Efficient Locality Approximation from Time

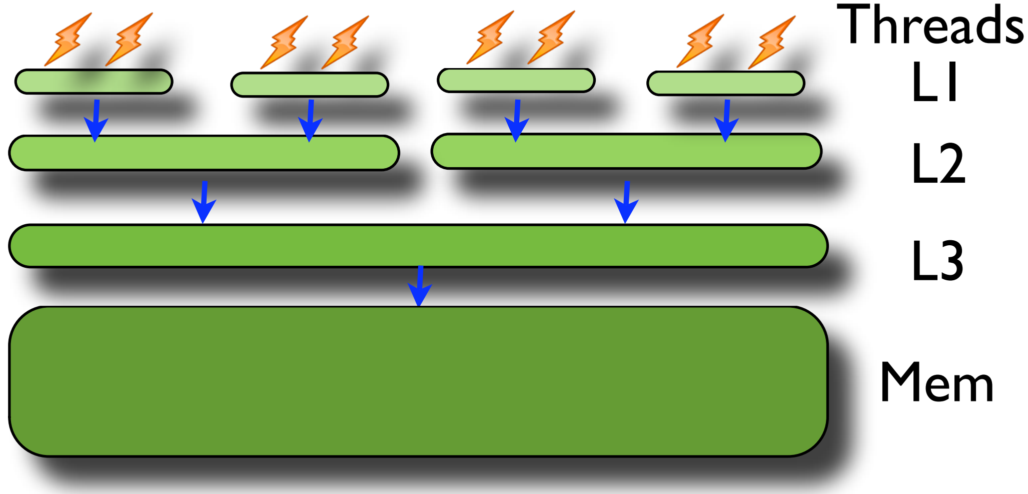
Xipeng Shen

The College of William and Mary

Joint work with Jonathan Shaw
at Shaw Technologies Inc., Tualatin, OR

Locality is Important

- Traditional reasons: memory wall, deeper memory hierarchy.
- New trends: more common and complex cache sharing.



A Locality Model

- Reuse distance (LRU stack distance)
  - *Def: number of distinct elements between reuse*
    `[Mattson et. al. 1970]`

Rd = 2

bcaacb

- Connection with cache
  - Rd > cache size → a likely cache miss

Appeal of Reuse Distance

Rd = 2

bcaacb

More rigorous & machine independent

	Reuse distance		Cache miss rate
Granularity	Point to point	✔	Interval
Accuracy	Exact	✔	Average
Adaptive to cache sizes	Yes	✔	No

Many Uses in Research

- Study cache reuses [Ding+:SC04,Huang+:ASPLOS05]
- Guide and evaluate program transformation [Almasi+:MSP02, Ding+:PLDI03]
- Predict locality phases [Shen+:ASPLOS04]
- Discover locality-improving refactoring [Beyls+:HPCC06]
- Model cache sharing [Chandra+:HPCA05, Jiang+:EuroPar08]
- Insert cache hints [Beyls+:JSA05]
- Manage superpages [Cascaval+:PACT05]
- Guide memory disambiguation [Fang+:PACT05]
- Predict program performance [Marin+:SIGMETRICS04,Zhong +:TOC07]
- Model reference affinity [Zhong+:PLDI04]
- ... ...

Properties of Reuse Distance

	Reuse distance		Cache miss rate	
Granularity	Point to point	✔	Interval	
Accuracy	Exact	✔	Average	
Adaptive to cache sizes	Yes	✔	No	
Practical uses	Few		Many	✔

Our objective: Making reuse distance faster to obtain.

Outline

- Reuse distance measurement
- Efficient approximation of reuse distance (17X speedup)
  - Algorithmic extensions (1 order of magnitude less)
  - Implementation optimizations (3.3X speedup)
- Evaluation tool: trace generator
- Evaluation
- Conclusions

Previous Research

T: execution time

N: data size

```
1970 (Mattson+)
1975 (Bennett+)
1981 (Olken)
1991 (Kim+)
1993 (Sugumar+)
2002 (Almasi+)
2003 (Ding+)
```

$O(T*N)$

$O(T*loglogN)$

But measuring 1-*min* execution still takes several *hours!*

A Different Path

- Key obstacle: Counting out repetitive references in an arbitrarily long interval.

- Previous methods

  implement the definition of reuse distance:

  "Counting" distinct data.

- Our approach

  uses some "cheap" program behavior to statistically approximate reuse distance.

The "Cheap" Behavior

- Time distance (TD)
  - *Def*: number of elements between reuse.

Td = 5

bcaacb

O(T)

- Reuse distance (RD)
  - *Def*: number of distinct elements between reuse.

