1

Neighborhood Prefetching

David M. Koppelman

Electrical & Computer Engineering Department

Louisiana State University

Baton Rouge, LA U.S.A.

Formatted 0:02, 17 October 2000 from pren˙ppact.
1

Hardware Prefetching

Prefetch:

Fetching a cache line in anticipation of its use.

Motivation:

Cache misses becoming larger component of execution time.

Multiprocessors add interconnect and protocol delays.

Challenges:

Prefetching addresses that will be used. (Easy)

Avoiding addresses that won't be used. (Hard)

Formatted 0:02, 17 October 2000 from pren˙ppact.

Address Determination in Some Existing Schemes

Sequential

In£nite sequential sequence.

Sequential sequence indicated in **bold**.

100, 3000, **101**, 2000, 3000, **102**, **103**, 6, . . .

Stride

In£nite stride sequence.

Stride 10 sequence indicated in **bold**.

100, 3000, **110**, 2000, 3000, **120**, **130**, 6, . . .

Real address reference sequences are £nite, guaranteeing useless prefetches.

Unneeded Prefetch Problems

Unneeded Prefetch Problems in All Systems

Resource Consumption. (Cache ports, interconnect bandwidth, etc.)

Eviction of useful lines.

Unneeded Prefetch Problems in Multiprocessors

Adds to false sharing.

Formatted 0:02, 17 October 2000 from pren˙ppact.

Neighborhood Prefetch

Advantages

- Larger variety of address sequences than stride.
- Less tendency to prefetch unneeded lines.

Formatted 0:02, 17 October 2000 from pren˙ppact.

Neighborhood Prefetch

Execution of following code fragment . . .

```
!  Iteration 1:  r2 = 0x2000, Line Size 0x100 Characters
!
A: load r1, [r2]
   ...
B: load r3, [r2+0x500]
   ...
C: load r4, [r2-0x700]
```

. . . generates . . .

Address Seq.: 0x2000, . . ., 0x2500, . . ., 0x1900, . . .

Formatted 0:02, 17 October 2000 from pren˙ppact.

Neighborhood Prefetch Terminology

Term:		*Example*
Base:	Address used as reference.	0x2000
Initiator:	Instruction accessing base address.	`A: load r1, [r2]`
Offset:	Distance from base, in lines.	0, . . ., +5, . . . -7
Neighborhood:	Set of small-magnitude offsets.	$\{0, +5, -7\}$

Formatted 0:02, 17 October 2000 from pren˙ppact.

Consider . . .

```
!  Iteration 1:  r2 = 0x2000 (Base), Line Size 0x100 Characters
!
A: load r1, [r2]  ! (Initiator)
   ...
B: load r3, [r2+0x500]
   ...
C: load r4, [r2-0x700]
```

. . . generates . . .

Address Seq.: 0x2000, . . ., 0x2500, . . ., 0x1900, . . .

Neighborhood: $\{0, +5, -7\}$

Notice that the neighborhood . . .

Consider . . . and

```
!  Iteration 2:  r2 = 0x8000 (Base), Line Size 0x100 Characters
!
A: load r1, [r2]  ! (Initiator)
   ...
B: load r3, [r2+0x500]
   ...
C: load r4, [r2-0x700]
```

. . . generates . . .

Address Seq.: 0x8000, . . ., 0x8500, . . ., 0x7900, . . .

Neighborhood: $\{0, +5, -7\}$

Notice that the neighborhood . . . is the same in both cases.

Base address and neighborhood used to construct prefetch addresses.

Neighborhood Prefetch Outline

Hardware monitors addresses that miss the cache.

Hardware determines neighborhoods and stores them in PC-indexed table.

Table checked on miss; if entry found used for prefetching.

Formatted 0:02, 17 October 2000 from pren˙ppact.

Neighborhood Prefetching Advantages

Superset (loosely) of existing schemes.

- Can detect sequential patterns: $\{0, 1, 2, 3, \ldots\}$.
- Can detect stride patterns: $\{0, 10, 20, 30, \ldots\}$.
- Can detect arbitrary patterns: $\{0, 10, 20, 50\}$.

Better at avoiding unneeded and harmful prefetches.

- Prefetch candidates based on past usage by instruction.
- Other schemes inevitably fetch unneeded, possibly harmful, lines.
- Particularly important in multiprocessors: less false sharing.

Formatted 0:02, 17 October 2000 from pren˙ppact.

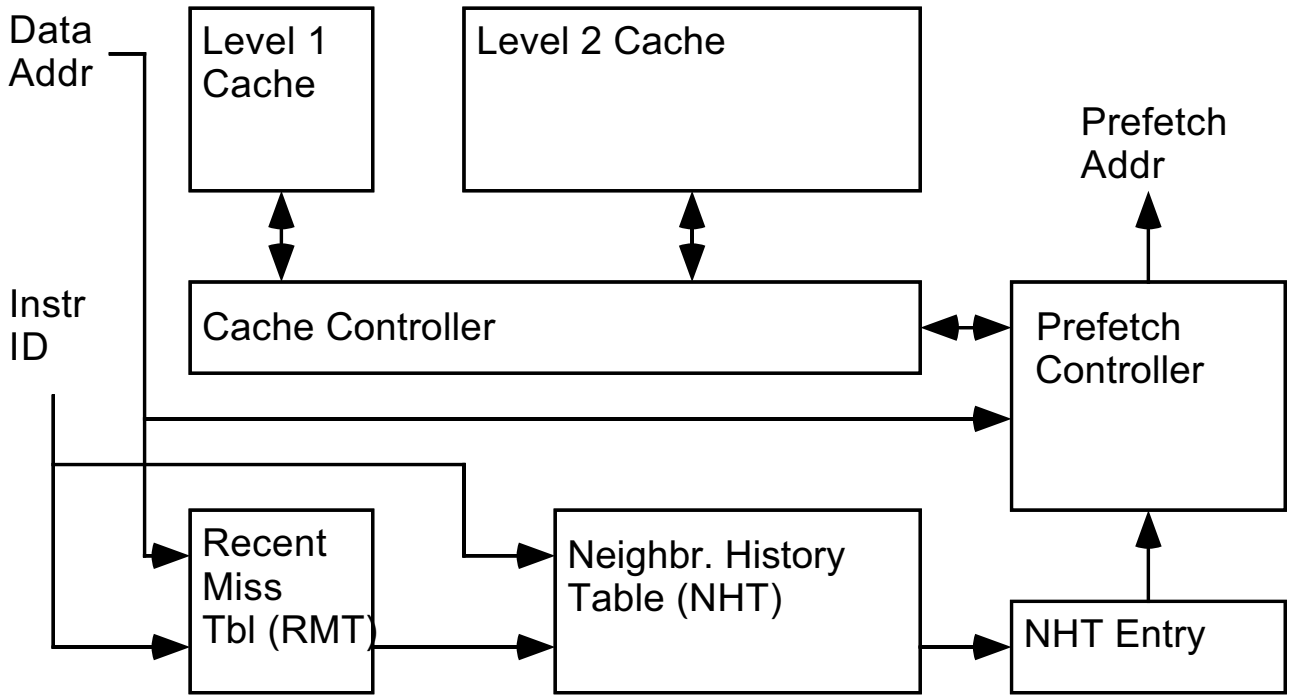
Neighborhood Prefetch Hardware

Main Parts

Recent Miss Table (RMT)—for determining neighborhoods.

