# Bit-Level Perceptron Prediction for Indirect Branches

Elba Garza
Samira Mirbagher-Ajorpaz
Tahsin Ahmad Khan
Daniel A. Jiménez

{elba, parisamir, tahsinkhan, djimenez}@tamu.edu



# Overview

- Background & Motivation
- Bit-Level Branch Prediction for Indirect Branches (BLBP)
- Testing Methodology
- Results
- Summary



# Indirect Branches

- Branch instructions with no **fixed** target address; target unknown until runtime
- Help support:
  - Virtual Function Calls
  - Jump Tables
  - Switch-Case Statements
- Behavior ranges from *monomorphic* (mainly one target address) to very *polymorphic* (*n*-many target addresses)

ISA Examples:

x86:

```
jmp %r11
br.call b1
```

RISC-V:

```
jalr rd, rs1, imm12
```

ARM A64:

```
BR Xm
BLR Xm
```

**Immediate address calculation → control flow hazards!**



# IB-Mitigation Techniques

- **Solution:** Predict the target addresses!
- Both hardware & software-supported solutions exist
- Hardware:
  - Branch Target Buffer (BTB) **[Lee '84]** – Most well-known; maps PCs to MRU target address
  - Target Cache **[Chang '97]** – Pattern & path-based address history predictor
  - Virtual Program Counter **[Kim '07]** – "Devirtualization" to sequence of conditional branches; reuses existing BP
  - ITTAGE **[Seznec '06]** – Based on TAGE predictor



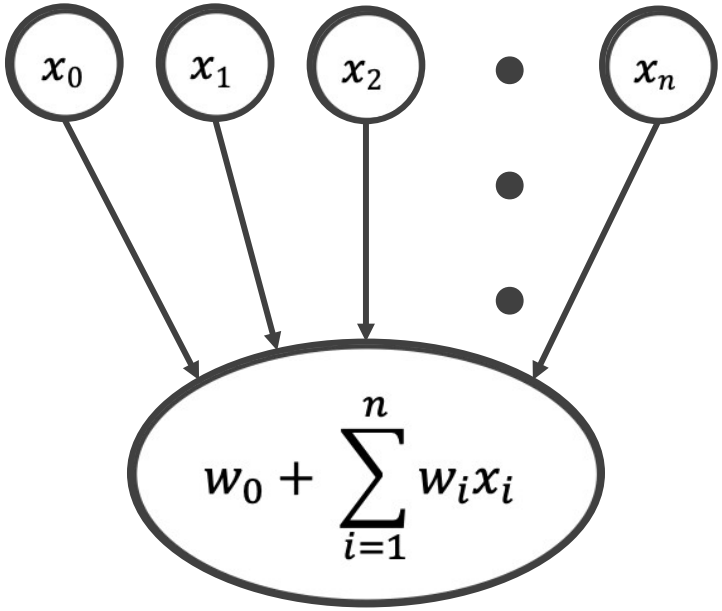
# IB-Mitigation Techniques

- Software:
  - Compiler-driven & profile-driven IB/virtual function call minimizations **[Hölzle '94]**
  - Static address prediction; profiling-driven **[Wall '91]**
  - Just-in-time IB address precomputation **[Roth '99]**
  - Dynamic predication prediction **[Joao '08]**
  - Value-Based BTB Indexing; profiling-driven **[Farooq '10]**



# Perceptron Prediction

- Based on perceptron **[Block '62]**, most basic neural-learning structure
- Comprised of a single-layer vector of integer weights, $w_0 w_1 w_2 \ldots w_n$
- Output computed as dot product of weight vector with input vector, $x_1 x_2 \ldots x_n$
- Weights updated with perceptron learning after computing output

