Internship subject (Master 2)

Hierarchical Parallelization

Advisor: Christophe Alias, christophe.alias@ens-lyon.fr

Place: Laboratoire de l'Informatique du Parallélisme (LIP)
École Normale Supérieure de Lyon

Context

Technological trends in circuit design force the duplication of computation units to obtain more performances. Splitting a program into parallel units and setup the synchronisations is bug-prone and expensive. Hence, the trend to rely on a *parallelizing compiler* to automate this task. Numerous problems must be addresses by such a compiler. Where is the parallelism in the program? Which parts must be parallelized and how? How to allocate the hardware resources?

The *polyhedral model* addresses these issues in the same unifying framework. Under certain conditions, the iterations of `for` loops may be summarized by polyhedra. Then, a geometric reasonning allows to build compiler analysis (e.g. dependence, scheduling) thanks to linear programming and geometric operations (union, intersection, projection). But the precision comes with a cost: the complexity of polyhedral analysis limits the size of compilable programs to a few tens of lines!

Objective

In this internship, we propose to study how to scale the polyhedral model. We will leverage a *divide-and-conquer* approach: the program is divided into subprograms, each subprogram is analyzed separately, then we "gather the pieces". We will focus on *loop tiling*, a polyhedral loop transformation which divides the iteration domain of nested `for` loops into atomic blocks.

1. Study the possible loop tilings on a composition of polyhedral programs (e.g. several matrix multiplication). Figure out how *local* loop tilings may be composed to form a *global* loop tiling. Specify the properties of a "good" composition.

2. Propose an algorithm for hierarchical loop tiling, which fulfills the properties formalized in step 1.

3. Implement and test your algorithm. The approach will be validated experimentally on the benchmarks of the polyhedral community [1].

Requirements. Notions in compilation and parallelism.

References

[1] Louis-Noël Pouchet. Polybench: The polyhedral benchmark suite. *URL: http://www. cs. ucla. edu/˜ pouchet/software/polybench/[cited July,]*, 2012.