Activity Report 2017

Team CAMUS

Compilation pour les Architectures MUlti-coeurS

RESEARCH CENTER
Nancy - Grand Est

THEME
Architecture, Languages and Compilation

Table of contents

Team CAMUS

Creation of the Team: 2009 July 01

Keywords:

Computer Science and Digital Science:

A1.1.1. - Multicore, Manycore
A1.1.4. - High performance computing
A2.1.1. - Semantics of programming languages
A2.1.6. - Concurrent programming
A2.2.1. - Static analysis
A2.2.3. - Run-time systems
A2.2.4. - Parallel architectures
A2.2.5. - GPGPU, FPGA, etc.
A2.2.6. - Adaptive compilation

Other Research Topics and Application Domains:

B4.5.1. - Green computing
B6.1.1. - Software engineering
B6.6. - Embedded systems

1. Personnel

Research Scientists

Arthur Charguéraud [Inria, Researcher]
Jens Gustedt [Inria, Senior Researcher, HDR]

Faculty Members

Philippe Clauss [Team leader, Univ de Strasbourg, Professor, HDR]
Cédric Bastoul [Univ de Strasbourg, Professor, HDR]
Alain Ketterlin [Univ de Strasbourg, Associate Professor]
Vincent Loechner [Univ de Strasbourg, Associate Professor]
Nicolas Magaud [Univ de Strasbourg, Associate Professor]
Julien Narboux [Univ de Strasbourg, Associate Professor]
Éric Violard [Univ de Strasbourg, Associate Professor, HDR]

Post-Doctoral Fellows

Manuel Selva [Inria]
Julien Pagès [Univ. Strasbourg, from Sept 2016, until Aug 2017]

PhD Students

Yann Barsamian [Univ de Strasbourg]
Paul Godard [Caldera]
Salwa Kobeissi [Inria, from Sep 2017]
Harenome Ranaivoarivony-Razanajato [Univ de Strasbourg]
Mariem Saied [Univ de Strasbourg, until Aug 2017]
Daniel Salas [INSERM]
Maxime Schmitt [Univ de Strasbourg]

Technical staff

Maxime Mogé [Inria]

Intern

Stan Wilhelm [Inria, until Jun 2017]

Administrative Assistants

Véronique Constant [Inria]

Ouiza Herbi [Inria]

2. Overall Objectives

2.1. Overall Objectives

The CAMUS team is focusing on developing, adapting and extending automatic parallelizing and optimizing techniques, as well as proof and certification methods, for the efficient use of current and future multicore processors.

The team's research activities are organized into five main issues that are closely related to reach the following objectives: performance, correction and productivity. These issues are: static parallelization and optimization of programs (where all statically detected parallelisms are expressed as well as all "hypothetical" parallelisms which would be eventually taken advantage of at runtime), profiling and execution behavior modeling (where expressive representation models of the program execution behavior will be used as engines for dynamic parallelizing processes), dynamic parallelization and optimization of programs (such transformation processes running inside a virtual machine), and finally program transformations proof (where the correction of many static and dynamic program transformations has to be ensured).

3. Research Program

3.1. Research Directions

The various objectives we are expecting to reach are directly related to the search of adequacy between the sofware and the new multicore processors evolution. They also correspond to the main research directions suggested by Hall, Padua and Pingali in [32]. Performance, correction and productivity must be the users' perceived effects. They will be the consequences of research works dealing with the following issues:

- Issue 1: Static Parallelization and Optimization
- Issue 2: Profiling and Execution Behavior Modeling
- Issue 3: Dynamic Program Parallelization and Optimization, Virtual Machine
- Issue 4: Proof of Program Transformations for Multicores

Efficient and correct applications development for multicore processors needs stepping in every application development phase, from the initial conception to the final run.

Upstream, all potential parallelism of the application has to be exhibited. Here static analysis and transformation approaches (issue 1) must be processed, resulting in a *multi-parallel* intermediate code advising the running virtual machine about all the parallelism that can be taken advantage of. However the compiler does not have much knowledge about the execution environment. It obviously knows the instruction set, it can be aware of the number of available cores, but it does not know the effective available resources at any time during the execution (memory, number of free cores, etc.).

That is the reason why a "virtual machine" mechanism will have to adapt the application to the resources (issue 3). Moreover the compiler will be able to take advantage only of a part of the parallelism induced by the application. Indeed some program information (variables values, accessed memory adresses, etc.) being available only at runtime, another part of the available parallelism will have to be generated on-the-fly during the execution, here also, thanks to a dynamic mechanism.

This on-the-fly parallelism extraction will be performed using speculative behavior models (issue 2), such models allowing to generate speculative parallel code (issue 3). Between our behavior modeling objectives, we can add the behavior monitoring, or profiling, of a program version. Indeed current and future architectures complexity avoids assuming an optimal behavior regarding a given program version. A monitoring process will allow to select on-the-fly the best parallelization.

These different parallelizing steps are schematized on figure 1.

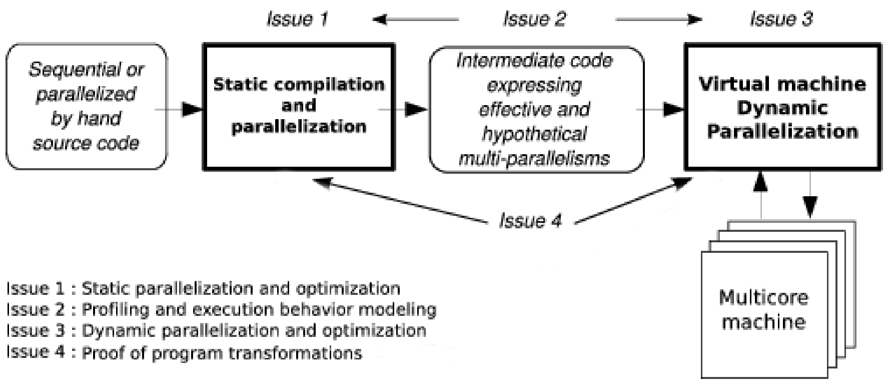
Issue 1
Issue 2
Issue 3
Sequential or parallelized by hand source code
Static compilation and parallelization
Intermediate code expressing effective and hypothetical multi-parallelisms
Virtual machine Dynamic Parallelization
Issue 4
Multicore machine
Issue 1 : Static parallelization and optimization
Issue 2 : Profiling and execution behavior modeling
Issue 3 : Dynamic parallelization and optimization
Issue 4 : Proof of program transformations

Figure 1. Automatic parallelizing steps for multicore architectures

Our project lies on the conception of a production chain for efficient execution of an application on a multicore architecture. Each link of this chain has to be formally verified in order to ensure correction as well as efficiency. More precisely, it has to be ensured that the compiler produces a correct intermediate code, and that the virtual machine actually performs the parallel execution semantically equivalent to the source code: every transformation applied to the application, either statically by the compiler or dynamically by the virtual machine, must preserve the initial semantics. They must be proved formally (issue 4).

In the following, those different issues are detailed while forming our global and long term vision of what has to be done.

3.2. Static Parallelization and Optimization

Participants: Vincent Loechner, Philippe Clauss, Éric Violard, Cédric Bastoul, Arthur Charguéraud.

Static optimizations, from source code at compile time, benefit from two decades of research in automatic parallelization: many works address the parallelization of loop nests accessing multi-dimensional arrays, and these works are now mature enough to generate efficient parallel code [30]. Low-level optimizations, in the assembly code generated by the compiler, have also been extensively dealt for single-core and require few adaptations to support multicore architectures. Concerning multicore specific parallelization, we propose to explore two research directions to take full advantage of these architectures: adapting parallelization to multicore architecture and expressing many potential parallelisms.

3.3. Profiling and Execution Behavior Modeling

Participants: Alain Ketterlin, Philippe Clauss, Manuel Selva.

The increasing complexity of programs and hardware architectures makes it ever harder to characterize beforehand a given program's run time behavior. The sophistication of current compilers and the variety of transformations they are able to apply cannot hide their intrinsic limitations. As new abstractions like transactional memories appear, the dynamic behavior of a program strongly conditions its observed performance. All these reasons explain why empirical studies of sequential and parallel program executions have been considered increasingly relevant. Such studies aim at characterizing various facets of one or several program runs, *e.g.*, memory behavior, execution phases, etc. In some cases, such studies characterize more the compiler than the program itself. These works are of tremendous importance to highlight all aspects that escape static analysis, even though their results may have a narrow scope, due to the possible incompleteness of their input data sets.