Rd = 2

bcaacb

TD → RD : Intuition

- Is it possible?

b . . . . b

TD=5 → RD=1, 2, 3, or 4 ?   *No idea.*

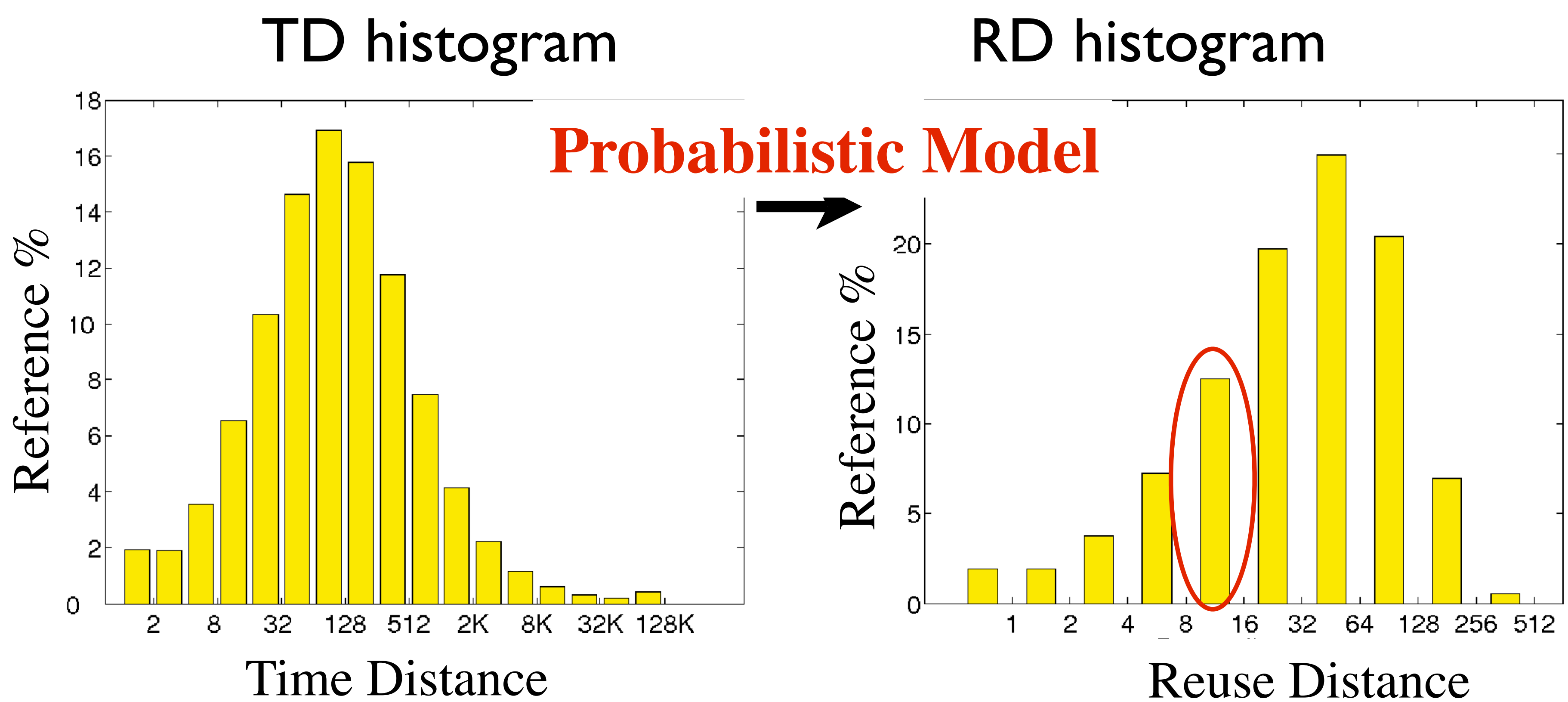
- What if we know the following:

  totally 4 reuses; one TD is 5, three TDs are all 1.

