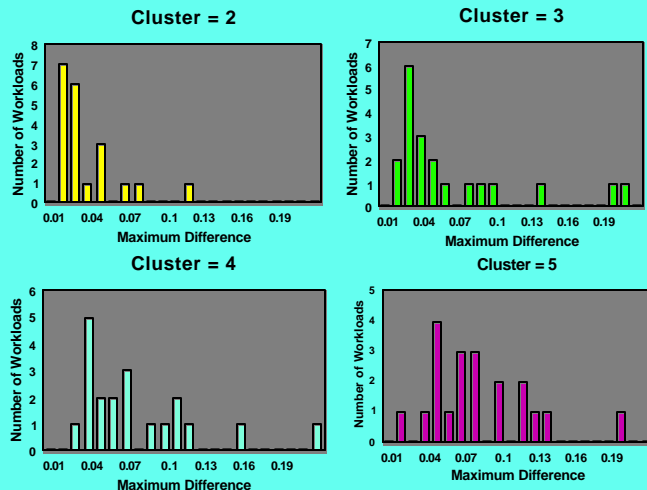
Find Value of a That is 'best fit' from cluster = Data. Then generate Hit/Miss sequences for larger Cluster

$$X_i \begin{matrix} \text{Hit} & \text{Miss} \\ \begin{pmatrix} 1-ap & ap \\ a-ap & 1-a+ap \end{pmatrix} \end{matrix} X_{i+1}$$

Theory: Predicting the Shape of a Spectrogram From a Smaller Cluster Spectrogram

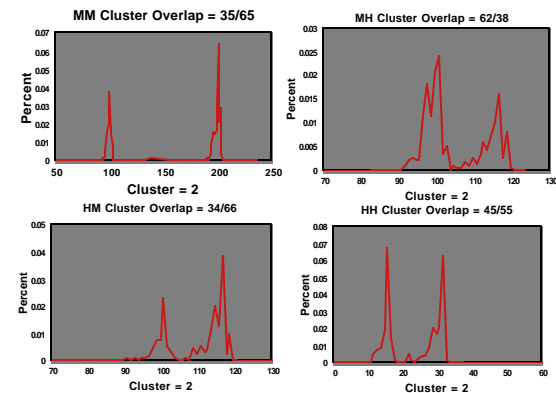
Cluster = 2	HH	HM	MH	MM
Real	.306	.175	.188	.331
Predicted	.308	.182	.182	.328

CL 3	HHH	HHM	HMH	HMM	MHH	MHM	MMH	MMM
Real	.208	.105	.090	.098	.103	.094	.083	.219
Pred	.193	.115	.065	.117	.115	.068	.117	.210



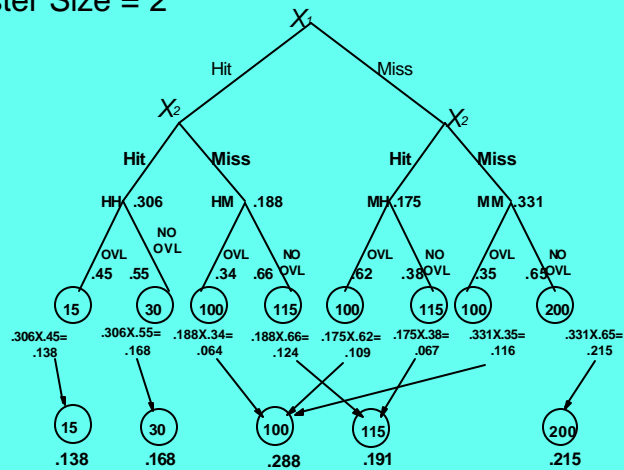
Kolmogorov-Smirnov Test For Predicted Hit/Miss Probabilities For Clusters = 2, 3, 4, and 5.