Neighborhood History Table (NHT)—stores neighborhoods.

Prefetch Controller—handles prefetches.



Formatted 0:02, 17 October 2000 from pren˙ppact.

Neighborhood Prefetching Hardware

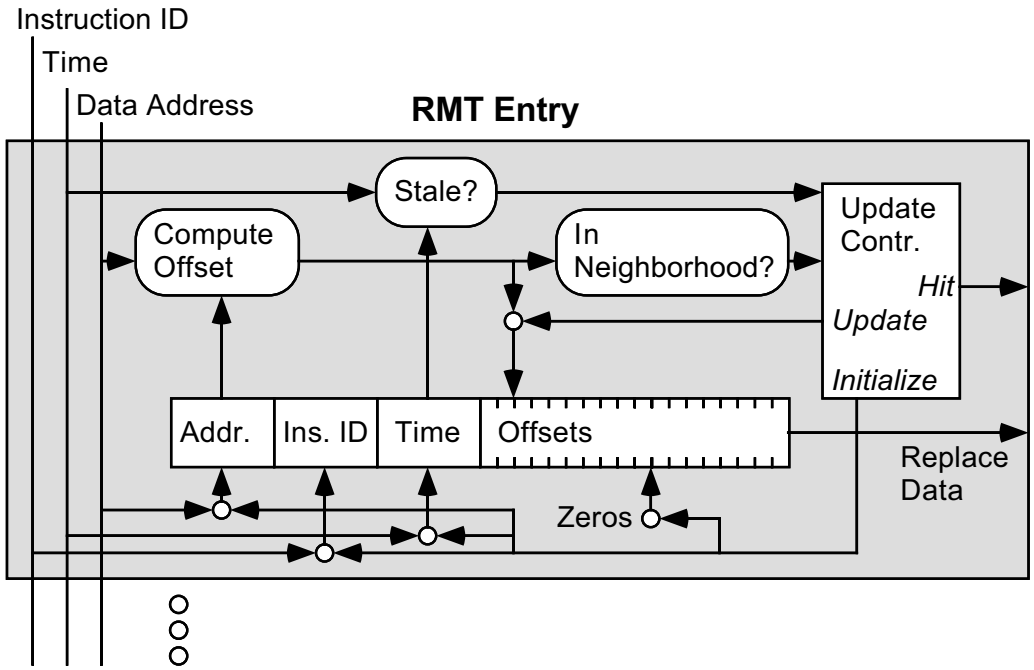
13

Recent Miss Table (RMT)

Entries for recently encountered neighborhoods.

Each entry holds:

- Base Address.
- Offsets.
- Initiating instruction.
- And other data.



On miss update existing entry or create new ones.

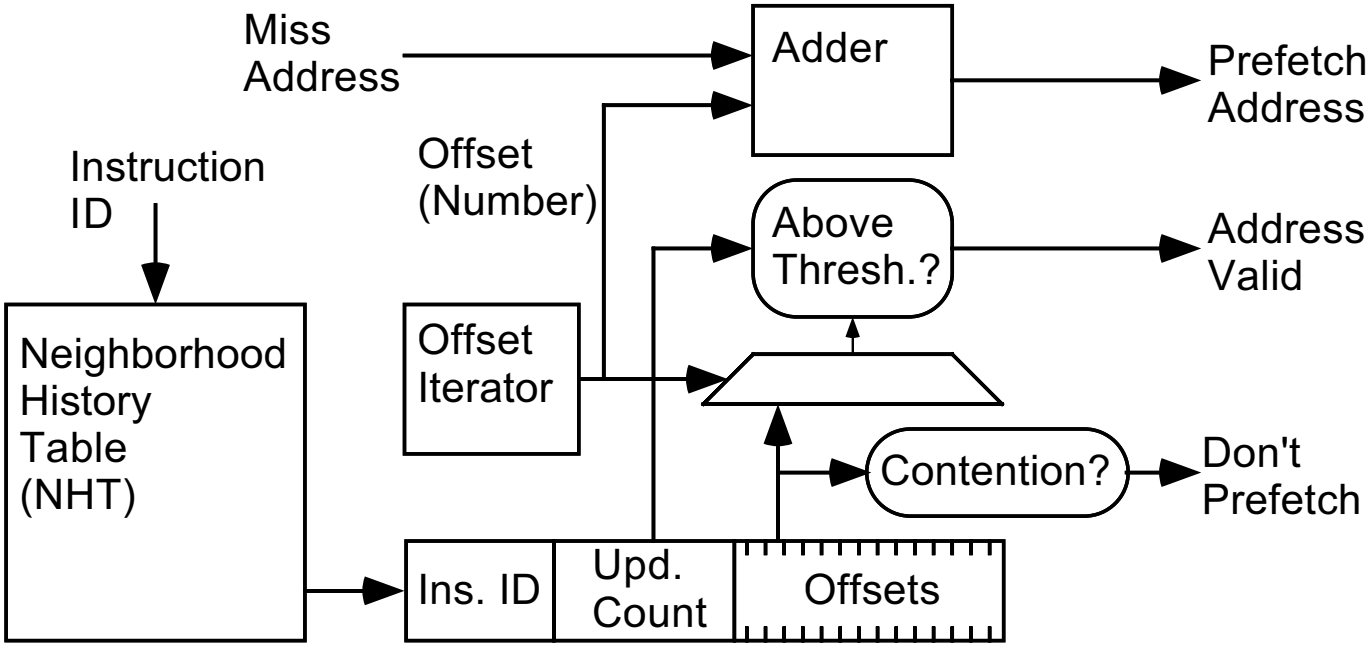
Entries occasionally moved to neighborhood history table.

Formatted 0:02, 17 October 2000 from pren˙ppact.

13

Neighborhood History Table

14



Neighborhood History Table

Entry for memory access instructions.

Retrieved on miss using address of missing instruction.

Used to construct prefetch addresses.

Formatted 0:02, 17 October 2000 from pren˙ppact.

14

Evaluation

15

Evaluated by execution driven simulation using Proteus.

Machine model:

- Sixteen-node multiprocessor.
- Full-map directory-based cache coherence.
- In-order execution, four-way issue, SPARC ISA.
- Virtual memory, nonblocking stores, blocking loads.
- Mesh topology.

Execution Bottlenecks

- Cache and Network Latency (all accesses)
- Hot Spots at Memories (application dependent)

Formatted 0:02, 17 October 2000 from pren˙ppact.