  ↳ **bxxxxb** → RD=1

$\boxed{\text{TD histogram}}$

Problem to Solve

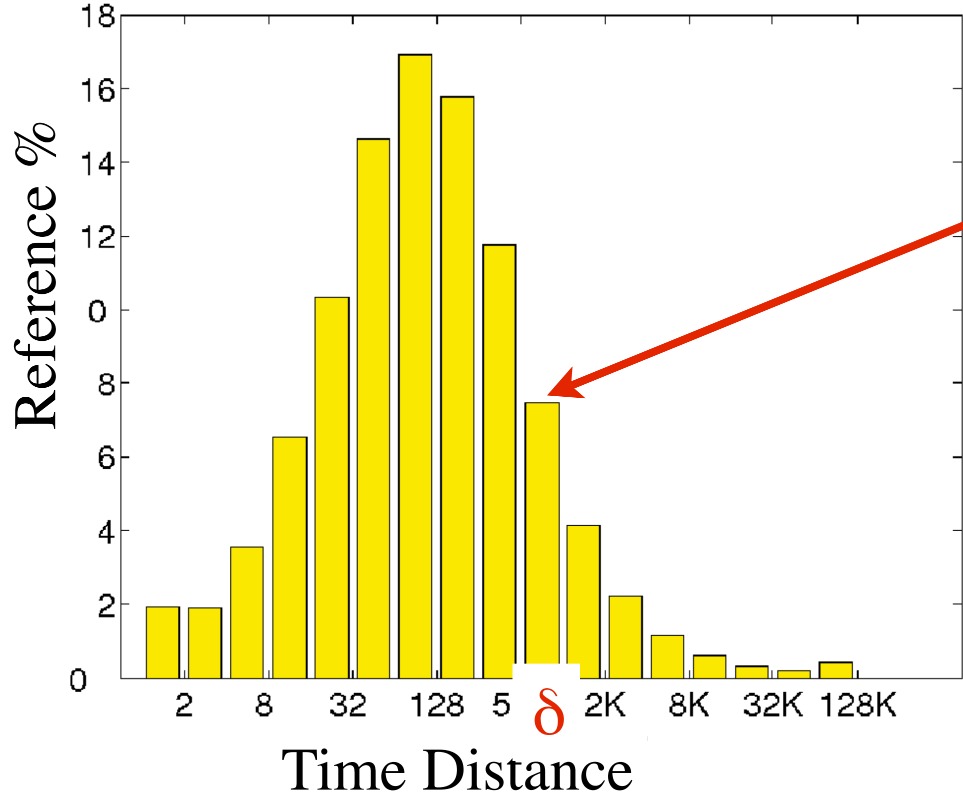


"Locality approximation from time", Shen+: POPL'07.

Connection between TD and RD

[Shen+:POPL'07]

Expectation of the probability for a variable to



$P_T(\delta)$

TD histogram

Connection between TD and RD

[Shen+:POPL'07]

Expectation of the probability for a variable to appear in a Δ-long interval:

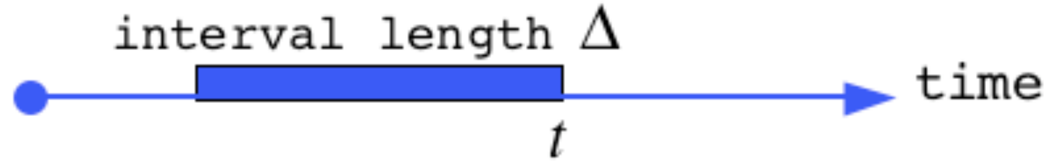
$$p(\Delta) = \sum_{\tau=1}^{\Delta} \sum_{\delta=\tau+1}^{T} \frac{1}{N-1} \underbrace{P_T}_{\text{TD histogram}}(\delta)$$

Probability for the interval to have *k* distinct variables (Bernoulli process):

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

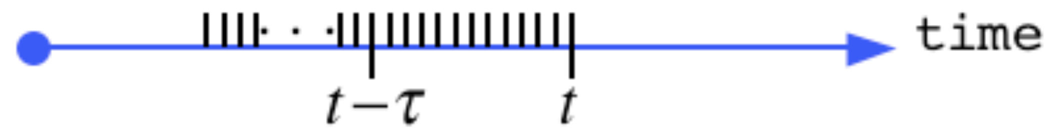
Compute p(Δ)

- p(Δ): a variable to appear in a Δ-long interval.



$$p(\Delta)=\sum_{\tau=1}^{\Delta}\sum_{\delta=\tau+1}^{T}\frac{1}{N\text{-}1}P_T(\delta)$$

- p’(τ): a variable’s last access before *t* is at time *(t*-τ*)*.



$$p(\Delta) = \sum_{\tau=1}^{\Delta} p'(\tau)$$

- The following is proved in [Shen+:POPL07]

$$p'(\tau) = \sum_{\delta=\tau+1}^{T} P_T(\delta)/(N\text{-}1)$$

Implementation Issues

- Scale: The model applies to every access, but not to histograms.
  - The width of a bar must be 1.
- Overhead: high cost in measuring time distance.
  - Bookkeeping and buffer boundary checking at every memory access.

