Session 3: Design Tradeoff Analysis

Characterization, Tracing, and Optimization of Commercial I/O Workloads
H. Huang, M. Teshome, J. Casmira and D.R. Kaeli
Northeastern University Computer Architecture Research Laboratory
Brian Garrett and William Zahavi
EMC Corporation

Trace-driven Performance Exploration of a PowerPC 601 OLTP Workload on Wide Superscalar Processors
J.H. Moreno, M. Moudgill, J.D. Wellman, P.Bose, L. Trevillyan
IBM T.J. Watson Research Center

Performance Analysis of Shadow Directory Prefetching for TPC-C
Dan Friendly - University of Michigan
Mark Charney - IBM Research

Evaluating Branch Prediction Methods for an S390 Processor using Traces from Commercial Application Workloads
Rolf B. Hilgendorf, IBM Entwicklung GmbH, Boeblingen, Germany
Gerald J. Heim, Wilhelm Schichard-Institut fuer Informatik, Universitaet Tuebingen, Tuebingen, Germany

Characterization, Tracing and Optimization of Commercial I/O Workloads

David Kaeli
Northeastern University Computer Architecture Research Laboratory
Boston, MA
kaeli@ece.neu.edu

Participants in this project

- NUCAR
  - Hua Huang
  - Melaku Teshome
  - Jason Casmira
- EMC Corporation
  - Brian Garrett
  - William Zahavi

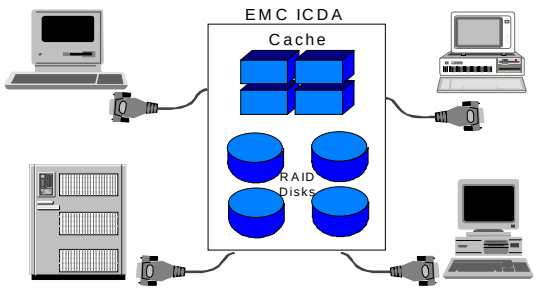
Overview

- Introduction
- ICDA architecture and performance
- I/O tracing and workload characterization
- I/O caching
- Performance modeling and results
- Summary and future work

Introduction

- Commercial applications (e.g., transaction processing) have become heavily dependent on high speed, reliable, mass storage (TB - PB)
- Integrated Cached Disk Arrays (EMC Symmetrix) have become the design of choice to supply high-end storage needs for business

ICDA Architecture



ICDA Architectures

- Fault tolerant and recoverable
  - RAID
  - Battery backed up cache
  - Dual redundant controllers
  - Hot pluggable disks and controllers
- High performance and scaleable
  - Large amount of cache memory (4 GB)
  - Aggressive caching schemes
  - Multi-host connect
  - Partitionable

ICDA Performance

- Cache architecture
  - Aggressive prefetching
  - Fully associative organization
  - Complete I/O space directory
- Our work investigates improved cache organization and prefetching to improve I/O response time
- We begin by capturing reference pattern traces of I/O workloads

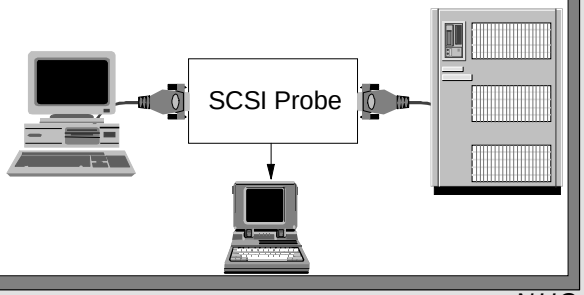
NUCAR

I/O Tracing

- Options
  - Host-based tracing (e.g., TNF on Solaris)
  - ICDA resident tracing (e.g., EMC Symmetrix)
  - SCSI and Fiberchannel tracing (e.g., I-Tech, Ancott)
- We currently use an SCSI-bus sniffer to capture I/O traces
  - Non-intrusive
  - Ability to capture multiple I/O streams
  - Developed a complete set of post processing tools

NUCAR

SCSI Bus Tracing



NUCAR

I/O Reference Trace Characteristics

- Variable spatial locality
- Long range temporal locality
- Reasons
  - Main memory acts as a filter
  - Spatial locality captured in a page
  - Majority of the references are for data (versus instructions)

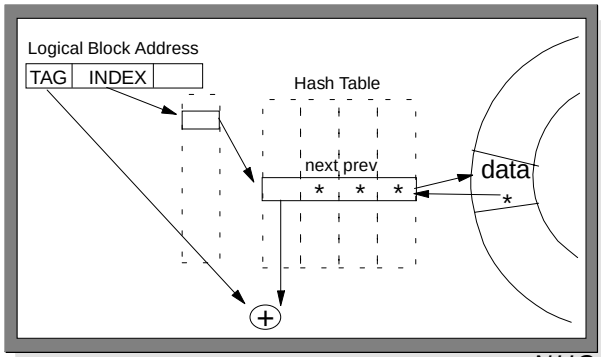
NUCAR

I/O Caching

- Currently commercial ICDA designs employ a fully associative LIFO or LRU replacement and a full I/O space directory
- Provides very fast lookup, but this implementation is very expensive
- Instead we want to reduce this cost by using:
  - a set-associate lookup
  - an inverted page table organization
  - efficient linked structure for performing lookup

NUCAR

Inverted Page Table



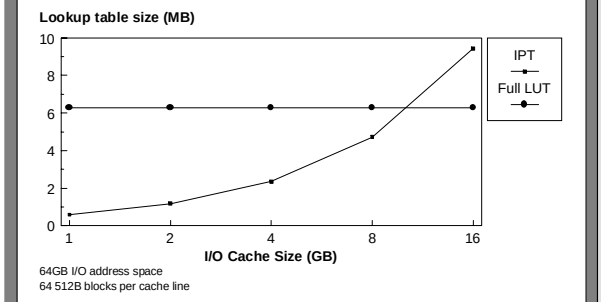
NUCAR

I/O Caching

- Existing LUT strategies
  - table size is proportional to the size of the addressable disk space
- Linked-List Lookup Structure
  - uses an inverted page table structure
  - worst case lookup time is bound by the number of hash classes and chain length
  - table size is proportional to the size of the cache
  - have also investigated tree-based structures

NUCAR

Lookup Table Size Growth



64GB I/O address space
64 512B blocks per cache line

NUCAR

Cache Prefetching

- Detect sequential access
- Miss prefetching
  - prefetch next sequential block on a miss
- Tagged prefetching
  - miss prefetching + prefetching on first reference to prefetched lines
- Prefetching can introduce cache pollution

NUCAR

Prefetch Buffer Filter

- Small, fully associative, table to act as both a stream buffer and a victim cache
- Increases the associativity of hot cache lines
- Uses a Least Recently Prefetched replacement algorithm

NUCAR

Workloads

- DSSLOAD
  - Decision Support System Loading
  - Restoring a large database system
  - Highly sequential behavior (approx. 70%)
  - Approximately 50% reads, 50% writes
  - Fixed I/O sizes (32 blocks)
- DSSQUERY
  - Decision Support System Query
  - Querying a large database system
  - Highly sequential behavior (approx. 70%)
  - All read accesses
  - Fixed I/O sizes (32 blocks)

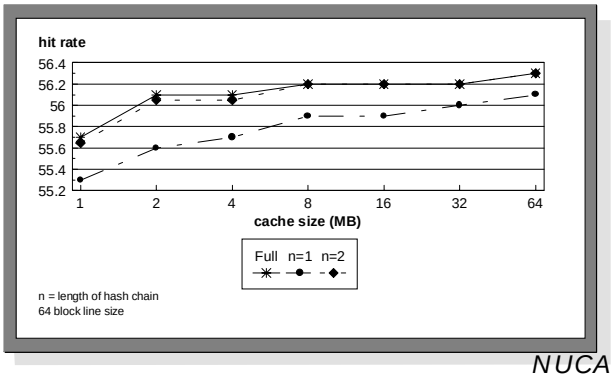
NUCAR

Workloads

- OLTP
  - On-Line Transaction Processing
  - Point of sale transactions
  - Smaller amount of sequential behavior (approx. 30%)
  - Smaller and variable I/O sizes (avg. 8 blocks)
  - Approximately 75% reads, 25% writes

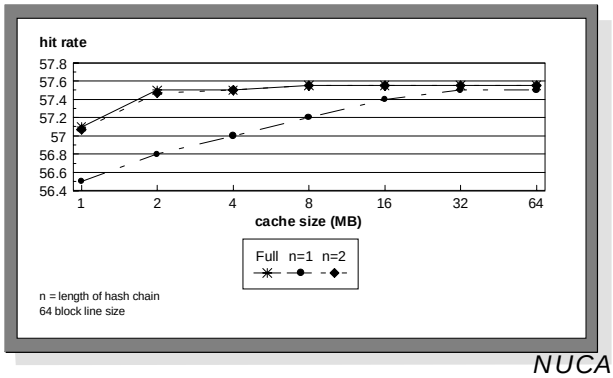
NUCAR

DSSLOAD



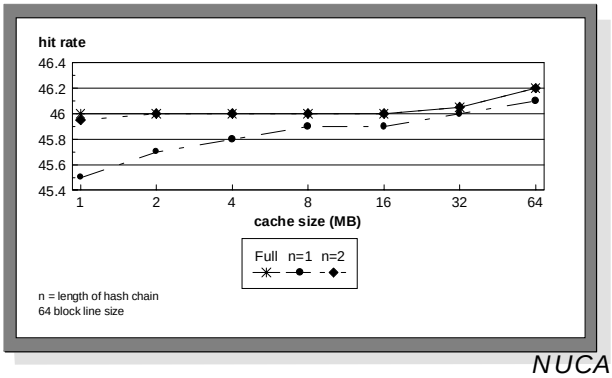
NUCAR

DSSQUERY



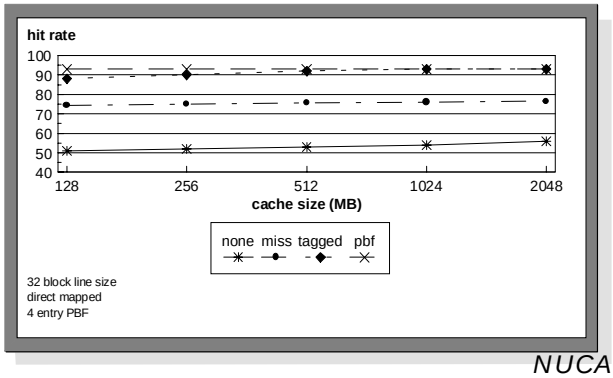
NUCAR

OLTP



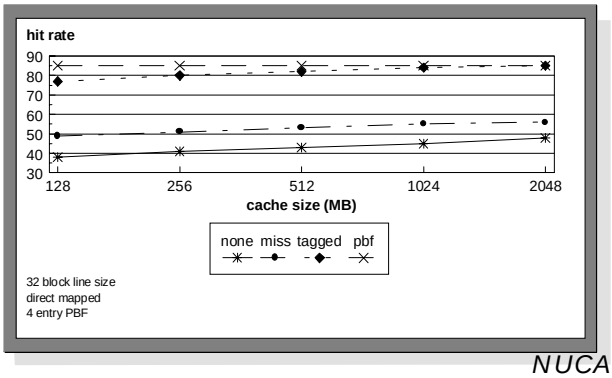
NUCAR

DSSLOAD



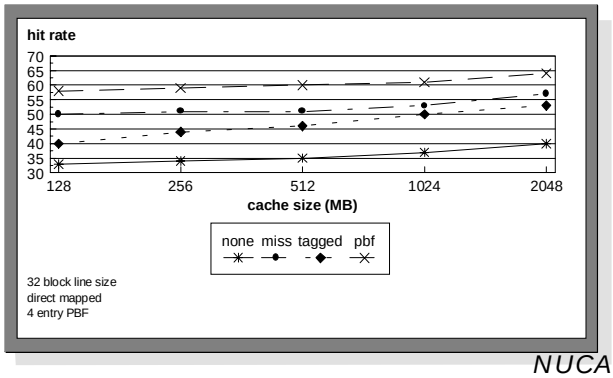
NUCAR

DSSQUERY



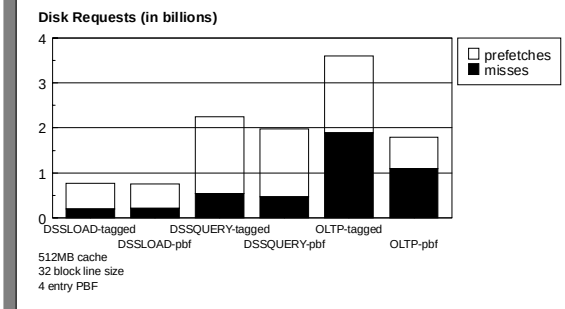
NUCAR

OLTP



NUCAR

Disk Traffic



NUCAR

Summary

- I/O commercial workloads present a different set of characteristics than those present in processor-based workloads
- IPT cache organizations can provide similar hit rates to full table implementations and can reduce memory requirements
- Prefetching and filtering can greatly enhance low associativity cache organizations

