Mini Assignment 2 : Compilers II

OM SITAPARA : CS16BTECH11036

October 2018

Contents

1 Understating Polyhedral Compilation and Polly

Polyhedral Compilation : Polyhedral Compilation uses polyhedral representations because hey can be manipulated or optimized with algorithms whose complexity depends on their structure and not on the number of elements they represent. Furthermore, generic and compact solutions can be designed that depend on program parameters (e.g., loop bounds, tile sizes, array bounds).[3]

Polly : Polly is a loop optimizer for LLVM. Starting from LLVM-IR it detects and extracts interesting loop kernels. For each kernel a mathematical model is derived which precisely describes the individual computations and memory accesses in the kernels. Within Polly a variety of analysis and code transformations are performed on this mathematical model. After all optimization's have been derived and applied, optimized LLVM-IR is regenerated and inserted into the LLVM-IR module.[2]

1.1 Diffrent stages of LLVM-Optimization

1.1.1 Canonicalization

This pass tries to achieve maximum elimination of raw IR using scalar optimization's. This consists of passes like -mem2reg, -instcombine, -cfgsimplify, or early loop unrolling.

1.1.2 Inliner-Cycle

This phase consists of three parts : Scalar Simplification, Simple Loop Optimizations and Inliner. The primary goal of this passes is canonicalization without the loss of semantics. In the Inliner pass the LLVM inliner tries to inline the functions and canonicalize them.

1.1.3 Target Specialization

This is the last stage where the optimization's are done keeping the respective hardware architecture in mind. The main transformation done by this is vectorization along with architecture specific loop unrolling.

1.2 Scheduling Polly

Polly can be scheduled at any of this two positions using the flag **-polly-position** with values as **early** or **before-vectorizer**. The default is early.

Between Canonicalization and Inliner-Cycle : Here the generated LLVM-IR is similar to raw IR so its better understandable.

Between Inliner-Cycle and Target Specialization : Here LLVM has already done a lot of optimizations the effect of polly is less.

2 Optimizations Using Polly

- constant propagation through arrays
- remove dead loop iterations
- optimize loops for cache locality
- optimize arrays
- apply advanced automatic parallelization
- drive vectorization
- software pipeline

[1]

3 Polly Performance

The comparison of matrix multiplication code in following conditions:

- **No Optimization** : 70 sec.
- **With O3** : 35 sec.
- **With O3 + polly** : 0.8 secs.

4 Analysis of Polly

- The first thing I observed if that if suppose there is a loop which does some computation but there is not use of that computation in later code than polly simply discards that. Eg, Suppose there is a loop from 1 to 100 where there is printf statement till 50 and after that there is no use of further computation than it creates a loop for 1 to 50.
- If there are two loops in sequential than it merges both the loops using phi nodes and looper goes to the max of both and computes for them respectively.
- The length of the IR increases and the polly enters basic blocks as vector.body etc

5 Understanding SCoP and Dependences

5.1 SCoP

SCoP : Static Control Part
A Scop is the polyhedral representation of a control flow region detected by the Scop detection. It is generated by translating the LLVM-IR and abstracting its effects.[4]A SCoP is defined as a maximal set of consecutive statements, where loop bounds and conditionals are affine functions of the surrounding loop iterators and the parameters (constants whose values are unknown at compilation time).[5]

5.2 Dependences

Suppose in a program the statement X is dependent on the result of statement Y than we say that X is dependent on Y. There are mainly two types of dependence :

- **Data Dependence** : True Dependence, Anti Dependence, Output Dependence
- **Control Dependence**

[6]

References

[1] *Optimizations by polly*. URL: https://github.com/llvm-mirror/polly.

[2] *Polly Info*. URL: http://polly.llvm.org/docs/Architecture.html.

[3] *Polyhedral Compilation*. URL: http://polyhedral.info/.

[4] *SCoP*. URL: https://polly.llvm.org/doxygen/classpolly_1_1Scop.html#details.

[5] *SCoP*. URL: http://web.cs.ucla.edu/~pouchet/doc/cc-article.10.pdf.

[6] Wikipedia. *Dependence*. URL: https://en.wikipedia.org/wiki/Loop_dependence_analysis.