15

Base Con£guration Parameters

16

Simulation Parameter	Value
System Size	16 processors
Network Topology	4×4 mesh
VM Page Size	2^{12} bytes
TLB Capacity	64 entries
TLB Replacement	LRU, fully assoc.
Cache Size	2^7 sets
Cache Associativity	8, LRU Repl.
Cache Line Size	64 bytes
L1 Cache Hit Latency	1 cycle
L2 Cache Hit Latency	7 cycles
Total L2 Miss Latency	50 (min), 135 (typ)
Directory Size	full map
Completion Buffer	5 stores
Raw Memory Latency	10 cycles
Protocol Message Size	8 bytes (plus data)
Network Interface Width	4 bytes
Network Link Width	4 bytes
Hop Latency	20 cycles (plus waiting)
Neighborhood Size	16 offsets
NHT Size	256 entries
RMT Size	8 entries

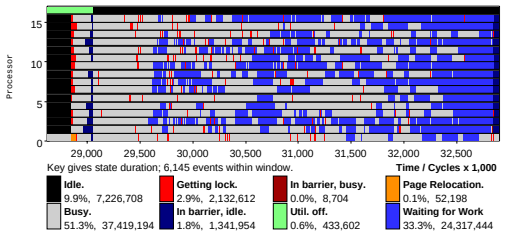
16

16

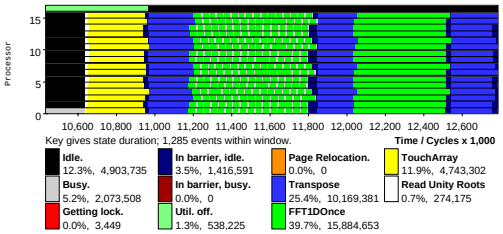
SPLASH-2 Benchmarks Used

17

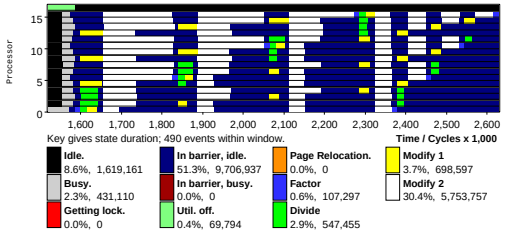
Cholesky - Processor States



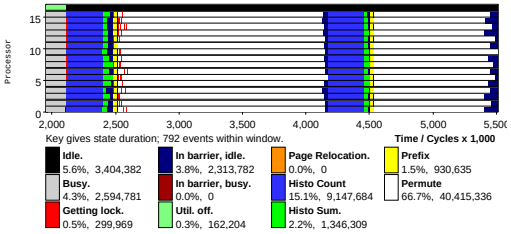
FFT: Processor States (65536 elements)



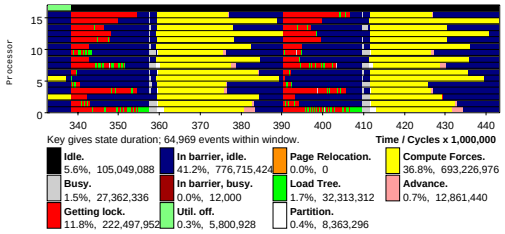
LU: Processor States (128 x 128)



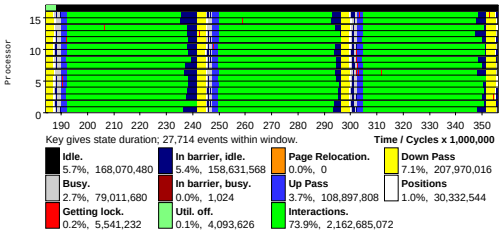
Processor States, Radix 262,144 Keys



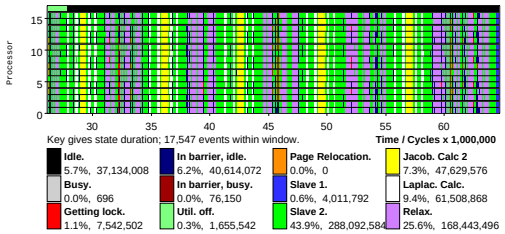
Processor States: Barnes, 16384 Particles



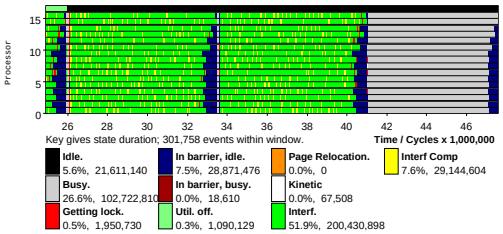
Processors States: FMM, 16384 Particles



Processor States: Ocean, 258 x 258



Processor States: Water N Squared, 512 Molecu



Formatted 0:02, 17 October 2000 from pren˙ppact.

17

Evaluation

18

Ran SPLASH-2 Benchmarks

Compared:

- Neighborhood Prefetch (Cache Size $\frac{7}{8}$ Other Systems)
- Adaptive Sequential Prefetch (Preupgrades, Tuned for System)
- Conventional System

Experiments Presented

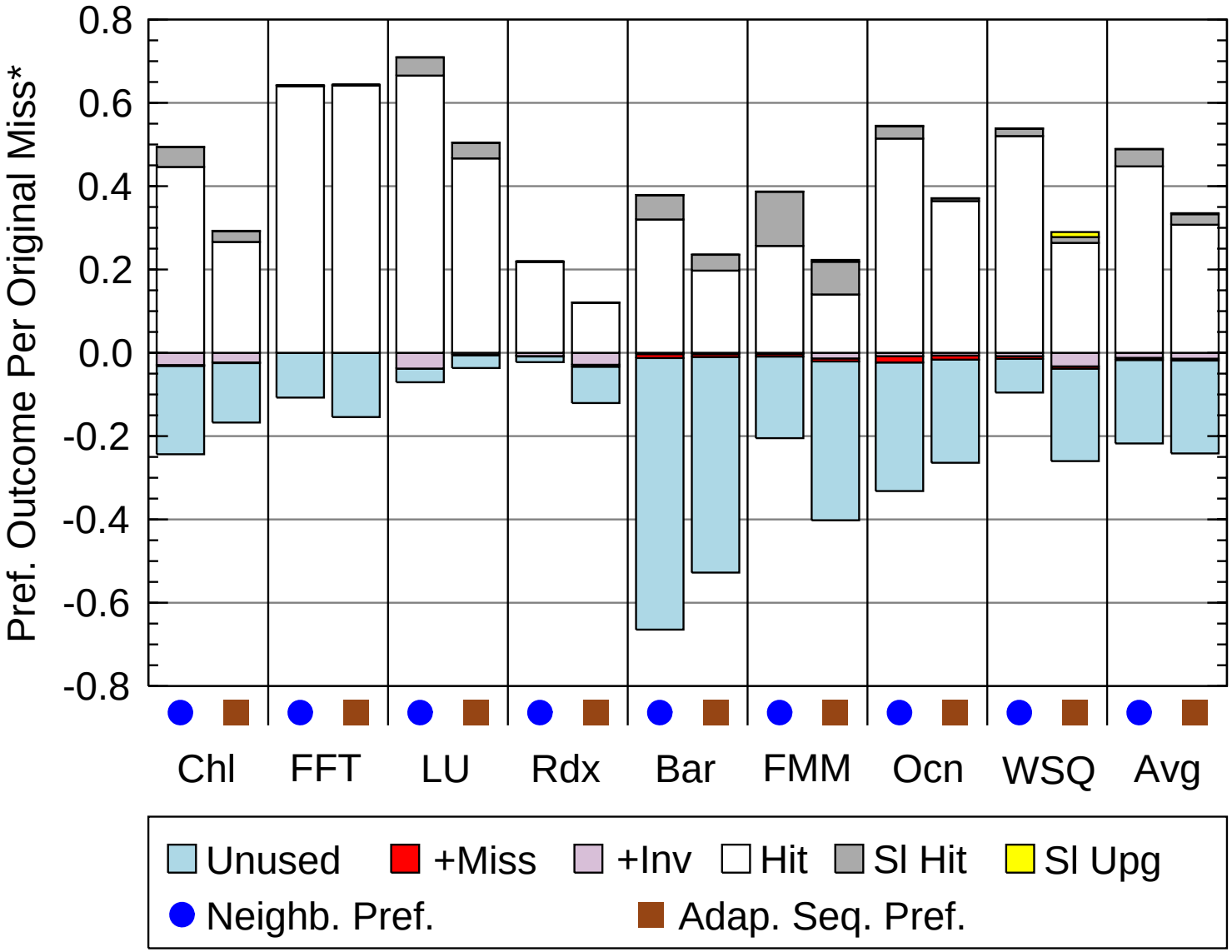
- Base: 64KB: 8 way assoc. $\times$ 64 byte lines $\times$ 128 sets.
- Cache Size: 8KB (2^4 sets) to 1MB (2^{11} sets).
- Line Size: 8 bytes to 1024 bytes.