Future Directions

- Cycle-based ICDA model
- Adaptive prefetching schemes
- Multi-stream caching
- COTS NOWs to perform multimedia caching

Trace-driven performance exploration of a PowerPC 601 OLTP workload on wide superscalar processors

J. H. Moreno, M. Moudgill, J.D. Wellman, P.Bose, L. Trevillyan
IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598

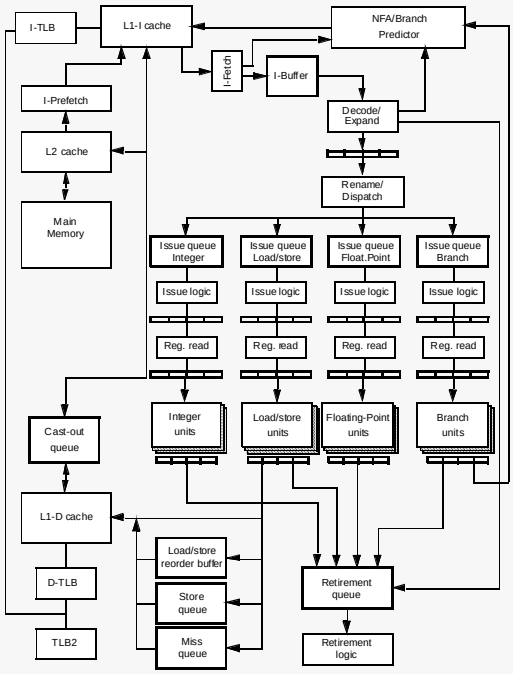
Why yet another superscalar processor model and performance evaluation?

- Superscalar processors continue dominating the field
  - No apparent likelihood of ending superscalar paradigm in near future
  - Continuing improvements in features and capabilities
  - Certain aspects getting easier due to number of transistors available
  - Existing programs (binary compatibility)
- Need for evaluating new implementation challenges
  - High frequency objectives
  - New structures and algorithms
  - Wider instruction issue
- Need to understand impact of various classes of workloads
  - "Commercial" workloads
  - ....

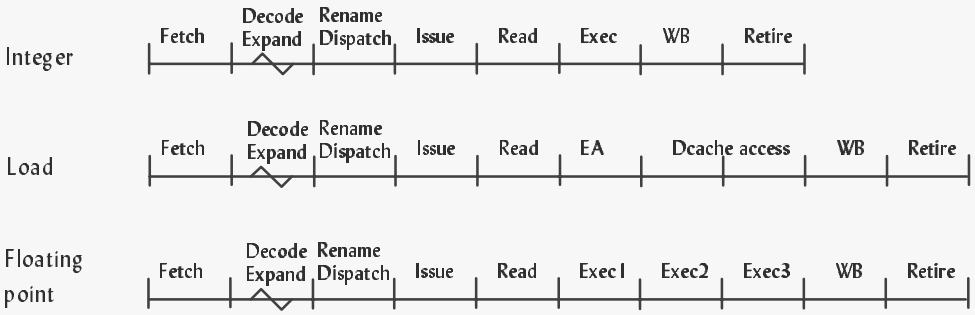
The MET: Microarchitecture Exploration Toolset

- Collection of tools for exploration of microarchitecture features
- Trace-driven and execution-driven tools
- Fast simulation: >300 M inst/hour
- Intended to support early exploration of processor organizations
  - detailed model of generalized pipeline
  - *trends* among results instead of their magnitudes

Processor organization



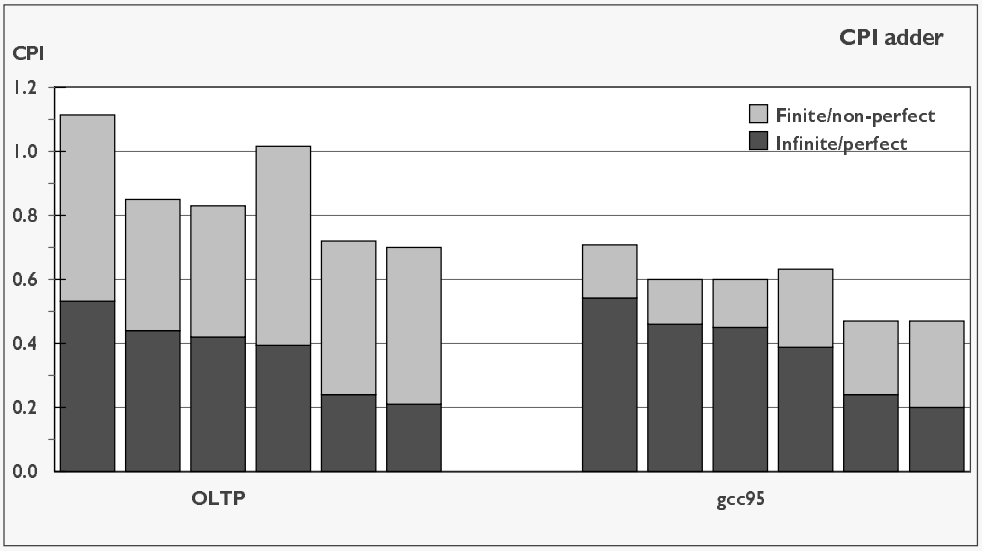
Pipeline stages



OLTP and GCC traces

	OLTP	GCC
Length	172 M instructions, user and kernel space	1212 M instructions, user space
Branch instructions	18.9 %	21.6 %
Branches taken	44.3 %	56.3 %
Instrs. in kernel space	22.1 %	n/a
Memory access instructions	34.8 %	27.1 %
Load/store multiple instructions	1.6 %	1.6 %
String instructions	1.4 %	0.3 %
Load/store w/update instrs.	1.7 %	2.8 %
Average block size	5.3 instrs.	
Mispredicted instructions, addresses	No	Yes

CPI adder for various processor configurations



- Much larger adder in the case of OLTP workload

Miss rates (per 1000 instructions): 64K L1, 2M L2, 128 entries TLB, 8K entries BHT

	OLTP	GCC
I1	21.3	8.3
I2	3.1	0.02
D1	22.4	9.8
D2	1.8	0.02
I TLB	1.8	~0
D TLB	4.5	1.5
Conditional branch misprediction	5.3 %	7.0 %

Exploration space (in this presentation)

Issue policy	Width	Cache size	Branch prediction
	Fetch/Dispatch/Retire	L1-I, L1-D, L2	
Class-order	4/4/6	64K, 64K, 2M	8192 entry BHT, 4096 BTAC
Out-of-order	8/8/12	128K, 128K, 4M	Perfect
	12/12/16	128K, 128K, Inf	
		Inf, Inf, Inf	

Widths	Units	Ports	Queues	Physical registers
Fetch/ Dispatch/ Retire	FX/FP/LS/BR	Data cache and TLB	Issue/ Retire/ IBuf	GPR/FPR/CCR/SPR
4/4/6	3/2/2/2	2	20(12)/128/24	80/80/32/64
8/8/12	6/4/4/4	4	40/160/48	128/128/64/96
12/12/16	8/4/6/4	6	60/160/72	128/128/64/96

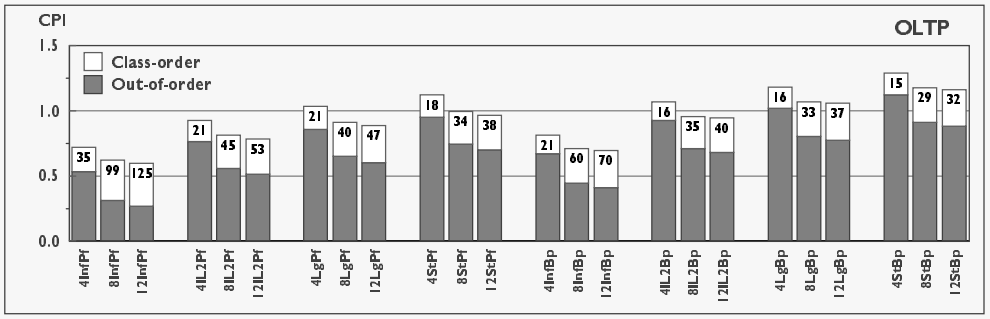
Other parameters (examples)

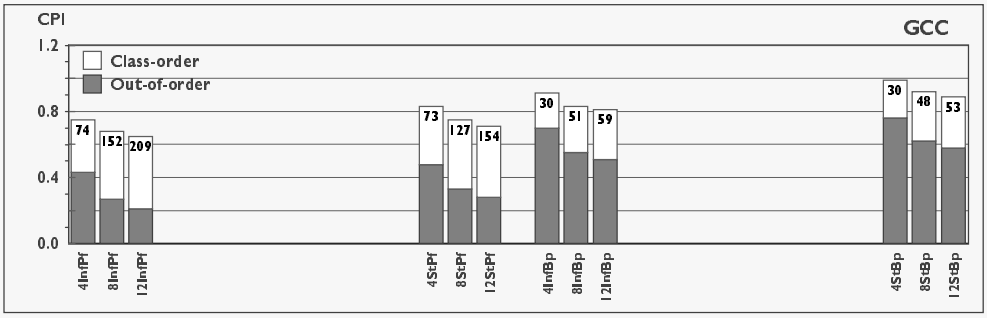
Sizes	I-prefetch buffer (entries)	4
	Miss queue, cast-out queue (entries)	8
	Store queue, reorder buffer (entries)	31
	D/I-TLBs (entries)	128
	TLB2 (entries)	1024
	L1-I/D, L2-cache line size (bytes)	128
	Page size (bytes)	4096

Latencies	I-prefetch buffer latency (cycles)	1
	D/I-TLBs miss penalty (cycles)	4
	TLB2 miss penalty (cycles)	40
	L1-I/D cache miss penalty (cycles)	8. 7
	L2 cache miss penalty (cycles)	40

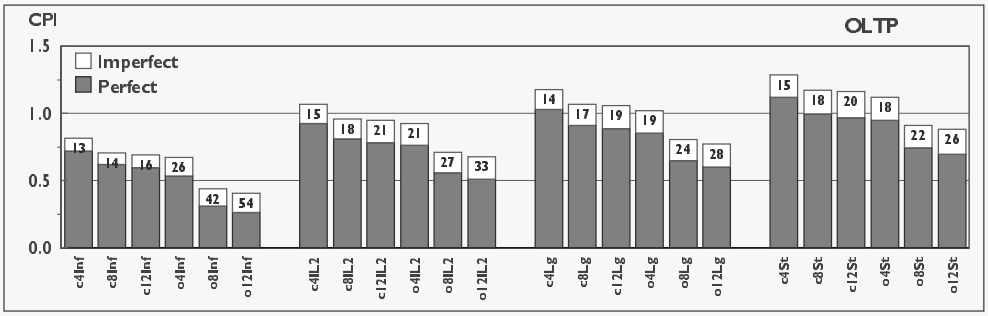
Branch prediction	BTAC (entries)	4096
	LR stack size (entries)	32
	Branch history table (entries)	8192