Outline

- Reuse distance measurement
- Efficient approximation of reuse distance (17X speedup)
  - Algorithmic extensions (1 order of magnitude less)
  - Implementation optimizations (3.3X speedup)
- Evaluation tool: trace generator
- Evaluation
- Conclusions

Algorithm Extension

- Basic extension: assume all references in a bar have the same $p(\Delta)$, denoted as $p(b_i)$.

$$p(\Delta)=\sum_{\tau=1}^{\Delta}\sum_{\delta=\tau+1}^{T}\frac{1}{N\text{-}1}P_T(\delta) \quad \rightarrow \quad p(b_i)=\sum_{\tau=1}^{\frac{\overleftarrow{b_i}+\overrightarrow{b_i}}{2}}\sum_{\delta=\tau+1}^{T}\frac{1}{N-1}P_T(\delta)$$

$$p(k,\Delta)=\binom{N}{k}p(\Delta)^k(1\text{-}p(\Delta))^{N\text{-}k} \quad \rightarrow \quad p(k,b_i)=\binom{N}{k}p(b_i)^k(1-p(b_i))^{N-k}$$

- Time complexity: $O(L_T^3)$

  L_T : number of bars in a TD histogram.

Algorithmic Optimizations

- Decompose $p(b_i)$ into 3 sub-equations to remove redundant computations.

$$
\begin{aligned}
p(b_i) &= P_2(b_i)/2 + \sum_{j=0}^{i-1} P_2(b_j) \\
P_2(b_i) &= [\sum_{j=i+1}^{L_T} P_1(b_j) \frac{\overrightarrow{b_i} - \overleftarrow{b_i}}{\overrightarrow{b_j} - \overleftarrow{b_j}} \sum_{\tau=\overleftarrow{b_j}}^{\overrightarrow{b_j}-1} \frac{1}{\tau - 1}] \\
&\quad + P_1(b_i) \frac{1}{\overrightarrow{b_i} - \overleftarrow{b_i}} \sum_{\tau=\overleftarrow{b_i}+1}^{\overrightarrow{b_i}-1} \frac{\tau - \overleftarrow{b_i}}{\tau - 1} \\
P_1(b_i) &= \frac{\overleftarrow{b_i} + \overrightarrow{b_i} - 3}{2(N-1)} P_T(b_i).
\end{aligned}
$$

Algorithmic Optimizations

- Further optimizations
  - mathematical approximation

$$\sum_{i=m_1}^{m_2} \frac{1}{i} \simeq ln\frac{m_2+0.5}{m_1-0.5}$$

  - statistical approximation
    - Normal distribution with table-lookup for binomial distribution calculation
- Time complexity

$$O(L_T^3) \rightarrow O(L_T^2)$$

Details in [Shen+:LCPC'08].

Measure TD

- Invocation to record function after every load/store.

Basic record function

```
Procedure RecordMemAcc (addr)
    buffer [index++] = addr;

    if (index== BUFFERSIZE) then
        ProcessBuff();
    endif

end
```

After optimization

```
Procedure RecordMemAcc (addr)
    buffer [index++] = addr;
end
```

