COMPILERS 2 - CS3423

MINI-ASSIGNMENT 2
SAI HARSHA KOTTAPALLI
CS17BTECH11036

Polly

About

Polly is a loop optimizer for LLVM. It generates a mathematical model by analyzing the given program for by finding loops which might be optimizable.
This mathematical model describes the memory accesses and computations, which is then used for applying optimization(fusion, fission, loop reversal , etc) to get a new llvm IR.

Architecture

Polly role in LLVM pass pipeline can be explained with these three stages:

1. Canonicalization
   This stage focuses on simplifying and better analysis. Passes such as -mem2reg, -instcombine, -cfgsimplify, or early loop unrolling come under this stage.
2. Inliner cycle
   Comprises of Scalar Simplification passes, Simple Loop Optimizations, and the Inliner itself whose main objective is to ensure semantic information and further simplification of code.
3. Target Specification
   Based on available target specific features we exploit to obtain a better IR representation. Example - Vectorization

   So, Polly can be used before (1) - early optimizer or before (3) .

This might be because inliner might miss some optimizations when polly enhances the code according to it.

Results

benchmark - [matmul.c]

Optimization	Time (in seconds)
None	29.735
O3 optimization	25.809
polly	0.492

SCOP

These pragmas are a way to let the compiler know which where polly can analyze and try to optimize code. Based on these scops, *-polly-scops -analyze* gives us its findings from analysis that is MustWriteAccess, ReadAccess, schedule, etc.
The findings are self-explanatory, MustWriteAccess are necessary writes around which polly opt is done.

Dependencies

Now based on different scops in the program, dependencies with respect to them are shown for each region.
RAW, WAR, WAW were shown when analyzing the dependencies.

Dependencies

Now based on different scops in the program, dependencies with respect to them are show

Analysis

1. Only scop regions are optimized
2. Tiling was used for matrix multiplication loop to split the iterations into chunks(with the help of getelementptr)
3. Declarations were vectorized
4. Size of the generated IR file is much larger
5. SCEV has been used for getting the loop bounds and steps.
6. More metadata related to polly was generated (maybe for debugging/ further opt usage)