Using Branch Handling Hardware to Support Profile-Driven Optimization

Benefited Areas of Profile-Driven Scope Enhancement Optimization

- Instruction & Loop Scheduling
- Data Preloading
- Function in-lining
- Instruction Cache Performance Enhancement

Profiling Methods

- Traditional Software Dynamic *(compile-run-recompile)*
- Hardware Dynamic *(compile-use-recompile)*
  - Existing, e.g. Branch handling Hardware
  - Special, e.g. Profile Buffer
- Static Estimation

Drawbacks of Traditional *compile-run-recompile* method

- Execution of the probe-inserted profiled version of the program is too slow (2 to 30 times slower).
- Test Input Suites Need to be carefully chosen, so inconvenient.

Solution using Branch Handling Hardware

- Slowdown imperceptible (0.4% to 4.6% increase)
- Elimination of test input suite choosing
- Not bad accuracy

Implementation Outline of Branch Hardware Profiling

- A program is compiled with a special Identifier token, indicating it contains a table of CFG arcs.
- During execution, the kernel reads the buffer at every context switch, or more frequently to increment the arc counters of WCFG.

  Target address <---> Arc Destination
  Buffer tag <---> Arc Source
  Prediction <---> Arc Weight

- On exit, the arc table is updated on disk

Implementation Details of 2-Level Profiling

- Zeroing HR after each context-switch dumping doesn't solve over-counting.
- *Dirty-bit* can be added for each HRT entry.
  - Accurate for frequent branches.
  - But not for moderate to infrequent branches.
- *Marker bit* can be used instead. As the branch updates its history, this bit shifts to the left bit by bit.

Implementation details of 1-Level Profiling

- *Dirty bit* is still used to indicate validity of branch history
- $\hat{w} = N/(\overline{B} * d)$;
  - $\hat{w}$: average touch times of each dirty entry of BTB
  - **N** : total number of instructions executed (from hardware monitor)
  - $\overline{B}$: average number of instructions per basic block (from compiler)
  - **d**: number of dirty entries in BTB (read from BTB)

Profiling Error Quantification

- **Trace selection error**

1. TSE=0;

2. forall blocks $i \in$ WCFG(*actual*) do:

(a) $j \leftarrow$ trace_successor(WCFG(actual), i)

(b) $k \leftarrow$ trace_successor(WCFG(estimate), i)

(c) if $j \neq k$ then $TSE \leftarrow TSE + W_i * |w_{ij} - w_{ik}|$

3. enddo

- **Distribution of arc weight error versus block weights**

$$\Delta w_i = \max_{j \in succ(i)} | w_{ij} - \hat{w}_{ij} |; \qquad f_{arcs}(W) = \sum_{i\,s.t.\,Wi = W} \Delta w_i$$