Compiler Management of

Global and Dynamic Data Reuse

Chen Ding

Computer Science Department
University of Rochester
Rochester, New York

Computer Science @ Rochester

- *Established in 1974*
  - *traditionally focused in AI, vision, natural languages, computer systems*
  - *Xerox workstations in late 70s*
  - *largest shared-memory parallel machine in late 80s*
  - *small department (14 faculty members)*
- *System group in 2003*
  - *computer science: Chen Ding, Sandhya Dwarkadas, Amy Murphy, Michael Scott, and Kai Shen*
  - *computer engineering: David Albonesi, Michael Huang, Wendi Heinzelman*
  - *7 papers in FCRC 2003 (4 ISCA, 2 PPoPP, 1 PLDI)*
- *Students involved in the compiler work*
  - *Yutao Zhong, Yongkang Zhu, Xipeng Shen, Maks Orlovich, and Ahren Studer*

Background

- *Computer speed improvement*
  - *Moore's law*
  - *supercomputers [Allen&Kennedy, Optimizing Compilers, p.2]*
- *Memory gap (wall, cliff, ..)*
- *Insufficient memory bandwidth*
  - *CPU speed improves 60% a year on average*
  - *memory bandwidth ~28% a year [Burger+ ISCA' 96]*
- *Our research*
  - *understanding large-scale data behavior*

Machine and Program Balance

by Callahan, Cocke, and Kennedy
in JPDC' 88
extended by Ding and Kennedy
in IPDPS' 00

Performance Model

- *Machine balance*
  - *max load/store bandwidth divided by max flop rate*
- *Program balance*
  - *total loads/stores divided by total floating-point ops*
- *Consequences*
  - $\mathcal{B}_{machine} = \mathcal{B}_{program}$ *full CPU and bandwidth utilization*
  - $\mathcal{B}_{machine} < \mathcal{B}_{program}$ *CPU idle*
- *Extensions [Ding&Kennedy, IPDPS' 00] to appear in JPDC*
  - *multi-level memory hierarchy*
  - *ratio of demand over supply*

Memory-Bandwidth Bottleneck

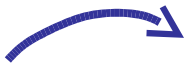
- *Ratios of demand to supply*

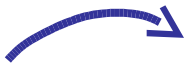
Applications	Ratio: demand/supply		
	Reg BW	Cache BW	Mem BW
Convolution	1.6	1.3	6.5
Dmxpy	2.1	2.1	10.5
Matrix Mul.	2.1	0.25	0.05
FFT	2.1	0.8	3.4
SP	2.7	1.6	6.1
Sweep3D	3.8	2.3	9.8

- *Memory bandwidth is most limited*
- *Maximal CPU utilization: 10% to 33%*
- *The imbalance is getting worse*
- *Need to minimize total memory transfer*

Software Solutions

- *Reorder the computation*
  - *fusing the computation on the same data*
- *Reorganize the data*
  - *grouping data used by the same computation*

a x y y ……a x e y ……  *a a x x y y y …e …*

a x y y ……x a e y ……  *a b c c …b a e c…*

- *Scale matters*
  - *long temporal distance*
  - *large data volumes*

Outline

- *Memory optimization in scientific programs*
  - *reuse-based loop fusion*
  - *affinity-based data regrouping*
  - *dynamic data packing*
- *Reuse pattern in complex programs*
  - *miss rate prediction*
- *Current work*

Computation Fusion
and
Data Regrouping

[Ding&Kennedy IPDPS' 01 Best Paper, LCPC' 99]
to appear in JPDC

Example Fusion

```
for i=2, N
  a[i] = f(a[i-1])
end for

a[1] = a[N]
a[2] = 0.0

for i=3, N
  b[i] = g(a[i-2])
end for
```

```
for i=2, N
  a[i]=f(a[i-1])
  if (i==3)
    a[2]=0.0
  else if (i==N)
    a[1]= a[N]
  end if

  if (i>2 && i<N)
    b[i+1] = g(a[i-1])
  end if
end for

b[3] = g(a[1])
```

- *loop embedding, loop splitting, interleaving+alignment*

Reuse-Based Fusion

- *Previous work*
  - *fusing loops of the same shape*
    - *e.g. same iteration counts, same number of levels, and perfectly nested*
- *Reuse-based fusion*
  - *reuse based*
  - *shape independent*
- *Multi-level fusion*
  - *minimize the number of outer loops*
- *Optimal fusion for bandwidth*
  - *hyper-graph formulation of data sharing*
  - *an NP-hard problem*