Formatted 0:02, 17 October 2000 from pren˙ppact.

18

Base: Prefetch Outcome

19

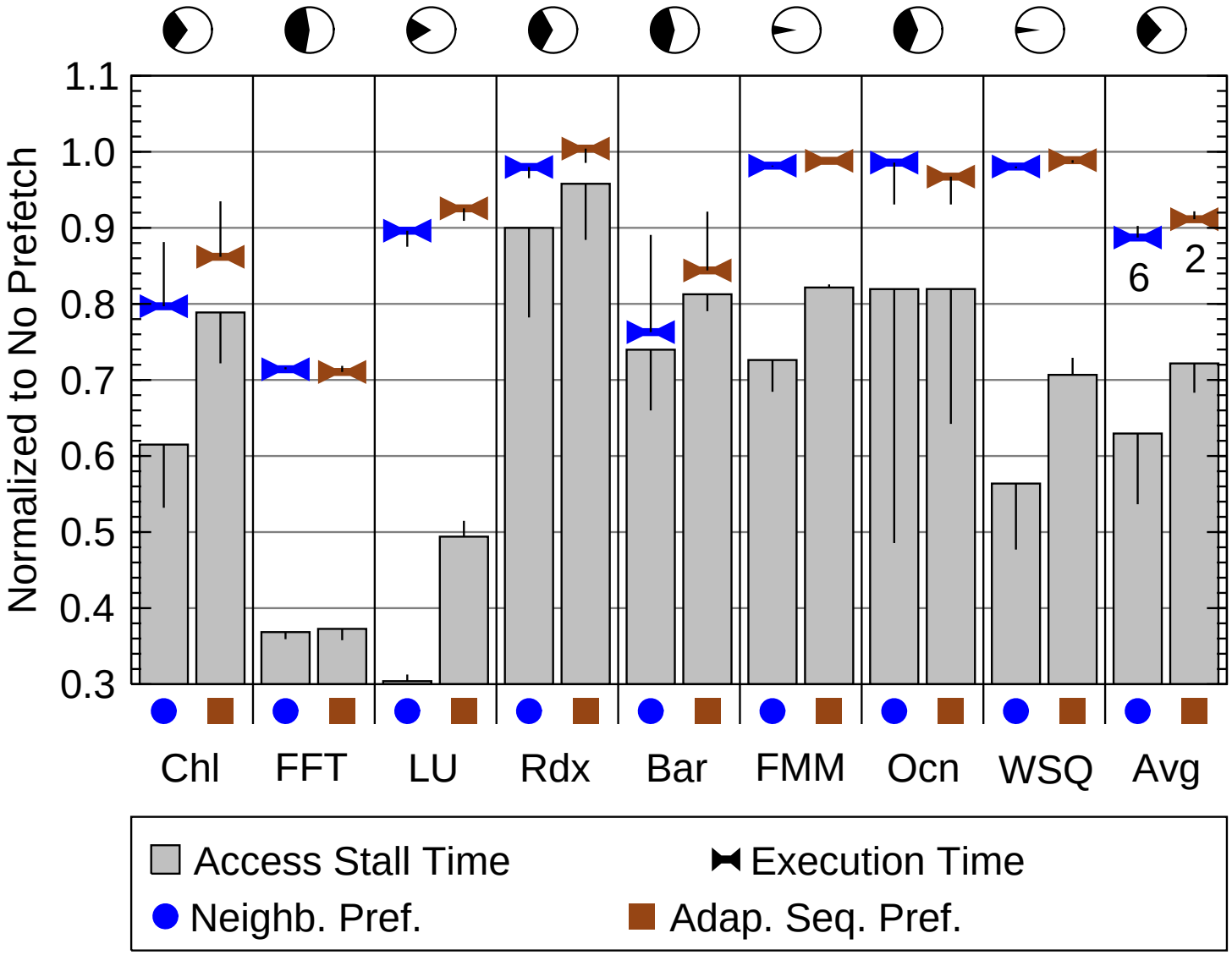


Formatted 0:02, 17 October 2000 from pren˙ppact.

19

Base: Normalized Stall and Execution Time

20

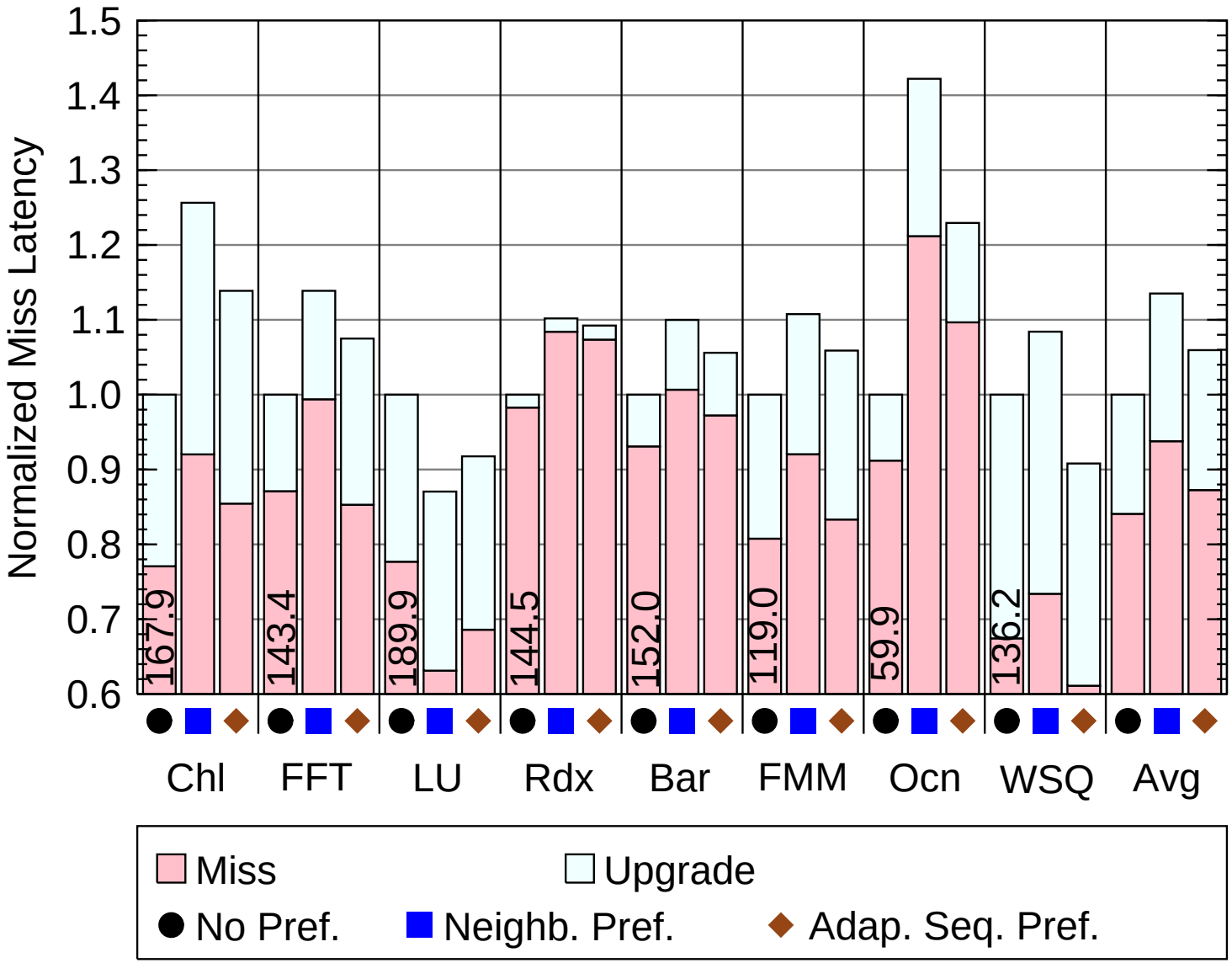


Formatted 0:02, 17 October 2000 from pren˙ppact.

