Introduction to Polly

CS3423/CS6240: Mini Assignment 2

Due 12th Monday, November 2018 11:59 PM

Note : This is an individual Assignment

This mini assignment introduces you to the **polly** tool in llvm. This assignment is meant to show the kind of advanced optimizations modern compilers can do.

The about page of polly explains it as

"Polly is a high-level loop and data-locality optimizer and optimization infrastructure for LLVM. It uses an abstract mathematical representation based on integer polyhedra to analyze and optimize the memory access pattern of a program."

In this Assignment you are expected to try some simple transformations with polly and learn the kind of optimizations that are possible and the limitations of polly.

Tasks to do:

- First thing to do is to install LLVM with polly. You can follow the below steps to install LLVM with polly
    - You can use the attached **polly.sh** script to install llvm with polly (this script builds only clang and opt for llvm, and so it builds faster).
    - While installation is happening, you can read about the architecture of polly.
    - Try reading about polyhedral compilation in general. You are not expected to understand polyhedral compilation completely. You can just try to get an intuitive idea of which part of the program is being optimized and what kind of optimizations are possible.
- After installation, as a first step with polly, you can try optimizing some basic programs with polly.
    - You can take a basic matrix multiplication program as given in this link. The commands to use polly with clang are given in this page.
    - Try optimizing with **-O3** with and without polly on clang and report on any improvements.

- Next step is to try running the individual pieces of polly.
    - You can follow the this link to try to look at the individual pieces of information that polly provides.
        - The first step in that link asks you to generate LLVM IR. Use the following command instead of the one given in the page.
          clang -S -emit-llvm -Xclang -disable-O0-optnone matmul.c -o matmul.s
        - This change is because an attribute called **optnone** was introduced in clang 5.0 which prevents optimization passes from being run on functions. This flag **(-Xclang -disable-O0-optnone)** prevents the **optnone** attribute from being added to the functions when generating LLVM IR.
    - Try understanding what the words **SCoP** or **dependences** mean (This step is not mandatory, but you can get extra points if you do this) and explain intuitively.
    - Run polly transformations like tiling and parallelization on a simple code of your choice and analyze the transformed IR. Report on your findings.
        - You can see the all the polly options available by running the following command: **opt -help-hidden | grep polly**

What to submit:

You have to submit a report on your study of polly's features. Specifically, you are expected to submit the following:

- Your understanding of the architecture of polly and its optimization capabilities.
- The amount of improvement in running time that you got on any code when optimized using polly with clang
- Your intuitive understanding of **SCoP**s and **dependences**. (Again, this step is not mandatory, but you get extra points).
- Your analysis of a transformed code using polly. Here you can explain what kind of optimization happened and how it improves the performance (if any). You don't have to explain how the transformation is happening (this is beyond the scope of this course).
    - You can explain the transformed code using an example if you like.

Create a report in latex or lyx of your findings and submit it as **<your-roll-no>.pdf**