Authors:

Michal Ryšavý, xrysav17

Jozef Senko, xsenko01

Topic 26

Optimizing Compilers - Chapter 6 (Creating Coarse-Grained Parallelism)

Abstract:

In this presentation, I will give an introduction to transformations for symmetric multiprocessor machines.

At the begining we will describe normal compilers transforms which are targeted to vector and superscalar architectures. Problem of this achitecture is that a basic synchronization element is the barrier, which forces all processes to reach a predetermined point before execution can go on. However disadvantage of this procedure is that it can slowdown whole program. But coarse-grained parallelism tends to be one of granularity.

After this short part of presentation we are going to talk in detail about problems of this architecture and its solutions. For example we are going to show method how put variables to seperable namespaces and this solution accelerate whole program.

The methods of coarse-grained parallelism uses some methods which put every exposed variable to seperable namespace. When variables are in seperable namespace, program can slightly use parallelism.
For example a scalar variable in some loop can be privatizabled if every path from the loop entry to a use of that variable inside the loop passes through a definition of our variable.

Then we show another important method which accelerate loops, so called loop distribution. This optimalization eliminates carried dependencies in loops. Consequence of this solution is that we have to add some extra barriers (conditions) to a code. As you will see many carried dependencies are due to array alignment issues. And we are going to talk about methods which removes all dependencies (by changing references) and allow us to use parallelism.

If there will be still some time in our presentation, we would like to talk about other examples of creating coarse-grained parallelism, like loop fusion, loop interchange, etc.

After explaining this usefull methods we would like to say our opinions about accelerate by this optimalizations.

And at the end of this presentation we would like to leave space for questions of audience.