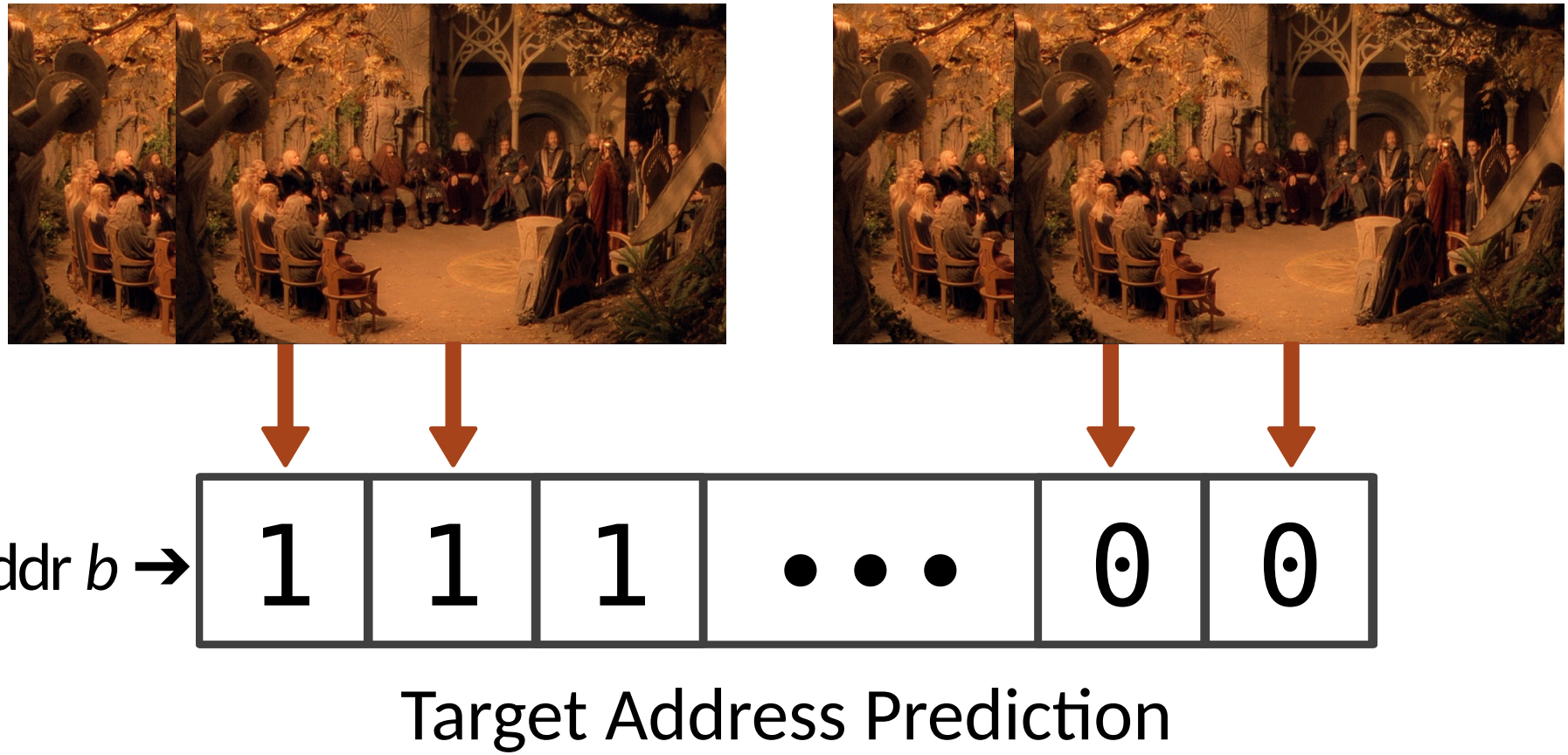


# Overview

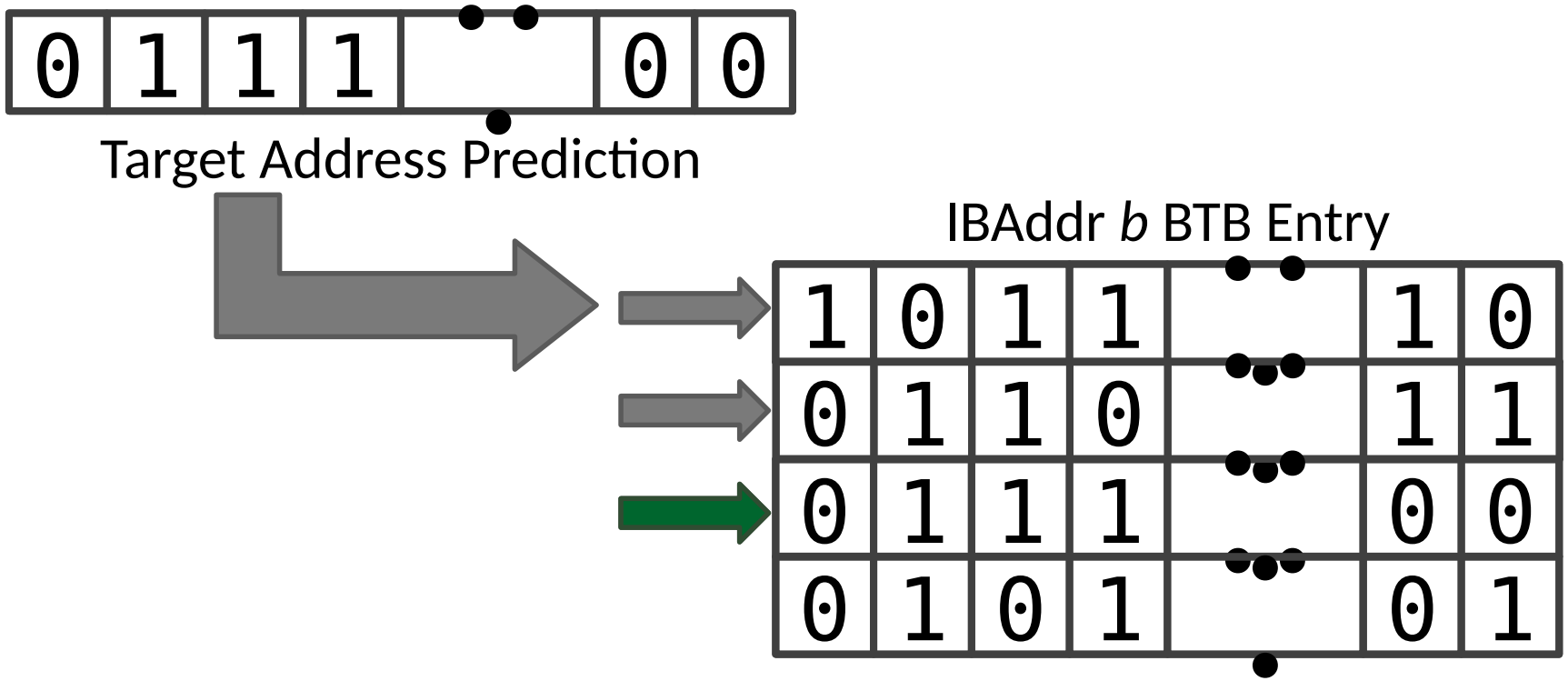
- Background & Motivation
- Bit-Level Branch Prediction for Indirect Branches (BLBP)
- Testing Methodology
- Results
- Summary