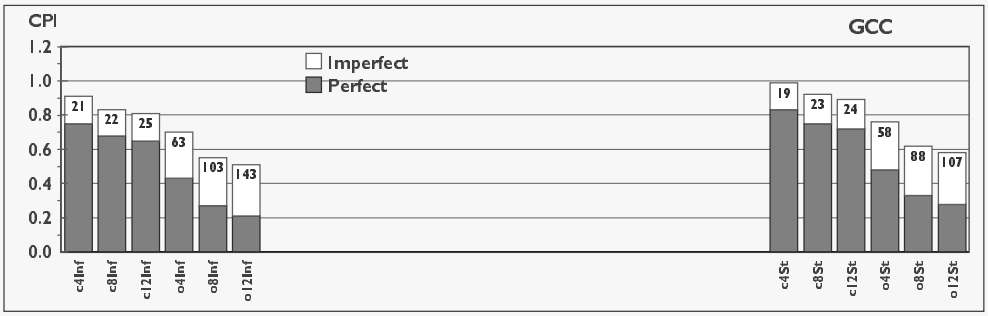
CPI adders due to issue policy (as % of base case)



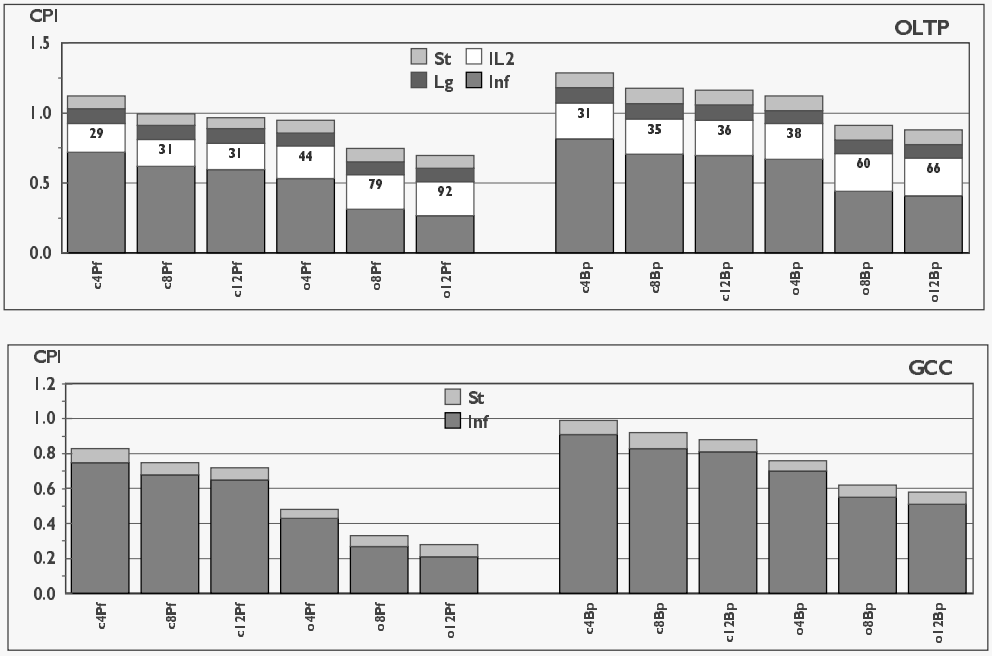


CPI adders due to branch prediction (as % of base case)

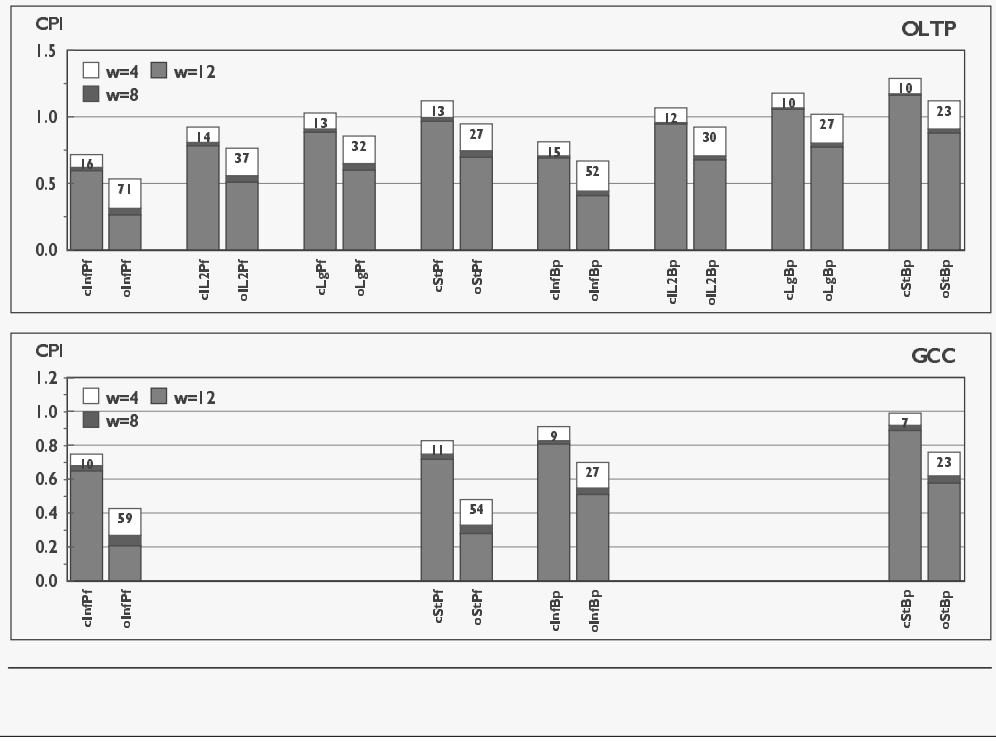




CPI adders due to cache size (as % of base case)



CPI adders due to processor width (as % of base case)



In OLTP workload

- "Least-aggressive" configurations considered
  - 15 to 32% degradation due to class-order issue
    - more severe degradation expected for in-order policy
  - 15 to 26% degradation due to imperfect branch predictor
  - 30 to 66% degradation due to finite L1 cache (128K)
  - 10 to 23% degradation due to processor width
  - Diminishing benefits beyond dispatching eight operations per cycle
    - conventional instruction fetching mechanism
- Still many microarchitecture issues to investigate in detail

Observations

- Clear differences in OLTP behavior relative to GCC
  - memory penalties in OLTP shadow other effects
- Caveats due to use of traces
  - length
  - number of traces (just one in this presentation)
  - observability
  - in OLTP
    - no mispredicted paths
    - time scaling
  - in GCC
    - no kernel code

Summary

- Environment for early exploration
  - fast, flexible
  - *trends* among aggressive superscalar organizations
- Behavior of OLTP workload
  - very different from others (i.e., SPEC)
  - different microarchitecture tradeoffs
- Aggressive superscalar
  - buildable?
  - need to quantify potential performance from realizable implementation
  - need to identify/develop features that provide better return

CPI results in OLTP workload

Issue policy	Width	Bp: 2-bit branch history table (8192 entries)				Pf: Perfect branch predictor			
		Inf	IL2	Lg	St	Inf	IL2	Lg	St
c: Class-order	4	0.82	1.07	1.18	1.29	0.72	0.93	1.03	1.12
	8	0.71	0.96	1.07	1.18	0.62	0.81	0.91	1.00
	12	0.70	0.95	1.06	1.17	0.60	0.79	0.89	0.97
o: Out-of-order	4	0.67	0.93	1.02	1.12	0.53	0.77	0.86	0.95
	8	0.44	0.71	0.81	0.91	0.31	0.56	0.65	0.75
	12	0.41	0.68	0.77	0.88	0.27	0.51	0.60	0.70

Performance Analysis of Shadow Directory Prefetching for TPC-C

Daniel H. Friendly
ites@eecs.umich.edu

High Performance Systems Group
Advanced Computer Architecture Laboratory
University of Michigan
Ann Arbor, MI 48109-2122

Mark Charney
charney@watson.ibm.com

IBM
T.J. Watson Research Center
Yorktown Heights, NY

February 1, 1998

Outline

- Shadow Directory Prefetching (SDP)
- Experimental Model
- Results
- Conclusions

SDP - The Concept

- Proposed by Puzak and Charney '97
- Recognize and exploit recurring memory access patterns
- Maintain history of missed cache lines
- Miss to line in history initiates prefetching of subsequent lines

SDP - The Implementation

- Capture miss history in L2 directory
  - Add Next Miss Address to each L2 directory entry
  - Add Last Miss Registers in L2

- Each L2 request
  - Returns requested line
  - Initiates prefetch of next miss address

- Confirmation message sent to L2 on prefetch use

Example

L2 Directory

Tag	NMA	Val.
A	B	0
B	C	0
C	??	0

Tag	NMA	Val.
A	B	1
B	??	0
C	??	0

Tag	NMA	Val.
A	B	1
B	C	1
C	??	0

L2 sees request for line A:

- Address of A held in last miss register

L2 sees request for line B:

- A's next miss address is updated with line address of B
- Address of B held in last miss register

L2 sees request for line C:

- B's next miss address is updated with line address of C
- Address of C held in last miss register

Example (continued)

Later...

- L2 sees request for line A

  L2 returns line A to L1

  A's next miss address field contains address of B

  Begin prefetch of line B

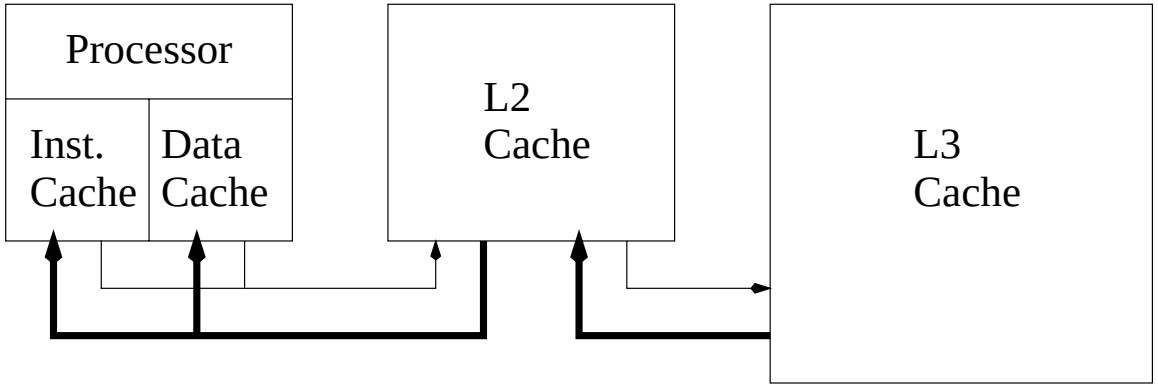
- L1 uses line B and sends confirmation to L2

  B's next miss address field contains address of C

  Begin prefetch of line C

- L1 uses line C and sends confirmation to L2...

