Analyzing Branch Mispredictions
Bernardo Toninho, Ligia Nistor and Filipe Militão

Introduction

- **Branch Prediction**: fundamental component of modern pipelined architectures, keeps pipeline full in presence of changes in control flow.
- **System Performance** is highly sensitive to predictor accuracy. **Mispredictions** require flushing the entire pipeline!

**Branch Prediction in a Nutshell**

- Filter a program's global branch behavior to exploit **branch correlations** (e.g. common history, path).
- Train a **prediction mechanism** (e.g. 2-bit counter, perceptron) with the filtered behavior stream.

**Analyzed Predictors**

- **GSHARE**: global history, combined with the PC to train a table of 2-bit counters.
- **LOCAL**: local history of a branch combined with the PC to train a table of 2-bit counters.
- **TAGE**: variable length **global history** combined with PC indexes into several **cache-like** prediction tables.

**Questions**

- How do **indexing collisions** affect predictions?
- Are there problematic **classes of branches**?
- Are the captured **correlations** useful?
- Are mispredictions **isolated** or **clustered**?

**Contributions**

In-depth analysis of "state of the practice" predictors:
- Global history, 2-bit counter (**GShare**)
- Local history, 2-bit counter (**PaC**, **Local**)
- New analysis methods to measure:
  - Correlation (decomposing into correlation sets)
  - Training stream characteristics (2-bit counter adequacy)
- New predictor to reduce clustered mispredictions

An in-depth analysis of "state of the practice" predictors:

- Global history, 2-bit counter (**GShare**)
- Local history, 2-bit counter (**PaC**, **Local**)
- New analysis methods to measure:
  - Correlation (decomposing into correlation sets)
  - Training stream characteristics (2-bit counter adequacy)
- New predictor to reduce clustered mispredictions

Analysis and Results

- **Clustered Mispredictions**
  - Definition: when a predictor is wrong in consecutive guesses.
  - How large and frequent are misprediction clusters?
- **How can we use this to improve prediction rates?**
  - Keep track of past consecutive mispredictions
  - "Flip" prediction when a cluster is found
  - Problem: when to stop flipping?
    - **ideal/optimal**: without making errors
    - **worst case**: after causing a misprediction
  - Implementations (invert after 2 wrongs):
    - gshare++: stop when first error is made
    - gshare+: track cluster size average and stop after sum of average and std. deviation or after mistake

**Correlation Analysis**

- Determine the sets of correlated predictions
- Correlations are linked to collisions in table index.
- We decomposed **GShare** into:
  - Base: BHR = PC % TABLE_SIZE
  - Unlimited: BHR = PC
  - No collisions: (BHR, PC)
  - Path history: (path, BHR, PC)
- We decomposed **Local** into:
  - Limited: (BHR[PC % TABLE_SIZE] = PC) % TABLE_SIZE
  - Unlimited: (BHR[PC] = PC)
  - No collisions: (BHR[PC], PC)

**Training Stream Analysis**

- 1-bit counter is a **last value** predictor
- N-bit counter is a **last value** predictor with a slack of \(N\)
- Tolerates alternating sequences
- Exploits repeating values
- Analyzed stream characteristics:
  - Length of consecutive repeating values (bursts)
  - Transition (noise) length between bursts
  - Burst value switches
  - Warm-up value sensitivity (how long until initial value is irrelevant?)

**Conclusions**

- Simple methods to exploit misprediction clusters yield small gains. Cluster distribution causes high inversion penalty.
- Performance gains due to decreasing indexing collisions are not significant overall
- Irregular/random patterns are generally the main cause of mispredictions (despite being a small class of branches)
- Training Stream/Correlation Analysis:
  - We observed some instances of **destructive interference**
  - Speculative predictions due to deep pipelines and superscalar execution cause errors that could have been easily predicted if the updates were committed sooner
  - Initial values affect predictions for a short time
  - 2-bit counter adequately models most patterns

**Misprediction Rates**

**Branch Classification**

- Classify branches according to their behavior (static, single execution, alternating, loops, block patterns, complex patterns, others):

**Clustered Mispredictions**

- Definition: when a predictor is wrong in consecutive guesses.
- How large and frequent are misprediction clusters?

**Correlation Analysis**

- Determine the sets of correlated predictions
- Correlations are linked to collisions in table index.
- We decomposed **GShare** into:
  - Base: BHR = PC % TABLE_SIZE
  - Unlimited: BHR = PC
  - No collisions: (BHR, PC)
  - Path history: (path, BHR, PC)
- We decomposed **Local** into:
  - Limited: (BHR[PC % TABLE_SIZE] = PC) % TABLE_SIZE
  - Unlimited: (BHR[PC] = PC)
  - No collisions: (BHR[PC], PC)

**Training Stream Analysis**

- 1-bit counter is a **last value** predictor
- N-bit counter is a **last value** predictor with a slack of \(N\)
- Tolerates alternating sequences
- Exploits repeating values
- Analyzed stream characteristics:
  - Length of consecutive repeating values (bursts)
  - Transition (noise) length between bursts
  - Burst value switches
  - Warm-up value sensitivity (how long until initial value is irrelevant?)

**Conclusions**

- Simple methods to exploit misprediction clusters yield small gains. Cluster distribution causes high inversion penalty.
- Performance gains due to decreasing indexing collisions are not significant overall
- Irregular/random patterns are generally the main cause of mispredictions (despite being a small class of branches)
- Training Stream/Correlation Analysis:
  - We observed some instances of **destructive interference**
  - Speculative predictions due to deep pipelines and superscalar execution cause errors that could have been easily predicted if the updates were committed sooner
  - Initial values affect predictions for a short time
  - 2-bit counter adequately models most patterns

**Misprediction Rates**

**Branch Classification**

- Classify branches according to their behavior (static, single execution, alternating, loops, block patterns, complex patterns, others):