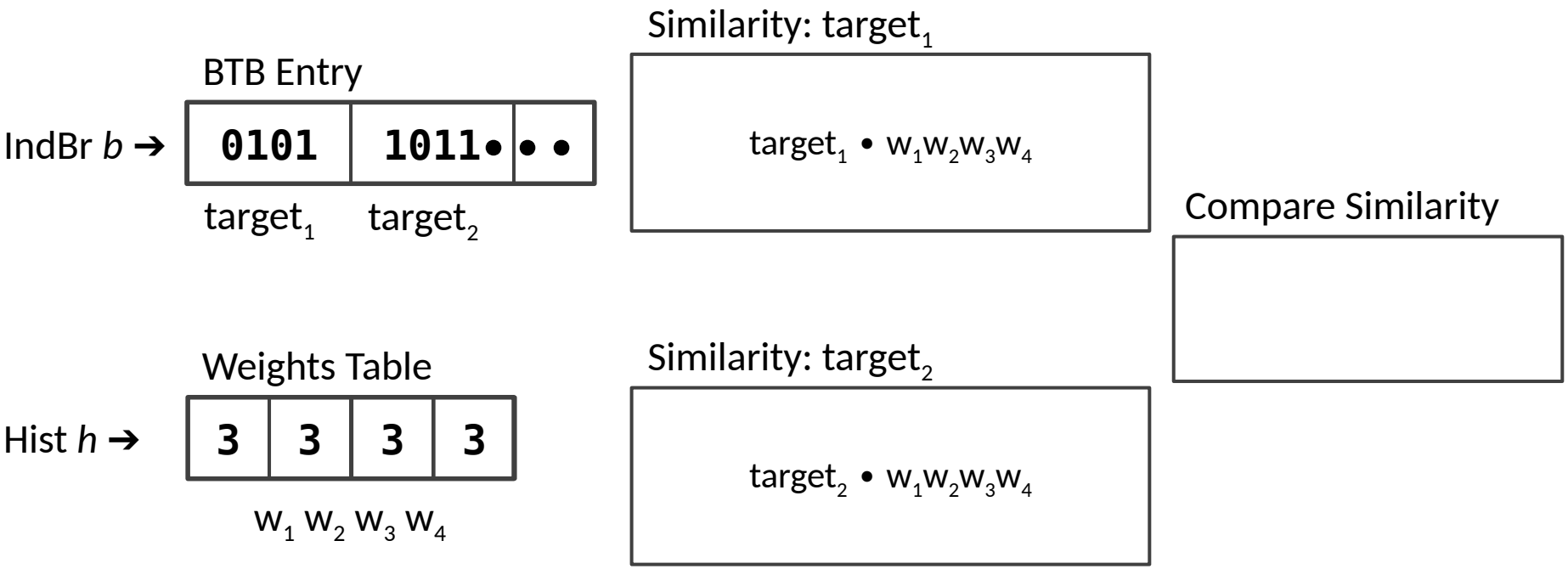
# BLBP Big Picture



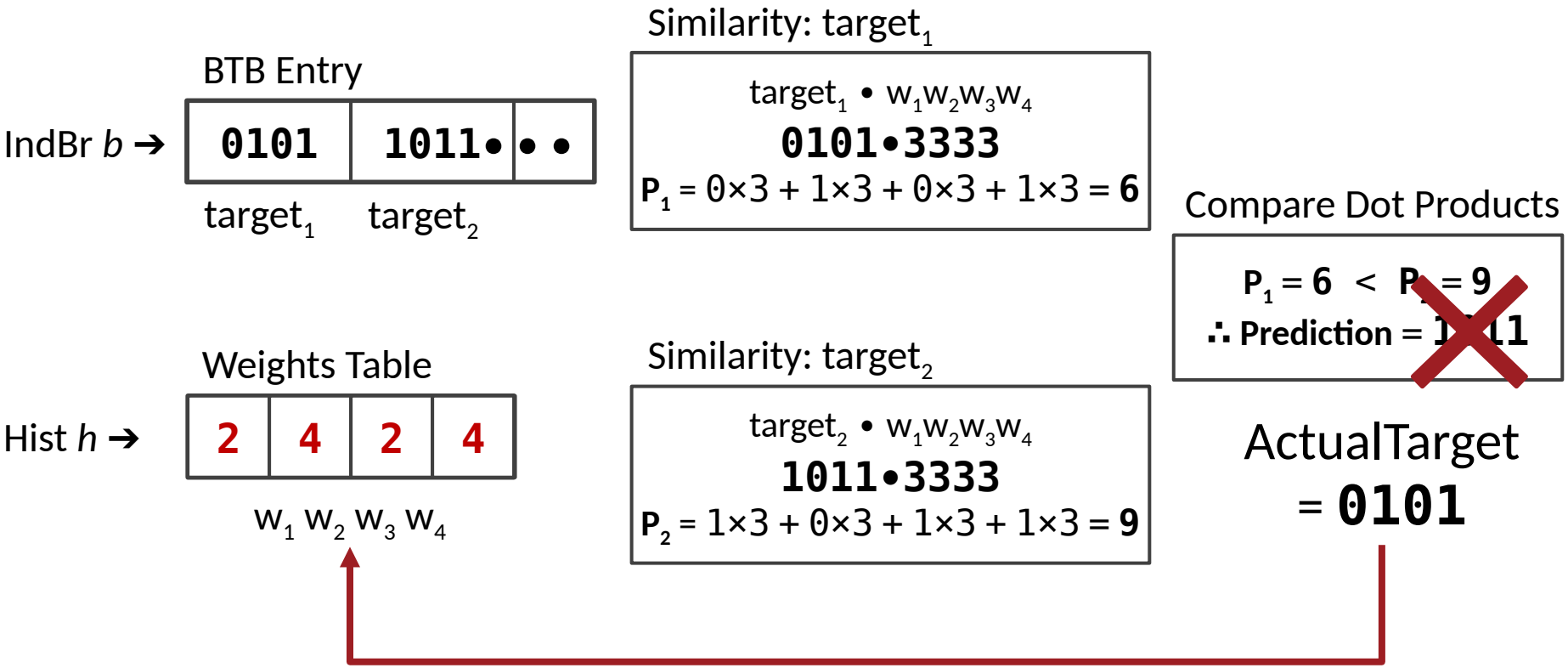
# BLBP Big Picture