3.4. Dynamic Parallelization and Optimization, Virtual Machine

Participants: Manuel Selva, Juan Manuel Martinez Caamaño, Luis Esteban Campostrini, Artiom Baloian, Mariem Saied, Daniel Salas, Philippe Clauss, Jens Gustedt, Vincent Loechner, Alain Ketterlin.

This link in the programming chain has become essential with the advent of the new multicore architectures. Still being considered as secondary with mono-core architectures, dynamic analysis and optimization are now one of the keys for controling those new mechanisms complexity. From now on, performed instructions are not only dedicated to the application functionalities, but also to its control and its transformation, and so in its own interest. Behaving like a computer virus, such a process should rather be qualified as a "vitamin". It perfectly knows the current characteristics of the execution environment and owns some qualitative information thanks to a behavior modeling process (issue 2). It appends a significant part of optimizing ability compared to a static compiler, while observing live resources availability evolution.

3.5. Proof of Program Transformations for Multicores

Participants: Éric Violard, Alain Ketterlin, Julien Narboux, Nicolas Magaud, Arthur Charguéraud.

Our main objective consists in certifying the critical modules of our optimization tools (the compiler and the virtual machine). First we will prove the main loop transformation algorithms which constitute the core of our system.

The optimization process can be separated into two stages: the transformations consisting in optimizing the sequential code and in exhibiting parallelism, and those consisting in optimizing the parallel code itself. The first category of optimizations can be proved within a sequential semantics. For the other optimizations, we need to work within a concurrent semantics. We expect the first stage of optimizations to produce data-race free code. For the second stage of optimizations, we will first assume that the input code is data-race free. We will prove those transformations using Appel's concurrent separation logic [33]. Proving transformations involving program which are not data-race free will constitute a longer term research goal.

4. Application Domains

4.1. Application Domains

Performance being our main objective, our developments' target applications are characterized by intensive computation phases. Such applications are numerous in the domains of scientific computations, optimization, data mining and multimedia.

Applications involving intensive computations are necessarily high energy consumers. However this consumption can be significantly reduced thanks to optimization and parallelization. Although this issue is not our primary objective, we can expect some positive effects for the following reasons:

- Program parallelization tries to distribute the workload equally among the cores. Thus an equivalent performance, or even a better performance, to a sequential higher frequency execution on one single core, can be obtained.
- Memory and memory accesses are high energy consumers. Lowering the memory consumption, lowering the number of memory accesses and maximizing the number of accesses in the low levels of the memory hierarchy (registers, cache memories) have a positive consequence on execution speed, but also on energy consumption.

5. Highlights of the Year

5.1. Highlights of the Year

5.1.1. Awards

A team composed of four CAMUS members (Cédric Bastoul, Vincent Loechner, Harenome Ranaivoarivony-Razanajato and Maxime Schmitt) participated to the Google Hash Code contest. They were ranked 9 during the qualification round, over more than 26000 participants from Europe, Middle-East and Africa, and qualified for the final. They were 34th at the final hosted in the Google Paris office.

6. New Software and Platforms

6.1. APOLLO

Automatic speculative POLyhedral Loop Optimizer
KEYWORD: Automatic parallelization
FUNCTIONAL DESCRIPTION: APOLLO is dedicated to automatic, dynamic and speculative parallelization of loop nests that cannot be handled efficiently at compile-time. It is composed of a static part consisting of specific passes in the LLVM compiler suite, plus a modified Clang frontend, and a dynamic part consisting of a runtime system. It can apply on-the-fly any kind of polyhedral transformations, including tiling, and can handle nonlinear loops, as while-loops referencing memory through pointers and indirections.

- Participants: Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño, Manuel Selva and Philippe Clauss
- Contact: Philippe Clauss
- URL: http://apollo.gforge.inria.fr

6.2. Clan

A Polyhedral Representation Extraction Tool for C-Based High Level Languages
KEYWORD: Polyhedral compilation
FUNCTIONAL DESCRIPTION: Clan is a free software and library which translates some particular parts of high level programs written in C, C++ or Java into a polyhedral representation called OpenScop. This representation may be manipulated by other tools to, e.g., achieve complex analyses or program restructurations (for optimization, parallelization or any other kind of manipulation). It has been created to avoid tedious and error-prone input file writing for polyhedral tools (such as CLooG, LeTSeE, Candl etc.). Using Clan, the user has to deal with source codes based on C grammar only (as C, C++ or Java). Clan is notably the frontend of the two major high-level compilers Pluto and PoCC.

- Participants: Cédric Bastoul and Imèn Fassi
- Contact: Cédric Bastoul
- URL: http://icps.u-strasbg.fr/people/bastoul/public_html/development/clan/

6.3. Clay

Chunky Loop Alteration wizardrY

FUNCTIONAL DESCRIPTION: Clay is a free software and library devoted to semi-automatic optimization using the polyhedral model. It can input a high-level program or its polyhedral representation and transform it according to a transformation script. Classic loop transformations primitives are provided. Clay is able to check for the legality of the complete sequence of transformation and to suggest corrections to the user if the original semantics is not preserved.

- Participant: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://icps.u-strasbg.fr/people/bastoul/public_html/development/clay/

6.4. CLooG

Code Generator in the Polyhedral Model

FUNCTIONAL DESCRIPTION: CLooG is a free software and library to generate code (or an abstract syntax tree of a code) for scanning Z-polyhedra. That is, it finds a code (e.g. in C, FORTRAN...) that reaches each integral point of one or more parameterized polyhedra. CLooG has been originally written to solve the code generation problem for optimizing compilers based on the polyhedral model. Nevertheless it is used now in various area e.g. to build control automata for high-level synthesis or to find the best polynomial approximation of a function. CLooG may help in any situation where scanning polyhedra matters. While the user has full control on generated code quality, CLooG is designed to avoid control overhead and to produce a very effective code. CLooG is widely used (including by GCC and LLVM compilers), disseminated (it is installed by default by the main Linux distributions) and considered as the state of the art in polyhedral code generation.

RELEASE FUNCTIONAL DESCRIPTION: It mostly solves building and offers a better OpenScop support.

- Participant: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://www.cloog.org

6.5. IBB

Iterate-But-Better

FUNCTIONAL DESCRIPTION: IBB is a source-to-source xfor compiler which automatically translates any C source code containing xfor-loops into an equivalent source code where xfor-loops have been transformed into equivalent for-loops.

RELEASE FUNCTIONAL DESCRIPTION: The IBB compiler has been improved in some aspects in 2014: loop bounds can now be min and max functions, IBB uses the OpenScop format to encode statements and iteration domains.

- Participants: Cédric Bastoul, Imèn Fassi and Philippe Clauss
- Contact: Philippe Clauss
- URL: http://xfor.gforge.inria.fr

6.6. OpenScop

A Specification and a Library for Data Exchange in Polyhedral Compilation Tools

FUNCTIONAL DESCRIPTION: OpenScop is an open specification that defines a file format and a set of data structures to represent a static control part (SCoP for short), i.e., a program part that can be represented in the polyhedral model. The goal of OpenScop is to provide a common interface to the different polyhedral compilation tools in order to simplify their interaction. To help the tool developers to adopt this specification, OpenScop comes with an example library (under 3-clause BSD license) that provides an implementation of the most important functionalities necessary to work with OpenScop.

- Participant: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://icps.u-strasbg.fr/people/bastoul/public_html/development/openscop/

6.7. PolyLib

The Polyhedral Library
KEYWORDS: Rational polyhedra - Library - Polyhedral compilation
SCIENTIFIC DESCRIPTION: A C library used in polyhedral compilation, as a basic tool used to analyze, transform, optimize polyhedral loop nests. Has been shipped in the polyhedral tools Cloog and Pluto.
FUNCTIONAL DESCRIPTION: PolyLib is a C library of polyhedral functions, that can manipulate unions of rational polyhedra of any dimension. It was the first to provide an implementation of the computation of parametric vertices of a parametric polyhedron, and the computation of an Ehrhart polynomial (expressing the number of integer points contained in a parametric polytope) based on an interpolation method. Vincent Loechner is the maintainer of this software.

