15-740 Project Proposal: Instruction Prefetching and Object-Oriented Programs

An-Cheng Huang
<pach@cs.cmu.edu>

Leejay Wu
<lw2j@cs.cmu.edu>

October 12, 1999

1 Introduction

Object-oriented programming, or OOP, exhibits behavior that differs significantly from that of traditional imperative non-object-oriented languages [4]; these differences range from shorter function lengths, to greater use of indirect jumps.

It is therefore reasonable to expect that traditional architectures well-suited for traditional programming may be non-optimal for programs written with such a dissimilar style.

One factor is that instruction cache misses occur significantly more often with OO-style programs [4]. This, combined with more frequent function calls, suggests that hiding the latency may improve IPC moreso on these programs than with more traditional C programs.

One approach is that of cooperative prefetching [7], an improvement over completely hardware-based methods for instruction prefetching. These older methods include next-N-line-prefetching, target-line prefetching, wrong-path prefetching, and Markov prefetching, ([14], [15], [9], [6], all cited respectively in [7])).

2 Objectives

- To assess the relative impact of these prefetching schemes on C and C++ programs.
- To consider possible optimizations for C++ programs, such as predicting virtual function call addresses and prefetching appropriately [11], as time allows.

http://www.cs.cmu.edu/~pach/740/proj740.html

3 Logistics

3.1 Methodology

Our tests will be run on the SimpleScalar [1, 2, 3] simulator. For benchmarking purposes, the OOCSB suite and possibly other publicly-available C and C++ programs should suffice, such as those comprising a recent SPEC suite.

The first priority was ensuring that C++ compilation can actually be done with this simulator. We found that SimpleScalar supports a superset of MIPS-IV, and with some minor bug-fixing and the addition of libg++ is capable of supporting C++ development and testing [3].

3.2 Schedule

The following is only an estimated schedule.

Week Beginning	**An-Cheng**	**Leejay**
October 10	Finish proposal. Write web page. Build OOCSB suite on SimpleScalar.	
October 17	Run initial simulations.	
October 24	Implement simple prefetching techniques.	
October 31	Benchmark selected programs on modified system.	
November 7	Analyze results. Attempt C++-targetted optimizations as appropriate.	
November 14	Poster presentation.	
November 21	Finishing touches on poster.	

The entire path is critical, since it's very linear. For this reason, we have not divided the tasks.

3.2.1 Milestone

We intend to have results showing that, I-cache prefetching improves C++ program performance to a greater degree than C, due to the typically higher miss rates. We also intend to attempt prefetching based on virtual function call prediction, followed by benchmarking.

3.3 Literature Search

In addition to papers cited above, various other papers were used to gain a brief overview of some of the issues involved with object-oriented programming. Predicting the targets of indirect branches is one approach [5, 11]. Another approach towards avoiding unneeded saves and restores was that of "dead" values [8]. Prefetching methods have been examined for saving memory latency when dealing with linked data structures [10, 13]. A variety of methods was considered in [12].

3.4 Resources

We need, and have the usage of, a big-endian machine on which to build SimpleScalar, along with libg++ and the cross-compiler.

We have a cross-compiled binary implementation of the SPEC95 suite.

We may need to forego sleep for prolonged periods of time.

3.5 Stuff So Far

We have obtained SimpleScalar, and built our development environment on a Solaris/SPARC server.

References

[1] Todd M. Austin. A user's and hacker's guide to the SimpleScalar architectural research tool set. `ftp://ftp.cs.wisc.edu/sohi/Code/simplescalar`, January 1997.

[2] Todd M. Austin and Doug Burger. SimpleScalar tutorial. `ftp://ftp.cs.wisc.edu/sohi/Code/simplescalar`.

[3] Doug Burger and Todd M. Austin. The SimpleScalar tool set, version 2.0. Technical report, University of Wisconsin-Madison, June 1997.

[4] Brad Calder, Dirk Grunwald, and Benjamin Zorn. Quantifying behavioral differences between c and c++ programs. *Journal of Programming Languages*, 2(4), 1994.

[5] Po-Yung Chang, Eric Hao, and Yale N. Patt. Target prediction for indirect jumps. *Proceedings of the 24th International Symposium on Computer Architecture*, 1997.

[6] Doug Joseph and Dirk Grunwald. Prefetching using markov predictors. *Proceedings of the 24th International Symposium on Computer architecture*, June 1997.

[7] Chi-Keung Luk and Todd C. Mowry. Cooperative prefetching: Compiler and hardware support for effective instruction prefetching in modern processors. *Proceedings of Micro-31*, December 1998.

[8] Milo M. Martin, Amir Roth, and Charles N. Fischer. Exploiting dead value information. *Proceedings of Micro-30*, December 1997.

[9] Jim Pierce and Trevor Mudge. Wrong-path instruction prefetching. *Proceedings of the 29th Annual IEEE/ACM International Symposium on Microarchitecture*, 1996.

[10] Amir Roth, Andreas Moshovos, and Gurindar S. Sohi. Dependence based prefetching for linked data structures. *Proceedings of ASPLOS-8*, October 1998.

[11] Amir Roth, Andreas Moshovos, and Gurindar S. Sohi. Improving virtual function call target prediction via dependence-based pre-computation. *Proceedings of ICS-99*, June 1999.

[12] Amir Roth and Gurindar S. Sohi. New methods for exploiting program structure and behavior in computer architecture. *Proceedings of IWIA-98*, October 1998.

[13] Amir Roth and Gurindar S. Sohi. Effective jump-pointer prefetching for linked data structures. *Proceedings of the 26th International Symposium on Computer Architecture*, May 1999.

[14] Alan J. Smith. Sequential program prefetching in memory hierarchies. *IEEE Computer*, 11(2), December 1978.

[15] J. E. Smith and W.-C. Hsu. Prefetching in supercomputer instruction caches. *Supercomputing '92*, July 1992.