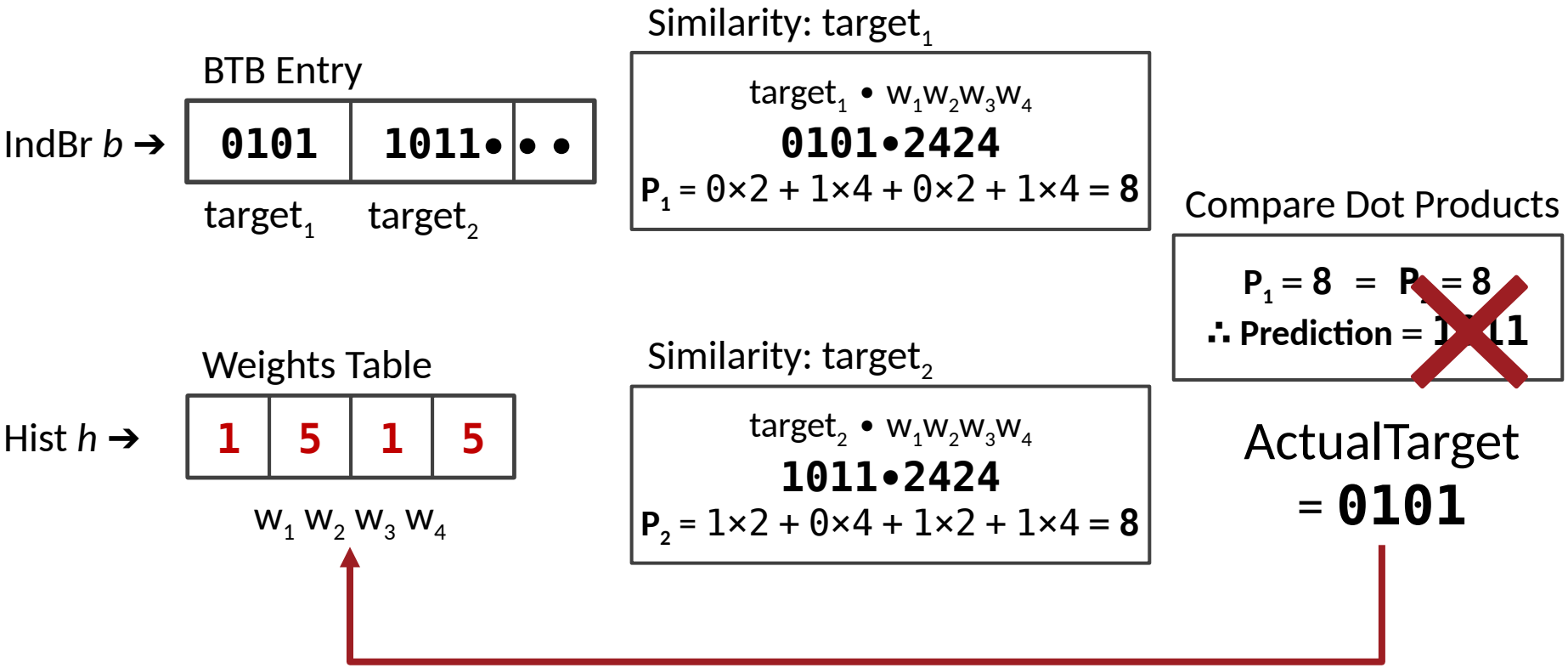
# BLBP: Example, Round 0



# BLBP: Example, Round 1



# BLBP: Example, Round 2




Training:

If bit *k* in ActualTarget = 0: $w_k = w_k - 1$

If bit *k* in ActualTarget = 1: $w_k = w_k + 1$

# BLBP: Example, Round 3

BTB Entry

IndBr *b* ➔ | **0101** | **1011•** | **• •** |

$target_1$ $target_2$

Similarity: $target_1$

$target_1 \bullet w_1w_2w_3w_4$

**0101•1515**

$P_1 = 0\times1 + 1\times5 + 0\times1 + 1\times5 = \mathbf{10}$

Compare Dot Products

$P_1 = \mathbf{10} = P_2 = \mathbf{7}$

∴ **Prediction = 0 (N)T**

Weights Table

Hist *h* ➔ | **0** | **6** | **0** | **6** |

$w_1$ $w_2$ $w_3$ $w_4$

Similarity: $target_2$

$target_2 \bullet w_1w_2w_3w_4$

**1011•1515**

$P_2 = 1\times1 + 0\times5 + 1\times1 + 1\times5 = \mathbf{7}$

ActualTarget = **0101**

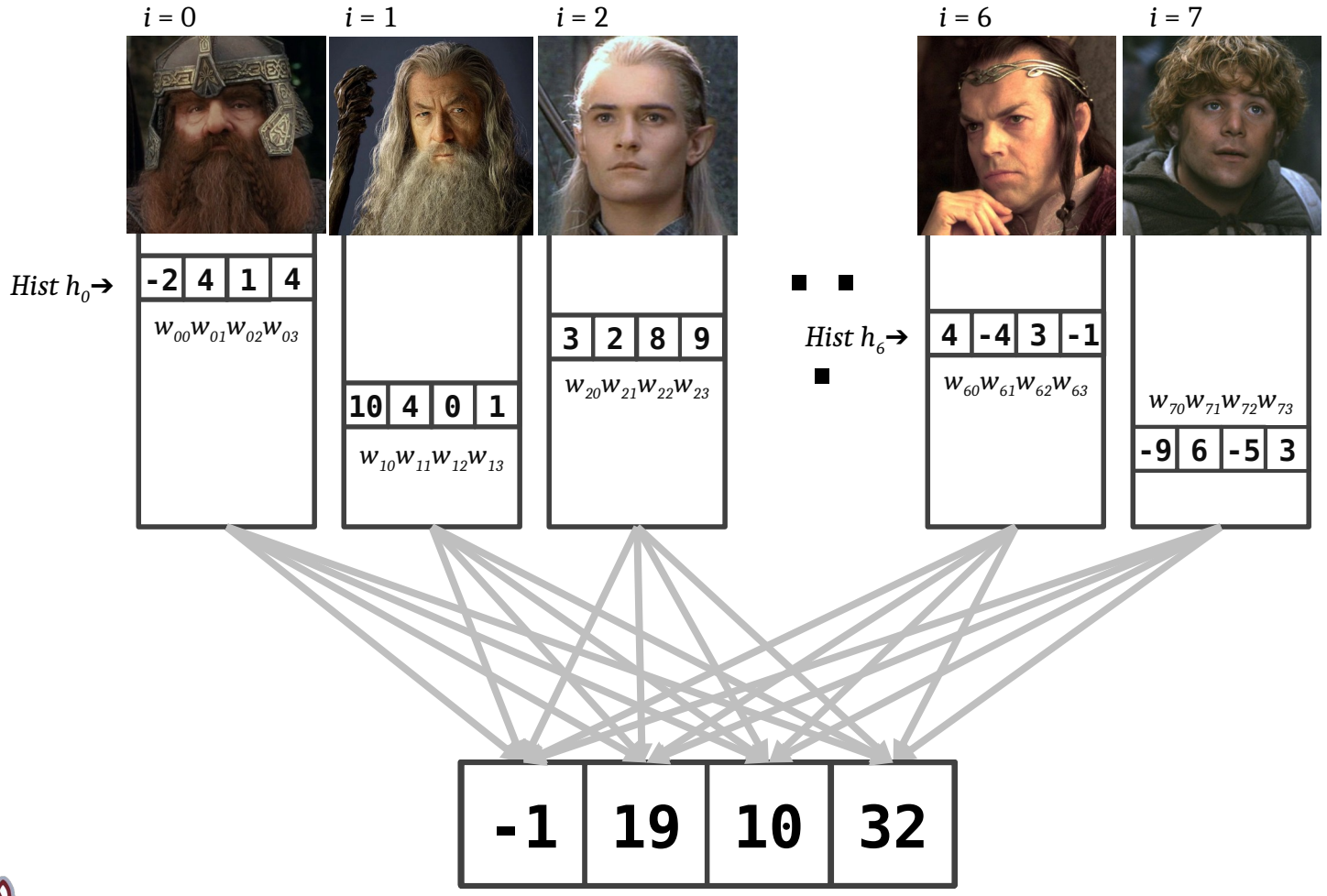


Training:

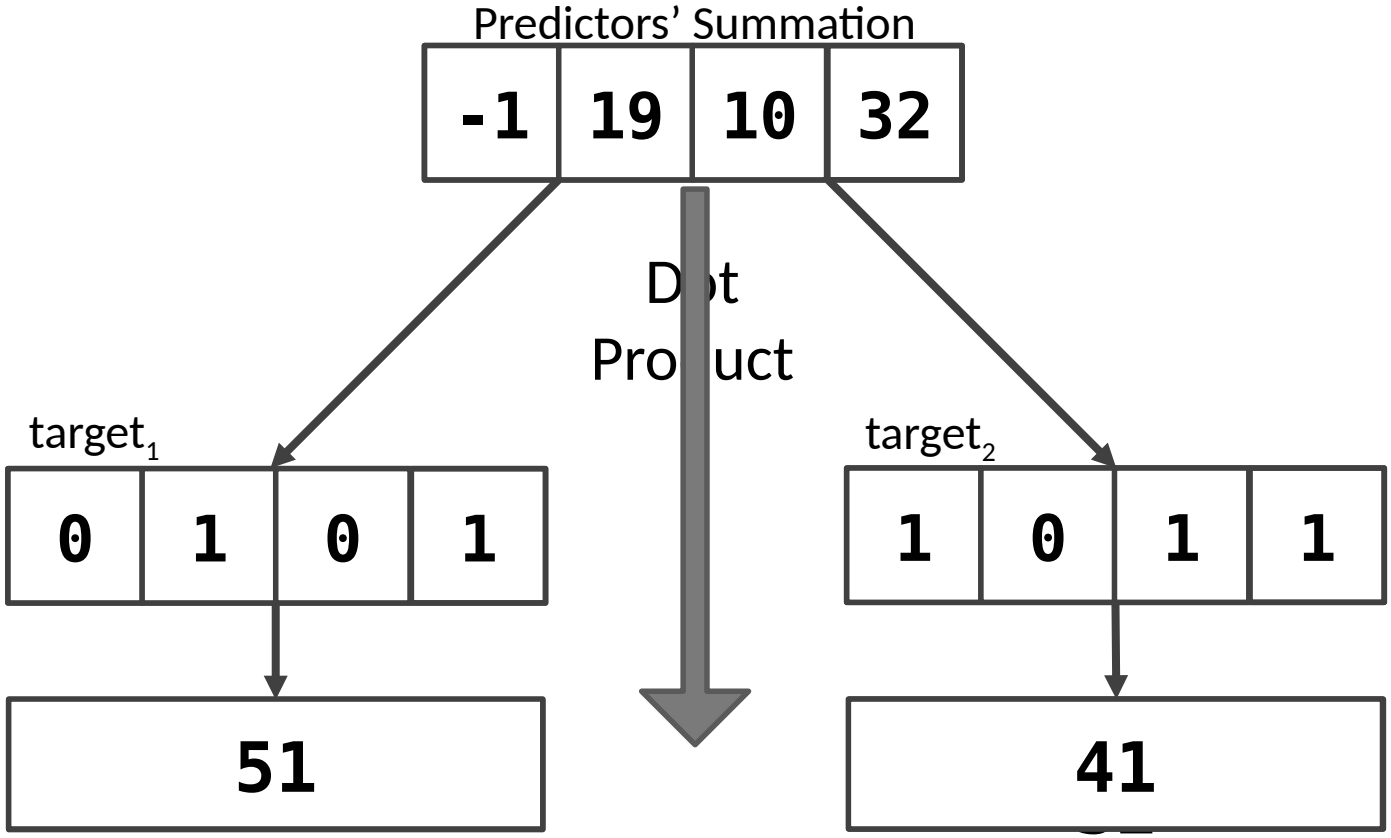
If bit *k* in ActualTarget = 0: $w_k = w_k - 1$

