Politenico di Milano

Dipartimento Elettronica, Informazione e Bioingegneria

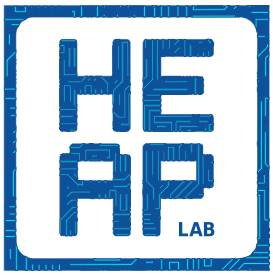
HEAPLab Project Report

Performance Estimation for TAFFO via LLVM-MCA

Author:
Marco Prosdocimi

Supervisors:
Stefano Cherubin
Daniele Cattaneo

September 2, 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.
Unfortunately, I'm not always able to render loops with a single exit block in this version of the tool this kind of loops are skipped.
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 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

I run the tool on several tests get the MCA analysis for AMD A8-6600K a x86 CPU 64 bit.

3.1 Polybench benchmarks

The Polybench benchmarks are 30 different tests that have one loop that dominates all the execution times. I wrote a shell script that runs those tests with and without TAFFO and use my tools to estimate the number of machine cycles.

See Appendix A for more detail about sinlgle tests.

Table 1: summary of Polybench benchmarks

Result of LLVM-MCA	
Total number of tests improved by TAFFO	11
Average of machine cycle reduced by TAFFO	1130

3.2 Axbench benchmarks

For these tests, instead of using the shell script tool, i wrote a Make file that compiles the code with and without TAFFO. I generating two different assembly code. Then i use the LLVM-MCA to estimate the machine cycles.

Table 2: machine cycle measured by use LLVM MCA for Axbench benchmarks

Test Name	Total number of CC	Total number of CC with TAFFO
Blackscholes	6402	8571
fft	36718	11281
jmeint	1720	1521
sobel	27094	11149
inversek2j	102512	44571

4 Conclusions and future study

The tool created is useful in the development of TAFFO but certainly needs some improvements. The challenging management of some loops makes the tool not entirely usable. In the future, we will have to change the "Loop Extractor" LLVM pass to be able to manage those loops that do not have a single exit block. This problem is present in some Axebench, so not all tests are fully reliable yet. In order to see the best TAFFO improvement, it is necessary to do some tests with machines without FPU (Floating point

unit). Repeat Axbench and Polybench benchmarks in this machine type can be a useful study.

5 Appendix A Polybenc benchmarks

Test Name	Total number of CC	Total number of CC with TAFFO
correlation.c	11379	12851
covariance.c	7086	7910
2mm.c	18940	16261
3mm.c	20772	17490
atax.c	8660	8791
bicg.c	14088	13855
doitgen.c	11557	14282
mvt.c	10634	11150
gemm.c	13633	12105
gemver.c	10101	9892
gesummv.c	10776	9467
symm.c	12764	13384
syr2k.c	13609	12082
syrk.c	10509	9582
trmm.c	7641	7458
cholesky.c	10705	10425
durbin.c	4789	6089
gramschmidt.c	11949	12399
lu.c	10705	10425
ludcmp.c	13660	15862
trisolv.c	4884	5688
deriche.c	14938	22465
floyd-warshall.c	16627	17054
nussinov.c	8506	9807
adi.c	7968	8615
fdtd-2d.c	13527	15870
heat-3d.c	14331	15655
jacobi-1d.c	5874	6283
jacobi-2d.c	7883	8562
seidel-2d.c	8700	9811