- Participant: Vincent Loechner
- Contact: Vincent Loechner
- URL: http://icps.u-strasbg.fr/PolyLib/

6.8. ORWL

Ordered Read-Write Lock
KEYWORDS: Task scheduling - Deadlock detection
FUNCTIONAL DESCRIPTION: ORWL is a reference implementation of the Ordered Read-Write Lock tools. The macro definitions and tools for programming in C99 that have been implemented for ORWL have been separated out into a toolbox called P99.

- Participants: Jens Gustedt, Mariem Saied and Stéphane Vialle
- Contact: Jens Gustedt
- Publications: Iterative Computations with Ordered Read-Write Locks - Automatic, Abstracted and Portable Topology-Aware Thread Placement - Resource-Centered Distributed Processing of Large Histopathology Images - Automatic Code Generation for Iterative Multi-dimensional Stencil Computations

6.9. P99

KEYWORD: Macro programming
FUNCTIONAL DESCRIPTION: P99 is a suite of macro and function definitions that ease the programming in modern C, minimum C99. By using tools from C99 and C11 we implement default arguments for functions, scope bound resource management, transparent allocation and initialization.

- Participants: Jens Gustedt, Mariem Saied and Stéphane Vialle
- Contact: Jens Gustedt
- URL: https://gforge.inria.fr/projects/p99/

6.10. stdatomic

standard atomic library
KEYWORD: Atomic access
SCIENTIFIC DESCRIPTION: We present a new algorithm and implementation of a lock primitive that is based on Linux' native lock interface, the futex system call. It allows us to assemble compiler support for atomic data structures that can not be handled through specific hardware instructions. Such a tool is needed for C11's atomics interface because here an _Atomic qualification can be attached to almost any data type. Our lock data structure for that purpose meets very specific criteria concerning its field of operation and its performance. By that we are able to outperform gcc's libatomic library by around 60%.
FUNCTIONAL DESCRIPTION: This implementation builds entirely on the two gcc ABIs for atomics. It doesn't even attempt to go down to assembly level by itself. We provide all function interfaces that the two gcc ABIs and the C standard need. For compilers that don't offer the direct language support for atomics this provides a syntactically reduced but fully functional approach to atomic operations.

- Author: Jens Gustedt
- Contact: Jens Gustedt
- Publications: Futex based locks for C11's generic atomics - Futex based locks for C11's generic atomics (extended abstract)
- URL: http://stdatomic.gforge.inria.fr/

6.11. musl

KEYWORDS: Standards - Library
SCIENTIFIC DESCRIPTION: musl provides consistent quality and implementation behavior from tiny embedded systems to full-fledged servers. Minimal machine-specific code means less chance of breakage on minority architectures and better success with "write once run everywhere" C development.

musl's efficiency is unparalleled in Linux libc implementations. Designed from the ground up for static linking, musl carefully avoids pulling in large amounts of code or data that the application will not use. Dynamic linking is also efficient, by integrating the entire standard library implementation, including threads, math, and even the dynamic linker itself into a single shared object, most of the startup time and memory overhead of dynamic linking have been eliminated.
FUNCTIONAL DESCRIPTION: We participate in the development of musl, a re-implementation of the C library as it is described by the C and POSIX standards. It is lightweight, fast, simple, free, and strives to be correct in the sense of standards-conformance and safety. Musl is production quality code that is mainly used in the area of embedded device. It gains more market share also in other area, e.g. there are now Linux distributions that are based on musl instead of Gnu LibC.

- Participant: Jens Gustedt
- Contact: Jens Gustedt
- URL: http://www.musl-libc.org/

6.12. Modular C

KEYWORDS: Programming language - Modularity

FUNCTIONAL DESCRIPTION: The change to the C language is minimal since we only add one feature, composed identifiers, to the core language. Our modules can import other modules as long as the import relation remains acyclic and a module can refer to its own identifiers and those of the imported modules through freely chosen abbreviations. Other than traditional C include, our import directive ensures complete encapsulation between modules. The abbreviation scheme allows to seamlessly replace an imported module by another one with equivalent interface. In addition to the export of symbols, we provide parameterized code injection through the import of "snippets". This implements a mechanism that allows for code reuse, similar to X macros or templates. Additional features of our proposal are a simple dynamic module initialization scheme, a structured approach to the C library and a migration path for existing software projects.

- Author: Jens Gustedt
- Contact: Jens Gustedt
- Publications: Modular C - Arbogast: Higher order AD for special functions with Modular C - Futex based locks for C11's generic atomics
- URL: http://cmod.gforge.inria.fr/

6.13. arbogast

KEYWORD: Automatic differentiation
SCIENTIFIC DESCRIPTION: This high-level toolbox for the calculus with Taylor polynomials is named after L.F.A. Arbogast (1759-1803), a French mathematician from Strasbourg (Alsace), for his pioneering work in derivation calculus. Its modular structure ensures unmatched efficiency for computing higher order Taylor polynomials. In particular it permits compilers to apply sophisticated vector parallelization to the derivation of nearly unmodified application code.
FUNCTIONAL DESCRIPTION: Arbogast is based on a well-defined extension of the C programming language, Modular C, and places itself between tools that proceed by operator overloading on one side and by rewriting, on the other. The approach is best described as contextualization of C code because it permits the programmer to place his code in different contexts – usual math or AD – to reinterpret it as a usual C function or as a differential operator. Because of the type generic features of modern C, all specializations can be delegated to the compiler.

- Author: Jens Gustedt
- Contact: Jens Gustedt
- Publications: Arbogast: Higher order AD for special functions with Modular C - Arbogast – Origine d'un outil de dérivation automatique
- URL: https://gforge.inria.fr/projects/arbo

6.14. CFML

Interactive program verification using characteristic formulae
KEYWORDS: Coq - Software Verification - Deductive program verification - Separation Logic
FUNCTIONAL DESCRIPTION: The CFML tool supports the verification of OCaml programs through interactive Coq proofs. CFML proofs establish the full functional correctness of the code with respect to a specification. They may also be used to formally establish bounds on the asymptotic complexity of the code. The tool is made of two parts: on the one hand, a characteristic formula generator implemented as an OCaml program that parses OCaml code and produces Coq formulae, and, on the other hand, a Coq library that provides notation and tactics for manipulating characteristic formulae interactively in Coq.

- Participants: Arthur Charguéraud, Armaël Guéneau and François Pottier
- Contact: Arthur Charguéraud
- URL: http://www.chargueraud.org/softs/cfml/

6.15. TLC

TLC Coq library

KEYWORDS: Coq - Library

FUNCTIONAL DESCRIPTION: TLC is a general purpose Coq library that provides an alternative to Coq's standard library. TLC takes as axiom extensionality, classical logic and indefinite description (Hilbert's epsilon). These axioms allow for significantly simpler formal definitions in many cases. TLC takes advantage of the type class mechanism. In particular, this allows for common operators and lemma names for all container data structures and all order relations. TLC includes the optimal fixed point combinator, which can be used for building arbitrarily-complex recursive and co-recursive definitions. Last, TLC provides a collection of tactics that enhance the default tactics provided by Coq. These tactics help constructing more concise and more robust proof scripts.

- Contact: Arthur Charguéraud
- URL: http://www.chargueraud.org/softs/tlc/

7. New Results

7.1. Automatic (Un-)Collapsing of Non-Rectangular Loops

Participants: Philippe Clauss, Ervin Altıntaş, Matthieu Kuhn.

Loop collapsing is a well-known loop transformation which combines some loops that are perfectly nested into one single loop. It allows to take advantage of the whole amount of parallelism exhibited by the collapsed loops, and provides a perfect load balancing of iterations among the parallel threads.

However, in the current implementations of this loop optimization, as the ones of the OpenMP language, automatic loop collapsing is limited to loops with constant loop bounds that define rectangular iteration spaces, although load imbalance is a particularly crucial issue with non-rectangular loops. The OpenMP language addresses load balance mostly through dynamic runtime scheduling of the parallel threads. Nevertheless, this runtime schedule introduces some unavoidable execution-time overhead, while preventing to exploit the entire parallelism of all the parallel loops.