20

Base: Miss Latency

21

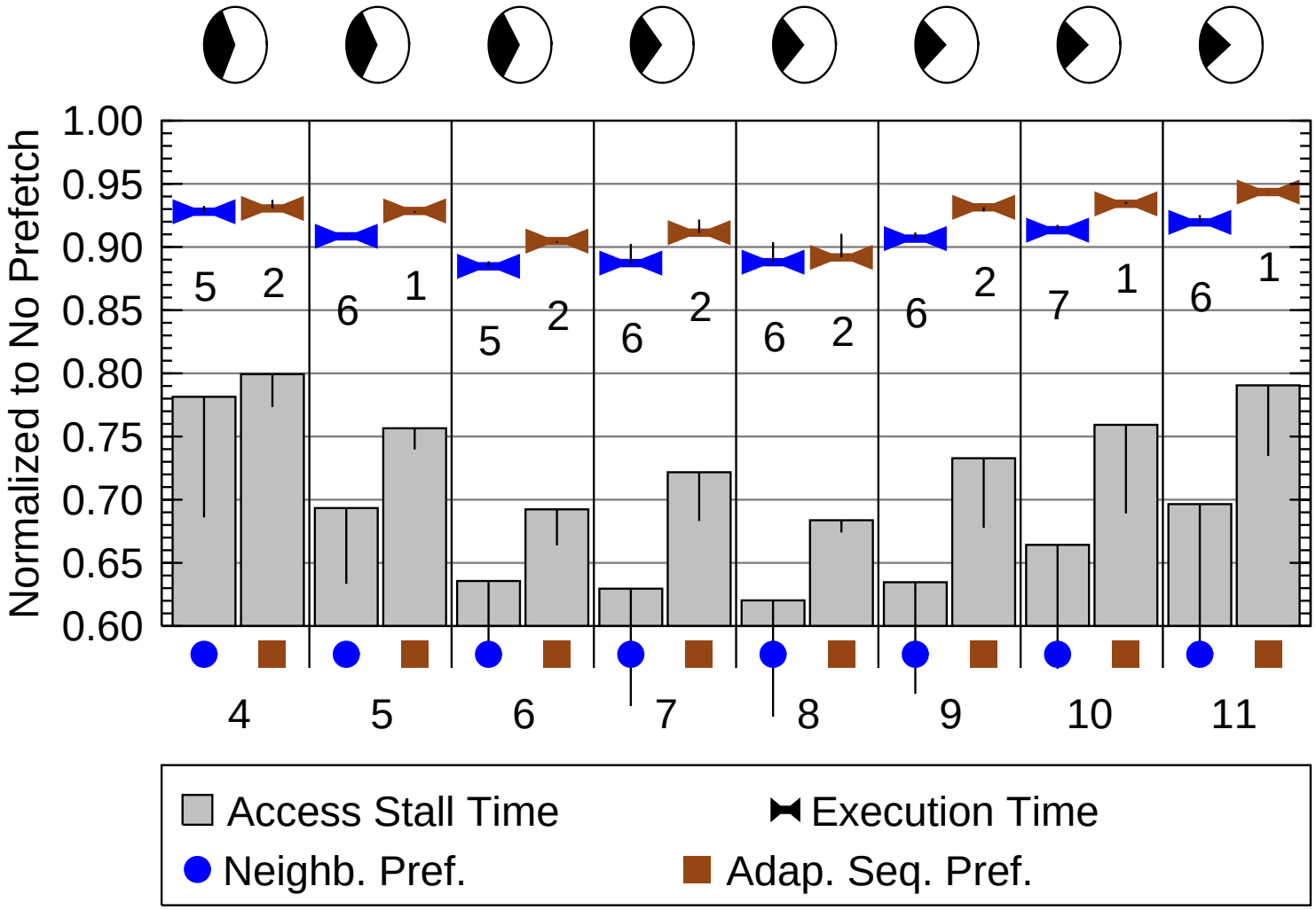


Formatted 0:02, 17 October 2000 from pren˙ppact.

