Load Balancing in Decoupled Look-ahead via Do-It-Yourself Branches

Raj Parihar, Michael C. Huang

Advanced Computer Architecture Lab (ACAL)
Dept. of Electrical & Computer Engineering

University of Rochester, Rochester, NY
Motivation

- Despite the proliferation of multi-core and multi-thread, high single thread performance is still an important design goal.
- Modern programs do not lack implicit parallelism (ILP).
- Real challenge: Exploit implicit parallelism w/o undue costs.
- Decoupled look-ahead: One effective approach to exploit implicit parallelism.
Baseline Decoupled Look-ahead System

- Look-ahead thread runs on a separate core and maintains its data in local L1, no writeback to L2
- Sends execution based branch hints through BOQ and also helps prefetching in the shared L2 cache
Idea of Do-It-Yourself Branches

- Skip branch based, side-effect free code modules that do not contribute to the look-ahead purposes
- Main thread: Relies on its own branch predictor and cache prefetcher in the DIY code regions
- Look-ahead thread: Skips or dithers DIY modules
Overall Performance Impact

- Baseline look-ahead speedup over single-thread: 1.33x
- Final speedup of look-ahead over single-thread: 1.53x
- DIY + payload tuned look-ahead performance improvement: 15%
Summary

- Decoupled look-ahead can uncover significant implicit parallelism with moderate hardware support
  - However, look-ahead thread often becomes the new bottleneck

- Fortunately, look-ahead lends itself to various optimizations due to lack of correctness constraints
  - Do-It-Yourself branch based modules can be skipped in the look-ahead thread w/o affecting the quality of the look-ahead
  - Look-ahead thread’s payload can be tuned for various phases

- New decoupled look-ahead speedup over single-thread: 1.53x

- A compelling technique for turbo boosting