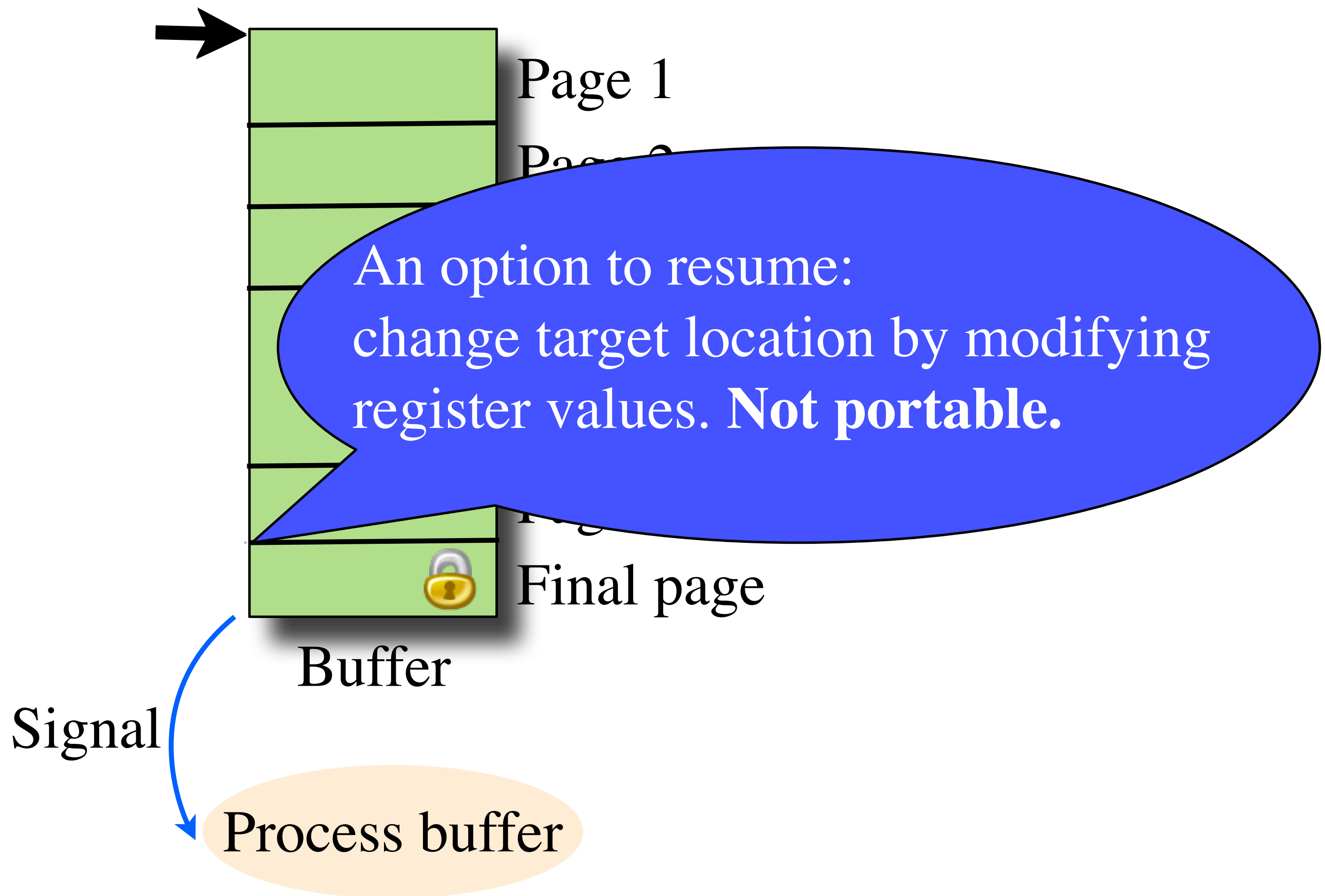
- Fewer operations
- Fewer branch miss predictions
- Amenable to runtime inlining
- 3.3X speedup

MMU Control

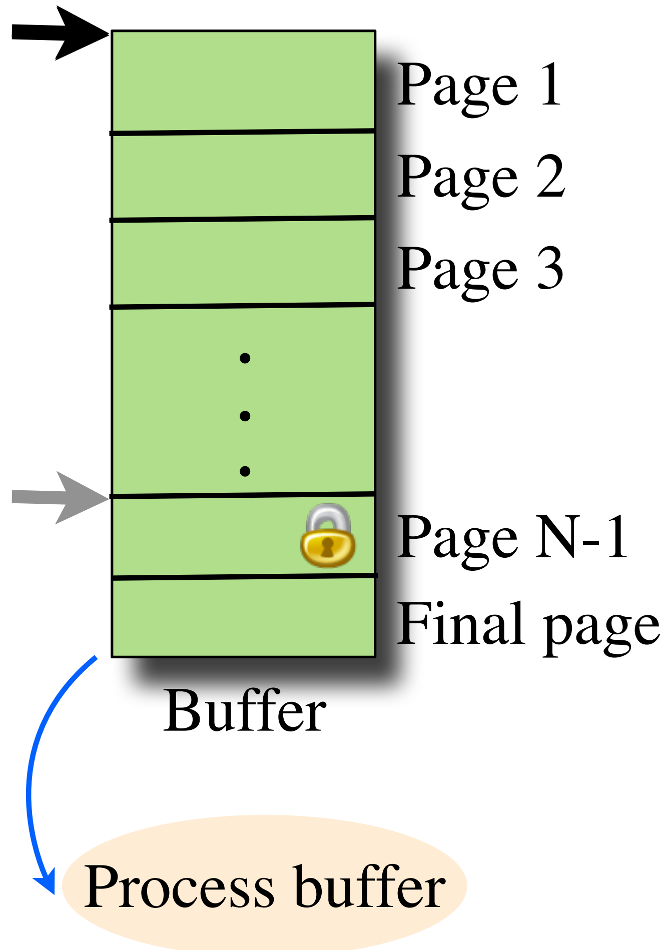
- Typical scheme:
  - Control page permission & modify registers
  - Not portable across architectures.
- Our approach:
  - 2-page scheme
  - Close & open permissions of the final 2 pages alternatively

Details in [Shen+:LCPC'08].

