Machine Learning Driven Compiler Tuning

クルス アヨロア, アーナルド ホセ

https://doi.org/10.15017/1654900

（別紙様式 2）

氏　　名　：アーナルド　ホセ　クルス　アヨロア

論 文 名　：Machine Learning Driven Compiler Tuning
（機械学習を用いたコンパイラ最適化技術）

区　　分　：甲

論　文　内　容　の　要　旨

It is the job of the compiler to translate human-readable code into machine code that makes efficient use of hardware resources. The code translation and optimization process is a complex one, because the compiler has to accommodate the program to the available hardware resources while at the same time preserving the original functionality. Compilers have been the subject of research for several decades, and there are already well established optimization strategies that can result in high speedups. Still, modern compilers fall short in selecting appropriate optimizations that yield the highest speedups due to the size of the optimization space, and because of the particularities of the source code, the target hardware and even the compiler itself.

This work explores the application of machine learning (ML) for compiler tuning, to improve the compiler's capacity at selecting beneficial optimization strategies for a given input program. Our highest success was at predicting when to apply vectorization, a very powerful compiler optimization that targets data parallelization and efficient memory use but which can also be detrimental if used incorrectly. We were able to train a ML predictor with 93% of accuracy at guessing from a high level description of our benchmark programs whether vectorization would be successful, resulting in a median 70% program speedup over Intel's ICC compiler. Another unique study was conducted at re-targeting our vectorization predictor from reducing execution time to reducing energy consumption. Our experiments yielded an average 64% decrease in energy consumption and only 5% decrease in energy in the case of mispredicitons.

In predicting multiple optimizations, we were able to identify the four main challenges in making a ML approach practical, and proposed techniques to ameliorate this situation. First, we designed an optimization space pruning technique that makes regression-based predictors feasible by bounding the prediction time on the size of the training set. We also proposed a multi-predictor modeling technique that increased prediction performance by 40% for our benchmarks. In addition, we tested using code generators as an alternative to hand coded benchmarks for training and testing predictors. Other contributions included multi-dimensional data visualization techniques to aid in characterizing programs, and a more rigorous predictor evaluation scheme. Finally, in order to help move this research filed forward, we opened the TeaBowl project to share our results with the compiler researcher community.