If bit *k* in ActualTarget = 1: $w_k = w_k + 1$

# BLBP: Multiple Predictors' Target Selection



# BLBP: Multiple Predictors' Target Selection




BLBP Target Prediction

# Overview

- Background & Motivation
- Bit-Level Branch Prediction for Indirect Branches (BLBP)
- Testing Methodology
- Results
- Summary

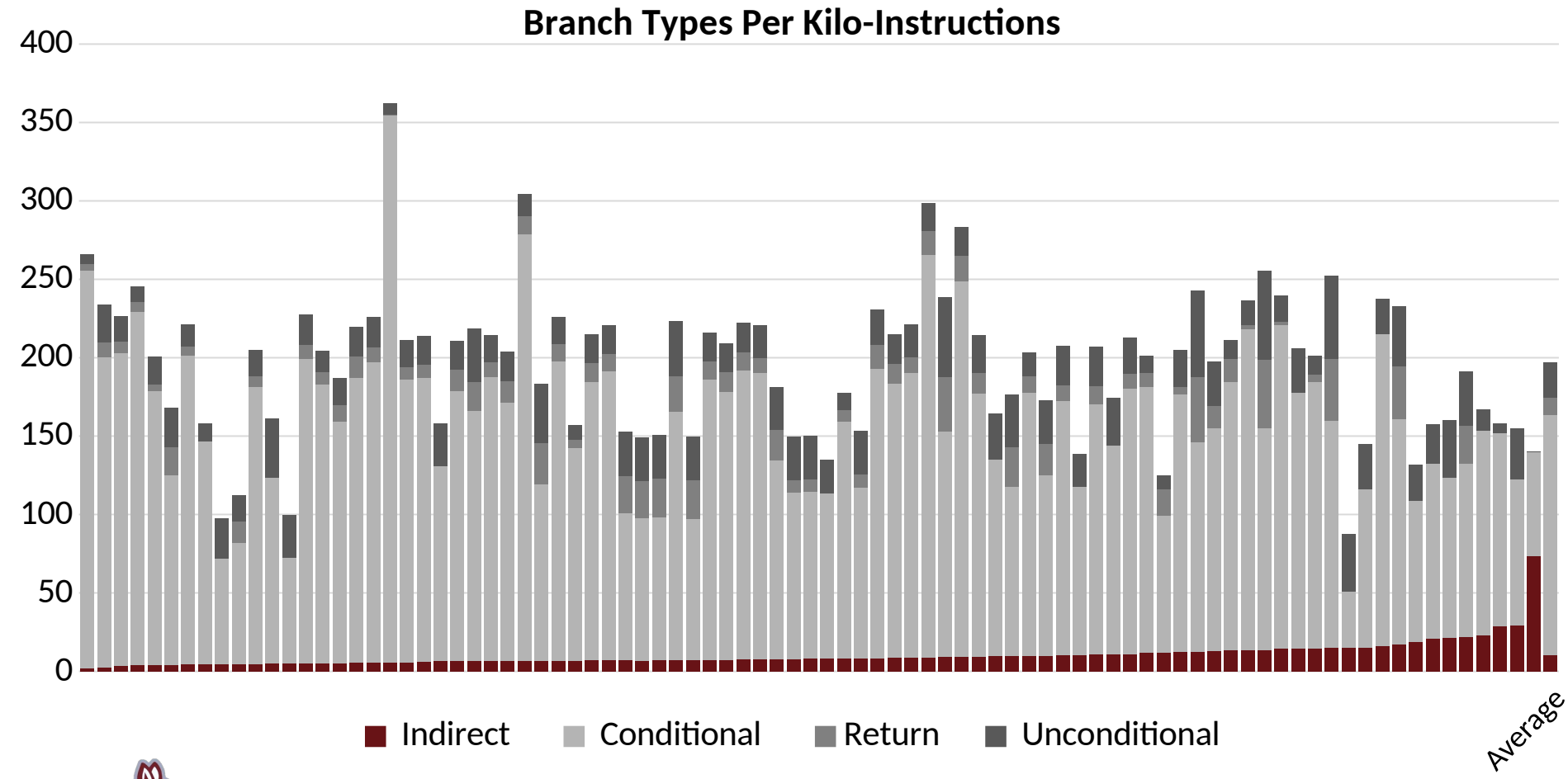


# Infrastructure & Implementation

- Augmented simulation infrastructure from CBP-5, adding BTB and indirect predictors
- Used suite of 88 workloads sourced from:
  - JILP Branch Prediction Competition **[CBP-5 '16]**
  - SPEC CPU2017 / SPEC CPU2006 / SPEC CPU2000
- Selected if indirect branches averaged >1 MPKI using infinite-sized BTB
- Comparison Indirect Predictors:
  - BTB
  - VPC
  - ITTAGE