We propose a technique to automatically collapse any perfectly nested loops defining non-rectangular iteration spaces, whose bounds are linear functions of the loop iterators. Such spaces may be triangular, tetrahedral, trapezoidal, rhomboidal or parallelepiped. Our solution is based on original mathematical results addressing the inversion of a multi-variate polynomial that defines a ranking of the integer points contained in a convex polyhedron.

We show on a set of non-rectangular loop nests that our technique allows to generate parallel OpenMP codes that outperform the original parallel loop nests, parallelized either by using options "static" or "dynamic" of the OpenMP-schedule clause. A conference paper presenting these results, co-authored by Philippe Clauss, Ervin Altıntaş (Master student) and Matthieu Kuhn (Inria Bordeaux Sud-Ouest, team HIEPACS), has been published at the International Parallel and Distributed Processing Symposium (IPDPS) [15].

We are currently developing a technique to also provide good load balancing when parallelizing non-rectangular loops carrying dependences. This new technique has been called *loop uncollapsing*. The idea is to split the outermost parallel loop into two nested loops, such that the new outermost loop, when parallelized, results in well-balanced parallel threads.

7.2. Code-Bones for Fast and Flexible Runtime Code Generation

Participants: Juan Manuel Martinez Caamaño, Manuel Selva, Philippe Clauss.

We have developed a new runtime code generation technique for speculative loop optimization and parallelization. The main benefit of this technique, compared to previous approaches, is to enable advanced optimizing loop transformations at runtime with an acceptable time overhead. The loop transformations that may be applied are those handled by the polyhedral model. The proposed code generation strategy is based on the generation of *code-bones* at compile-time, which are parametrized code snippets either dedicated to speculation management or to computations of the original target program. These code bones are then instantiated and assembled at runtime to constitute the speculatively-optimized code, as soon as an optimizing polyhedral transformation has been determined. Their granularity threshold is sufficient to apply any polyhedral transformation, while still enabling fast runtime code generation. This approach has been implemented in the speculative loop parallelizing framework Apollo, and has been more recently extended to also support loops exhibiting a non-linear behavior thanks to a modeling using “tubes”. The whole approach has been published in Concurrency and Computation: Practice and Experience [11].

7.3. Formal Proofs about Explicitly Parallel Programs with Clocks

Participants: Alain Ketterlin, Éric Violard, Tomofumi Yuki, Paul Feautrier.

We have continued this year our work on formalizing the *happens-before* relation in explicitly parallel programs of the X10 family. Our goal is to define, for certain classes of programs, a relation between instances of elementary instructions that guarantees that one instance necessarily executes before another. Our toy language includes static-control counted loops and conditionals, as well as the usual `finish` and `async` parallel constructs. Moreover, parallel activities can synchronize though the use of *clocks*, which are barriers with dynamic membership. Clocks partition the execution into phases, and profoundly modify the happens-before relation.

This year’s work has focused on correctly accounting for the possibility to define specific activities that execute irrespective of the discipline of the clock in scope, so-called *detached* activities. The presence of such activities modifies the notion of phase number, because they let their instructions execute across a range of clock-phases. Our generic notion of phase *ranking* had to be modified. Similarly, the natural semantics we defined had to be slightly modified to correctly represent the parallel execution of both clocked and detached activities. In practice, almost every lemma of the Coq proof has changed, and new definitions were introduced. The new definition of happens-before preserves all desirable properties: it is correct and complete, and is a strict partial order. There is one unpleasant aspect of detached activities that had a strong impact on happens-before: the possibility of deadlocks. A significant part of new definitions and lemmas are devoted to explicit the conditions under which programs terminate. A useful outcome of this part of the mechanization is a static, compile-time deadlock detection criterion.

Most of this work has been described in a paper currently under submission (this paper will be on HAL as soon as anonymity constraints permit). However, the diversity of themes covered in this research (compilation of static-control programs, especially those that fit the polyhedral model, but also semantic modeling of explicitly parallel programs, and formal proofs) make us contemplate the redaction of a much longer paper, which we plan to start at the beginning of next year. At the same time, this work (especially the part about deadlocks) led us to start designing an happens-before relation for a language where multiple clocks can share (part of) their scopes. We hope to be able to advance the formalization of this new family of languages in the near future.

7.4. High-Performance Particle-in-Cell Simulations

Participants: Arthur Charguéraud, Yann Barsamian, Alain Ketterlin.

Yann Barsamian’s PhD thesis focuses on the development of efficient programs for Particle-in-Cell (PIC) simulations, with application to plasma physics. Typically, a simulation involves a cluster of machines, each machine hosting several cores, and each core being able to execute vectorized instructions (SIMD). The challenge is to efficiently exploit these three levels of parallelism. Regarding the processing on one given multicore machine, existing algorithms either suffer from suboptimal execution time, due to sorting operations or use of atomic instructions, or suffer from suboptimal space usage. We have developed a novel parallel algorithm for PIC simulations on multicore hardware that features asymptotically-optimal memory consumption, and that does not perform unnecessary accesses to the main memory. The algorithm relies on the use of *chunk bags*, i.e., linked lists of fixed-capacity arrays, for storing particles and allowing to process them efficiently using SIMD instructions. Practical results show excellent scalability on the classical Landau damping and two-stream instability test cases. A paper was published at PPAM [12].

7.5. Granularity Control for Parallel Programs

Participant: Arthur Charguéraud.

Arthur Charguéraud contributes to the ERC DeepSea project, which is hosted at Inria Paris (team Gallium). With his co-authors, he focused this year on the development of techniques for controlling granularity in parallel programs. Granularity control is an essential problem because creating too many tasks may induce overwhelming overheads, while creating too few tasks may harm the ability to process tasks in parallel. Granularity control turns out to be especially challenging for nested parallel programs, i.e., programs in which parallel constructs such as fork-join or parallel-loops can be nested arbitrarily. This year, the DeepSea team investigated two different approaches.

The first one is based on the use of asymptotic complexity functions provided by the programmer, combined with runtime measurements to estimate the constant factors that apply. Combining these two sources of information allows to predict with reasonable accuracy the execution time of tasks. Such predictions may be used to guide the generation of tasks, by sequentializing computations of sufficiently-small size. An analysis is developed, establishing that task creation overheads are indeed bounded to a small fraction of the total runtime. These results extend prior work by the same authors [29], extending them with a carefully-designed algorithm for ensuring convergence of the estimation of the constant factors deduced from the measures, even in the face of noise and cache effects, which are taken into account in the analysis. The approach is demonstrated on a range of benchmarks taken from the state-of-the-art PBBS benchmark suite. These results were submitted to an international conference.

The second approach is based on an instrumentation of the runtime system. The idea is to process parallel function calls just like normal function calls, by pushing a frame on the stack, and only subsequently promoting these frames as threads that might get scheduled on other cores. The promotion of frames takes place at regular time interval, hence the name *heartbeat scheduling* given to the approach. Unlike in prior approaches such as *lazy scheduling*, in which promotion is guided by the work load of the system, hearbeat scheduling can be proved to induce only small scheduling overheads, and to not reduce asymptotically the amount of parallelism inherent to the parallel program. The theory behind the approach is formalized in Coq. It is also implemented through instrumented C++ programs, and evaluated on PBBS benchmarks. A paper describing this approach was submitted to an international conference.

7.6. Program verification and formal languages

Participant: Arthur Charguéraud.

- A. Charguéraud and François Pottier (Inria Paris) extended their formalization of the correctness and asymptotic complexity of the classic Union Find data structure, which features the bound expressed in terms of the inverse Ackermann function. The proof, conducted using CFML extended with time credits, was refined using a slightly more complex potential function, allowing to derive a simpler and richer interface for the data structure. This work appeared in the Journal of Automated Reasoning (JAR) [9].
- A. Charguéraud and F. Pottier have developed an extension of Separation Logic with temporary read-only permissions. This mechanism allows to temporarily convert any assertion (or “permission”) to a read-only form. Unlike with fractional permissions, no accounting is required: the proposed read-only permissions can be freely duplicated and discarded. Where mutable data structures are temporarily accessed only for reading, the proposed read-only permissions enable more concise specifications and proofs. All the metatheory is verified in Coq. An article was presented at ESOP [14].