Experimental Model



- Wide buses
- Fast Caches

Configuration Details

Processor Core

- 4-issue limited out of order superscalar, 2 FXU, 1 FPU, 1 LD/ST, 1 BRN, 8K-entry BHT, 8K-entry BTB, 10-entry link stack

L1 Caches

- 1 cycle load-use, 16K, 2-way assoc., 128B lines, Write through, Hit under miss, 256K-entry 2-way assoc. TLBs, 8-entry prefetch buffer

Unified L2 Cache

- 10 cycle hit time, 2M, 4-way assoc., 128B lines, Write back, Hit under miss, Dual ported directory, Maintains strict inclusion

Buses

- 64B reply buses, 32B request/store buses, Common L1-L2 buses clocked 2:1, L2-L3 buses clocked 3:1

TPC-C Workload

- PowerPC 604 Oracle trace
- 100 million instructions
- Base miss ratios

  I-Cache: 21.8% (18.4 million accesses)

  D-Cache: 9.6% (39 million accesses)

  L2 Cache: 3.7% (21.3 million accesses)

Baseline Results

Prefetch Algorithm	CPI	Improvement
No Prefetch	1.69	–
Next Sequential	1.62	4.1%
Shadow Directory	1.56	7.7%
Infinite L1s	0.97	42.6%

- SDP shows significant improvements over next sequential

Coverage and Accuracy

Coverage - Ratio of prefetch uses to base case misses

Accuracy - Ratio of prefetch uses to prefetches received

	I-Cache		D-Cache	
Prefetch Algorithm	I-Cov	I-Acc	D-Cov	D-Acc
Next Sequential	40%	77%	7%	22%
Shadow Directory	48%	73%	24%	47%

- SDP much more effective for data prefetching

Bus Utilization

- Prefetching tradeoff is lower latency but increased bandwidth
- L1-L2 reply bus is a bottleneck
  - 4 cycle latency for full line transfer
  - Not pipelined

Prefetch Algorithm	I-Traffic	D-Traffic	Reply Bus Util.
No Prefetch	100%	100%	17%
Next Sequential	121%	125%	23%
Shadow Directory	122%	130%	24%

Reduce Reply Bus Contention

- Keep prefetches from delaying demand requests
- Inhibit prefetch transfer if demand request pending
  - Demand request in L2 array
  - Reply coming from L3

Skip Distance

More than 10% of useful prefetches arrive "late"

- Alter shadow directory mapping
  - Instead of miss to next miss

    Map miss to N-th miss
  - With miss pattern A, B, C, D and skip distance of 1

    A maps to C, B maps to D

- Reduce latency of useful prefetches

Results

Prefetch Algorithm	CPI	Improvement
No Prefetch	1.69	–
Next Sequential	1.62	4.1%
Next Seq., Lower bus priority	1.60	5.3%
Shadow Directory	1.56	7.7%
SDP, Lower bus priority	1.54	8.9%
SDP, Lower Bus Priority, Skip 1	1.52	10.1%
Infinite L1s	0.97	42.6%

Conclusions

- SDP shown to boost performance by 10%
  - 25% of infinite performance
- SDP is almost twice as effective as next sequential
  - Data prefetching significantly improved

Evaluating Branch Prediction Methods for S/390 Processors using Traces from Commercial Application Workloads

R.B. Hilgendorf, IBM Entwicklung GmbH, Boeblingen, Germany

HILGENDO @ de.IBM.com

G.J.Heim, Wilhelm Schickard-Institut fuer Informatik, Universitaet Tuebingen, Tuebingen, Germany

HEIM @ Informatik.Uni-Tuebingen.de

OUTLINE

- Motivation
- The Traces
- Specifics of the ESA/390 Instruction Set
- Prefetch Prediction
- Results for Local History Prediction
- Results for Global History Prediction
- Hybrid Prediction
- Prediction for Changing Target Addresses
- Conclusion

Motivation

- Branch Prediction Studies for S/390 were done mainly during the 80's
- Can new Algorithms developed by
  - Yale Patt and his Students
  - Scott McFarling
  - and others

  be used for S/390 Processors
- How comparable are S/390 Workloads to traditional Specmark Evaluation

Trace Description

T1: Transaction Processing like Database Queries in Warehouse Management and Stock Keeping

T2: Interactive Usage in Program Development like Searches, Editing, Compilation, and Testing

T3: Transaction Processing like Hotel-Reservation Banking and Order Processing

T4: Batch-Jobs compiled from 130 single Jobs including Link- and Run- Steps as well as Assist Programs for Data Manipulation

Trace Statistics

Trace	T1	T2	T3	T4
# dynamic Instructions	1300881	1325359	1309178	1667468
# dynamic Branches	285528	321441	312865	359550
# static Branches	19176	27878	21202	15491
# Interrupts	35	65	97	115

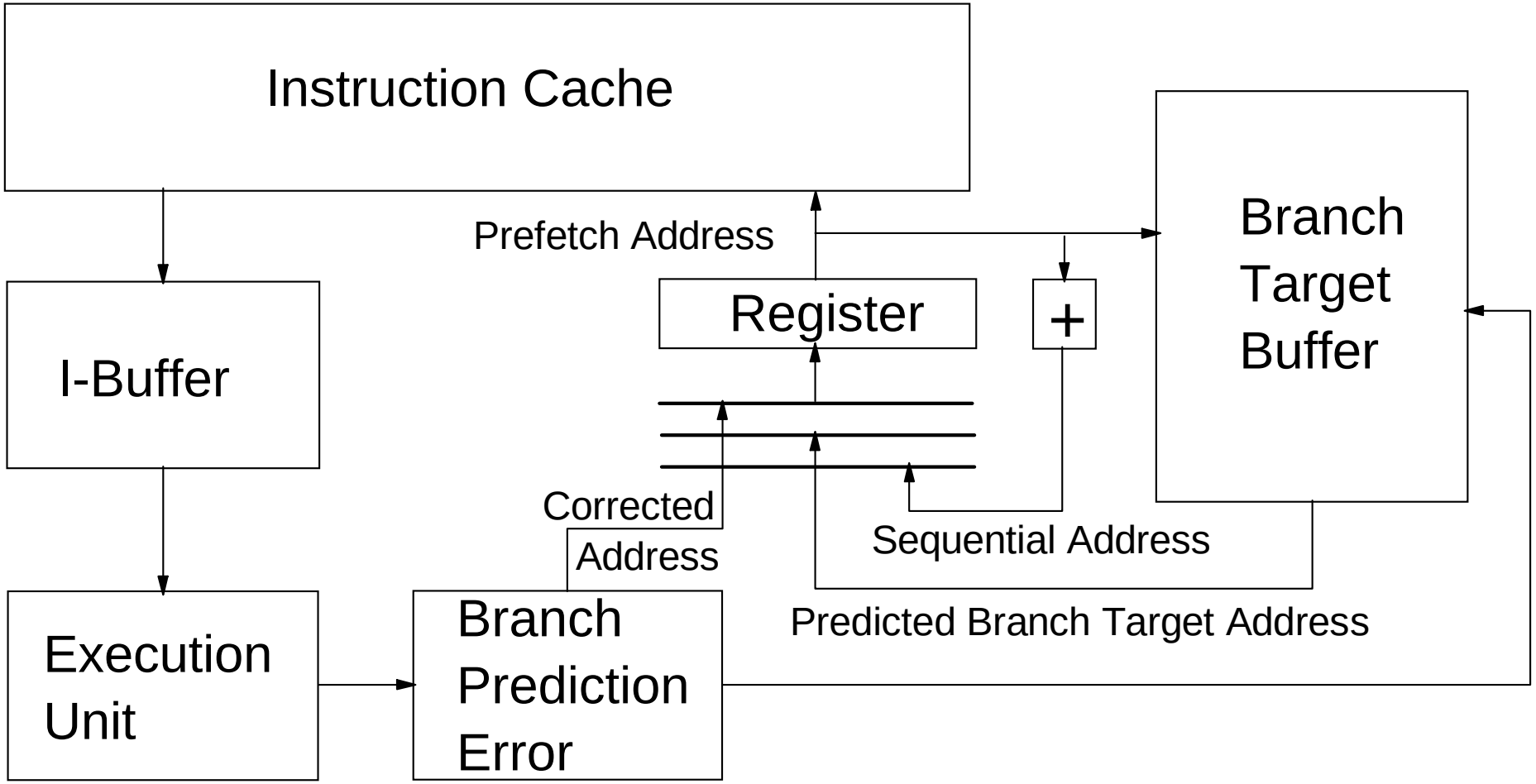
# of dynamic Executions	1	2 - 3	4 - 8	8 -15	16 - 31	32 - 63	64 -127	>12
# static Branches	3402	3663	2402	1743	1532	1323	981	44

7065 2402 6024

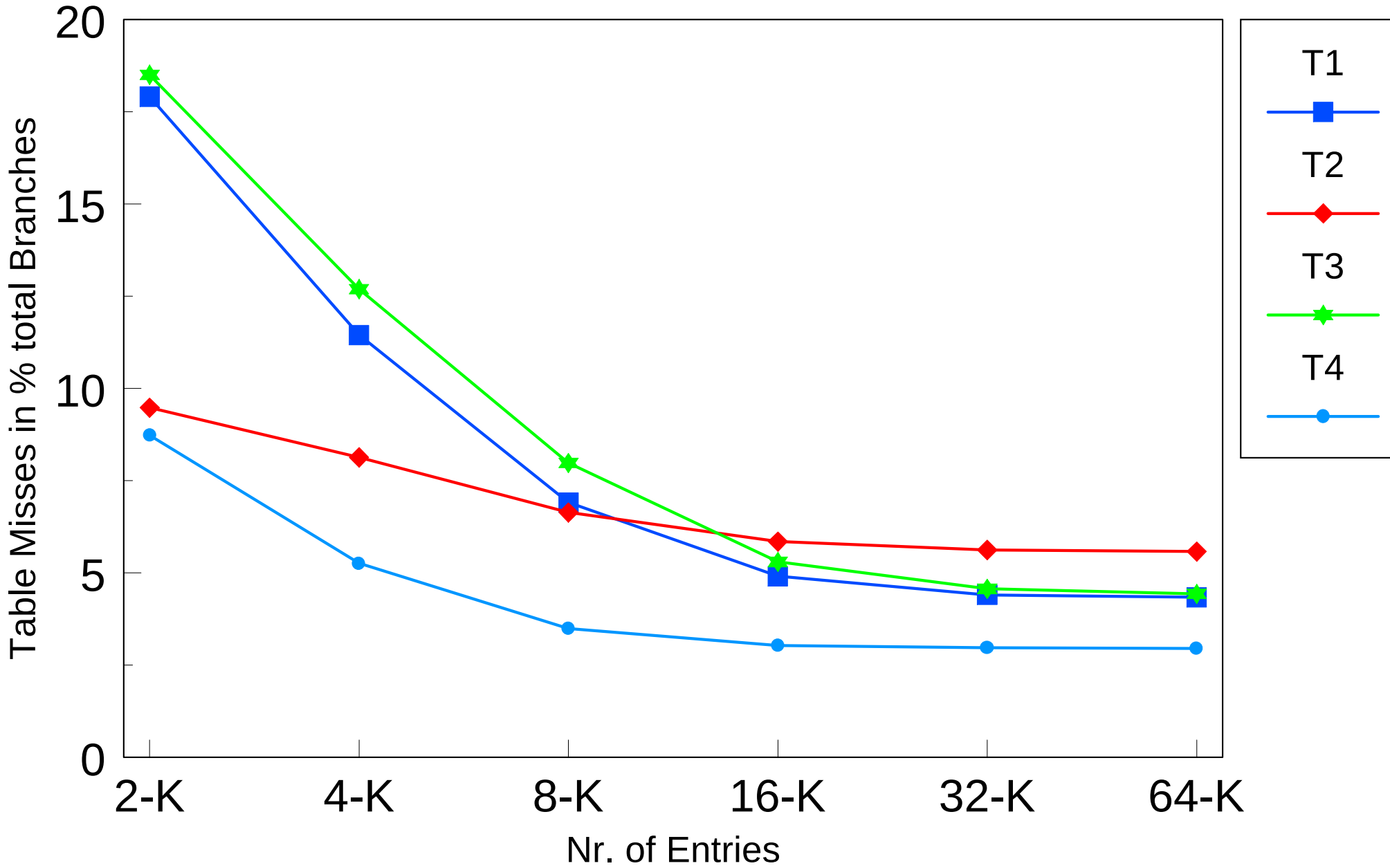
Specifics of ESA/390 ISA

- 17 different Branch Instructions
  - Majority of the Branches in the Traces are coded as Indirect Branches
    - Target Address taken from a General Purpose Register
    - Target Address generated by Address Calculation
- Conditional Branch may be coded as to Branch-Always
- Unconditional Branch may be coded as Not-To-Branch
- No explicit Call- and/or Return- Instructions