Data Regrouping

- *Cache-block utilization*
  - *high-end machines use large cache blocks*
  - *use one integer in a 64-byte cache block*
    - *6% bandwidth and cache utilization*
- *Data regrouping*
  - *group " useful" data into the same cache block*
- *Questions*
  - *what does " useful" mean?*
  - *can we regroup data across array and object boundary?*
  - *can we regroup data during execution?*

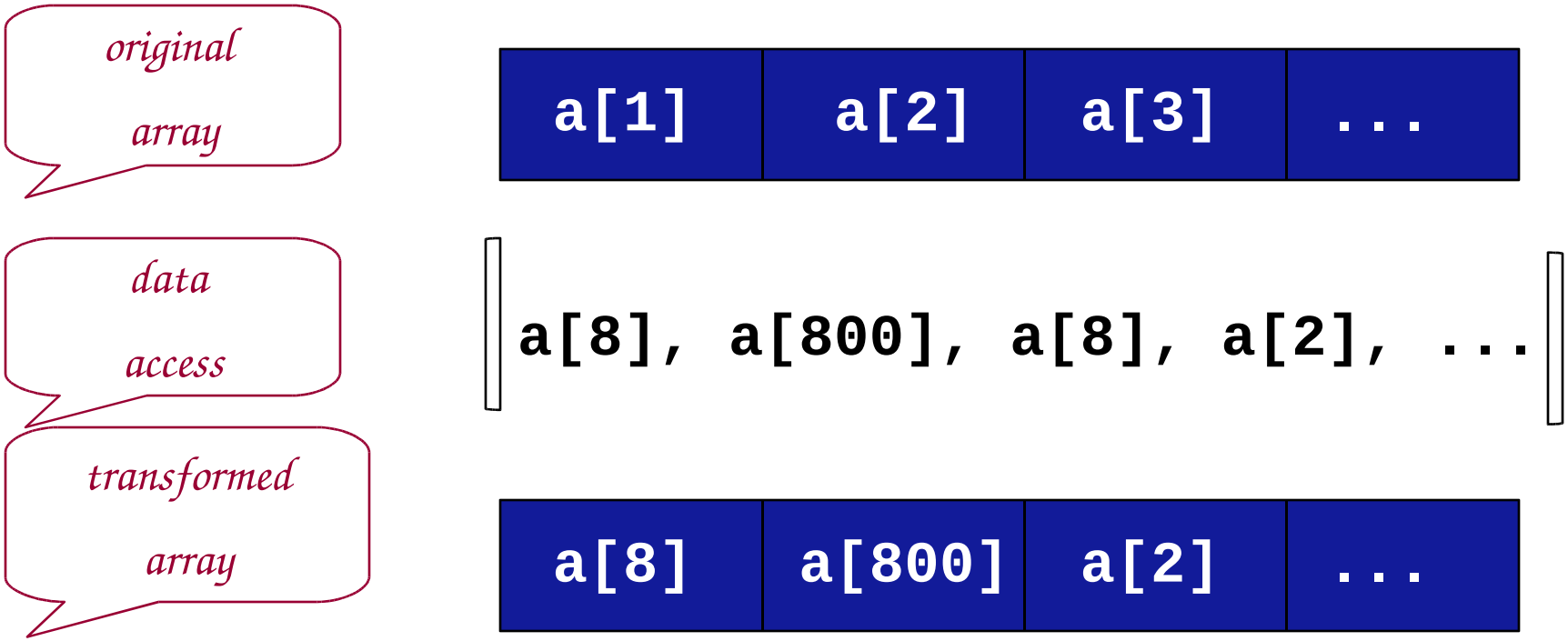
Reference Affinity

- *Definition*
  - *data that are always used together belong to the same affinity group*
  - *reflective, symmetric, transitive*
- *An example*
  - *a y y x ……x a y e x x ……e e y e*
  - *affinity groups: {a,x}, {y}, and {e}*
- *Comparison with frequency models*
  - *access frequency: {x,y,e} and {a}*
  - *pairwise frequency/affinity: {a, y, e}, {x}*
  - *frequently used ≠ frequently used together*

