E0255 (2025)

ASSIGNMENT 2

Deadline: Apr 18, 2025 5pm.

This assignment is available at

https://www.csa.iisc.ac.in/~udayb/e0255/2025/asst-2.txt

Instead of storing a copy of this file, please check at this URL in case there

are updates or clarifications.

Test cases: https://www.csa.iisc.ac.in/~udayb/e0255/2025/asst-2/interchange.mlir

DESCRIPTION

Develop a loop interchange pass in MLIR driven by an analytical cost model that

optimizes for locality (spatial, temporal - both self and group) and parallelism

for multicores (so as to minimize the frequence of synchronization). The pass

should be implemented on the Affine dialect in MLIR, which ensures a certain

structure to the loads, stores, and loop bounds that facilitate dependence

analysis and easy transformation. The pass should be able to handle both

perfectly nested loop nests and imperfectly nested ones. The pass does not need

to perform interchange in the presence of if conditionals and non-rectangular

loop nests, but should still handle them by detecting those and bailing out,

i.e., it should not crash or abort/assert on any input. You are free to make use

of any existing infrastructure in MLIR to build this pass.

The pass should be triggered by the command line flag -affine-loop-interchange.

Test cases specifying the input and output IR, which could be automatically

tested via FileCheck will be provided. Your pass shouldn't require any

additional command line flags for validation on the provided test cases. (You

are free to have additional flags for your own debugging purposes.)

You should use the git version of LLVM project as of commit id

b9d27ac252265839354fffeacaa8f39377ed7424 (Mar 17, 2025) as the base version.

Please clone from

https://github.com/llvm/llvm-project at that commit

id.

Simple example:

Input IR snippet:

affine.for %i = 0 to %N {

affine.for %j = 0 to %M {

%v = affine.load %A[%j, %i] : memref<?x?xf32>

%vd = arith.addf %v, %v : f32

affine.store %vd, %A[%j, %i] : memref<?x?xf32>

}

}

Expected output IR snippet:

affine.for %i = 0 to %M {

affine.for %j = 0 to %N {

%v = affine.load %A[%i, %j] : memref<?x?xf32>

%vd = arith.addf %v, %v : f32

affine.store %vd, %A[%i, %j] : memref<?x?xf32>

}

}

WHAT TO SUBMIT

Please submit your assignment as a single git commit patch file that should

apply on the base version along with a README.txt. Once you have created a

single commit to contain your implementation, such a patch can be generated

using:

$ git format-patch HEAD~

The README.txt can include any short notes or observations you

made or to document any limitations of your implementation that make it fall

short of the assignment requirements. Please make sure you use the same coding

style as the rest of MLIR. Automatic formatting can be performed using

clang-format (or via git clang-format HEAD~ once committed).

Both files should be emailed to udayb@iisc.ac.in by the deadline mentioned on

the top. We recommend using the CSA clserv server.

REFERENCES

MLIR documentation: https://mlir.llvm.org