Complex data structures scheduling

Laure Gonnord – Maître de conférences Université Lyon1/LIP
et Lionel Morel, Ingénieur de Recherche, CEA, et Tomofumi Yuki, CR Inria, Rennes

M2 2018

General context

The rise of embedded systems and high performance computers generated new problems in high-level code optimization, especially those around loops : optimization of embedded applications and program transformations for high-level synthesis (HLS). Moreover, everything involving data storage is of prime importance as it impacts power consumption, performance, and for hardware, chip area. Thus, there is an increasing need for better scheduling techniques for all kinds of programs, especially those manipulating a huge amount of data.

So far, the polyhedral model [3], a framework introduced in the late eighties, has been successfully applied to a range of these compilation problems, such as (semi-)automatic parallelization and code generation [2] or data movement optimization [1]. However, this powerful model hits its limit as soon as we are faced with irregular programs (general `while` loops, unpredictable conditions). As a consequence, these powerful techniques have, for the moment, only seen their use in limited (but still important) niches, because of the intrinsic restrictions.

The long term objective is to give a general way to reason and manipulate programs with general control flow and complex data structures.

Objectives of the internship

Benchmarking extraction

The first objective of the internship is to extract from actual programs some “kernels” for which computing a scheduling is challenging. Apart from non-regular programs/kernels (general `while` loops), we want to extract and characterize middle-sized kernels with code operating on :

— Trees (binary trees, general trees) : tree traversals, with or without cuts, like in *deep learning* or symbolic computing programs ;
— Linked lists, maps. Maps are definitely under interest since they are an essential step toward *sparse matrices*, which are a common object in scientific computing (numerical simulations).

In fact, as shown previously, for these kinds of data structures (or at least the former ones), it seems quite easy to properly define a notion of *dependence*, in a quite compact way. However combining data structures and control is challenging since the dependence graphs may quickly become infinite thus hard to formally manipulate.

The objective of this first task is to construct a benchmark suite of interesting programs to schedule/optimize. We might also modify benchmarks from the polyhedral literature to operate on lists and/or trees instead of arrays. This task will be performed from a basis already done during a previous internship.

A new language for code optimisation

Once we have a collection of benchmarks, the next step is to define a common abstraction of the kernels and important properties (such as dependences) to reason about within the compiler. The original source program contains a lot of implementation details and language specific “noise” that can complicate static analysis. This abstraction can take a form of a language of its own, and hence we call them a mini-language. We envison this language to incorporate high-level operators on complex data structures inspired by functional languages (e.g., inductively defined data structures, “à la ML”).

LIP, Lyon et CEA Tech - LIST, Grenoble et Inria Rennes
Contact E-mail : Laure.Gonnord@ens-lyon.fr et Lionel.Morel@cea.fr et Tomofumi.yuki@inria.fr

This activity will consist of defining a clean “abstract” semantics for our language, in order to avoid specific implementation matters, but also permit the expression of the intrinsic parallelism of both control and data structures. The challenging operations such as, dependence analysis, instruction scheduling, and code generation, will have to be performed on this mini-language. We expect the candidate to have enough time to at least express the activity of scheduling with respect to the semantics of the new language. Formally, we aim to express the notion of dependence, like the “happens-before” relation of the polyhedral model. We expect the intern to work on the following tasks :

- Define semantics for the mini-language.
- Develop a notion of dependence, e.g., the happens-before relation defined for various other languages.
- Develop a notion of schedule, i.e., a compact characterization of when dynamic operations in the programs written in the mini-language should be executed.
- Code generation given a program and its schedule.
- Formulating the legal space of schedules given the program and its dependences.

Note that we do not expect the intern to complete all the items listed above during this internship. We are likely to work on the items in the order listed above, which also matches the expected difficulty, and expect to have initial results on expressing the schedules.

The candidate should be familiar with Abstract Interpretation and other formal methods such as SMT-solvers and/or rewriting.

Remark : This internship proposal is the first step of a more general research project financed by the National Agency of Research (ANR). Successful intern is likely to have opportunities to continue as a PhD student for the same project.

Références

[1] Alain Darte and Alexandre Isoard. Exact and approximated data-reuse optimizations for tiling with parametric sizes. In *Proceedings of the 24th International Conference on Compiler Construction*, CC ’15, pages 151–170, April 2015.

[2] Paul Feautrier. Some efficient solutions to the affine scheduling problem, I, one-dimensional time. *International Journal of Parallel Programming*, 21(5) :313–348, October 1992.

[3] Paul Feautrier and Christian Lengauer. The polyhedron model. In David Padua, editor, *Encyclopedia of Parallel Programming*. Springer, 2011.