More on Data Regrouping

- *Multi-dimension data regrouping*
  - *exploits reference affinity at all levels*
  - *guarantees a consistent solution*
- *Partial and dynamic reference affinity*
  - *data are accessed together at different times*
  - *NP-hard problems [Thabit, Rice' 81, Kennedy&Kremer, TOPLAS' 98]*
  - *optimal solution is machine-dependent*
- *Dynamic data packing [Ding&Kennedy, PLDI' 99]*
  - *adaptive changing data layout during execution*
  - *found to be cost effective [Mellor-Crummey+ ICS' 99, Strout+ PLDI' 03]*

Example packing



Software remapping:

a[i] a[remap[i]]

a[b[i]] a[remap[b[i]]]

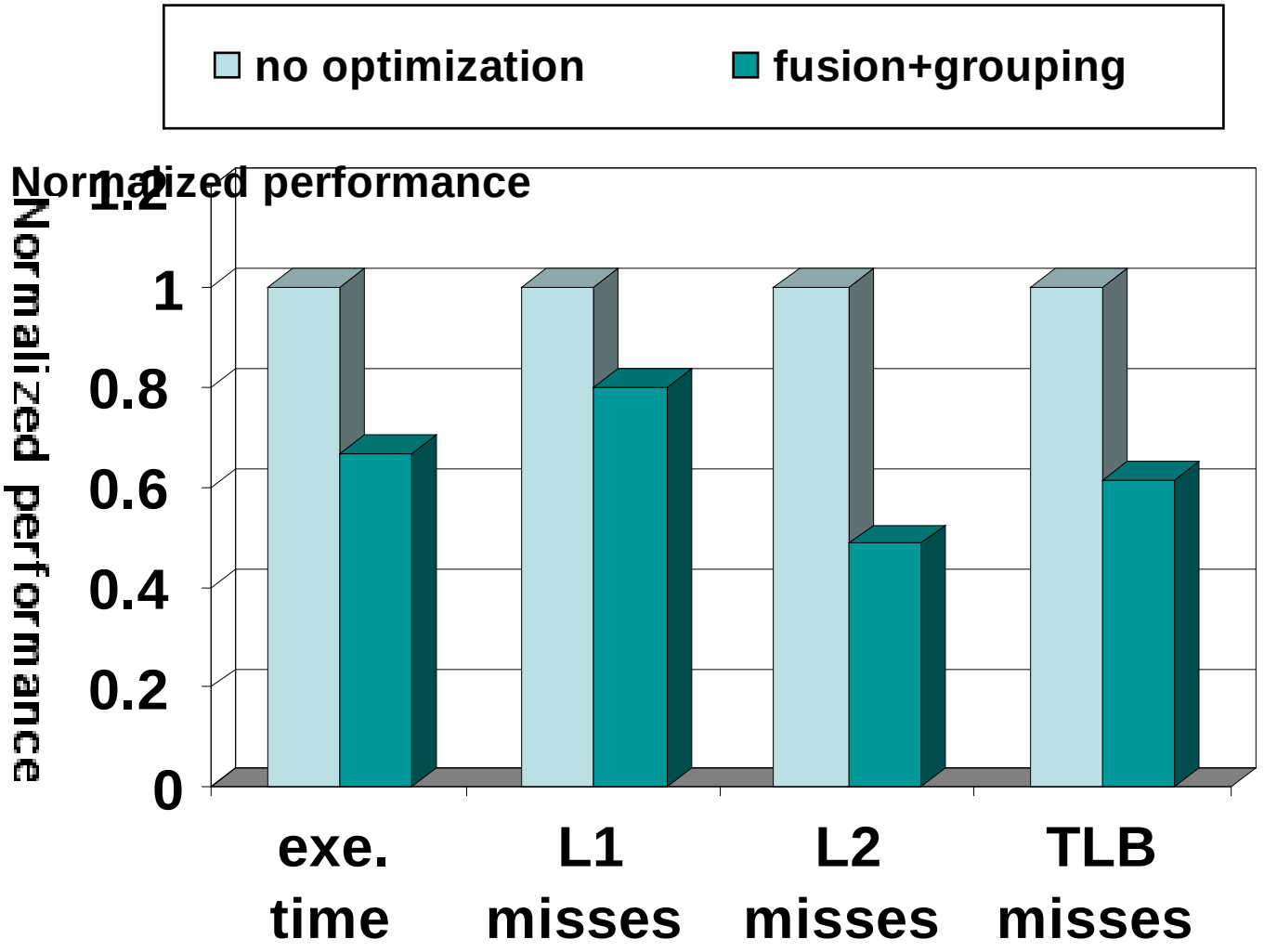
Dynamic Optimizations

- *Locality grouping & Dynamic packing*
  - *run-time versions of computation fusion & data grouping*
  - *linear time and space cost*
- *Compiler support*
  - *analyze data indirections*
  - *find all optimization candidates*
  - *use run-time maps to guarantee correctness*
  - *remove unnecessary remappings*
    - *map reuse*
    - *reference update*
- *The first set of compiler-generated run-time transformations*

NAS/SP

- *Benchmark application from NASA*
  - *computational fluid dynamics (CFD)*
  - *class B input, 102x102x102*
  - *218 loops in 67 loop nests, distributed into 482 loops*
  - *15 global arrays, split into 42 arrays*
    - *e.g. a(3,n) -> a1(n), a2(n), a3(n)*
- *Optimizations*
  - *fused into a dozen loop nests*
  - *grouped into 17 new arrays, e.g.*
    - *{ainv[n,n,n], us[n,n,n], qs[n,n,n], u[n,n,n,1-5]}*
