Abstract

Tuning compiler optimization for a given application of particular computer architecture is not an easy task, because modern computer architecture reaches higher levels of compiler optimization. These modern compilers usually provide a larger number of optimization techniques. By applying all these techniques to a given application degrade the program performance as well as more time consuming. The performance of the program measured by time and space depends on the machine architecture, problem domain and the settings of the compiler. The brute-force method of trying all possible combinations would be infeasible, as its complexity $O(2^n)$ even for "on-off" optimizations. Even though many existing techniques are available to search the space of compiler options to find optimal settings, most of those approaches can be expensive and time consuming. In this paper, machine learning algorithm has been modified and used to reduce the complexity of selecting suitable compiler options for programs running on a specific hardware platform. This machine learning algorithm is compared with advanced combined elimination strategy to determine tuning time and normalized tuning time. The experiment is conducted on core i7 processor. These algorithms are tested with different mibench benchmark applications. It has been observed that performance achieved by a machine learning algorithm is better than advanced combined elimination strategy algorithm.
Evaluation of Tuning and Normalized Tuning Time using an Effective and Automated Framework pertaining to Benchmark Applications

References
- GCC online documentation http://gcc.gnu.org/onlinedocs/
- www.networktheroy.co.uk
- www.sourceforge.net
- Open64: an open source optimizing compiler. www.open64.net
- ROSE: www.rosecompiler.org

Index Terms

Computer Science

Information Sciences

Keywords
Machine learning  Program features  Compiler optimization  Mibench