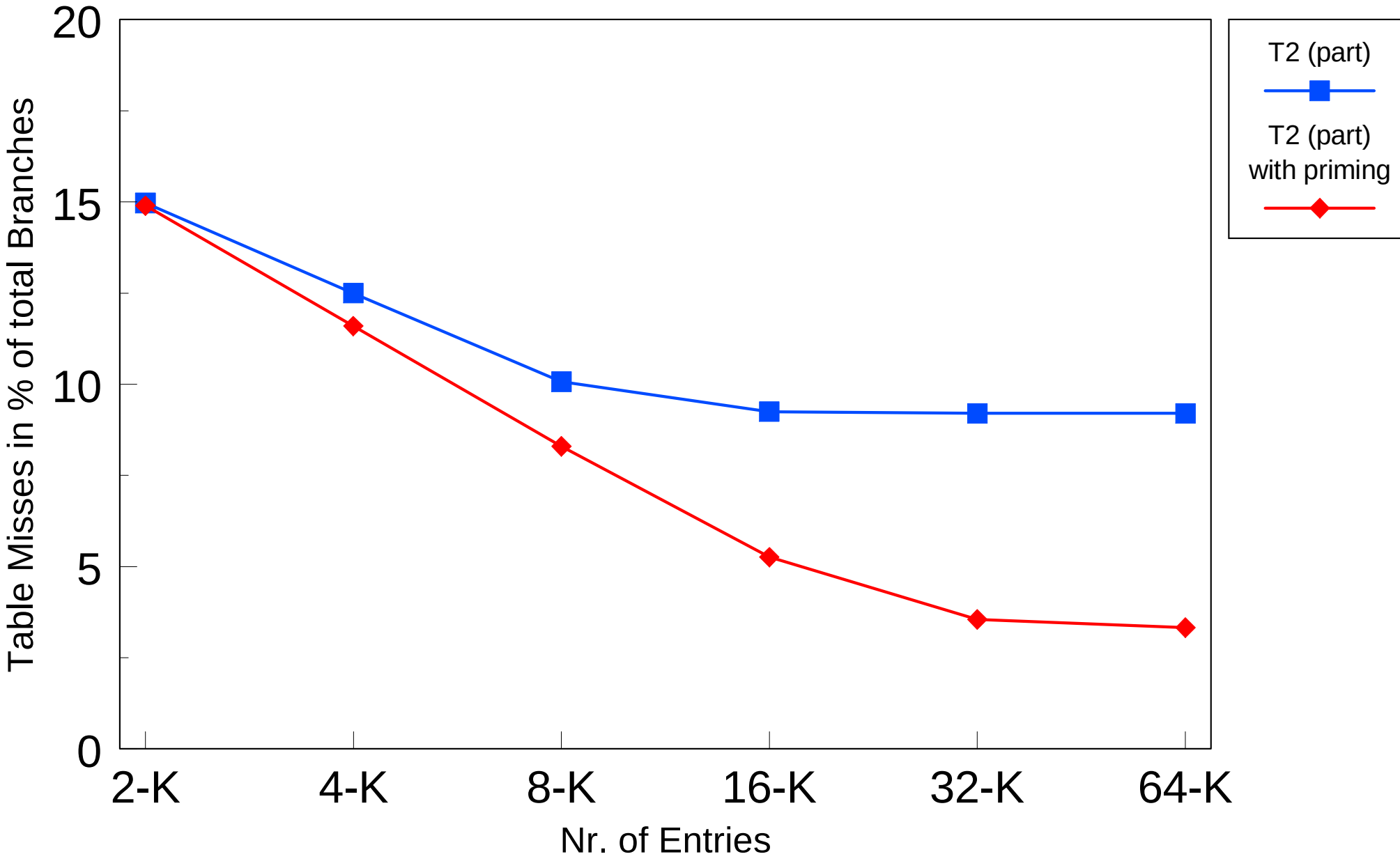
Prefetch Prediction (Principle)



