Politenico di Milano

Dipartimento Elettronica, Informazione e Bioingegneria

HEAPLab Project Report

Performance Estimation for TAFFO via LLVM-MCA

Author:
Marco Prosdocimi

Supervisors:
Stefano Cherubin
Daniele Cattaneo

July 23, 2020

Abstract

Use LLVM-MCA to compare the fixed-point code produced by TAFFO with the original floating-point code for all loops in the code. LLVM-MCA is a tool that simulates the inner behavior of the CPU to estimate the performance of a machine code snippet. TAFFO is an autotuning framework, based on LLVM 8, which tries to replace floating-point operations with fixed-point operations as much as possible.

1 Introduction

The purpose of the project is to write a tool to find body loops in a c program, compile them with and without TAFFO, and use LLVM-MCA checks if TAFFO improves the execution of the loops by reducing the total amount of machine cycles required.

2 Design and Implementation

The tool is divided into two different parts: the Loops Finder and the LLVM-MCA analysis.

2.1 Loop Extractor

By using the LLVM API, I was able to write an LLVM Pass that searches loops in code and add an Assembly inline comment in the header and the exit block.
The Assembly comments are "LLVM-MCA-BEGIN" and "LLVM-MCA-END". They are used in the LLVM-MCA analysis. In the event of nested loops, the pass marks only the most inner one.
Before running this pass, we use the loop simplify pass to try to have simple form loops, with a pre-header and just one exit block.
OTP executes the Loop Finder Pass after the TAFFO ones.

2.2 LLVM-MCA analysis

I had to modify the main of LLV-MCA so that he shows the total machine cycle count for all marked regions. LLVM-MCA is designed to print several different analysis concerned the different region that was marked in the code. For this project, I was interested only in the total amount of machine cycle for all the regions. In the file ”llvm-mca-mod.c” I underline the originally LLVM-MCA code and the few lines that I change.

2.3 Use a heuristic approach to different weight the loops

WIP.

2.4 Tool Steps

The tool follows thees main step:

1. By using CLANG, it compiles the code and emits the LLVM IR.
2. To generate the TAFFO IR file, OPT runs all TAFFO passes, and then the Loop Extractor pass.
3. For the non TAFFO IR file, it executes only Loop Extractor pass.
4. The two IR files are compiled using LLC.
5. Uses the mod version of MCA on both the assembly.
6. Print the results of the analysis for each of the two assembly file.

3 Experimental Results (WIP)

I run the tool on several tests that are used for TAFFO.

3.1 Polybench benchmarks

The Polybench benchmarks are 30 different tests that have one loop that dominates all the execution times. I run tests and get the MCA analysis for two different processors: an INTEL-I7 x86-64 CPU processor and an INTEL-CORE2 x86-64.

Result of LLVM-MCA		
	INTEL X86 CORE2	INTEL X86 I7
Total number of test improved by TAFFO	10	20
Average of total amount of machine cycle reduced by TAFFO	WIP	WIP

4 Conclusions

WIP