This PhD thesis project aims the combination of two research streams of our team, namely to integrate models for coarse grained parallelism with (fine-grained) polyhedral models of compilation.

For coarse grained parallelism, with our work on ORWL (ordered read-write locks) [1], we have proposed a framework dedicated to a large set of applications characterized by iterative computations. ORWL guarantees important properties related to equity of all compute tasks and liveness. Currently, we are applying this framework to iterative computations that are decomposable into tasks that are guaranteed to have the same memory access pattern for all iterations. This execution scheme is typical in numerical simulations. Memory access classification is used to detect the data access order of the tasks and to schedule them according an autonomous access pattern.

Currently, the application code is instrumented either manually by means of pragmas (C), or automatically by overloading the access operator [] (C++). With these, the programmer identifies a suitable iteration loop, the decomposable tasks and annotates the memory accesses as being read-only or exclusive. Our framework then instruments the memory access pattern during the first three iterations, sets up a system of FIFOs that regulates the data accesses and then runs the mostly unmodified code for the remaining iterations. The polyhedral model is a well-known mathematical framework that describes iterations and memory accesses by means of integral points in polyhedra. State of the art compilers are able to use this framework to automatically reorganize nested iteration loops. Our Apollo platform has been developed to extend the scope of this model from static compilation to runtime speculative parallelization [2, 3]. It allows to instrument and predict memory access patterns dynamically, to choose optimized binary code variants, to execute code speculatively and to roll-back executions that violated the prediction if necessary.

### Assignement

This PhD thesis project aims the combination of two research streams of our team, namely to integrate models for coarse grained parallelism with (fine-grained) polyhedral models of compilation.

For coarse grained parallelism, with our work on ORWL (ordered read-write locks) [1], we have proposed a framework dedicated to a large set of applications characterized by iterative computations. ORWL guarantees important properties related to equity of all compute tasks and liveness. Currently, we are applying this framework to iterative computations that are decomposable into tasks that are guaranteed to have the same memory access pattern for all iterations. This execution scheme is typical in numerical simulations. Memory access classification is used to detect the data access order of the tasks and to schedule them according an autonomous access pattern.

Currently, the application code is instrumented either manually by means of pragmas (C), or automatically by overloading the access operator [] (C++). With these, the programmer identifies a suitable iteration loop, the decomposable tasks and annotates the memory accesses as being read-only or exclusive. Our framework then instruments the memory access pattern during the first three iterations, sets up a system of FIFOs that regulates the data accesses and then runs the mostly unmodified code for the remaining iterations. The polyhedral model is a well-known mathematical framework that describes iterations and memory accesses by means of integral points in polyhedra. State of the art compilers are able to use this framework to automatically reorganize nested iteration loops. Our Apollo platform has been developed to extend the scope of this model from static compilation to runtime speculative parallelization [2, 3]. It allows to instrument and predict memory access patterns dynamically, to choose optimized binary code variants, to execute code speculatively and to roll-back executions that violated the prediction if necessary.

### Main activities

The goal of this thesis is to formalize our approach of memory access classification in terms of the polyhedral model such that the classification phase can be automated and such that the execution...
phase can be verified. For this, it will be crucial to capture regular access patterns of the decomposed tasks as much as possible and to describe them as potentially intersecting polyhedra of the data domain. In addition, it will also be necessary to describe and manage irregular accesses that add to the more regular ones. This should lead to more competitive execution times for the classification phase, to a verification of the access pattern during execution of the decomposed tasks that can be performed in parallel, and to a rollback strategy that ensures coherent execution if accesses outside the predicted domain occur.

Skills

- good knowledge in parallel computing and algorithms, compilation, polyhedral models
- advanced level of programming, preferable in C, and some knowledge of compiler tools

Benefits package

- Subsidised catering service
- Partially-reimbursed public transport

Remuneration


Monthly salary after taxes: around 1596,05€ for 1st and 2nd year. 1678,99€ for 3rd year. (medical insurance included.)