Exploring Look-ahead Techniques to Improve Single Thread Performance

Raj Parihar

*Joint work with Alok Garg, Michael C. Huang
University of Rochester

- Small private research university
  - 4400 undergrad students
  - 2800 grad students
- Set on the Genesee river in western upstate New York
  - 350 miles from NYC
  - 70 miles from Niagara Falls
- ECE department
  - 25 faculty members
  - Student faculty ratio: 3.5:1
Motivation

- Single-thread performance still important design goal
- Bottlenecks in single thread performance
  - Branch mispredictions
  - Cache imperfections
  - Unscalable rename/decode
- Large body of work inspired from decoupled Access/Execute architecture
  - Slipstream, Run-ahead, Dual core execution, Microthreads
- One effective approach: Decoupled Look-ahead
Outline

• Motivation
• Baseline decoupled look-ahead
• Enhanced look-ahead techniques
  • Speculative parallel look-ahead
  • Self tuning decoupled look-ahead
  • Look-ahead in multicore system
• Additional details
• Summary
Two Approaches for Higher Performance

- Increase size of caches, branch prediction tables, register files
- Use of specialized hardware/accelerators
  - Spend extra transistors on specialized blocks

- Diminishing return
  - Hits plateau sooner or later
- Power hungry structures

- Achieves close to ideal performance
  - Branch outcome using dual-path execution
Baseline Decoupled Look-ahead

- Binary parser is used to generate skeleton from original program
- The skeleton runs on a separate core and
  - Maintains its own memory image in local L1, no write-back to shared L2
  - Sends all branch outcomes through FIFO queue and also helps prefetching

Baseline Look-ahead Analysis

- Comparing four systems
  - Baseline
  - Decoupled look-ahead
  - Ideal
  - Look-ahead alone

- Application categories
  - Bottlenecks removed
  - Speed of look-ahead not the problem
  - Look-ahead is the new bottleneck

(a) Integer applications.

(b) Floating-point applications.
Speculative Parallelization in Look-ahead

- Skeleton code offers more parallelism
  - Certain dependencies removed during slicing for skeleton

- Look-ahead is inherently error-tolerant
  - Can ignore dependence violations
  - Little to no support needed, unlike in conventional TLS

Assembly code from equake
Speedup of speculative parallel look-ahead

- 14 applications in which look-ahead is bottleneck
- Speedup of look-ahead systems over single thread
  - Decoupled look-ahead over single thread baseline: 1.61x
  - Speculative look-ahead over single thread baseline: 1.81x
- Speculative look-ahead over decoupled look-ahead: 1.13x
Self Tuning Look-ahead

- Strong vs weak dependences in look-ahead
- Genetic algorithm (GA) to determine the strength of dependences during the execution (runtime profiling)
Speedup of GA based Look-ahead

- GA based look-ahead achieves \textbf{1.15x} speedup over baseline look-ahead
  - Baseline look-ahead over single thread is already 1.61x
Look-ahead in Multicore

- Large number of cores in manycore systems would be idle due to low inherent scalability

```c
/* IS Kernels */
1 Do for i=1 to Imax {
2 if(master_thread)
3 modify_sequence_of_keys;
4 BARRIER(all_procs);
5 for ( i=0; i<NUM_KEYS; i++ )
6 compute_rank_of_each_key_locally;
7 lock(CS_lock);
8 update_global_key_array;
9 unlock(CS_lock);
10 BARRIER(all_procs);
11 if(master_thread)
12 perform_partial_verification;
13 }
```

*Complementing User-Level Coarse-Grain Parallelism with Implicit Speculative Parallelism, N Ioannou, M Cintra, MICRO, Dec ‘11

- Use some of the idle cores to do look-ahead for other active cores
Additional Details

• TLS+GA based look-ahead system
  • 4% speedup over TLS based look-ahead

• Energy implications
  • Energy overhead is about 5% compare to single thread
  • Major saving comes from avoiding caches stalls

• Design complexity
  • Look-ahead thread can be designed, implemented and validated in least complex manner i.e. No load-store reply, partial renaming, aggressive speculation with no rollback
Summary

• Decoupled look-ahead can significantly improve single-thread performance

• Look-ahead lends itself to various optimizations which are not possible in single thread otherwise
  • Skeleton construction removes dependences and increases parallelism
  • Hardware design is flexible and can be a simple extension of SMT
  • Occasional errors can be tolerated (approximate computing)

• Our techniques/findings can be applied more generically to approximate computing
  • We have shown it to work in look-ahead to improve single thread
References


• **Complementing User-Level Coarse-Grain Parallelism with Implicit Speculative Parallelism** Nikolas Ioannou, and Marcelo Cintra *Intl. Symp. on Microarchitecture (MICRO)*, p 284-295, December 2011