Predicting sequence of parameterized transform passes using Machine Learning techniques

Ashique Khudabukhsh (khudabukhsh@gmail.com)
Jayant Krishnamurthy (jayantk@cs.cmu.edu)

March 22, 2012

1 Project Description

The performance of code generated from an optimizing compiler depends not only on the optimizations performed, but also upon their order. However, it is typically unknown what sequence of optimizations should be applied to any given input program. This problem is exacerbated by parametrized transform passes, where performance depends on the chosen parameter configuration. A bad parameter configuration for such transform can adversely affect performance. For example, for a given program, a bad parameter configuration for loop unrolling may result in a code that runs slower than the unoptimized original code. Selecting an appropriate sequence of optimizations and their parameters is critical to achieving maximum performance in the compiled program.

Previously, machine learning has been used to predict a high-performance sequence of optimization passes for a given program or method[1, 2]. However, for computational tractability, these approaches severely limit the space of optimization sequences from which the prediction is drawn. In [1], the predicted optimization sequence must be a modification of an existing, hand-tuned strategy. In [2], a classifier is trained to predict an entire pass sequence; since a pass sequence is treated as an atomic unit, the predicted sequences must be relatively short (since the number of sequences grows exponentially in sequence length). As a consequence, these approaches are likely to find

suboptimal optimization sequences, since they explore only a small subset of the space of optimization sequences.

We propose to explore using machine learning to determine high-performance optimization sequences and parameters. Our approach is designed to be more scalable than previous approaches, and as a consequence will allow us to search longer sequences of optimizations. Ideally, this increased exploration will result in better compiled program performance.

We compute the overall effect of a sequence of passes in the following manner. For a given feature vector of code and a transform pass, we learn a model to predict the feature vector of the resulting code after applying the pass. For our first pass in the sequence, we start with the feature vector of our original, unoptimized code as input. For each subsequent pass, we treat the resulting intermediate feature vector obtained in previous step as input. This gives us the freedom to create a sequence of passes of arbitrary length. We also learn a model to predict the speedup factor of an intermediate feature vector. By using these two models, we can apply beam search to search through a large space of sequence of passes.

2 Goals

- 75 percent goal: Implement code to extract features from programs and measure program performance with different optimizations and parameter settings. Learn models to predict intermediate feature vectors and speedup factor. Evaluate performance of models (accuracy of intermediate code representations and running time predictions).
- 100 percent goal: Compare approach to prior work. Demonstrate that increased exploration ability improves program performance.
- 125 percent goal: Explore other search options. One possible direction is to use Iterative-Local-Search based parameter optimization tools such as ParamILS to predict globally efficient sequences.

3 Proposed Schedule

- March 26th - April 9th: Get SPEC benchmarks, write code to apply various optimizations to each program, storing the resulting compiled

code and measuring resulting performance.

- April 9th - April 19th: Implement machine learning models. This includes generating program features and trying several prediction algorithms (e.g., multiple regression, nearest neighbor).
- **Milestone**: Demonstrate preliminary prediction results for both intermediate program representations and program runtime.
- April 19th - May 2nd: Tune learned models to improve prediction accuracy. Compare learned models with prior work.

Project Webpage: `http://www.cs.cmu.edu/~jayantk/15745/`

4 Resources

Our project will be implemented using LLVM and our evaluation will use the SPEC benchmarks.

References

[1] F. Agakov, E. Bonilla, J. Cavazos, B. Franke, G. Fursin, M. F. P. O'Boyle, J. Thomson, M. Toussaint, and C. K. I. Williams. Using machine learning to focus iterative optimization. In *Proceedings of the International Symposium on Code Generation and Optimization*, pages 295–305, Washington, DC, USA, 2006. IEEE Computer Society.

[2] R. N. Sanchez, J. N. Amaral, D. Szafron, M. Pirvu, and M. G. Stoodley. Using machines to learn method-specific compilation strategies. In *CGO*, pages 257–266. IEEE, 2011.