- Armaël Guéneau, PhD student advised by A. Charguéraud and F. Pottier, has developed a Coq library formalizing the asymptotic notation (big-O), and has developed an extension of the CFML verification tool to allow specifying the asymptotic complexity of higher-order, imperative programs. This new feature has been tested on several classic examples of complexity analyses, including: nested loops in $O(n^3)$ and $O(nm)$, selection sort in $O(n^2)$, recursive functions in $O(n)$ and $O(2^n)$, binary search in $O(\log n)$, and Union-Find in $O(\alpha(n))$. A paper was submitted paper to an international conference.
- A. Charguéraud has made progress towards CFML 2.0, a reimplementation of CFML entirely inside Coq. In contrast, the initial version of CFML, developed in A. Charguéraud's PhD thesis, is based on an external tool that parses OCaml source code and produces Coq axioms describing their semantics. The new version will remove the need for axioms, thereby further reducing the trusted code base. Furthermore, CFML 2.0 provides a more general memory model, designed to also accomodate formal reasoning about C-style programs, in future work. In passing, A. Charguéraud performed a complete cleanup of the TLC Coq library, which is used extensively by CFML, leading to the beta release of TLC 2.0.
- A. Charguéraud, together with Alan Schmitt (Inria Rennes) and Thomas Wood (Imperial College), developed an interactive debugger for JavaScript. The interface, accessible as a webpage in a browser, allows to execute a given JavaScript program, following step by step the formal specification of JavaScript developed in prior work on *JsCert* [31]. Concretely, the tool acts as a double-debugger: one can visualize both the state of the interpreted program and the state of the interpreter program. This tool is intended for the JavaScript committee, VM developers, and other experts in JavaScript semantics. A paper describing the tool has been submitted, and the tool has been presented to the JavaScript standardization committee (ECMA) in November 2017.

7.7. Combining Locking and Data Management Interfaces

Participants: Jens Gustedt, Mariem Saied, Daniel Salas.

Handling data consistency in parallel and distributed settings is a challenging task, in particular if we want to allow for an easy to handle asynchronism between tasks. Our publication [1] shows how to produce deadlock-free iterative programs that implement strong overlapping between communication, IO and computation.

An implementation (ORWL) of our ideas of combining control and data management in C has been undertaken, see Section 6.8. In previous work it has demonstrated its efficiency for a large variety of platforms.

This year, wee have been able to use the knowledge of the communication structure of ORWL programs to map tasks to cores and thereby achieve interesting performance gains on multicore architectures, see [16]. We propose a topology-aware placement module that is based on the Hardware Locality framework, HWLOC, and that takes the characteristics of the application, of the runtime and of the architecture into account. The aim is double. On one hand we increase the abstraction and the portability of the framework, and on the other hand we enhance the performance of the model's runtime.

Within the framework of the thesis of Daniel Salas we have successfully applied ORWL to process large histopathology images. We are now able to treat such images distributed on several machines or shared in an accelerator (Xeon Phi) transparently for the user.

7.8. Automatic Generation of Adaptive Simulation Codes

Participants: Cédric Bastoul, Maxime Schmitt.

Compiler automatic optimization and parallelization techniques are well suited for some classes of simulation or signal processing applications, however they usually don't take into account neither domain-specific knowledge nor the possibility to change or to remove some computations to achieve "good enough" results. Quite differently, production simulation and signal processing codes have adaptive capabilities: they are designed to compute precise results only where it matters if the complete problem is not tractable or if the computation time must be short. In this research, we design a new way to provide adaptive capabilities to compute-intensive codes automatically, inspired by Adaptive Mesh Refinement a classical numerical analysis technique to achieve precise computation only in pertinent areas. It relies on domain-specific knowledge provided through special pragmas by the programmer in the input code and on polyhedral compilation techniques, to continuously regenerate at runtime a code that performs heavy computations only where it matters at every moment. A case study on a fluid simulation application shows that our strategy enables dramatic computation savings in the optimized portion of the application while maintaining good precision, with a minimal effort from the programmer.

This research direction started in 2015 and complements our other efforts on dynamic optimization. In 2016, we started a collaboration on this topic with Inria Nancy - Grand Est team TONUS, specialized on applied mathematics (contact: Philippe Helluy), to bring models and techniques from this field to compilers. This collaboration received the support from the excellence laboratory (LabEx) IRMIA through the funding of the thesis of Maxime Schmitt on this topic. Two papers on this new research direction has been accepted this year on this topic (IMPACT 2017 workshop, HiPC 2017 conference [20]).

7.9. Parallel Polyhedral Regions

Participants: Cédric Bastoul, Vincent Loechner, Harenome Ranaivoarivony-Razanajato.

Nowadays best performing automatic parallelizers and data locality optimizers for static control programs rely on the polyhedral model. State-of-the-art polyhedral compilers generate only one type of parallelism when targeting multicore shared memory architectures: parallel loops via the OpenMP `omp parallel for` directive.

We propose to explore how a polyhedral compiler could exploit parallel region constructs. Instead of initializing a new set of threads each time the code enters a parallel loop and synchronizing them when exiting it, the threads are initialized once for all at the entrance of the region of interest, and synchronized only when it is necessary.

Technically, the whole region containing parallel loops is embedded in an `omp parallel` construct. Inside the parallel region, the `single` construct is used when some code needs to be executed sequentially; the `for` construct is used to distribute loop iterations between threads. Thanks to the power of the polyhedral dependence analysis, we compute when it is valid to add the optional `nowait` clause, to omit the implicit barrier at the end of a worksharing construct and thus to reduce even more control overhead.

This work was published and presented at the HiPC 2017 conference [19].

7.10. Optimization of Sparse Triangular and Banded Matrix Codes

Participants: Vincent Loechner, Rachid Seghir, Toufik Baroudi.

This work is a collaboration between Vincent Loechner and Rachid Seghir from University of Batna (Algeria). Toufik Baroudi is a second year PhD student under his supervision. Rachid Seghir was visiting the CAMUS team from March 25th to April 8th, 2017.

In this work, we enabled static polyhedral optimization techniques to handle sparse matrix storage formats. When handling sparse triangular and banded matrices in their packed formats, such as in the LAPACK library band storage, loop nests bounds and array references of the resulting codes are not affine functions. We proposed to use a new 2d-packed layout and simple affine transformations to enable polyhedral optimization of sparse triangular and banded matrix operations. The effectiveness of our proposal was shown through an experimental study over a large set of linear algebra benchmarks.

These results were published in ACM TACO [8], and will be presented at the HiPEAC conference in January 2018.

8. Bilateral Contracts and Grants with Industry

8.1. NANO 2017/PSAIC

The CAMUS team is taking part of the NANO 2017 national research program and its sub-project PSAIC (Performance and Size Auto-tuning thru Iterative Compilation) with the company STMicroelectronics, which started in January 2015. Since the release of our automatic speculative parallelization framework Apollo, we have been working on an extension making Apollo usable as a advanced program profiling tool. We are currently working in extending Apollo to the memoization of the memory behavior for loops that are invoked several times.

8.2. Caldera

Vincent Loechner and Cédric Bastoul are involved in a collaboration with the French company Caldera (http://www.caldera.com), specialized in software development for wide image processing. The goal of this collaboration is the development of parallel and scalable image processing pipeline for industrial printing. The project started in September 2016 and involves a contract established between the ICube laboratory and the Caldera company. This contract includes the funding of the industrial thesis (CIFRE) of Paul Godard (started in September 2016) on the topic of the collaboration, under the supervision of Vincent Loechner and Cédric Bastoul.

9. Partnerships and Cooperations

9.1. National Initiatives

9.1.1. Inria Large Scale Initiative on Multicore

Philippe Clauss, Jens Gustedt, Alain Ketterlin, Cédric Bastoul and Vincent Loechner are involved in the Inria Project Lab entitled “Large scale multicore virtualization for performance scaling and portability” and regrouping several French researchers in compilers, parallel computing and program optimization [1]. The project started officially in January 2013. In this context and since January 2013, Philippe Clauss is co-advising with Erven Rohou of the Inria team PACAP, Nabil Hallou’s PhD thesis focusing on dynamic optimization of binary code. The PhD defense was held December the 18th 2017.