    - *{lhs[n,n,n,6-8], lhs[n,n,n,11-13]}*

NAS/SP


no optimization
fusion+grouping
Normalized performance
Normalized performance
1.2
1
0.8
0.6
0.4
0.2
0
exe. time
L1 misses
L2 misses
TLB misses

Comparison with SGI Compiler

programs	L2 misses			TLB misses			Speed up
	NoOpt	SGI	New	NoOpt	SGI	New	over SGI
Swim	1.00	1.10	0.94	1.00	1.60	1.05	1.14
Tomcatv	1.00	0.49	0.39	1.00	0.010	0.010	1.17
ADI	1.00	0.94	0.53	1.00	0.011	0.005	2.33
Swee p3D	1.00	0.99	0.16	1.00	1.00	0.04	1.93
NAS/SP	1.00	1.00	0.49	1.00	1.09	0.67	1.49
Average	1.00	0.90	0.50	1.00	0.74	0.35	1.61
Moldyn	1.00	0.99	0.19	1.00	0.77	0.10	3.02
Mesh	1.00	1.34	0.39	1.00	0.57	0.57	1.20
Magi	1.00	1.25	0.76	1.00	1.00	0.36	1.47
NAS/CG	1.00	0.95	0.15	1.00	0.97	0.03	4.36
Average	1.00	1.13	0.37	1.00	0.83	0.27	2.51

Other Fusion Studies

- *Early fusion studies*
  - *first uses [Wolfe UIUC' 82, Allen & Kennedy IEEE TC' 86]*
  - *complexity [Kennedy&McKinley Rice' 93, Darte PACT' 99]*
  - *heuristics [Gao+ LCPC' 92, Kennedy ICS' 01]*
  - *implementation [McKinley+ TOPLAS' 96, Manjikian&Abdelrahman 97, Lim+ PPoPP' 01]*
  - *array contraction [Gao+ LCPC' 92, Lim+ PPoPP' 01, Song+ ICS' 01]*
- *Aggressive loop blocking/tiling*
  - *shackling and slicing [Kodukula+ PLDI' 97, Pugh&Rosser LCPC' 99, Yi+ PLDI' 00]*
  - *time skewing [Song PLDI' 99, Wonnacott IPDPS' 00, Jin+ SC' 02]*
- *Recent work*
  - *manual fusion in C programs [Pingali+ ICS' 02]*
  - *compiler fusion of loops containing array indirection [Strout+ PLDI' 03]*