21

Cache Size: Normalized Stall and Execution Time

22

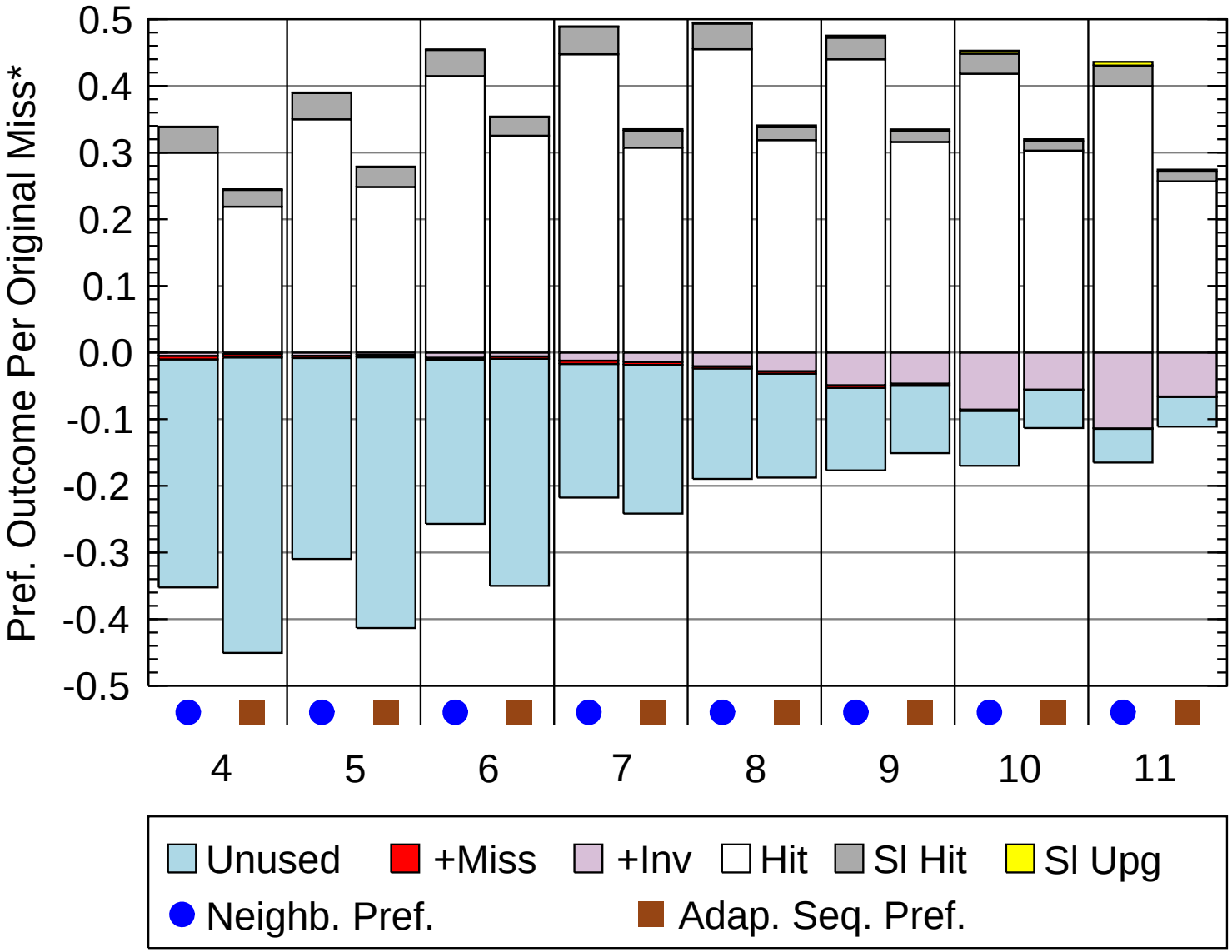


Formatted 0:02, 17 October 2000 from pren˙ppact.

22

Cache Size: Prefetch Outcome

23

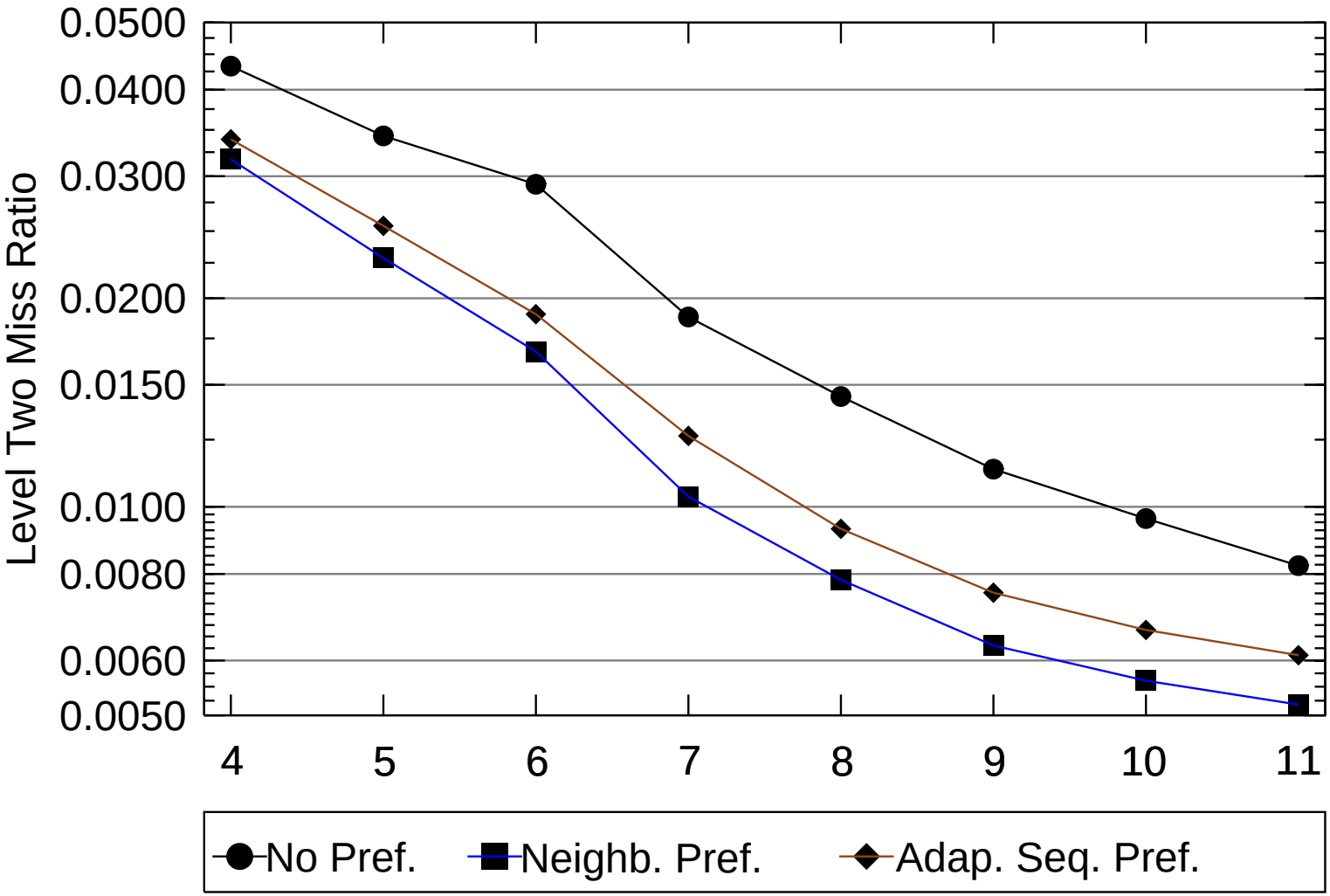


Formatted 0:02, 17 October 2000 from pren˙ppact.

