Department of Electrical and Computer Engineering

University of Rochester, Rochester, NY

Ph.D. Public Defense

Wednesday, July 27, 2016

10:30 AM

Computer Studies Building, Room 426

Accelerating Decoupled Look-ahead to Exploit Implicit Parallelism

Raj Parihar

Supervised by
Professor Michael C. Huang

Abstract

Despite the proliferation of multi-core and multi-threaded architectures, exploiting *implicit* parallelism for a single semantic thread is still a crucial component in achieving high performance. While a canonical out-of-order engine can effectively uncover implicit parallelism in sequential programs, its effectiveness is often hindered by instruction and data supply imperfections (manifested as branch mispredictions and cache misses). *Look-ahead* is a tried-and-true strategy to exploit implicit parallelism, but can have resource-inefficient implementations such as in a conventional, monolithic out-of-order core. A more decoupled approach with an independent, dedicated look-ahead thread on a separate thread context can be a more flexible and effective implementation, especially in a multi-core environment. While capable of generating significant performance gains, the look-ahead agent often becomes the new speed limit; thus, we explore a range of software and hardware based techniques for accelerating the look-ahead agent to exploit implicit parallelism.

Fortunately, the look-ahead thread has no hard correctness constraints and presents new opportunities for optimizations which are not present in traditional architecture. First, we explore speculative parallelization in the look-ahead thread which is especially suited for the task of accelerating the look-ahead agent. Second, we observe that not all dependences are equal, and some links in a dependence chain are *weak* enough that removing them in the look-ahead thread does not materially affect the quality of look-ahead. A trial-and-error approach and a genetic algorithm based *self-tuning* framework can reliably identify weak instructions to improve the speed of the look-ahead thread. We further tune look-ahead payload via skipping side-effect free, non-critical (we call them *Do-It-Yourself* or DIY) branches. Finally, we apply self-tuning principles in a multi-program environment to improve overall protection and utilization of shared caches which are often shared among multiple competing programs.

With a series of faithful simulation experiments and detailed insightful analysis, we show that while the two main drivers for single-thread performance – faster clocks and advancements in microarchitecture – have all but stopped in recent years, we can still uncover significant implicit parallelism using *intelligent* look-ahead techniques which brings impressive performance gain at relatively low cost.