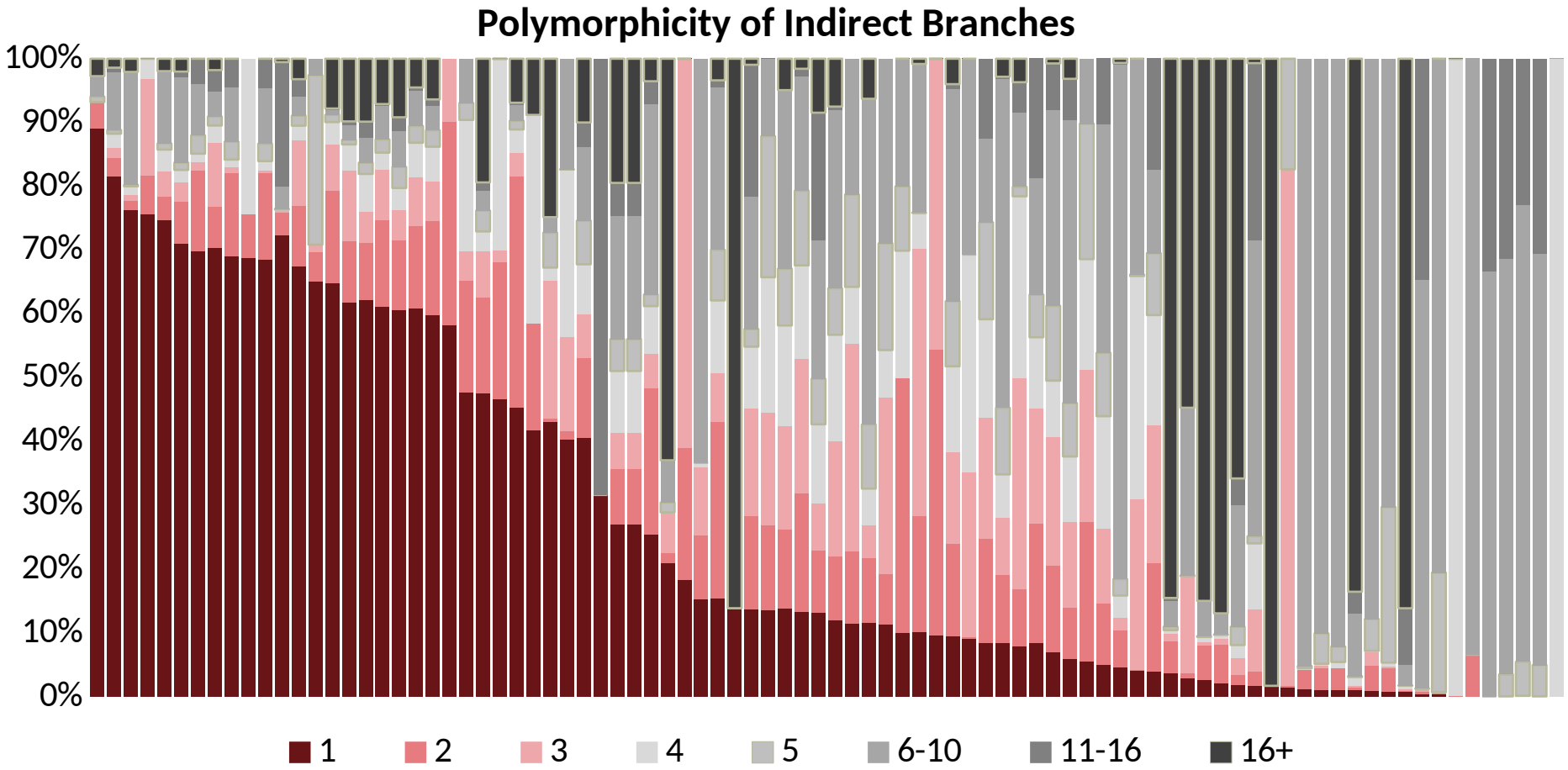
# Workload Characteristics






**Indirect branches are prevalent in tested workloads**

# Workload Characteristics




**Tested on mix of monomorphic & polymorphic workloads**

# Training/Optimization of BLBP

- Modified & tuned with the training workload selection from CBP-5
  - Mutually exclusive from 88 test workloads!
  - Various algorithm optimizations explored, implemented, and described in detail in paper.
- Even without focused tuning for workload suite, we still beat state of the art!