23

Cache Size: Miss Ratio

24

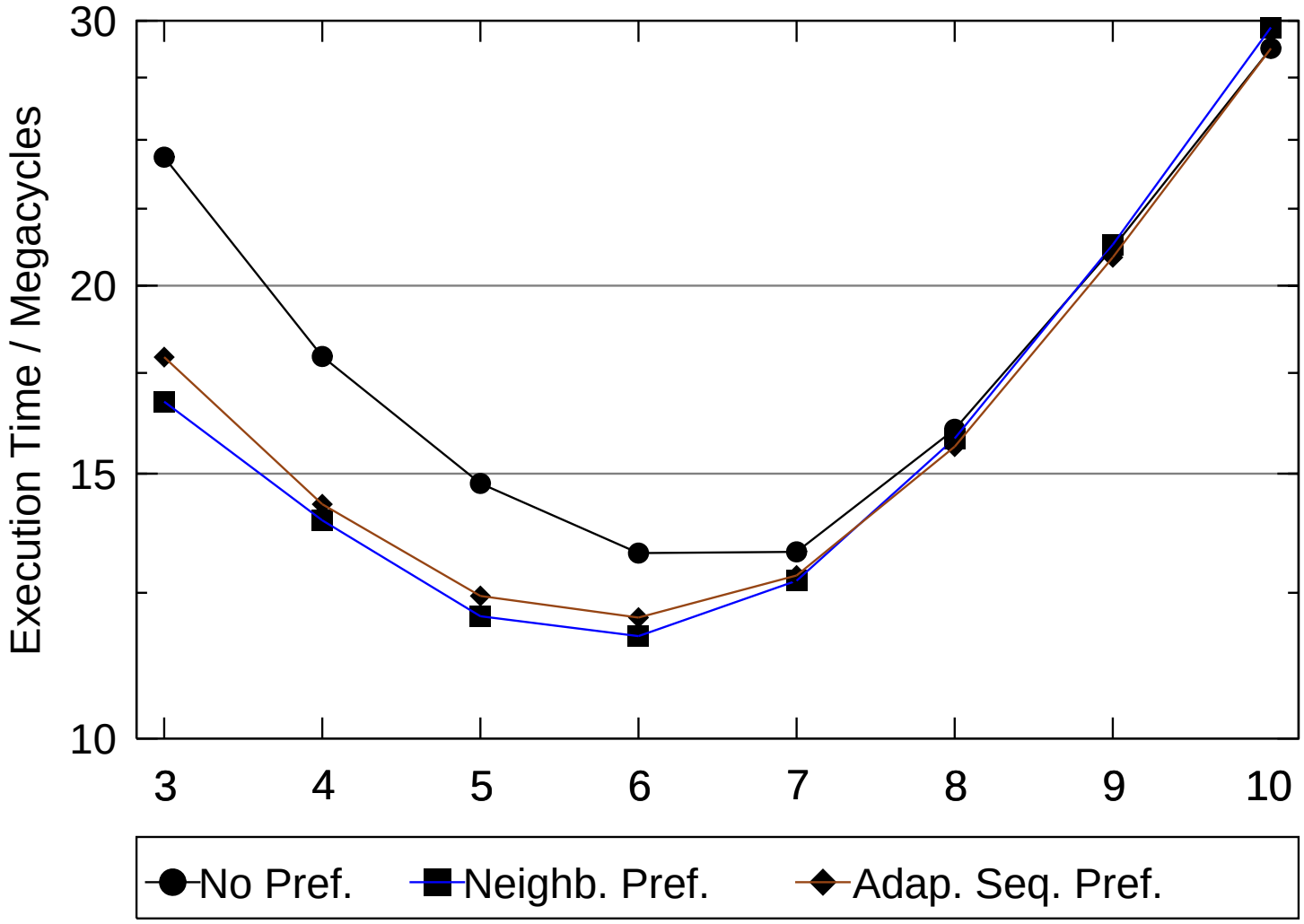


Formatted 0:02, 17 October 2000 from pren˙ppact.

24

Line Size: Execution Time

25

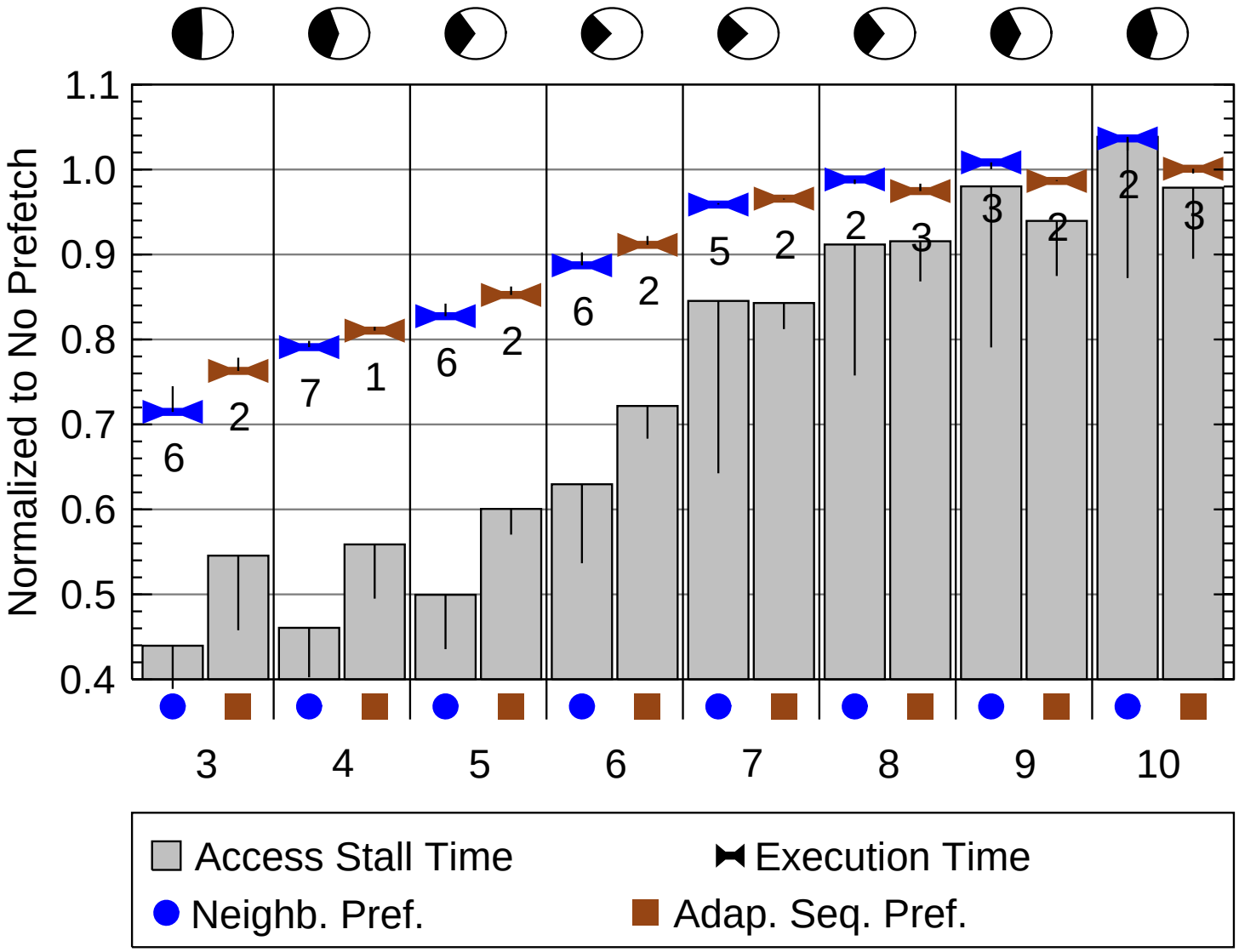


Formatted 0:02, 17 October 2000 from pren˙ppact.

