18-740 Fall 2013 Research Proposal: Tournament Prefetcher

Ryan MacDonald Edward Sears

Abstract

In this paper, we will propose a plan to research the performance benefit of a tournament style prefetcher. We will begin by defining the problem and describe why our idea can help sove this problem. Next, we will formally state our hypothesis and finish by describing the methodology of our research.

1. The Problem

Prefetching is a memory latency tolerance mechanism which aims to anticipate accesses to cache blocks and speculatively issue memory requests to those blocks with the intent of warming the cache for future execution. There are various types of prefetching strategies used in modern processors including stream, next-block, and Markov prefetchers. While each of these prefetching strategies has its advantages, none of them are optimal for all memory access patterns and workloads. See table 1. For example, a stream-based strategy may provide accurate, timely prefetches for a streaming access pattern, but it would underperform in the case of a more random, non-contiguous access pattern. Thus, modern processors require a more general purpose, hybrid prefetching mechanism to preserve optimal memory latency tolerance over a variety of memory demands within and between applications.

Prefetcher type	Advantages	Timeliness
Next-block	Simple to implement [1]	High
Stream-based	Can cover linear access patterns [1]	Medium
Markov	Can cover arbitrary access patterns [1]	Low

Table 1: Prefetcher Advantages

2. Novelty

Research has been conducted towards continuously monitoring prefetcher metrics including accuracy, timeliness, and cache pollution [2]; based on these measurements, prefetch behavior can be tuned and optimized by hardware at run-time. While this research has had success in that it is able to dynamically optimize prefetch aggression, it falls short in that it does not attempt to dynamically change its fundamental prefetching strategy. This research will be novel in that it will attempt to dynamically determine an optimal prefetching strategy(stream, next-block, Markov, etc.) over changing memory demands.

3. Idea

Our idea is to design, implement, and analyze the advantages and disadvantages of a hybrid, tournament prefetching mechanism which determines an optimal prefetching strategy at run-time based on the memory access pattern of the application currently running. One area of analysis will be which types of hardware prefetchers (stream, next-block, Markov, etc.) perform best together when placed in configuration with each other. A question this research will seek to answer is, "From M possible prefetching strategies, which N are optimal to include in a hybrid configuration?"

4. Hypothesis

We hypothesize that a prefetcher which dynamically takes into account performance metrics and tunes its strategy accordingly will outperform the accuracy, timeliness, and cache-pollutant behavior of a static, non-hybrid alternative. We anticipate that such a hybrid, tournament prefetcher will perform akin to the optimal static prefetching strategy for any given workload.

5. Methodology

In order to evaluate our hypothesis, we will need to extensively modify an existing processor simulator to model static and dynamic prefetching strategies. We plan to model and measure the accuracy, timeliness, and cache-pollutant behavior of the baseline simulator without prefetching, with each static prefetching strategy individually, and with various configurations of a hybrid, tournament prefetcher.

Modeling and measuring these metrics with each respective static prefetching mechanism should be relatively straightforward because they have already been documented and implemented [2], however, modeling the hybrid tournament prefetcher will be a bit more complicated. Our current design is that the hybrid prefetcher will instantiate various static prefetchers as well as a prediction unit, responsible for determining the optimal static prefetcher to use at any given time. This prediction unit will consist of a hardware structure which contains a speculative cache (s-cache) for each prefetcher strategy, keeping track of what the accuracy, timeliness, and cache blocks would be if the respective strategy had been used. The prediction unit will be updated in real time during program execution and, in this fashion, the optimal prefetching mechanism will be tracked. Thus, only one prefetcher will be issuing fetches to memory at a time; the others will merely be keeping track of what fetches they would've issued and updating their state accordingly.

We also wish to investigate how such a hybrid prefetcher could be integrated with previously explored throttling techniques. As a means to this end we hope to acquire, study, and modify the simulator used for this research [2].

6. Plan for final report

Milestone	Expected Progress
1	Finalized design for hybrid, tournament prefetcher; implemented, simulated, and benchmarked baseline, static prefetchers.
2	Implemented, simulated and benchmarked hybrid, tournament prefetcher; begin putting together final results, data, report and presentation.
Final Report	Integrate analysis of "best N static prefetchers" question and feedback directed throttling.
Moonshot goals	Implement hybrid prefetcher in RTL, use design compiler (DC) tools to further analyze performance, area, power, scalability; investigate possible use of ISA extensions and pragmas to explicitly allow the programmer to specify optimal prefetch strategy.

References

[1] Onur Mutlu, "Lecture 29: Prefetching," lecture notes for 18-447 at Carnegie Mellon University.

[2] S. Srinath, "Feedback directed prefetching: Improving the performance and bandwidth-efficiency of hardware prefetchers," *High Performance Computer Architecture*, vol. 17, pp. 63–74, 2007.