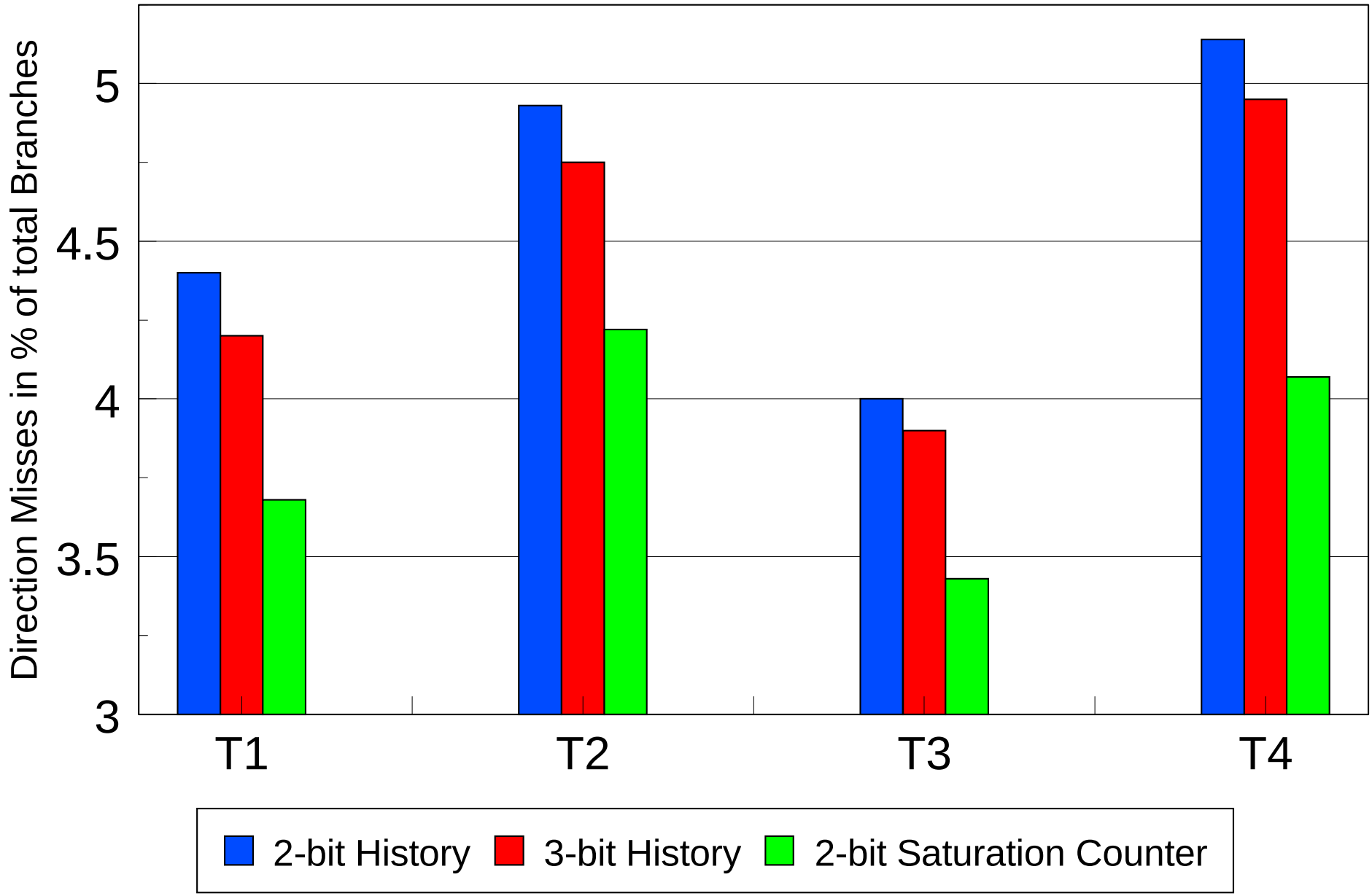
Influence of BTB Size



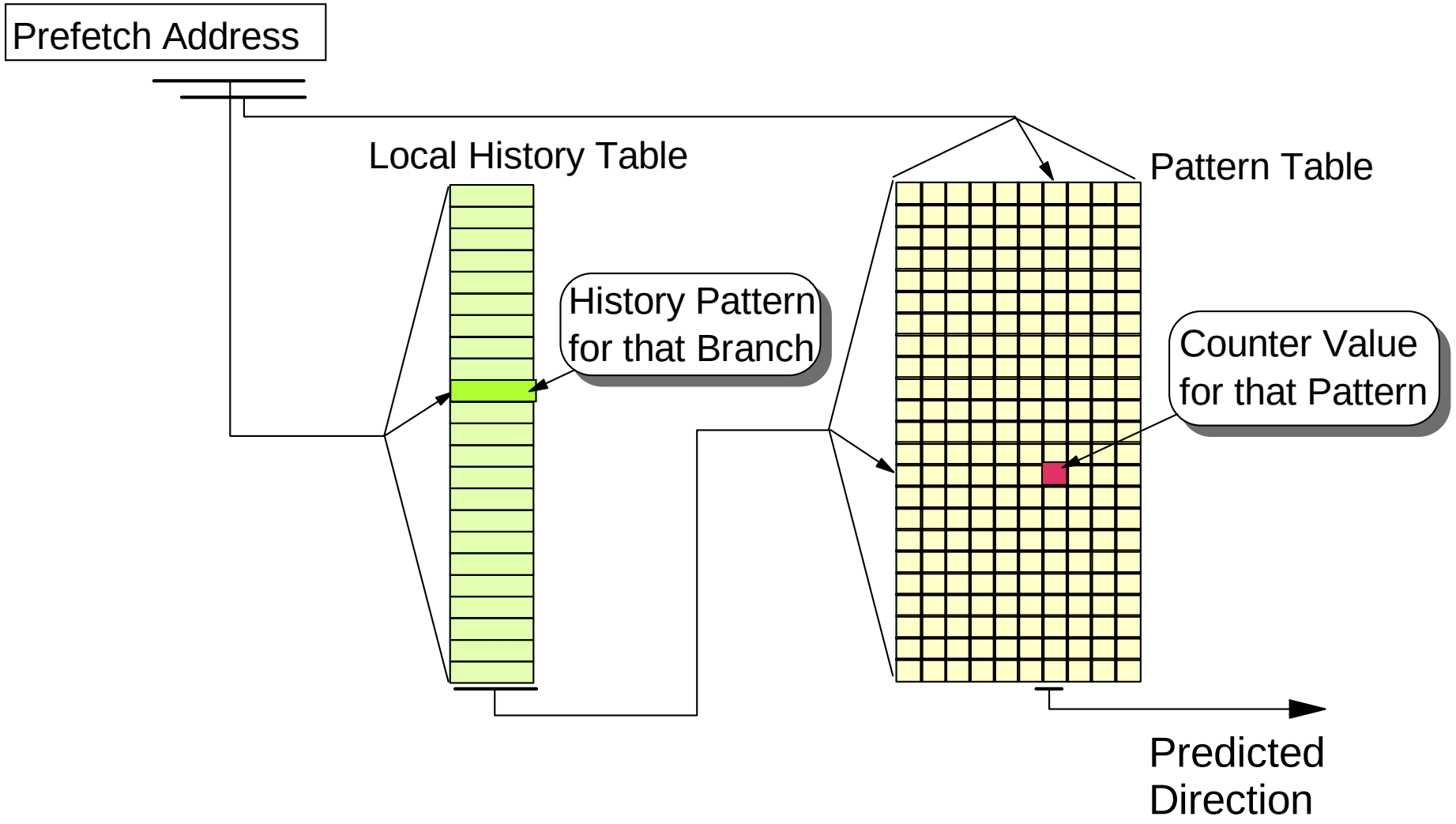
Influence of Priming on Table Misses



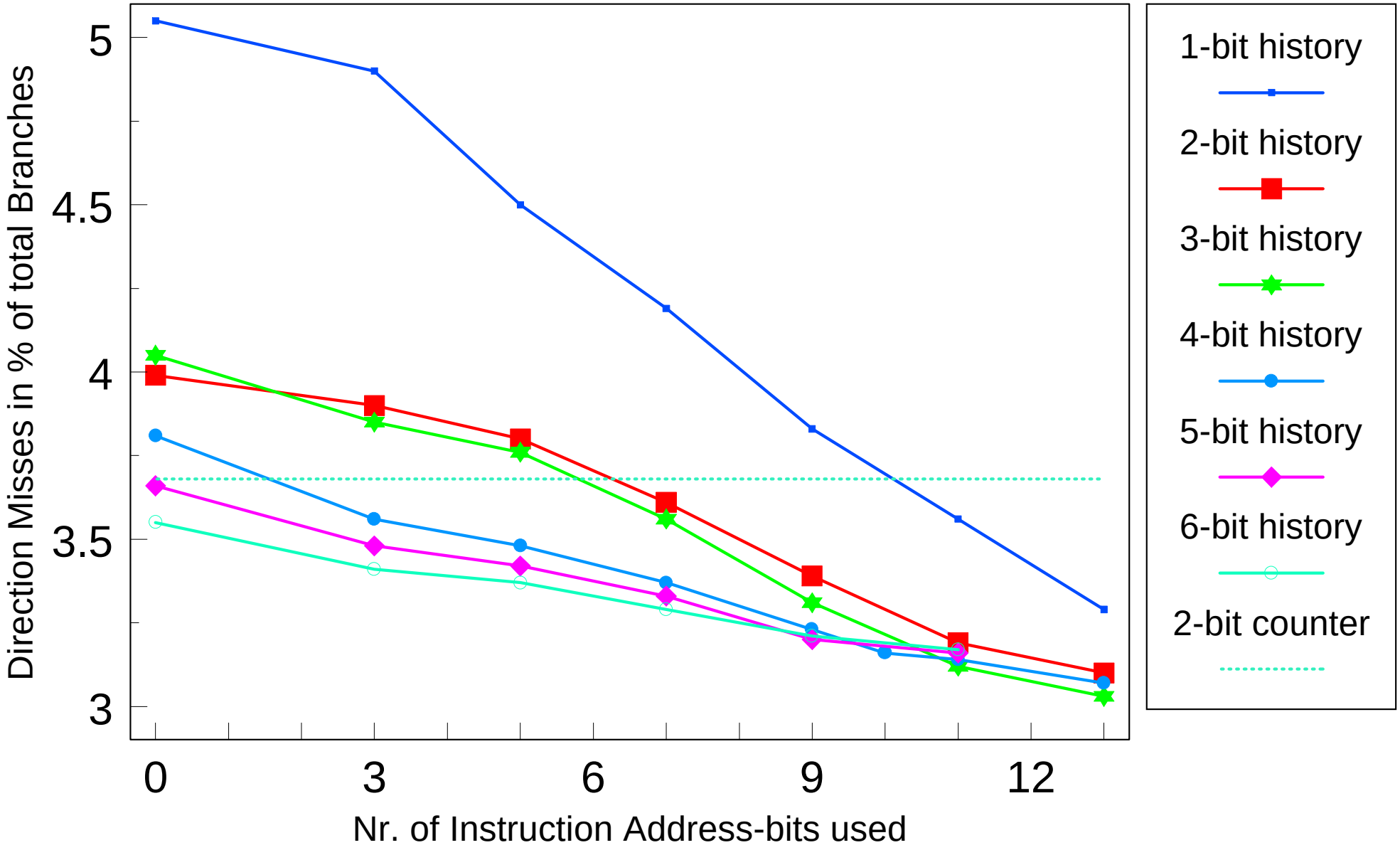
Local History: Static Predictions



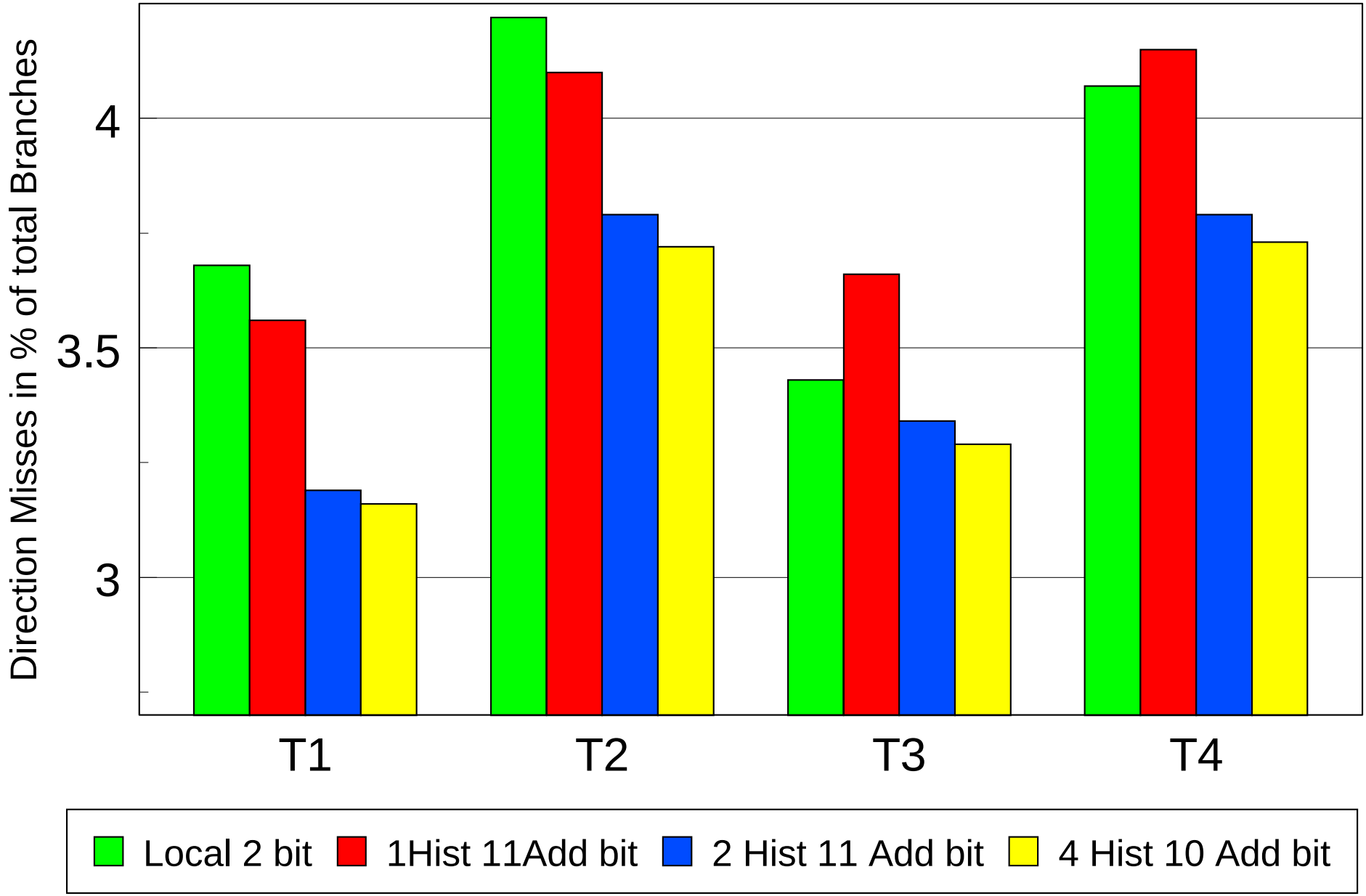
Local History with Adaptive Assignment (Principle)



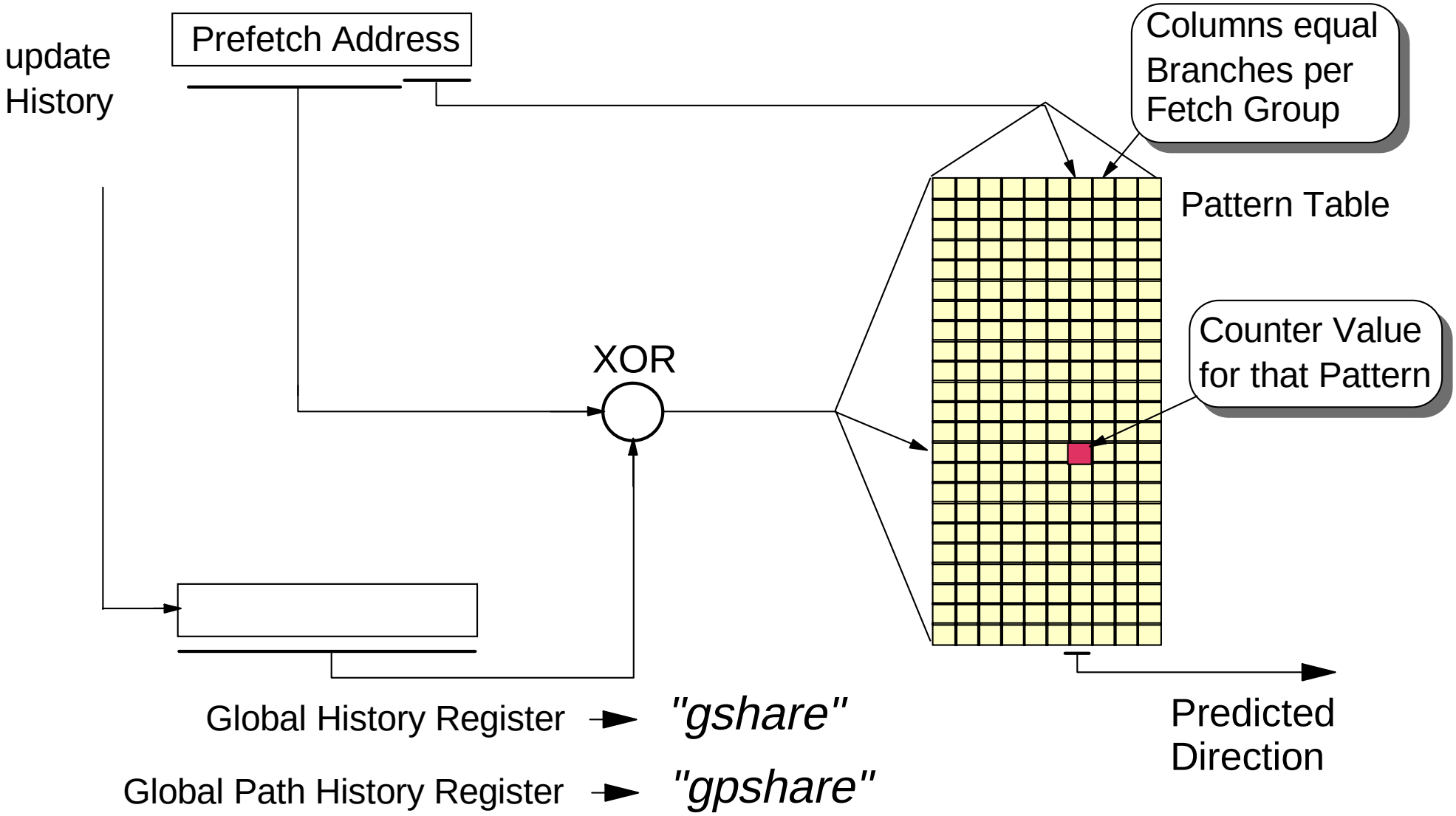
Local History with Adaptive Pattern Table