25

Line Size: Normalized Stall and Execution Time

26

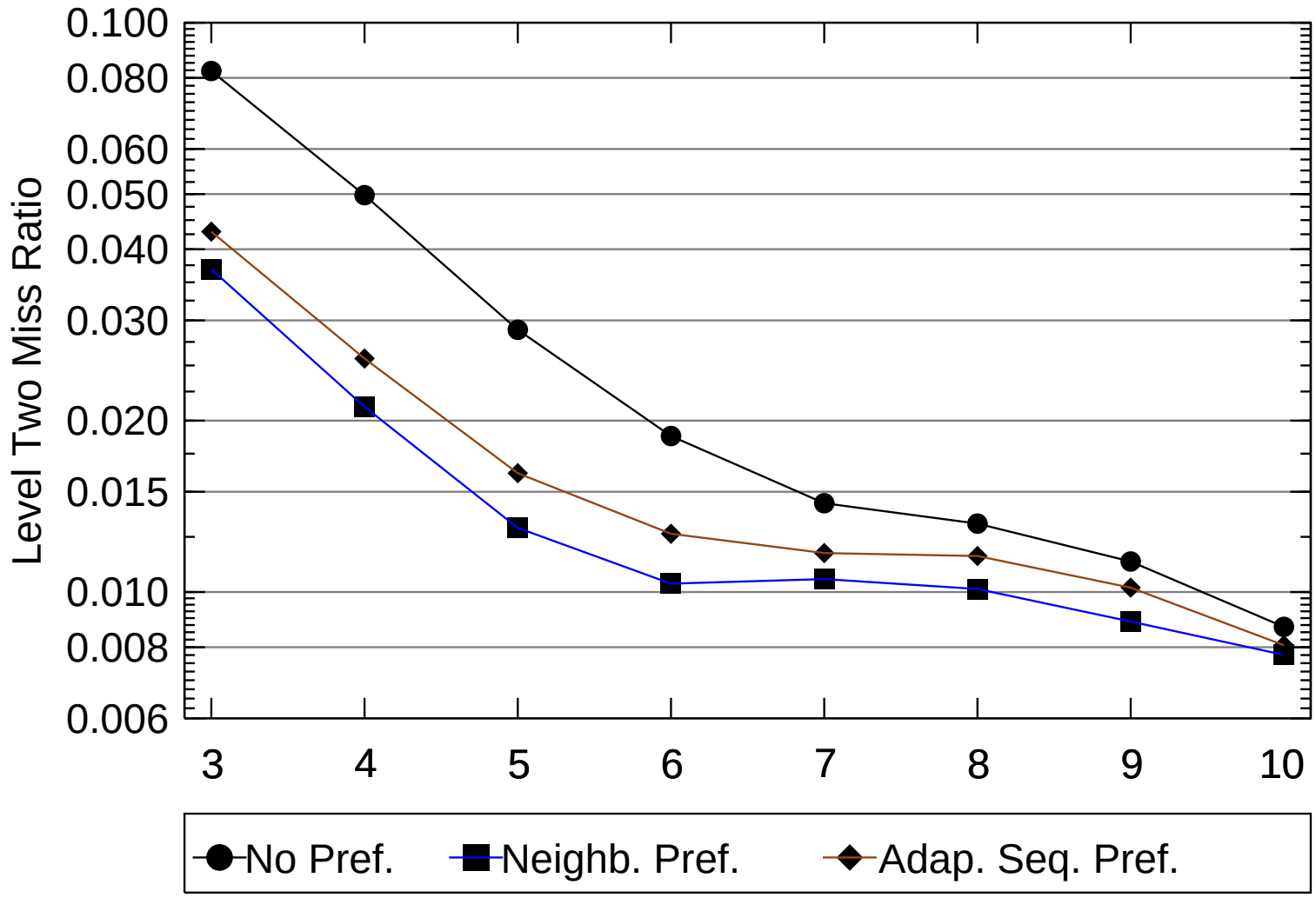


Formatted 0:02, 17 October 2000 from pren˙ppact.

26

Line Size: Miss Ratio

27

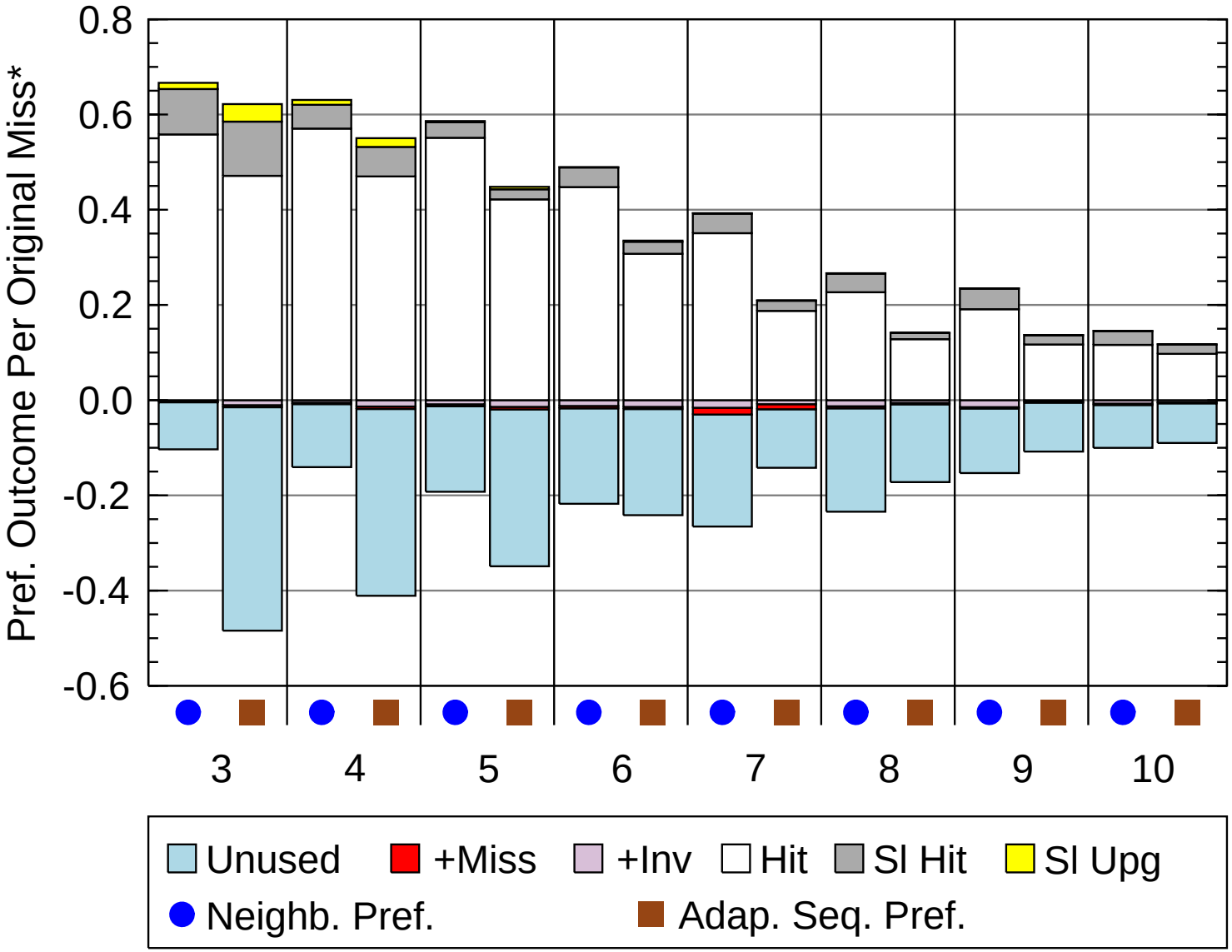


Formatted 0:02, 17 October 2000 from pren˙ppact.

27

Line Size: Prefetch Outcome

28

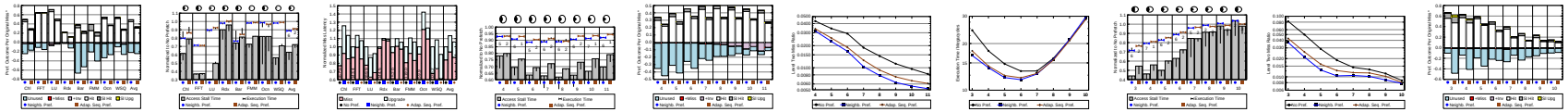


Formatted 0:02, 17 October 2000 from pren˙ppact.

28

Conclusions

29



Conclusions

Neighborhood Prefetch handles wider variety of reference patterns.

Less prefetching of unneeded lines.

Reduces miss ratio by nearly 50%.

Reduces execution time by 10% or more.

Performance better than adaptive sequential prefetch . . .
. . . though implementation considerably more complex.

Future Work: Prefetch on Hit

Wallpaper prefetch? (Repeat neighborhoods.)

Formatted 0:02, 17 October 2000 from pren˙ppact.

29