Assigning Confidence to Conditional Branch Predictions

Erik Jacobsen, Eric Rotenberg**, Jim Smith

Departments of Electrical and Computer Engineering and
**Computer Science
University of Wisconsin — Madison
Introduction

• Branch prediction and speculative execution is becoming common in high performance processors
• Typically, all branch predictions are acted upon
• May not want to speculate if the likelihood of a branch misprediction is high
• More generally:
  Vary how we act upon a branch prediction depending on the likelihood of a misprediction.
• Goal: develop hardware for assessing likelihood that a conditional branch prediction is correct
  - branch prediction confidence mechanisms
Applications

• Selective Dual Path Execution (Tim Heil)
• Guide instruction fetching in simultaneous multithreading
• Dynamic selector for N-way hybrid branch predictor (Jong-Hoon Shin)
• Branch prediction reverser (Karen Wells)

This talk: confidence mechanisms underlying these applications.
• Divide branch predictions into high and low confidence sets.
• Concentrate as many of the mispredicted branches as possible into the low confidence set, while keeping the low confidence set relatively small.
Simulation Methodology

- Trace-driven simulation using IBS benchmarks
  - both user and kernel code
  - benchmarks use X libraries
- Results are shown in aggregate
- Underlying predictor: gshare, 64K entries
  - ~4% aggregate mispredict rate
- Notes:
  - all results are ideal because they apply to the given input datasets only
  - choice of underlying predictor affects results
  - did not exhaust design space (must qualify “best”)
Static Confidence Experiment

1. Profile every static branch:
   - count the number of dynamic occurrences
   - count the number of mispredicts

2. Sort the static branches from worst to best mispredict rate

3. Going down the list of static branches, construct a graph:
   - Y-axis: cumulative % of total mispredicts
   - X-axis: cumulative % of total dynamic branches
**Static Confidence (Example)**

<table>
<thead>
<tr>
<th>Address</th>
<th>occurrences: # / cum. %</th>
<th>mispredicts: # / cum. %</th>
<th>mispredict rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2020</td>
<td>100 / 27%</td>
<td>98 / 69%</td>
<td>0.98</td>
</tr>
<tr>
<td>0xAF04</td>
<td>20 / 32%</td>
<td>18 / 81%</td>
<td>0.90</td>
</tr>
<tr>
<td>0x1234</td>
<td>50 / 46%</td>
<td>25 / 99%</td>
<td>0.50</td>
</tr>
<tr>
<td>0x982F</td>
<td>200 / 100%</td>
<td>2 / 100%</td>
<td>0.01</td>
</tr>
</tbody>
</table>

![Graph showing cumulative percentage of dynamic branches vs cumulative percentage of mispredictions]
Static Confidence Experiment

![Graph showing the relationship between % of Dynamic Branches and % of Mispredicts. The graph includes a point labeled (25.2, 70.6).]
Dynamic Confidence Mechanisms

1. Index into a hardware table using branch PC.

2. Each entry in the hardware table is a shift register containing history of prediction correctness:
   - CIR ("sir"): correct/incorrect register
   - on a mispredict, shift in a ‘1’; else shift in a ‘0’

3. Similar to static experiment, maintain a software table that keeps a profile for each possible CIR value:
   - count number of occurrences of each CIR value (N)
   - count number of mispredicts upon seeing that CIR (m)
   - construct graph similarly (sort CIRs, plot cum. %...)
Dynamic: Single-Level

Program Counter → global BHR

0101010101010101 → 16-bit CIR

reduction function → high/low confidence signal

for software measurement only...

2^{16} entries

N m m/N
Dynamic: Two-Level

Level 1
CIR History Table

n bit entries

2^m entries

Program Counter

Level 2
CIR History Table

p bit entries

2^n entries

global BHR

for software measurement only...

N  m  m/N

m bits

high/low confidence signal

reduction function
Parameters

• underlying predictor: 64K entry gshare

• CIR size (all levels) = 16 bits

• confidence table sizes:
  - first level: 64K entries
  - second level: 64K entries

• we vary indexing method, both levels

• Two sets of results:
  - limit study ("ideal") => use CIR patterns
  - implementations => use reduction functions
Single-Level Methods

% of Mispredicts vs % of Dynamic Branches for different methods:
- "static"
- "PC"
- "BHR"
- "BHRxorPC"

(c) Eric Rotenberg
Two-Level Methods

% of Dynamic Branches

% of Mispredicts

"static"

"PC-CIR"

"BHRxorPC-CIR"

"BHRxorPC-BHRxorCIRxorPC"
Best Single- and Two-Level

% of Dynamic Branches

% of Mispredicts

"static"
"BHRxorPC"
"BHRxorPC-CIR"

(c) Eric Rotenberg
“Assigning Confidence to Conditional Branch Predictions”
Real Implementations

• Reduction functions:
  - saturating counters
    Increment if correct, Decrement if incorrect.
  - resetting counters
    Increment if correct, Reset to 0 if incorrect.
  - ones counting
    Maintain CIRs in confidence table, count one bits.

• To limit graphs, we’ll show only the “best” single- and two-level methods:
  - PC xor BHR
  - PC xor BHR / CIR
Single-Level, Real Implementations

![Graph showing the relationship between the percentage of dynamic branches and the percentage of mispredictions for different branch prediction schemes. The graph includes lines for various schemes labeled as follows:

- "BHRxorPC"
- "BHRxorPC.ICnt"
- "BHRxorPC.Reset"
- "BHRxorPC.Sat"

Each line represents a different strategy for assigning confidence to conditional branch predictions. The x-axis represents the percentage of dynamic branches, while the y-axis represents the percentage of mispredictions. The graph demonstrates how different methods impact the accuracy of branch predictions.]
Two-Level, Real Implementations

![Graph showing the relationship between % of Mispredicts and % of Dynamic Branches for various branch prediction schemes.

- "BHRxorPC-CIR"
- "BHRxorPC-CIR.1Cnt"
- "BHRxorPC-CIR.Reset"
- "BHRxorPC-CIR.Sat"

(c) Eric Rotenberg

"Assigning Confidence to Conditional Branch Predictions"
Effect of CIR Table Initialization

% of Mispredicts

% of Dynamic Branches

"one"  
"zero"  
"lastbit"  
"random"
Summary

- Developed a technique for establishing high/low confidence sets (correct/incorrect history...)
- Can do much better than static confidence
- Introduced single-level and two-level dynamic schemes
  - CIR tables
  - found “best” indexing methods (BHRxorPC - CIR)
  - two-level gives little or no benefit over single-level
- Introduced reduction functions
  - resetting counters achieve close to ideal
- Result: a low confidence set of 20% covers 89% of all mispredicts
Using Predictor State for Confidence

Use predictor state itself for confidence?

% of mispredicts vs % of dynamic branches

(c) Eric Rotenberg

"rcnt.4-bit"
"sat.6-bit"
"sat.4-bit"
"sat.2-bit"