15745 Optimizing Compiler Project Proposal: Implementation of Decoupled Software Pipelining (DSWP) in the LLVM Compiler Infrastructure

1 Group Information

Fuyao Zhao: fuyaoz@cs.cmu.edu
Mark Hahnenberg: mhahnenb@andrew.cmu.edu

2 Project Web Page

http://www.cs.cmu.edu/~fuyaoz/courses/15741/index.html

3 Project Description

The goal of the project is to survey the idea of decoupled software pipelining [4], which is essentially a coarse-grained loop body thread level parallelization technique that has been implemented in the VELOCITY research compiler [1]. In this project, we'll first try to implement it in the LLVM compiler infrastructure and then investigate the performance and practicability of this technique.

- 75% goal: partially implement the original decoupled software pipelining [4] in LLVM. It will be functional, but has limited ability to deal with different kind of data structures.
- 100% goal: implement the full version of DSWP in LLVM.
- 125% goal: there could be two possible options to achieve 125% goal:
  - implement parallel-stage decoupled software pipelining [3].
  - implement speculative decoupled software pipelining [5].

4 Logistics

4.1 Plan

- Week 1 $\sim$ 2 (Preparation): Do literature research. We are going to going through the detail of the important papers. Understand how to use data dependency analysis in LLVM, and how to generated multiple thread code.
- Week 3: Begin to implementing DSWP.

- Week 4 (Milestone): Get initial prototype working to the point where it can do dependency analysis, thread partitioning, thread communication, and code generation. There might be some performance issues and bugs after this stage.
- Week 5: Finish the implementation. If everything is going well, then we'll begin to purse the 125% goal.
- Week 6: Refine the system, evaluate the our system on different benchmarks, finish the final report.

4.2 Getting started

We have read several papers, though not in complete detail. We've also collected a series of papers that could possibly be helpful. We found that the project poolalloc [2] could be helpful since it implements some dependency analysis and also could be a basic example of how to parallelize programs in LLVM using Cilk. We are not sure if we can get the source code of VELOCITY compiler, which could also be greatly helpful.

References

[1] Matthew John Bridges. *The velocity compiler: extracting efficient multicore execution from legacy sequential codes.* PhD thesis, Princeton, NJ, USA, 2008. AAI3332403.

[2] Christopher Arthur Lattner. *Macroscopic data structure analysis and optimization.* PhD thesis, Champaign, IL, USA, 2005. AAI3182303.

[3] Easwaran Raman, Guilherme Ottoni, Arun Raman, Matthew J. Bridges, and David I. August. Parallel-stage decoupled software pipelining. In *Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization*, CGO '08, pages 114–123, New York, NY, USA, 2008. ACM.

[4] Ram Rangan, Neil Vachharajani, Manish Vachharajani, and David I. August. Decoupled software pipelining with the synchronization array. In *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*, PACT '04, pages 177–188, Washington, DC, USA, 2004. IEEE Computer Society.

[5] Neil Vachharajani, Ram Rangan, Easwaran Raman, Matthew J. Bridges, Guilherme Ottoni, and David I. August. Speculative decoupled software pipelining. In *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, PACT '07, pages 49–59, Washington, DC, USA, 2007. IEEE Computer Society.