Diverge Branch

Frequently executed path
Not frequently executed path
Hard to predict path

Hard to predict

Insert select-μops
(φ-nodes SSA)
Diverge-Merge Processor (DMP)
[MICRO 2006, TOP PICKS 2007]

Frequently executed path  
Frequently executed path

Not frequently executed path
Profile-Assisted Compiler Support for Dynamic Predication in Diverge-Merge Processors

Hyesoon Kim
José A. Joao
Onur Mutlu*
Yale N. Patt

HPS Research Group
University of Texas at Austin

*Microsoft Research
Control-Flow Graphs

66% of mispredicted branches can be dynamically predicated by DMP.

Exact CFM points

Approximate CFM points
Diverge-Merge Processor (DMP)

- DMP can dynamically predicate complex branches (in addition to simple hammocks).

- The compiler identifies
  - Diverge branches
  - Control-flow merge (CFM) points

- The microarchitecture decides when and what to predicate dynamically.
Why hardware *and* compiler?

- Compiler-centric solution (static predication): predicated ISA, not adaptive, applicable to limited CFG.

- Microarchitecture-only solution: complex, expensive, limited in scope.

- Compiler-microarchitecture interaction
  - Each one does what it is good at.
Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points

Select Diverge Branches and CFM points

Code generation: mark the selected diverge branches and CFM points (ISA extensions)
Simple/Nested Hammocks: Alg-exact

CFM point = IPOSDOM(A)
Frequently-Hammocks: Alg-freq

- Use edge-profiling frequencies

\[ pT(X) \text{: conditional probability of reaching basic block } X \text{ if branch A is taken} \]

\[ pNT(X) \text{: conditional probability of reaching basic block } X \text{ if branch A is not taken} \]

- Stop at IPOSDOM or MAX_INSTR

- Compute \( p\text{Merging}(X) = pT(X) \times pNT(X) \)

\[ pT(H) = 1 \]

\[ pNT(H) = 0.5 \]

\[ p\text{Merging}(H) = 1 \times 0.5 = 0.5 \]

\( \text{CFM point candidate!} \)
Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points

- Simple/nested hammocks
- Chains of CFM points
- Short hammocks
- Frequently-hammocks
- Return CFM points
- Diverge loop branches

Select Diverge Branches and CFM points

Code generation: mark the selected diverge branches and CFM points (ISA extensions)
Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points

Select Diverge Branches and CFM points

• Heuristics
• Cost-benefit model

Code generation: mark the selected diverge branches and CFM points (ISA extensions)
Heuristics-Based Selection

Motivation:
- minimize overhead: wrong path
- maximize benefit: control-independence

Do not select:
- CFM points too far from the diverge-branch
- Hammocks with too many branches on each path
- Approximate CFM points with a low probability of merging
Cost-Benefit Model-Based Selection

<table>
<thead>
<tr>
<th>Confidence</th>
<th>Estimation</th>
<th>Cost</th>
<th>Benefit</th>
<th>Compared to baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Same</td>
</tr>
<tr>
<td>Low</td>
<td>Inaccurate</td>
<td>Overhead</td>
<td>None</td>
<td>Lose</td>
</tr>
<tr>
<td></td>
<td>Accurate</td>
<td>Overhead</td>
<td>No flush</td>
<td>possibly WIN</td>
</tr>
</tbody>
</table>

\[ A = \text{accuracy of the conf. estimator} = \frac{\text{mispredicted}}{\text{low_conf}}. \]

\[ \text{cost} = (1 - A) \times \text{overhead} + A \times (\text{overhead} - \text{misprediction_penalty}) \]
Cost-Benefit Model-Based Selection

Select if $\text{cost} < 0$

$A = \text{accuracy of the conf. estimator} = \text{mispredicted} / \text{low_conf.}$

$\text{cost} = (1 - A) \times \text{overhead}$

$+ A \times (\text{overhead} - \text{misprediction_penalty})$
Compiler Support

Analysis: Identify Diverge Branch Candidates and CFM points

Select Diverge Branches and CFM points
  • Heuristics
  • Cost-benefit model

Code generation: mark the selected diverge branches and CFM points (ISA extensions)
Methodology

- All compiler algorithms implemented on a binary analysis and annotation toolset.

- Cycle-accurate execution-driven simulation of a DMP processor:
  - Alpha ISA
  - Processor configuration
    - 16KB perceptron predictor
    - Minimum 25-cycle branch misprediction penalty
    - 8-wide, 512-entry instruction window
    - 2KB 12-bit history enhanced JRS confidence estimator
    - 32 predicate registers, 3 CFM registers
  - 12 SPEC CPU 2000 INT, 5 SPEC 95 INT
Heuristic-Based Selection
Cost-Benefit-Based Selection

The cost-benefit model is simpler and effective
Input Set Effects

The graph shows the IPC delta (%) for various programs, including gzip, vpr, gcc, mcf, crafty, parser, eon, perlbrk, gap, vortex, bzip2, twolf, comp, go, jpeg, li, m88ksim, and hmean. The heuristics-same, heuristics-diff, cost-same, and cost-diff are represented by different colored bars. The overall IPC delta is 19.8%.
Conclusion

- Compiler-microarchitecture interaction is good!
- DMP exploits frequently-hammocks.
- We developed new algorithms that select beneficial diverge-branches and CFM points.
- We proposed a new cost-benefit model for dynamic predication.
- DMP and our algorithms improve performance by 20%.
Thank You!

Questions?