Philippe Clauss, Jens Gustedt and Maxime Mogé are involved in the ADT Inria project ASNAP (*Accélération des Simulations Numériques pour l’Assistance Peropératoire*), in collaboration with the Inria team MIMESIS. The goal is to find opportunities in the SOFA simulation platform for applying automatic parallelization techniques developed by Camus. We are currently investigating two approaches. The first uses memory behavior memoization to generate a parallel code made of independent threads at runtime. The second uses ordered read-write locks (ORWL) to dynamically schedule a pipeline of parallel tasks.

9.1.2. ANR AJACS

Participant: Arthur Charguéraud [contact].

The AJACS research project is funded by the programme “Société de l’information et de la communication” of the ANR, from October 2014, until November 2018. http://ajacs.inria.fr/

[1] https://team.inria.fr/multicore

The goal of the AJACS project is to provide strong security and privacy guarantees on the client side for web application scripts implemented in JavaScript, the most widely used language for the Web. The proposal is to prove correct analyses for JavaScript programs, in particular information flow analyses that guarantee no secret information is leaked to malicious parties. The definition of sub-languages of JavaScript, with certified compilation techniques targeting them, will allow deriving more precise analyses. Another aspect of the proposal is the design and certification of security and privacy enforcement mechanisms for web applications, including the APIs used to program real-world applications. Arthur Charguéraud focuses on the description of a formal semantics for JavaScript, and the development of tools for interactively executing programs step-by-step according to the formal semantics.

Partners: team Celtique (Inria Rennes - Bretagne Atlantique), team Prosecco (Inria Paris), team Indes (Inria Sophia Antipolis - Méditerranée), and Imperial College (London).

9.1.3. ANR Vocal

Participant: Arthur Charguéraud [contact].

The Vocal research project is funded by the programme "Société de l'information et de la communication" of the ANR, for a period of 48 months, starting on October 1st, 2015. https://vocal.lri.fr/

The goal of the Vocal project is to develop the first formally verified library of efficient general-purpose data structures and algorithms. It targets the OCaml programming language, which allows for fairly efficient code and offers a simple programming model that eases reasoning about programs. The library will be readily available to implementers of safety-critical OCaml programs, such as Coq, Astrée, or Frama-C. It will provide the essential building blocks needed to significantly decrease the cost of developing safe software. The project intends to combine the strengths of three verification tools, namely Coq, Why3, and CFML. It will use Coq to obtain a common mathematical foundation for program specifications, as well as to verify purely functional components. It will use Why3 to verify a broad range of imperative programs with a high degree of proof automation. Finally, it will use CFML for formal reasoning about effectful higher-order functions and data structures making use of pointers and sharing.

Partners: team Gallium (Inria Paris), team DCS (Verimag), TrustInSoft, and OCamlPro.

9.2. European Initiatives

9.2.1. FP7 & H2020 Projects

Project acronym: ERC Deepsea

Project title: Parallel dynamic computations

Duration: Jun. 2013 - May 2018

Coordinator: Umut A. Acar

Other partners: Carnegie Mellon University

Abstract:

The objective of this project is to develop abstractions, algorithms and languages for parallelism and dynamic parallelism with applications to problems on large data sets. Umut A. Acar (affiliated to Carnegie Mellon University and Inria Paris - Rocquencourt) is the principal investigator of this ERC-funded project. The other main researchers involved are Mike Rainey (Inria, Gallium team), who is full-time on the project, and Arthur Charguéraud (Inria, Toccata Camus), who works part time on this project. Project website: http://deepsea.inria.fr/.

9.3. International Initiatives

9.3.1. Inria International Partners

9.3.1.1. Informal International Partners

The CAMUS team maintains regular contacts with the following entities:

- Reservoir Labs, New York, NY, USA
- University of Batna, Algeria
- Ohio State University, Colombus, USA
- Louisiana State University, Baton Rouge, USA
- Colorado State University, Fort Collins, USA
- Carnegie Mellon University, Pittsburgh, USA
- Indian Institute of Science (IIIS) Bangalore, India
- Barcelona Supercomputing Center, Barcelona, Spain

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Selection

10.1.1.1. Member of the Conference Program Committees

Philippe Clauss and Cédric Bastoul have been part of the program committee of IMPACT 2017 and 2018 (International Workshop on Polyhedral Compilation Techniques), held in conjunction with the international conference HiPEAC.

Cédric Bastoul has been part of the program committee of the international conference on Compiler Construction 2017 and 2018 (CC'2017 and CC'2018).

Cédric Bastoul and Vincent Loechner have been part of the program committee of the HIP3ES workshop 2017 and 2018 (International Workshop on High Performance Energy Efficient Embedded Systems), co-organized by Cédric Bastoul in conjunction with the international conference HiPEAC.

Arthur Charguéraud has been part of the program committee for the Conference on Verified Software: Theories, Tools, and Experiments (VSTTE 2017).

10.1.1.2. Reviewer

Philippe Clauss has been reviewer for the following conferences and workshops: IMPACT 2017 and 2018 (International Workshop on Polyhedral Compilation Techniques), CC 2017 (International Conference on Compiler Construction).

Cédric Bastoul has been reviewer for the following international conferences and workshops: CC 2017 and 2018 (International Conference on Compiler Construction), PARMA 2017 (International Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures), IMPACT 2017 and 2018 (International Workshop on Polyhedral Compilation Techniques), HIP3ES 2017 and 2018 (International Workshop on High Performance Energy Efficient Embedded Systems).

10.1.2. Journal

10.1.2.1. Member of the Editorial Boards

Since October 2001, J. Gustedt is Editor-in-Chief of the journal *Discrete Mathematics and Theoretical Computer Science* (DMTCS).

10.1.2.2. Reviewer - Reviewing Activities

Philippe Clauss has been reviewer for the following journals: Journal of Computer and System Sciences, Journal of Software: Practice and Experience, IEEE Transactions on Computers.

Cédric Bastoul has been reviewer for the following journals: Journal of Parallel, Emergent and Distributed Systems, and IEEE Transactions on Computers.

Arthur Charguéraud has been reviewer for JAR (Journal of Automated Reasoning), DMTCS (journal of Discrete Mathematics and Theoretical Computer Science), and JFP (Journal of Functional Programming).

Vincent Loechner has been reviewer for: JAR (Journal of Automated Reasoning, Springer), STTT (Int. J. on Software Tools for Technology Transfer, Springer), ComCom (Computer Communications, Elsevier).

10.1.3. Invited Talks

Philippe Clauss has been invited to give a talk at a seminar dedicated to Jean-Luc Gaudiot, organized by the French Computer Science Engineering school ENSIEE, Paris, September the 21st 2017. The topic of his talk was: *Le modèle polyédrique au delà de la compilation statique, des fonctions affines et des boucles*.

Arthur Charguéraud has been invited to give a talk at ENS Rennes, on November 21st, 2017, to present the CFML interactive program verification tool.

10.1.4. Scientific Expertise

Cédric Bastoul as been an expert for the French research ministry and the French finance ministry for the research tax credit programme.

10.1.5. Standardization

Since Nov. 2014, Jens Gustedt is a member of the ISO working group SC22-WG14 for the standardization of the C programming language and serves as co-editor of the standards document. He participates actively in the defect report processing, the planning of future versions of the standard, and publishes an ongoing document to track inconsistencies and improvements of the C threads interface.

