Accelerating Decoupled Look-ahead to Exploit Implicit Parallelism

Raj Parihar, Michael C. Huang
{parihar@ece, michael.huang@rochester.edu}

Advanced Computer Architecture Laboratory (ACAL), University of Rochester, Rochester, NY 14627

Motivation
- Despite the proliferation of multi-core, multi-threaded systems, single-thread performance is still an important processor design goal
- Modern programs do not lack instruction level parallelism (ILP)
- Real challenge: exploit implicit parallelism without undue costs
- One effective approach: Decoupled look-ahead

Look-ahead Acceleration via Speculative Parallelization
- Look-ahead binary (skeleton) offers more parallelism because certain dependencies are removed during slicing for skeleton
- Look-ahead is more error-tolerant due to lack of correctness constraint
  - Can ignore occasional dependence violations
  - Little to no support needed, unlike in conventional TLS

Look-ahead Acceleration via Weak Dependence Removal
- Not all instructions are equally important and critical for the final outcome; Plenty of weak instructions are present in a typical program
- Weak instructions can be removed safely from the look-ahead thread to speedup the look-ahead agent without degrading the quality of it

Weak Dependencies: Opportunities and Challenges
- Example of weak instructions: Inconsequential adjustments, load and store instructions that are (mostly) silent, Dynamic NOP instructions
- Challenges involved: Context-based, hard to identify and combine much like game Jenga, also interact with surrounding instructions

Experimental Analysis and Results
- Speedup of decoupled look-ahead over single-thread baseline: 1.53x
- Speedup of speculative parallel look-ahead over single-thread: 1.77x
- Speedup of conventional TLS over single-thread baseline: 1.07x
- Speculative parallel look-ahead over decoupled look-ahead: 1.13x

Self-tuned and Speculatively Parallel Look-ahead
- In some cases, self-tuned and speculatively parallel look-ahead techniques are synergistic (annm, art)
- Self-tuned + Speculative parallel look-ahead speedup: over single thread baseline 1.16x; over decoupled look-ahead baseline 1.20x

Summary and Insights
- Decoupled look-ahead can uncover significant implicit parallelism
- Look-ahead thread often becomes a new bottleneck
- Fortunately, look-ahead thread tends itself to various optimizations due to lack of hard correctness constraints
- Speculative parallelization is more beneficial in look-ahead thread compared to main program thread due to increased parallelism
- Weak instructions can be removed w/o affecting look-ahead quality
- Intelligent look-ahead technique is a promising solution in the era of flat frequency and modest microarchitecture scaling

References

Acknowledgements
NSF (Grant #CCF-0747124), HSFC (Grant # 61028004), Alok Garg