How do we determine Overlap? Measure overlap/no-overlap for cluster size = 2



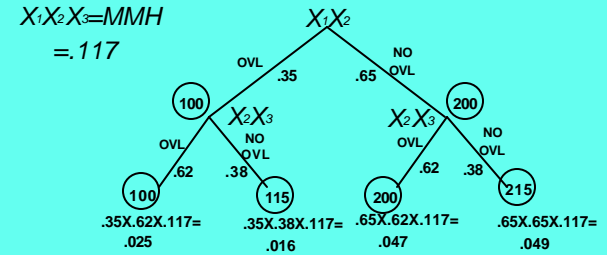
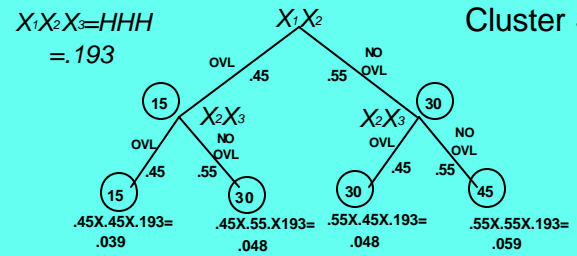
Individual Miss Spectrogram for Cluster Size = 2, L1=64KB, L2=256KB 15 Cycle Latency, L3 = 100 Cycles Latency, Data for OLTP

Cluster Size = 2

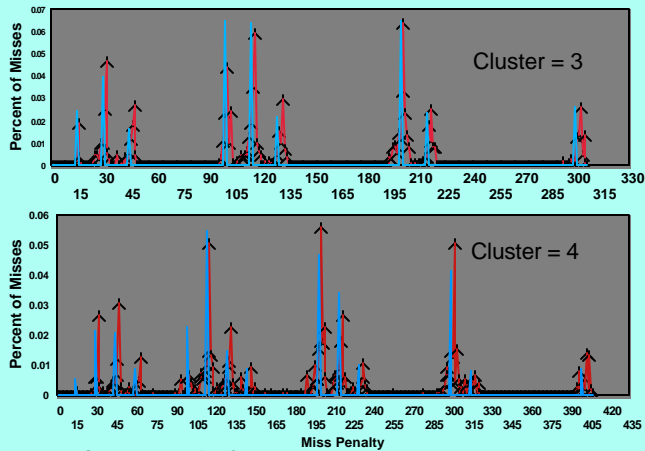


Constructing a Miss Spectrogram. HH, MM, HM, MH, and overlap Probabilities for Cluster = 2 Spectrogram. Data is for OLTP workload

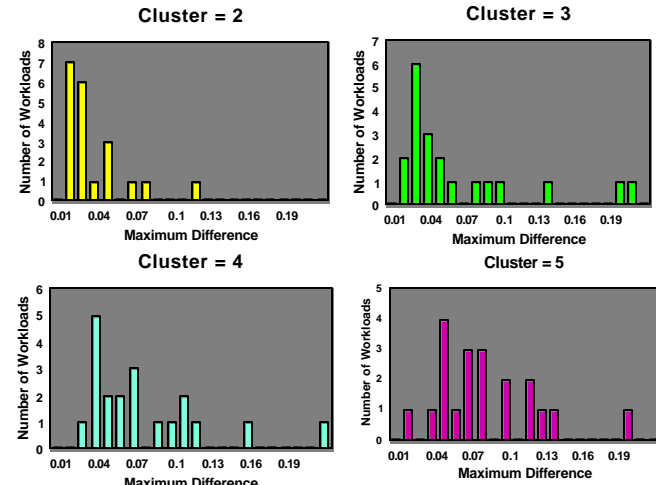
Cluster Size = 3



Overlap/No-Overlap Probability Tree For HHH and MMH. Data For OLTP



Predicted Spectrogram for Clusters = 3 and 4. L1=64KB, L2=256KB, 15 Cycles Latency, L3=100 Cycle Latency, Data For OLTP



Kolmogorov-Smirnov Test For Predicted Hit/Miss Probabilities For Clusters = 2, 3, 4, and 5.

Agreement between and theory and the experiment is very good.

Model uses six parameters

L2 H/M rate

A Correlation Parameter

4 Overlap/Non-Overlap parameters gleaned from
the four H/M patterns in a Cluster Size = 2

Further analysis needed to gain insight into overlap/non-overlap
Programming Structure, and Hardware Limitations

Future work needed to study

In-order versus Out-Of-Order Spectrogram differences
SMT, and MLP organizations.

Conclusions

Pipeline Spectroscopy allows us to apply Statistical
Probabilities to analyze miss behavior in a Multilevel Memory
Hierarchy

Same analysis possible to determine Branch Wrong Guess
penalty, software analysis, run ahead, or AGI Penalty

Pipeline Spectroscopy allows quantitative analysis of individual
miss penalties to considerable precision.

Provides means to quantitatively compare different prefetching
and memory hierarchy analysis