Shibo Xing

Dr. Xulong Tang

April 24, 2020

CS2210

Review of A Loop Transformation Theory And an Algorithm to Maximize Parallelism

This paper addresses a common concern of the compiler loop optimization which is that in what order and what kinds of kind of loop transformation should be performed in order to achieve a certain optimization goal, whether data locality or parallelism. It proposes to make various loop transformations into unimodular transformations and an algorithm that maximizes loop parallelism.

It first identifies the dilemma that compiler faces, which is whether to generate loop transformation type a-priori or ‘generate and test’, the former might be ineffective and the latter might be too expensive too execute. Then the approach proposed is to use distance and direction vectors to compute loop transformation matrix. Hence, even compound loop transformations could become a unimodular linear algebraic puzzle. They aim to use this transformation to maximize the degree of parallelization and data locality.

Then the paper goes on discussing the elementary loop transformations: interchange, reversal and permutation, and how to write transformation matrices to represent them. Along with the transformation, the legality of transformation is also articulated, which is examined by the value of the dependence vector. The transformation is only legal when the dependence vector has a positive lexicographical value. Legality determines the feasibility of a transformation. For example, permutation could only be performed only if any arbitrary permutation loop generates a lexicographically positive dependence vector.

For many of the transformations the authors discuss, they present the analysis of both the fine-grained and coarse parallelization effects, in which they talk about placing the DOALL loops in a ideal position to prevent parallelizing loops with small iteration counts.

The first half of the paper consists explanation of the elementary loop transformation and the elementary rubrics to compute and test those transformations. Since we’ve covered those content in the lectures, I found those rather straightforward and quite make sense. But later when the paper gets into the more advance transformation analysis and even the proof of the transformation, I began to lose track. This paper makes me realize the depth of the compiler optimization study, compared to the front-end procedure we learned in the first half of the class.