2-Page Scheme for Using MMU



2-Page Scheme for Using MMU



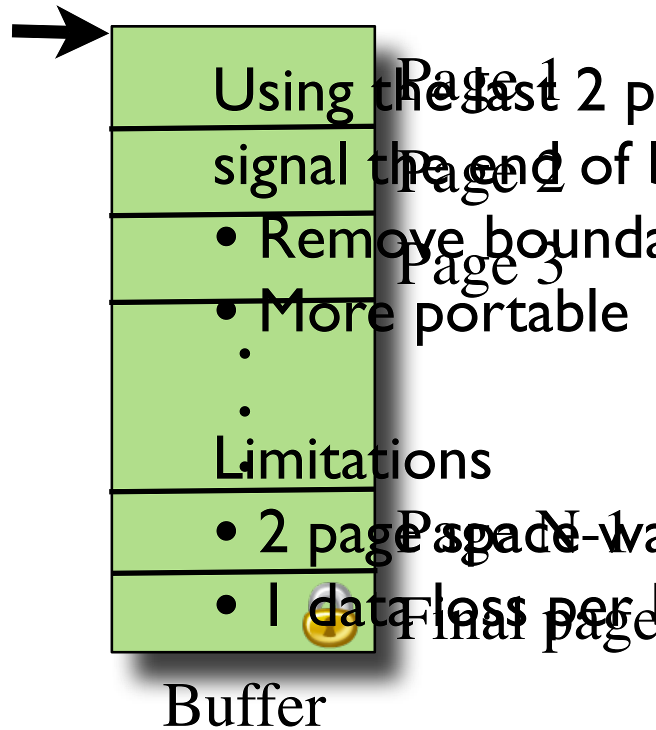
2-Page Scheme for Using MMU

Using the last 2 pages alternatively to signal the end of buffer

- Remove boundary check
- More portable

Limitations

- 2 page space-waste
- 1 data loss per buffer



Outline

- Reuse distance measurement
- Efficient approximation of reuse distance
  - Algorithmic extensions
  - Implementation optimizations
- Evaluation tool: trace generator
- Evaluation
- Conclusions

A Reverse Problem: Trace Generation

- Reuse distance measurement or approximation

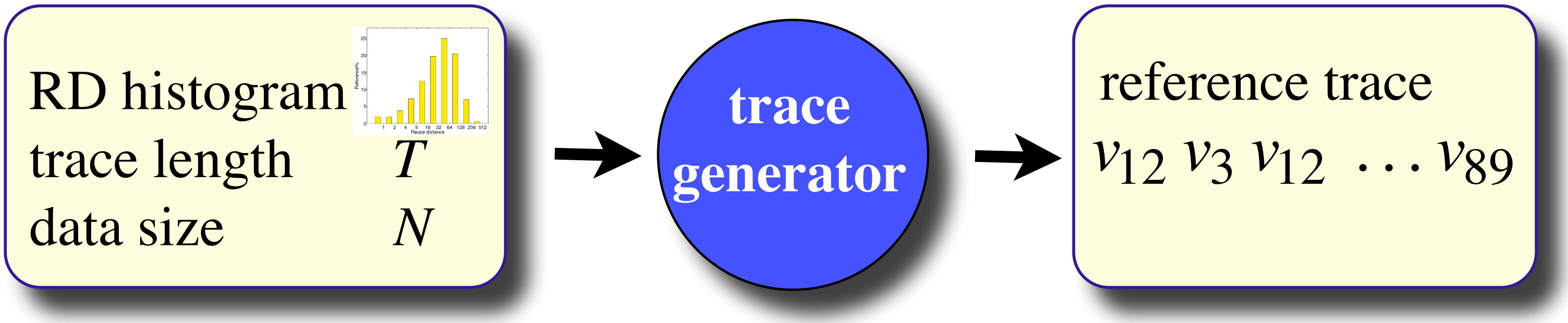
Trace $\longrightarrow$ Reuse distance

- Trace generator

Trace $\longleftarrow$ Reuse distance

Use: for evaluating locality techniques on various reuse patterns.