In 2017, he was the one of the main forces behind the elaboration of C17, the new version of the C standard that is expected to go into ballot in the member states end of 2017.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Licence : Philippe Clauss, Architecture des ordinateurs, 45h, Université de Strasbourg, France

Licence : Philippe Clauss, Systèmes d'exploitation, 40h, Université de Strasbourg, France

Master : Philippe Clauss, Compilation, 78h, Université de Strasbourg, France

Master : Philippe Clauss, Système et programmation temps-réel, 25h, Université de Strasbourg, France

Master : Philippe Clauss, Compilation avancée, 30h, Université de Strasbourg, France

2nd year engineering school: Jens Gustedt, programmation avancée, 20h, ENSIIE Strasbourg, France

Licence : Jens Gustedt, systèmes concurrents, 20h, Université de Strasbourg, France

Master : Jens Gustedt, parallélisme, 14h, M1, Université de Strasbourg, France

IUT d'Informatique : Alain Ketterlin, Architecture et programmation des mécanismes de base d'un système informatique, 68h, Université de Strasbourg, France

Licence : Alain Ketterlin, Algorithmique et programmation L1, 82h, Université de Strasbourg, France

Master (Informatique) : Alain Ketterlin, Ingénierie de la preuve en Coq, 18h, Université de Strasbourg, France

Master (Calcul Scientifique et Mathématiques de l'Information) : Alain Ketterlin, Compilation et optimisation, 26h, Université de Strasbourg, France

Licence : Cédric Bastoul, Computer architecture, 68h, L1 (IUT), Université de Strasbourg, France

Licence : Cédric Bastoul, Concurrent Systems, 20h, L3, Université de Strasbourg, France

Master : Cédric Bastoul, Compiler Design, 48h, M1, Université de Strasbourg, France

Master : Cédric Bastoul, Parallelism, 19h, M1, Université de Strasbourg, France

Master : Cédric Bastoul, Introduction to Research, 11h, L2+M1, Université de Strasbourg, France

Licence : Eric Violard, Programmation Fonctionnelle (licence informatique), 64h eq. TD, L2, Université de Strasbourg, France

Licence : Eric Violard, Architecture des Ordinateurs (licence informatique), 54h eq. TD, L2, Université de Strasbourg, France

Licence : Eric Violard, Logique et Programmation Logique (licence informatique), 34h eq. TD, L2, Université de Strasbourg, France

Licence : Eric Violard, Algorithmique et Structure de Données (licence mathématique), 39h eq. TD, L3, Université de Strasbourg, France

Licence : Eric Violard, Modèles de Calcul (licence informatique), 29h eq. TD, L1, Université de Strasbourg, France

Licence : Eric Violard, Systèmes Concurrents (licence informatique), 7h eq. TD, L3, Université de Strasbourg, France

Master : Arthur Charguéraud, Proof of Programs (MPRI), 12h, M2, Université Paris Diderot, France

Licence : Vincent Loechner, responsable pédagogique de la licence professionnelle SIL spécialité ARS (Administration de Réseaux et Services), 24h, L3, université de Strasbourg, France

Licence : Vincent Loechner, systèmes d'exploitation, 13h, L2, université de Strasbourg, France

Licence : Vincent Loechner, administration système et internet, 54h, L3, université de Strasbourg, France

Master : Vincent Loechner, calcul parallèle, 32h, M1, université de Strasbourg, France

Master : Vincent Loechner, OS embarqués, 34h, M2, université de Strasbourg, France

Master : Vincent Loechner, calcul parallèle, 30h, 3ième année école d'ingénieur (TPS), université de Strasbourg, France

10.2.2. Supervision

PhD: Nabil Hallou, *Dynamic binary optimizations*, University of Rennes, December the 18th 2017, Erven Rohou (PACAP team) and Philippe Clauss

PhD in progress: Salwa Kobeissi, *Dynamic parallelization of recursive functions by transformation into loops*, September 2017, Philippe Clauss

PhD in progress: Mariem Saied, *Ordered Read-Write Locks for Multicores and Accelerators*, since Nov 2013, Jens Gustedt & Gilles Muller.

PhD in progress: Daniel Salas, *Integration of the ORWL model into parallel applications for medical research*, since Mar 2015, Jens Gustedt & Isabelle Perseil.

PhD in progress: Yann Barsamian, *Optimization and parallelization of particle and semi-Lagrangian methods for multi species plasma simulations*, since Oct 2014, Eric Violard.

PhD in progress: Armaël Géneau, *Formal verification of complexity analyses*, since Sept 2016, co-advised by Arthur Charguéraud and François Pottier, from team Gallium (Inria Paris), where Armaël is located.

PhD in progress: Harenome Ranaivoarivony-Razanajato, *Hierarchical Parallelization and Optimization*, Oct. 2016, Cédric Bastoul and Vincent Loechner

PhD in progress: Maxime Schmitt, *Automatic Generation of Adaptive Codes*, September 2016, Cédric Bastoul and Philippe Helluy

PhD in progress : Paul Godard, *Parallelization and Scalability of a Graphical Pipeline for Professionnal Inkjet Printing*, Jun. 2016, Cédric Bastoul and Vincent Loechner

10.2.3. Juries

Philippe Clauss participated to the following PhD committees in 2017:

Date	**Candidate**	**Place**	**Role**
Dec. 11	Alexandre Maréchal	Université de Grenoble	Examiner
Dec. 18	Nabil Hallou	Université de Rennes	Co-advisor
Dec. 21	Jordy Ruiz	Université de Toulouse	Reviewer

Vincent Loechner participated as examiner to the PhD committee of Maroua Maalej, defended on Sept. 26th 2017 at Université Claude Bernard (Lyon 1).

10.3. Popularization

A. Charguéraud is one of the three organizers of the *Concours Castor informatique* http://castor-informatique.fr/. The purpose of the Concours Castor in to introduce pupils (from *CM1* to *Terminale*) to computer sciences. More than 500,000 teenagers played with the interactive exercises in November 2017.

Jens Gustedt is blogging about efficient programming, in particular about the C programming language. He also is an active member of the stackoverflow community a technical Q&A site for programming and related subjects.

Cédric Bastoul prepared activities and participated to *Fête de la Science* at University of Strasbourg in October 2017.

11. Bibliography

Major publications by the team in recent years

[1] P.-N. Clauss, J. Gustedt. *Iterative Computations with Ordered Read-Write Locks*, in "Journal of Parallel and Distributed Computing", 2010, vol. 70, n^{o} 5, pp. 496–504 [*DOI :* 10.1016/j.jpdc.2009.09.002], https://hal.inria.fr/inria-00330024

[2] J. Gustedt. *Futex based locks for C11's generic atomics*, Inria Nancy, December 2015, n^{o} RR-8818, https://hal.inria.fr/hal-01236734

[3] A. Jimborean, P. Clauss, J.-F. Dollinger, V. Loechner, M. Juan Manuel. *Dynamic and Speculative Polyhedral Parallelization Using Compiler-Generated Skeletons*, in "International Journal of Parallel Programming", August 2014, vol. 42, n^{o} 4, pp. 529-545, https://hal.inria.fr/hal-01003744

[4] A. Ketterlin, P. Clauss. *Prediction and trace compression of data access addresses through nested loop recognition*, in "6th annual IEEE/ACM international symposium on Code generation and optimization", Boston, USA, ACM, April 2008, pp. 94-103, http://dx.doi.org/10.1145/1356058.1356071

[5] A. Ketterlin, P. Clauss. *Profiling Data-Dependence to Assist Parallelization: Framework, Scope, and Optimization*, in "MICRO-45 – Proceedings of the 2012 IEEE/ACM 45th International Symposium on Microarchitecture", Vancouver, Canada, December 2012

[6] B. Pradelle, A. Ketterlin, P. Clauss. *Polyhedral parallelization of binary code*, in "ACM Transactions on Architecture and Code Optimization", January 2012, vol. 8, n^{o} 4, pp. 39:1–39:21 [*DOI :* 10.1145/2086696.2086718], http://hal.inria.fr/hal-00664370

[7] R. SEGHIR, V. LOECHNER, B. MEISTER. *Integer Affine Transformations of Parametric Z-polytopes and Applications to Loop Nest Optimization*, in "ACM Transactions on Architecture and Code Optimization", June 2012, vol. 9, n^{o} 2, pp. 8.1-8.27 [*DOI :* 10.1145/2207222.2207224], http://hal.inria.fr/inria-00582388

Publications of the year

Articles in International Peer-Reviewed Journals

[8] T. BAROUDI, R. SEGHIR, V. LOECHNER. *Optimization of Triangular and Banded Matrix Operations Using 2d-Packed Layouts*, in "ACM Transactions on Architecture and Code Optimization (TACO) ", December 2017, https://hal.inria.fr/hal-01633724