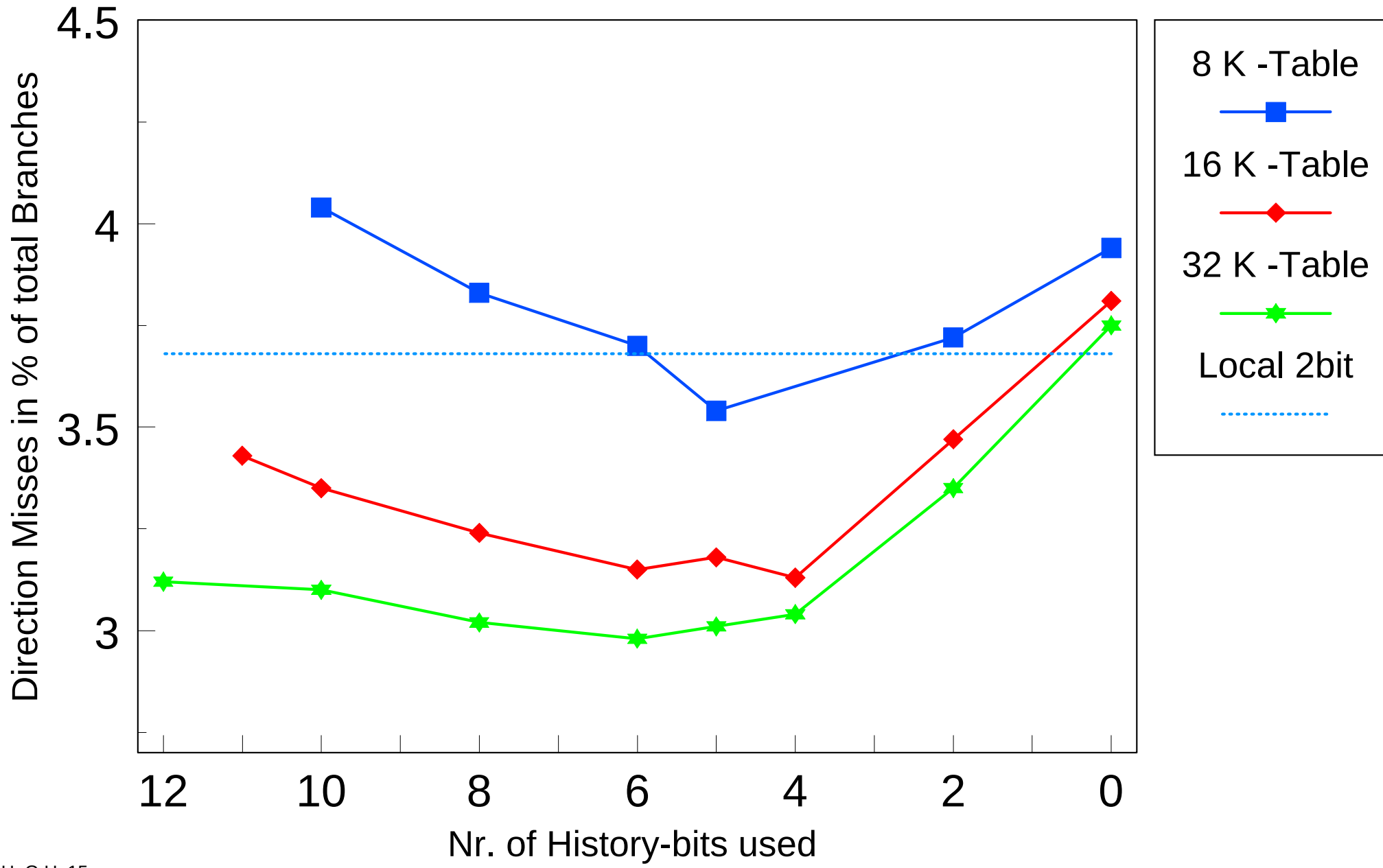
Local History with Adaptive Pattern Table



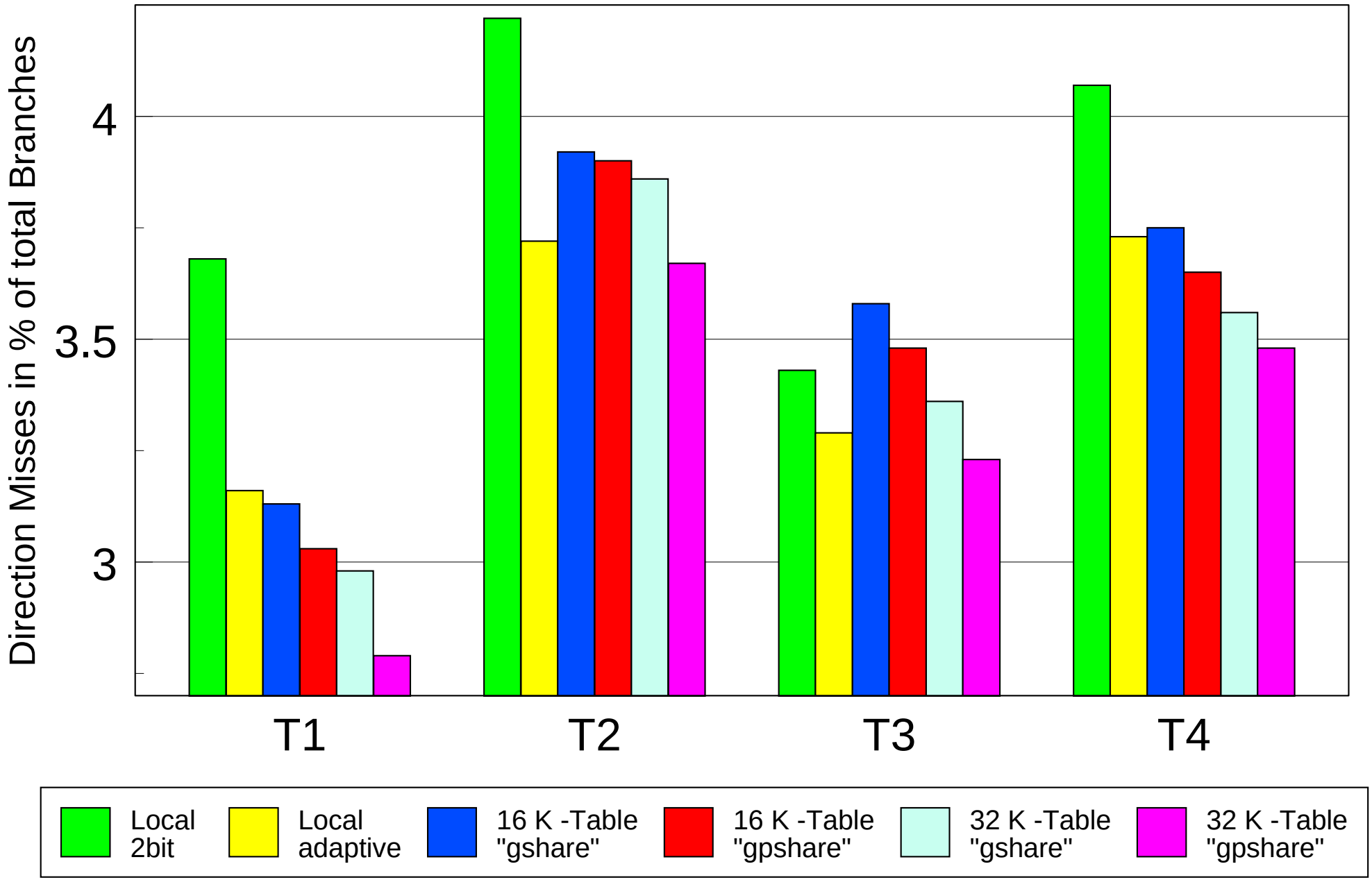
Using Global History to Access the Pattern Table (Principle)



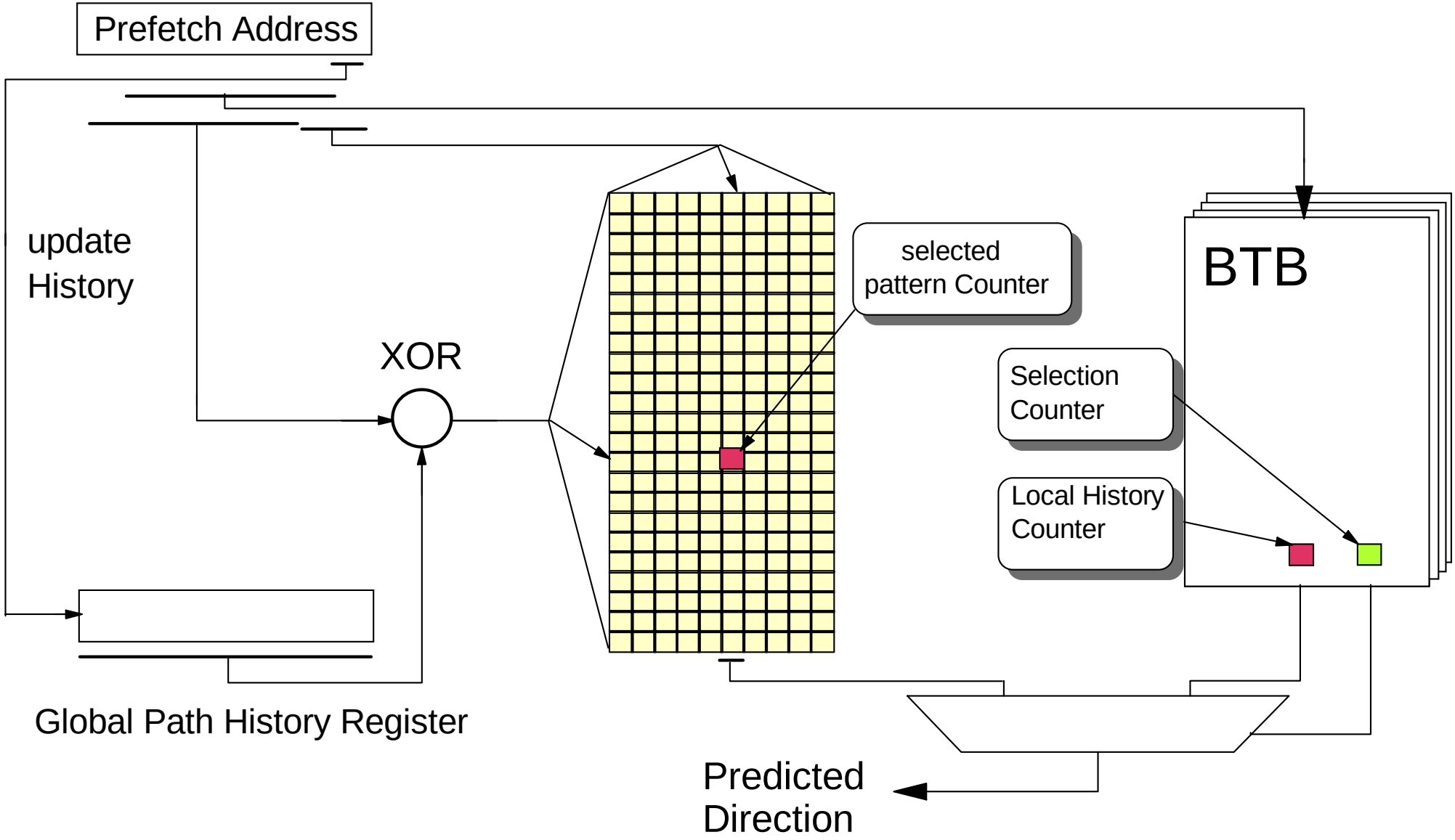
Global History Pattern "gshare"