A Reverse Problem: Trace Generation



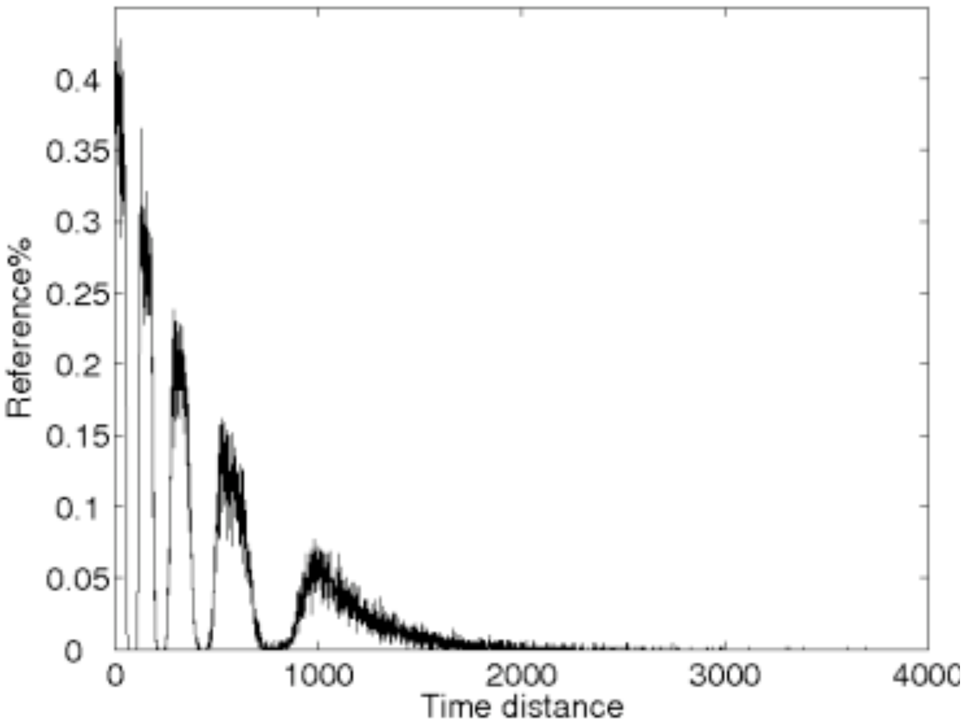
- Technique: a stochastic process
- Property: The generated trace meets input requirements *(proof in [Shen+:LCPC'08])*

Outline

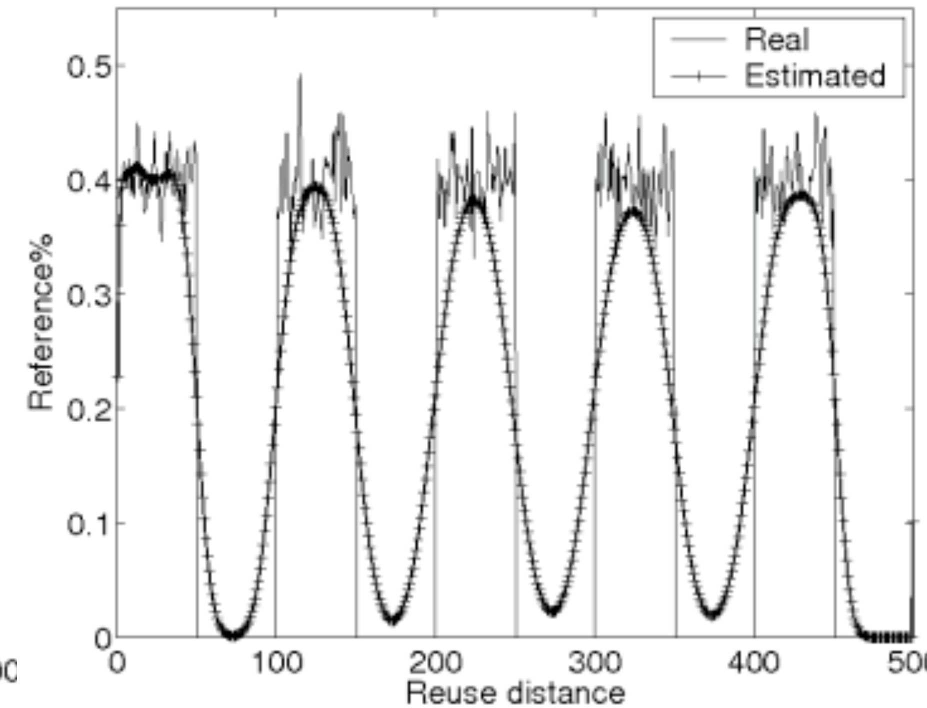
- Reuse distance measurement
- Efficient approximation of reuse distance
    - Algorithmic extensions
    - Implementation optimizations
- Evaluation tool: trace generator
- Evaluation
- Conclusions

Evaluation (Pulse-like reuse distributions)

Time Distance Histogram



Reuse Distance Histogram



RD Approximation on Synthetic Traces

	acc%
Normal (var=20)	92.8
Normal (var=100)	96.3
Normal (var=200)	95.8
Exponential	96.9
Average	**95.5**