[9] A. CHARGUÉRAUD, F. POTTIER. *Verifying the Correctness and Amortized Complexity of a Union-Find Implementation in Separation Logic with Time Credits*, in "Journal of Automated Reasoning", September 2017 [*DOI :* 10.1007/s10817-017-9431-7], https://hal.inria.fr/hal-01652785

[10] N. HALLOU, E. ROHOU, P. CLAUSS. *Runtime Vectorization Transformations of Binary Code*, in "International Journal of Parallel Programming", June 2017, vol. 8, n^{o} 6, pp. 1536 - 1565 [*DOI :* 10.1007/s10766-016-0480-z], https://hal.inria.fr/hal-01593216

[11] J. M. MARTINEZ CAAMAN˜O, M. SELVA, P. CLAUSS, A. BALOIAN, W. WOLFF. *Full runtime polyhedral optimizing loop transformations with the generation, instantiation, and scheduling of code-bones*, in "Concurrency and Computation: Practice and Experience", June 2017, vol. 29, n^{o} 15 [*DOI :* 10.1002/CPE.4192], https://hal.inria.fr/hal-01581093

International Conferences with Proceedings

[12] Y. A. BARSAMIAN, A. CHARGUÉRAUD, A. KETTERLIN. *A Space and Bandwidth Efficient Multicore Algorithm for the Particle-in-Cell Method*, in "PPAM 2017 - 12th International Conference on Parallel Processing and Applied Mathematics", Lublin, Poland, September 2017, pp. 1-12, https://hal.inria.fr/hal-01649172

[13] Y. A. BARSAMIAN, S. A. HIRSTOAGA, E. VIOLARD. *Efficient Data Structures for a Hybrid Parallel and Vectorized Particle-in-Cell Code*, in "IPDPSW 2017 - IEEE International Parallel and Distributed Processing Symposium Workshops", Lake Buena Vista, FL, United States, May 2017, pp. 1168-1177 [*DOI :* 10.1109/IPDPSW.2017.74], https://hal.inria.fr/hal-01504645

[14] A. CHARGUÉRAUD, F. POTTIER. *Temporary Read-Only Permissions for Separation Logic*, in "Proceedings of the 26th European Symposium on Programming (ESOP 2017)", Uppsala, Sweden, April 2017, https://hal.inria.fr/hal-01408657

[15] P. CLAUSS, E. ALTINTAS, M. KUHN. *Automatic Collapsing of Non-Rectangular Loops*, in "Parallel and Distributed Processing Symposium (IPDPS), 2017", Orlando, United States, I. INTERNATIONAL (editor), May 2017, pp. 778 - 787 [*DOI :* 10.1109/IPDPS.2017.34], https://hal.inria.fr/hal-01581081

[16] J. GUSTEDT, E. JEANNOT, F. MANSOURI. *Automatic, Abstracted and Portable Topology-Aware Thread Placement*, in "IEEE Cluster", Hawaï, United States, Cluster Computing (CLUSTER), 2017 IEEE International Conference on, September 2017, pp. 389 - 399 [*DOI :* 10.1109/CLUSTER.2017.71], https://hal.archives-ouvertes.fr/hal-01621936

[17] N. MAGAUD. *Transferring Arithmetic Decision Procedures (on Z) to Alternative Representations*, in "CoqPL 2017: The Third International Workshop on Coq for Programming Languages", Paris, France, January 2017, https://hal.inria.fr/hal-01518660

[18] L. MOREL, M. SELVA, K. MARQUET, C. SAYSSET, T. RISSET. *CalMAR -a Multi-Application Dataflow Runtime*, in "Thirteenth ACM International Conference on Embedded Software 2017, EMSOFT'17", Seoul, South Korea, October 2017 [*DOI :* 10.1145/3125503.3125562], https://hal.inria.fr/hal-01631691

[19] H. RAZANAJATO, C. BASTOUL, V. LOECHNER. *Lifting Barriers Using Parallel Polyhedral Regions*, in "HiPC 2017 - 24th International Conference on High Performance Computing, Data, and Analytics", Jaipur, India, IEEE, December 2017, https://hal.inria.fr/hal-01633839

[20] M. SCHMITT, P. HELLUY, C. BASTOUL. *Adaptive Code Refinement: A Compiler Technique and Extensions to Generate Self-Tuning Applications*, in "HiPC 2017 - 24th International Conference on High Performance Computing, Data, and Analytics", Jaipur, India, December 2017, https://hal.inria.fr/hal-01655459

Conferences without Proceedings

[21] J. M. MARTINEZ CAAMAÑO, A. SUKUMARAN-RAJAM, A. BALOIAN, M. SELVA, P. CLAUSS. *APOLLO: Automatic speculative POLyhedral Loop Optimizer*, in "IMPACT 2017 - 7th International Workshop on Polyhedral Compilation Techniques", Stockholm, Sweden, January 2017, 8 p. , https://hal.inria.fr/hal-01533692

[22] H. RAZANAJATO, V. LOECHNER, C. BASTOUL. *Splitting Polyhedra to Generate More Efficient Code: Efficient Code Generation in the Polyhedral Model is Harder Than We Thought*, in "IMPACT 2017, 7th International Workshop on Polyhedral Compilation Techniques", Stockholm, Sweden, January 2017, https://hal.inria.fr/hal-01505764

[23] M. SCHMITT, C. SABATER, C. BASTOUL. *Semi-Automatic Generation of Adaptive Codes*, in "IMPACT 2017 - 7th International Workshop on Polyhedral Compilation Techniques", Stockholm, Sweden, January 2017, pp. 1-7, https://hal.inria.fr/hal-01655456

Research Reports

[24] I. CHARPENTIER, J. GUSTEDT. *Arbogast: Higher order AD for special functions with Modular C*, Inria Nancy - Grand Est (Villers-lès-Nancy, France), August 2017, n^{o} RR-8907, 20 p. , https://hal.inria.fr/hal-01307750

Other Publications

[25] Y. A. BARSAMIAN, J. BERNIER, S. A. HIRSTOAGA, M. MEHRENBERGER. *Verification of 2D × 2D and two-species Vlasov-Poisson solvers*, December 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01668744

[26] Y. A. BARSAMIAN, S. A. HIRSTOAGA, E. VIOLARD. *Efficient Data Layouts for a Three-Dimensional Electrostatic Particle-in-Cell Code*, 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01664207

[27] A. CHARGUÉRAUD, J.-C. FILLIÂTRE, M. PEREIRA, F. POTTIER. *VOCAL – A Verified OCAml Library*, September 2017, ML Family Workshop 2017, https://hal.inria.fr/hal-01561094

[28] A. CHARGUÉRAUD, M. RAINEY. *Efficient Representations for Large Dynamic Sequences in ML*, September 2017, ML Family Workshop, Poster, https://hal.inria.fr/hal-01669407

References in notes

[29] U. A. ACAR, A. CHARGUÉRAUD, M. RAINEY. *Oracle-Guided Scheduling for Controlling Granularity in Implicitly Parallel Languages*, in "Journal of Functional Programming", November 2016, vol. 26 [*DOI :* 10.1017/S0956796816000101], https://hal.inria.fr/hal-01409069

[30] C. BASTOUL. *Code Generation in the Polyhedral Model Is Easier Than You Think*, in "PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques", Juan-les-Pins, France, 2004, pp. 7–16, https://hal.archives-ouvertes.fr/ccsd-00017260

[31] M. BODIN, A. CHARGUÉRAUD, D. FILARETTI, P. GARDNER, S. MAFFEIS, D. NAUDZIUNIENE, A. SCHMITT, G. SMITH. *A Trusted Mechanised JavaScript Specification*, in "Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", San Diego, USA, ACM Press, January 2014, http://hal.inria.fr/hal-00910135

[32] M. HALL, D. PADUA, K. PINGALI. *Compiler research: the next 50 years*, in "Commun. ACM", 2009, vol. 52, n^{o} 2, pp. 60–67, http://doi.acm.org/10.1145/1461928.1461946

[33] A. HOBOR, A. W. APPEL, F. Z. NARDELLI. *Oracle Semantics for Concurrent Separation Logic*, in "ESOP", 2008, pp. 353-367