Polyhedral compilation for a data-flow model of execution

Mots clés :

- Directeur de thèse : albert COHEN
- Co-encadrant(s) :
- Unité de recherche : Laboratoire inconnu!
- École doctorale : École Doctorale Informatique, Télécommunications, Électronique de Paris
- Domaine scientifique principal: Divers

Résumé du projet de recherche (Langue 1)

Future embedded and high-performance microprocessors will include, not 2 or 4, but hundreds of computing units, organized as heterogeneous clusters of general-purpose cores and more specialized accelerators. State-of-the-art parallelizing compilers are rarely capable of generating scalable and/or efficient code, for essential reasons deeply rooted into the semantics of programming languages, the undecidability of static analyses and the complexity of modern computer architectures. As a result, most programmers rely on low-productivity programming models that have not evolved significantly in two decades. These models suffer from well known flaws (non-deterministic, not compositional, not modular), and induce massive productivity and robustness losses compared to sequential programming languages. The dynamic data-flow model of execution is a parallel operational semantics for Kahn process networks. We are working on extending this model with more flexible scheduling constraints, such as transactional execution (for in-place operations, reductions, or more complex data structure operations). In this extended model, static guarantees must be obtained about determinism, boundedness of communication buffers and bounded time reactivity. The polyhedral model of compilation is an ideal tool to extract such properties automatically from a high-level description of the denotational invariants of the program. It is also a well known framework to design advanced code generation algorithms. Both assests make it very suitable to adapt the concurrency of the source program to the target architecture, leveraging a rich algebra of static data-flow properties. This thesis will start from the mapping of a polyhedral representation to a data-flow model of execution, then progressively extend the code generator and optimization heuristics to support other forms of scheduling constraints (transactions). The polyhedral representation itself will capture static properties from a data-flow extension of the source language, in the form of an extended set of OpenMP pragmas. Experiments and validation of the proposed algorithms will take place in the GNU Compiler Collection (GCC). The student will participate to the development of the Graphite optimization framework of this compiler.