Imperial Journal of Interdisciplinary Research (IJIR)
Vol-3, Issue-4, 2017
ISSN: 2454-1362, http://www.onlinejournal.in

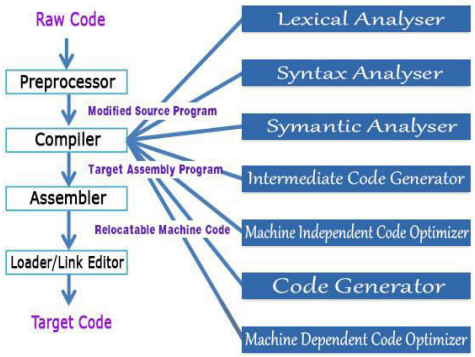
Machine learning to improve Compiler Heuristics

Palash Gupta [1], Himanshu Jain [2] & Prof Kannadasan R [3]

Abstract:*There is a need of developing systems which are fast and have a good accuracy to provide a solution to non-linear, troublesome issues in compilers, which are capable of being generated quickly and should also be reliable. Machine Learning gives us a chance to develop complicated models which are accurate in representing the complex problems.*

Keywords: *Machine Learning, Heuristics, Compiler, Loop unrolling, optimization, Memory, Processor.*

Literature Review:



Phases of a Compiler

A compiler can be understood as a special computer program which processes statements and transforms the source code of a specific programming language to another computer language also known as machine language that a computer's processor uses. The three main phases of a compiler include the Front end, Middle end and the Back end. Each phase takes input from the previous one, and each one has a unique representation of the source program and it further feeds its output to the next phase of the compiler.

Machine Learning can be understood as a process which has certain inputs which are mapped to a set of outputs, almost equivalent to a mathematical function. In a conventional compiler, the hard decisions are either not considered by any means – being mapped to a certain number for all inputs – or else are made by physically composed and tuned heuristics. Machine Learning can be utilized to supplant these heuristics with legitimate factual examination and demonstrating, which can better express the genuine way of the issue. Artificial Neural Network, Independent and Identically Distributed Model, Markov Model, are some of the most commonly used machine learning techniques applied in compilers.

In order to exploit and make the most of the computer architecture, Complier plays a crucial role. The optimization of compiler leads to maximizing the throughput from a microprocessor. The efficiency of a complier depends on heuristics and each and every new processor that is made requires generation of new heuristics. The generation of optimized heuristic can be achieved to a very extent by the help of machine learning. Loop unrolling is a simple transformation which can help achieve the same. The transformation can have good or bad effect on program.

The ability of a compiler to achieve great performance depends on the extent to which it can exploit the architecture of the system on which the complier is made to run on. There are several heuristics which are integrated with a compiler and it entirely depends on the complier to choose the most efficient out of all. The heuristic chosen by a complier entirely depends on the architecture and the organization of system. Heuristic generation by machine learning is one such way in which the heuristics of a complier are generated by artificial intelligence based on several factors, such are having the prerequisite knowledge, learning from environment and so on.

Loop unrolling is a method which will be focused, which has its roots through machine learning. The processor works in high frequency and achieves Instruction Level Parallelism(ILP) in which several instructions can be executed in parallel, all instructions being processed at the same time. The memory hierarchy of a system incorporates several types of memory, the one which is accessed easiest and the fastest by CPU is the smallest in size and the most expensive one, and as the speed to access the memory by CPU decreases, the size increases. Usually there is presence of cache memory between the CPU and the main memory, in order to fasten the process to access the data. In a single CPU cycle, there are hundreds of call to the memory, and by accessing the memory faster, the work can be done faster. There are many levels of

cache memory as well, for example, if there are 3 levels, level 1 will be accessed the fastest, then level 2 and finally level 3.

Loop unrolling involves a simple program of transformation in which the loop is replicated several number of times. It can be applied to any level and can optimize them simultaneously. It reduces the loop management overhead and has benefits of subexpression elimination. Its disadvantages include, that if the loop id too big that it cannot be fitted into the cache, then the program won't work. Additional loading may be required to load the cache with the instructions. When this technique is applied, many other effective techniques could not be applied with it. This reduces the chance of optimization.

Machine learning can play a major role in deciding the various parameters for the heuristics of complier. One of the methods to optimize is to reduce the number of parameters involved as the number of dimensions of learning space increases, the performance of machine learning decreases.

So basically, the few initial steps include, finding a loop which captures "performance feature in such a way that the optimization of loop is built on learning set. Then it needs to be decided if, loop unrolling needs to be applied or not. This is done by automated learning process. Now, the output or the formulation that is obtained is used as the heuristic for compiler.

Loop abstraction is used to find the main characteristics of loop that influence the execution of any processor. 5 classes of integer features are used, which include, Memory access, Arithmetic operations count, Size of the loop body, Control statements in the loop and Number of iterations. Not all these features are used, but only a part of them are used for a given complier and are applied on targeted processor.

The original loop and that of the unrolled loop is compared, then the performance is compared on the basis of various factor, mainly, execution time. Four cases are obtained, which are not significant, equal, improved and degraded. The next step involves partitioning the loop set into equivalent class. If the abstractions of two loops are found to be equal them they are categorized in the same class.

Now the unrolling decisions are represented using the decision tree. Decision tree is based on vectors in which each node represents a test which checks the value of a certain feature. Decision tree is a 2D representation in which various factors based on statistics are considered, and each of the node is a parameter, the leaves represent classes which have to be judged.

There are certain drawbacks of using machine learning in compiler heuristics. It is important to understand that it not just an issue of expelling a heuristic from a compiler, and socketing in a model in its place. There are noteworthy challenges to be conquered, both in the choice and use of components, and in the preparation procedure of building a model. To overcome these drawbacks, some of the possible solutions are: Overfitting, Cross-Validation, Underfitting.

The set and the steps through which a complier optimization is done, is with the help of hard coded order in heuristic stored in each complier. So basically, this could be improved as the complier is used in new machine, and there won't be a need to redesign or reengineer, the complier heuristic, with the help of unrolling loop, and the decision tree produced, statistically, it can be found the best optimization way. This will help achieve the efficient heuristics for every processor, and would increase its efficiency with time.

Conclusion:

The whole process of reviewing research papers gave us an in depth knowledge of existing methodology of the issues being faced by the present compilers and also few techniques which are still being researched for proper implementation. Machine Learning is a way through which we can resolve these issues. Loop unrolling is one of the methods which we have used for the design of compiler heuristics. Considering this to be primary result there are still many issues left to be addressed. We strongly believe that Machine Learning can help resolve the issues.

References:

[1] François Bodin, Antoine Monsifrot, and Ren´e Quiniou. A Machine Learning Approach to Automatic Production of Compiler the Heuristics.

[2] Gennady G. Pekhimenko. Machine Learning Algorithms for Choosing the Compiler Heuristics.

[3] John D. Thomson. Using Machine Learning to Automate Compiler Optimization.

[4] S.Kulkarni. Improving compiler optimization using machine learning.