Optimizing compilers: Handling Control Flow

Vendula Poncová (xponco00)
Martin Šifra (xsifra00)

November 2, 2014

One of the challenges of optimizing compilers for high-performance architectures can be vectorizing a program with control flow, specifically with conditional branches in loops. Control flow creates a new type of constraints on program transformations: a control dependence. There are two strategies for dealing with control. While *if-conversion* simply converts the control dependences to data dependences, *control dependence* includes control dependence edges in the dependence graph. That complicates the code generation process.

If-conversion is a composition of two methods: *branch relocation* and *branch removal*. Branch relocation can generate very complex guarding conditions, therefore it is suitable to simplify them. However, Boolean simplification is NP-complete, so we need to redefine the problem and find another way of computation. Once the branch relocation is finished, we can transform the conditional statements to vector statements. Considering the fact, that sometimes vectorization is not possible, we can apply a transformation called *if-reconstruction*, to prevent the performance degradation.

By establishing a control dependence as a type of constraint, we can prevent the undesirable effects of if-conversions. Control dependences between blocks of statements can be represented by a control dependence graph, that enables to generate the optimized code.