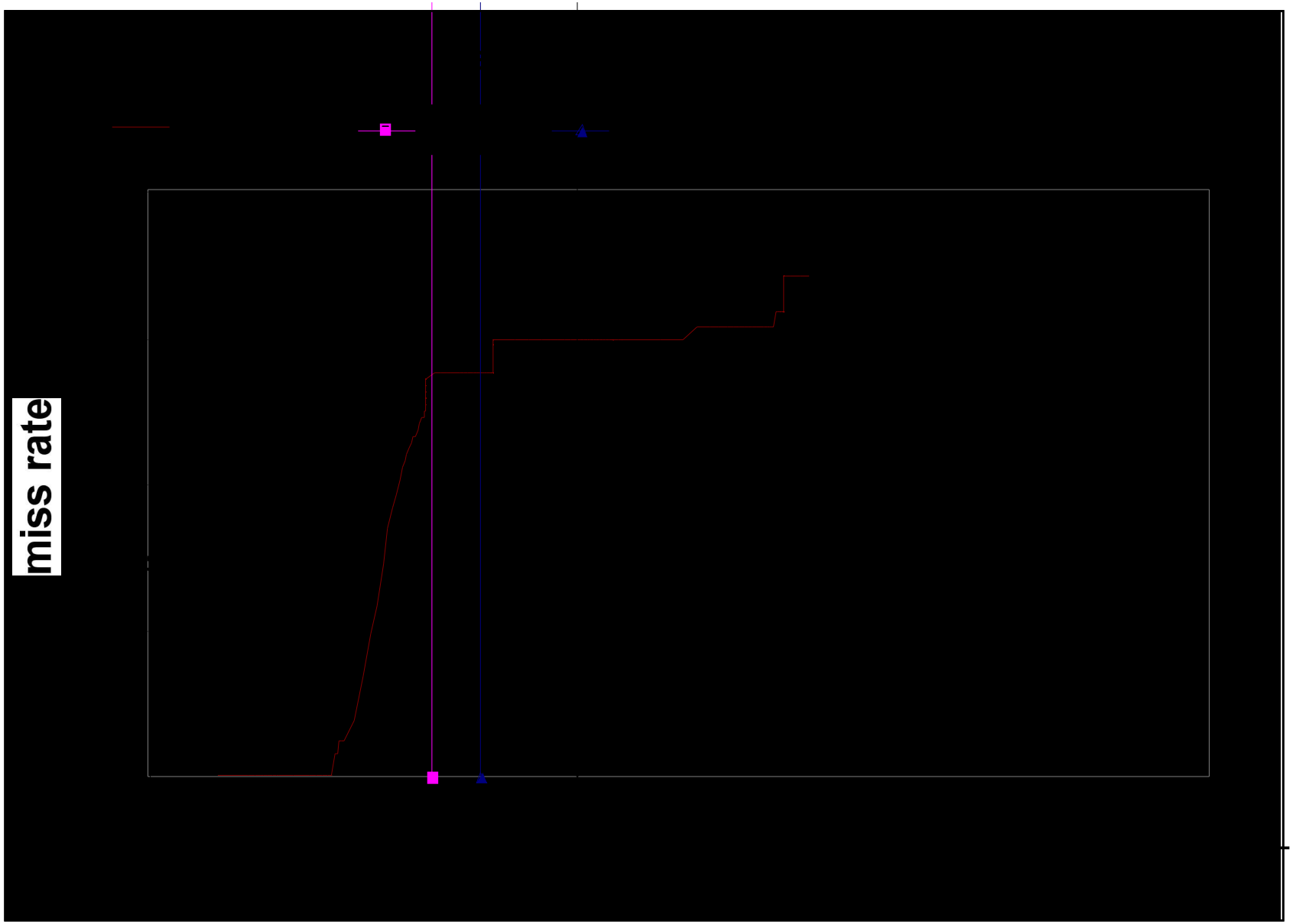
Data Locality Models

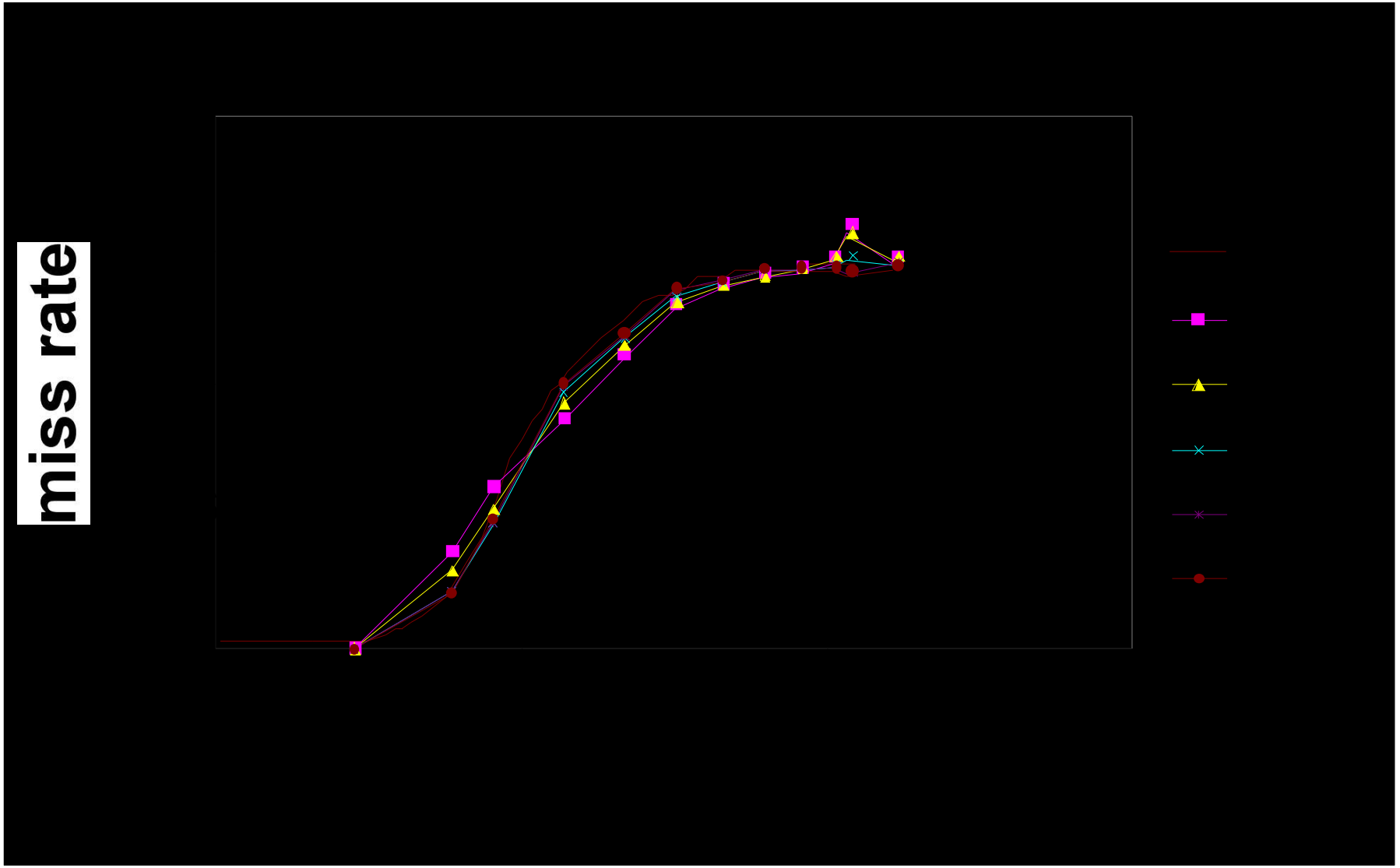
- *Frequency*
  - *frequency [Knuth 71, Cocke-Kennedy IBM74, Sarkar PLDI' 86, Seidl-Zorn ASPLOS' 98]*
  - *pair-wise affinity [Thabit 81, Calder+ ASPLOS' 98], hot data streams [Chilimbi+ PLDI' 01]*
  - *NP-hardness [Thabit 81]*
  - *the harsh limit of heuristics [Petrank-Rawitz POPL' 02]*
    - *observation: frequently used ≠ frequently used together*
- *Affinity groups*
  - *compile-time optimal [Ding&Kennedy IPDPS' 01]*
  - *hierarchical and consistent [Zhong+ LCPC' 03]*