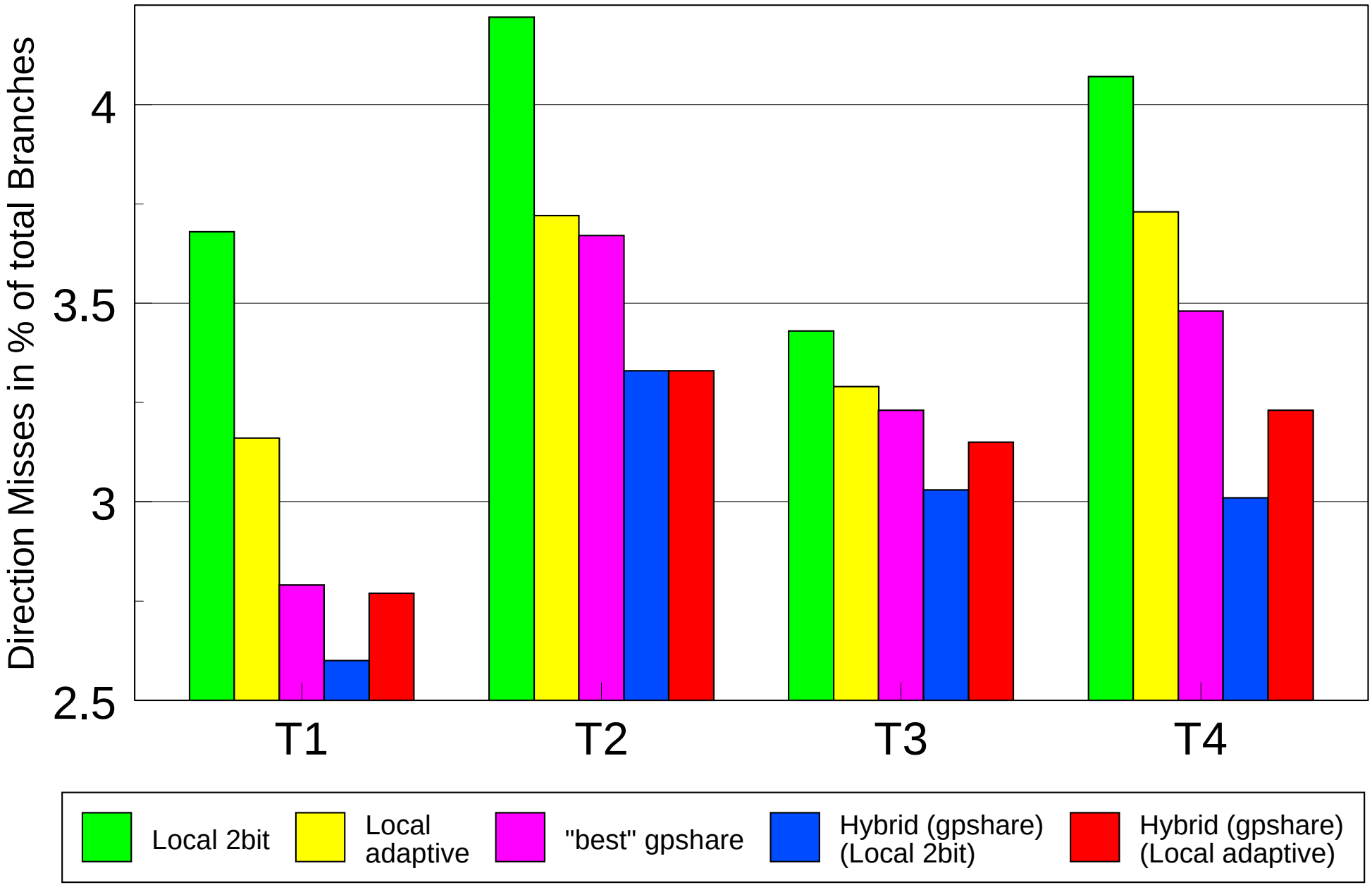
Comparing Global- and Path History



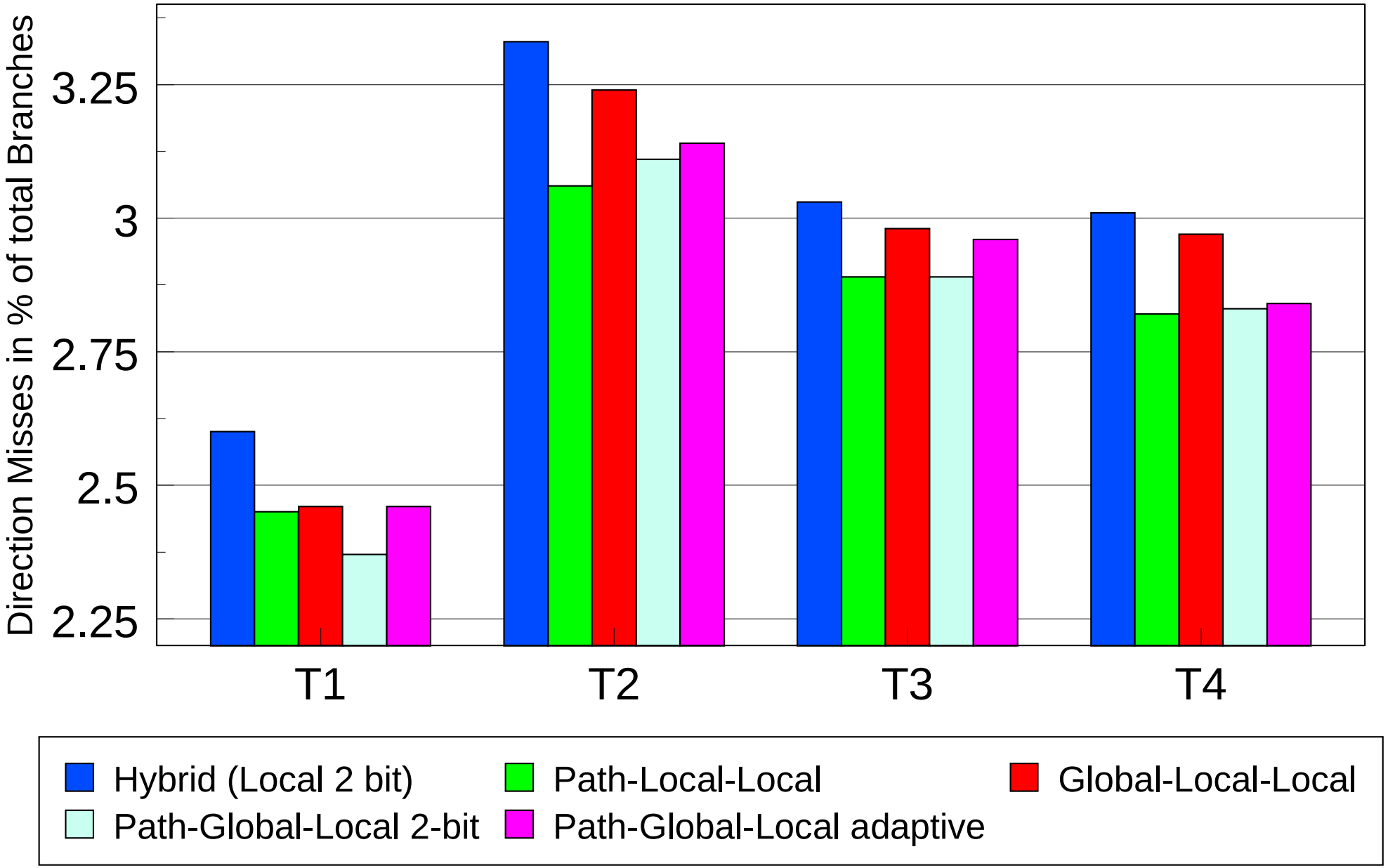
Hybrid Predictor (Principle)



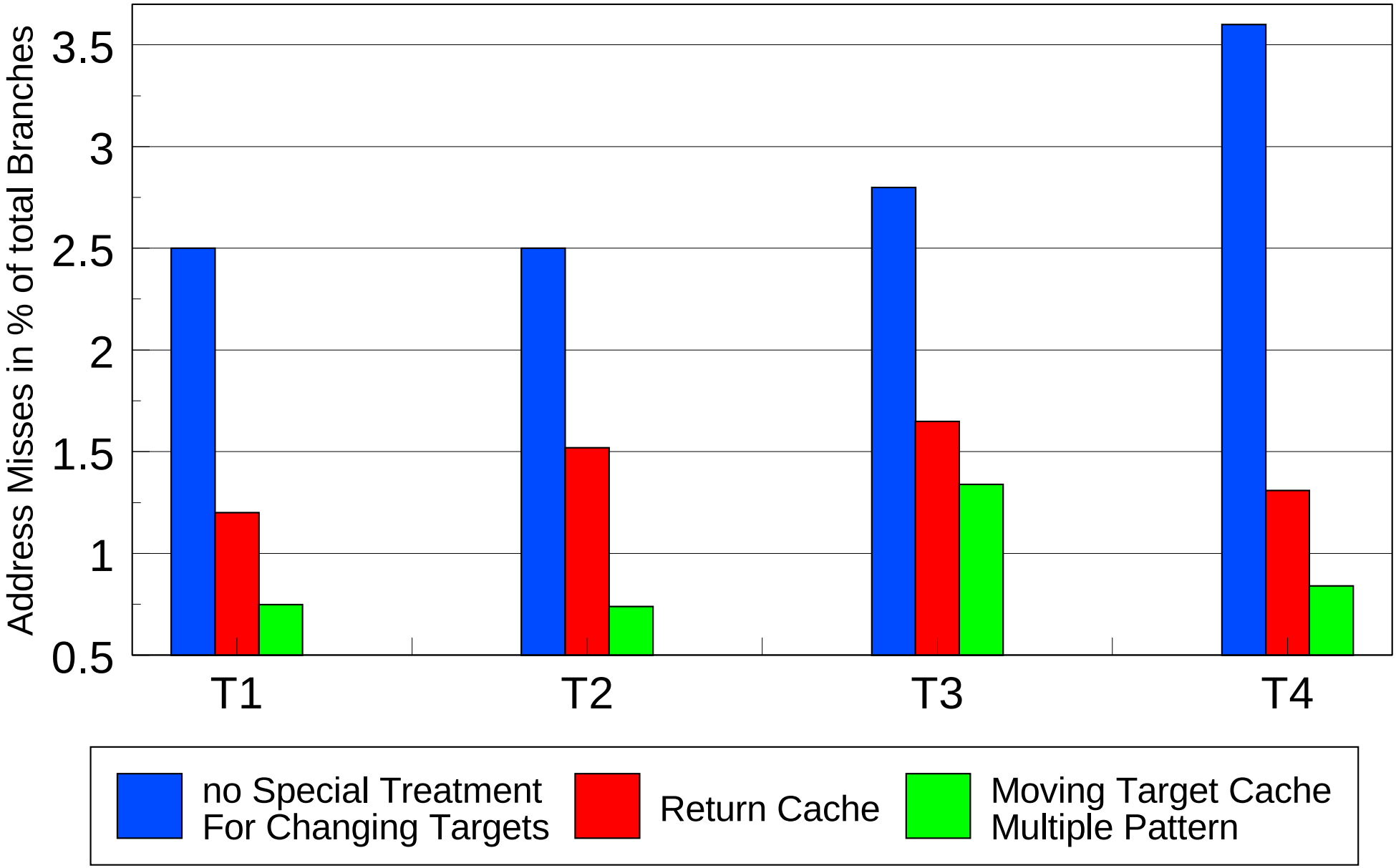
Hybrid Predictor



Hybrid Predictor using Multiple Predictors



Changing Target Treatment by Adding Cache



Conclusion

- The Branch Target Buffer should be as large as possible
- Local Adaptive History does not improve Prediction Correctness enough to pay for the Hardware added
- Modest 2 Way Hybrid Predictor seems to have the optimal Cost Performance
- To add Small Return- and Moving-Target-Caches improves the Prediction Correctness significantly