# Overview

- Background & Motivation
- Bit-Level Branch Prediction for Indirect Branches (BLBP)
- Testing Methodology
- Results
- Summary & Future Work



# Workload MPKI Results

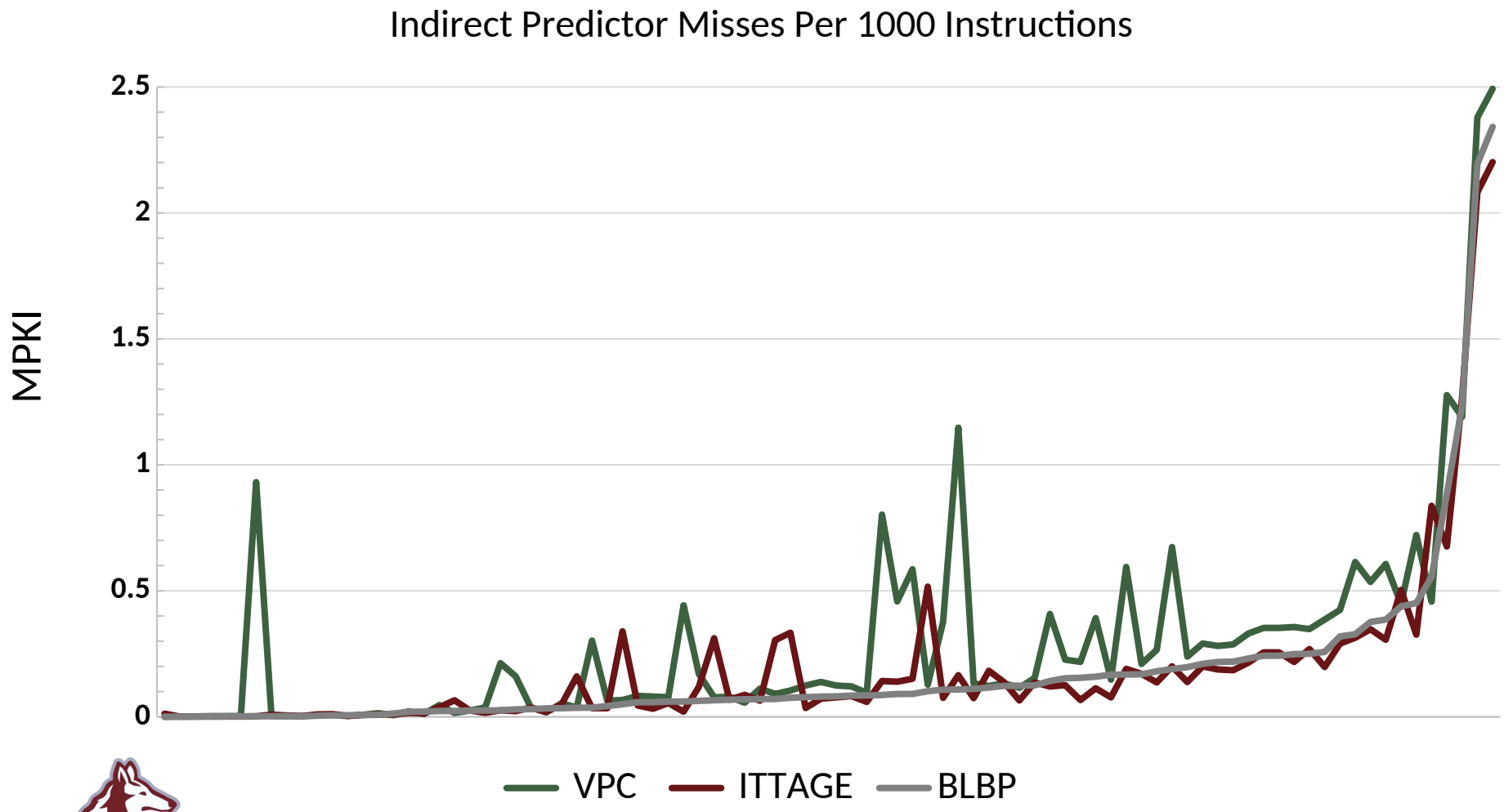
| Indirect Predictor | MPKI Arithmetic Mean |
|---|---|
| BTB | 3.40 |
| VPC | 0.29 |
| ITTAGE | 0.193 |
| BLBP | 0.183 |

**ITTAGE vs. BLBP: 0.193 → 0.183 MPKI**

**MPKI reduction of 5% over state-of-the-art!**



# Finer-Grained Results






# Overview

- Background & Motivation
- Bit-Level Branch Prediction for Indirect Branches (BLBP)
- Testing Methodology
- Results
- Summary



# Summary

- BLBP helps predict indirect branch target addresses at the **bit** level
- Set of sub-predictors works together to predict bit values, which are then compared to known target addresses
- BLBP then outputs the most similar target address to prediction



# SUPPLEMENTAL SLIDES



# BLBP: An Overview

1. Relies on set of perceptron-based subpredictors
2. Set works together to predict individual bits of indirect branch target
3. Learning relies on varying branch history subsets
4. Predicted bits are compared with previously observed targets of indirect branch
5. Outputs observed target whose bits match **most** closely with perceptron-predicted bits



# Hardware Budget

| Indirect Predictor | Total Hardware Budget | Details |
|---|---|---|
| BTB | 64 KB | 32K-entry, partially-tagged, direct-mapped |
| VPC | 128 KB | 32K-entry, partially-tagged, direct-mapped BTB; Multi-perspective Perceptron Predictor for BP |
| ITTAGE | 64 KB | as described in the original paper [25] |
| BLBP | 64.08 KB | 64-entry, 64-way set-associative, partially-tagged IBTB; 256 10-bit local histories; 630-bit global history; 8 correlating-weights tables; and 128-entry region array |



# Benchmark Breakdown

| Benchmark Sourse | Number | Details |
|---|---|---|
| SPEC CPU2000 | 1 | `252.eon` |
| SPEC CPU2006 | 12 | `400.perlbench, 403.gcc, 453.povray, 458.sjeng` |
| SPEC CPU2017 | 7 | `600.perlbench, 602.gcc, 623.xalancbmk` |
| CBP-5 Competition | 68 | Industry-sourced workloads offered by host Samsung. Benchmarks divided into two categories: `MOBILE` and `SERVER`. Trace lengths also in two categories: `SHORT` and `LONG`. |

