Complex data structures scheduling for optimizing compilers

Laure Gonnord – Maître de conférences HDR Université Lyon1/LIP
and Lionel Morel, Ingénieur de Recherche, CEA

Funded Phd Proposal, beginning: September 2018

General context

Having been limited to high-performance computers for a long time, parallel processors have gained wide-spread deployment into the consumer market with the number of cores steadily increasing both in personal computers and in smartphones. The problem of high-level code optimization is thus at the heart of program performance not only for scientific applications but also for mainstream, user-centric applications that tend to aggregate and manipulate large amounts of data. Algorithms manipulating this data are designed using various algorithmic patterns that hopefully express parallelism in a way that can fully exploit the parallelism available on the hardware.

At the heart of HPC and scientific computing applications, a lot of regular algorithms (e.g., `for` loops) manipulating regular data structures (e.g., matrices) have been shown to be optimizable using various compilation techniques. In this context, the polyhedral model [4], a framework introduced in the late eighties, has been successfully applied to a range of the arising compilation challenges, such as (semi-)automatic parallelization and code generation [3] or data movement optimization [2].

However, applications today are not limited to regular parallelism: many applications, both in scientific computing and in user-centric contexts are irregular programs (e.g., general `while` loops) that manipulate general data structures (lists, trees, maps). Parallelizing such irregular programs is an open research problem. The long term objective of the ANR project CODAS (`http://codas.ens-lyon.fr`) is to address this problem by giving a general framework for reasoning and manipulating programs with general control flow and complex data structures.

Objectives of the Phd

The objective of the thesis is to:

- Demonstrate that the state-of-the-art methods and tools miss some optimizations opportunities when dealing with irregular control flow (`while` loops) and data structures other than arrays (lists, trees, maps);
- Construct and solve an expressive intermediate representation that captures all data as well as control dependencies, under reasonable assumptions;
- Prototype code generation procedure that produces optimized code from this intermediate representation.

A work-in-progress master internship in Spring 2018 will construct the following bases for the Phd:

- It will provide a benchmark of interest for the Phd, namely a set of programs to schedule/optimize, 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 of interest since they are an essential step toward *sparse matrices*, which are a common object in scientific computing (numerical simulations).
- It will propose a mini-language of interest and a way to represent control and data dependencies in a compact way. Formally, we aim to express the notion of dependency, like the “happens-before” relation of the polyhedral model.

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

The Phd student will pursue these works in the following directions:

- Consolidate the formal bases: language semantics, semantics of dependency, extensions, algorithm to solve the dependencies. He/She might take inspiration from our previous work [1].
- Formulate the legal space of schedules given the program and its dependencies.
- Propose a code generation algorithm given a program and its schedule.

All the algorithms and approaches will be implemented and heavily tested to demonstrate both pertinence and scalability of the approach.

Keywords: Compilation, formal methods, parallelism, data-structures, code generation.

Expected skills

- The candidate should be familiar with Abstract Interpretation and other formal methods such as SMT-solvers and/or rewriting.
- The candidate will develop non-trivial software, thus skills in programming languages like Ocaml and/or C++ as well as proficiency in Unix-like environments are required.
- Knowledge in compiler construction is a plus.

Phd environment

The PhD student will be co-supervised by:

- Laure Gonnord, Associate Professor, HDR. Dr Gonnord is specialized in abstract interpretation and formal methods applied to software design and compilation, with a special focus on scalable techniques.
- Lionel Morel, Research Engineer at CEA-List, Grenoble, PhD. Dr Morel has worked for 15 years on programming models and runtimes for dataflow languages and compilation of regular programming structures. Recently he is involved in designing code-generation approaches for the security of embedded applications.

The PhD project will take place fully at the LIP laboratory, in Lyon, France and will benefit from the environment of the newly created CASH[1] team. The team works on novel approaches to extract and express parallelism from imperative programs to intermediate representations. This Phd is clearly one major milestone for CASH.

References

[1] Christophe Alias, Carsten Fuhs, and Laure Gonnord. Estimation of Parallel Complexity with Rewriting Techniques. In *Proceedings of the 15th International Workshop on Termination*, WST '16, September 2016.

[2] 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.

[3] 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.

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

[1] `http://www.ens-lyon.fr/LIP/CASH/`