Evaluation on Real Benchmarks

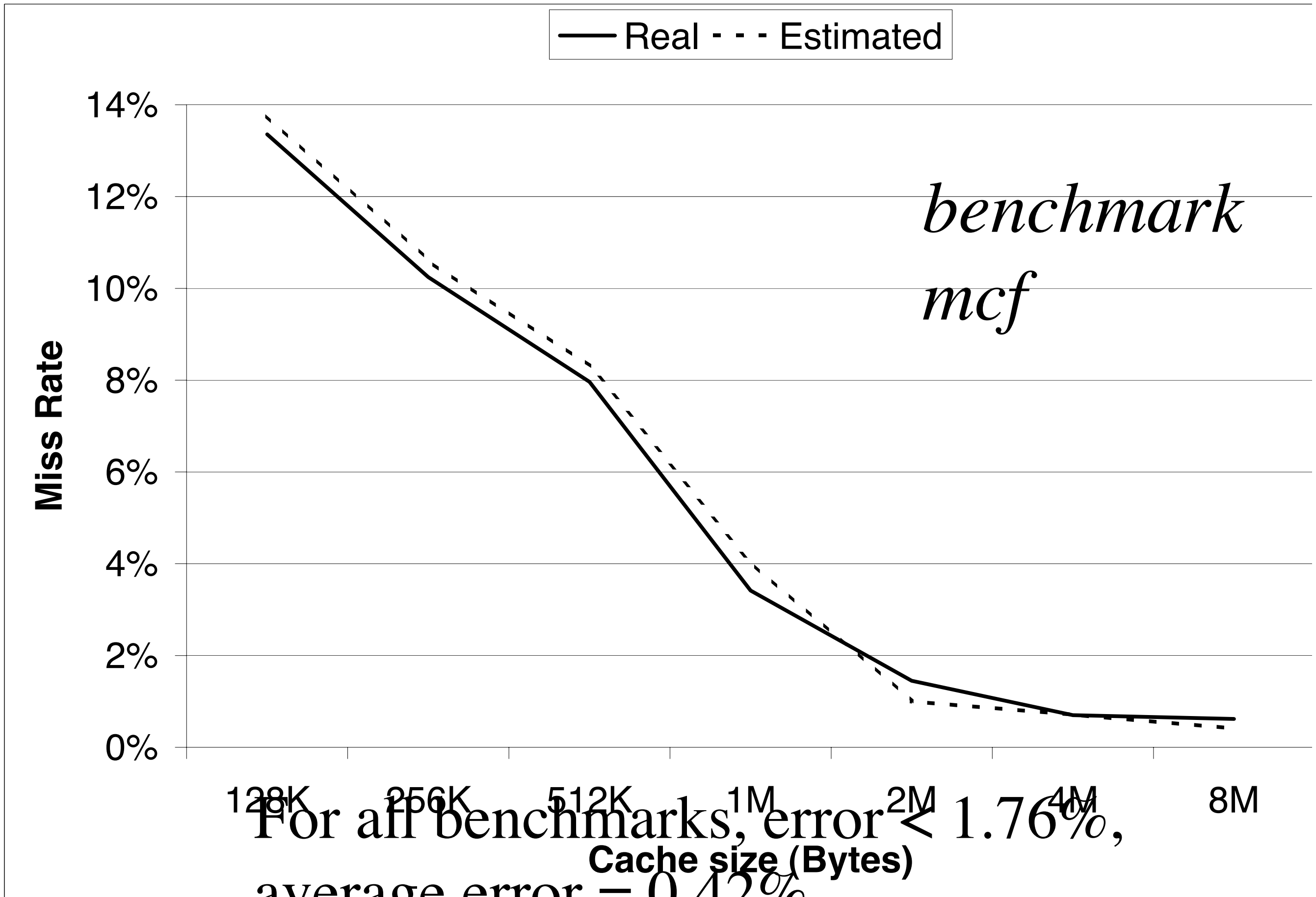
CPU	Intel Xeon 2GHz
Instrumentor	PIN 3.4
Compiler	GCC 3.4.4 (“-O3”)
HW perf. measure	PAPI 3.2
Benchmarks	SPEC CPU2000 ref

Baseline: Ding+:PLDI’03.

Results on Real Benchmarks

Programs	Element	
	acc%	speedup
gcc	89.0	21.2X
gzip	99.0	19.0X
mcf	42.6	8.3X
twolf	88.2	5.9X
ammp	95.8	14.3X
applu	86.1	19.0X
equake	57.6	23.7X
mesa	97.3	26.3X
mgrid	89.7	20.6X
Average	**82.8**	**17.6X**

Uses in Cache Miss Rate Estimation



For all benchmarks, error < 1.76%, average error = 0.42%.

(Details in Shen+:TR902)

Conclusions

- Strong connection exists between time and locality.
- Reuse distance can be approximated from time efficiently.

* Details in:

"Locality approximation from time", POPL'07.

"Adaptive software speculation for enhancing the efficiency of behavior-oriented parallelization", LCPC'08.

Acknowledgment

- Chen Ding (U of Rochester) suggested the use of 2 pages for MMU control.
- Brian Meeker (U of Rochester) collected preliminary data in the early stage.
- Supported by NSF.