Miss Rate Prediction Across All Program Inputs

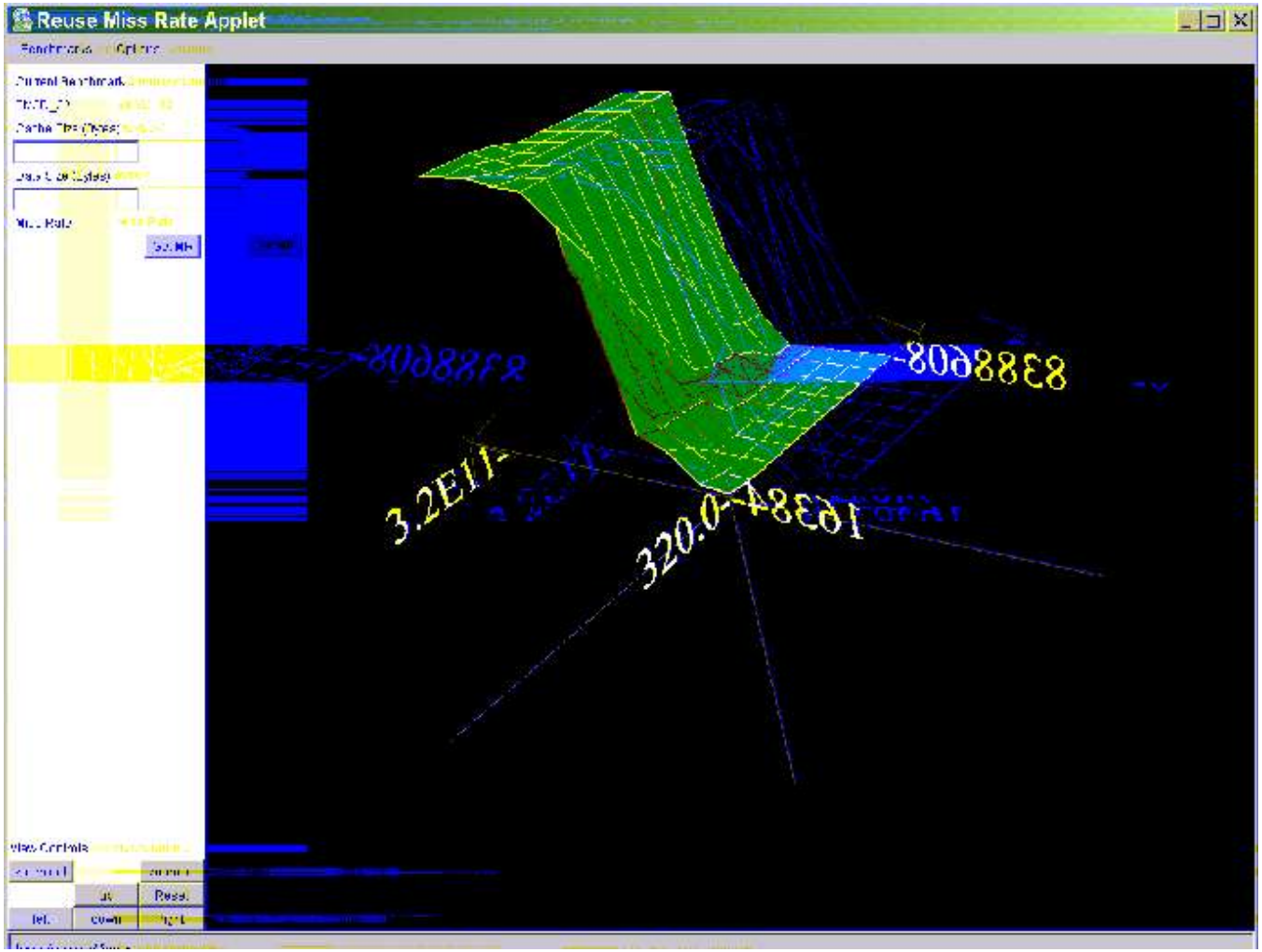
[Zhong, Dropsho, & Ding, PACT 2003]

Based on Reuse Signature Pattern
[Ding&Zhong PLDI 2003]


miss rate


miss rate

A Web-based Interactive Tool



- *http://www.cs.rochester.edu/research/locality*

Summary

- *Long distance reuses*
  - *reflects affinity relation of data*
  - *determines cache utilization and bandwidth demand*
- *Improvement*
  - *applying computation fusion and data regrouping across whole programs and at run time*
- *Analysis and prediction*
  - *correlation and prediction of reuse signatures*
- *On-going research*
  - *limit of program locality*
  - *program phase analysis*
  - *fine-grained and dynamic data management*

The End

Thank you