
# Activity Report 2018

# Team CAMUS

## Compilation pour les Architectures MUlti-coeurS

Inria teams are typically groups of researchers working on the definition of a common project, and objectives, with the goal to arrive at the creation of a project-team. Such project-teams may include other partners (universities or research institutions).

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
A2.2.4. - Parallel architectures
A2.2.5. - Run-time systems
A2.2.6. - GPGPU, FPGA...
A2.2.7. - Adaptive compilation

**Other Research Topics and Application Domains:**

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

# 1. Team, Visitors, External Collaborators

**Research Scientists**

Bérenger Bramas [Inria, Researcher, from Oct 2018]
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 Fellow**

Manuel Selva [Inria, until Jan 2018]

**PhD Students**

Yann Barsamian [Univ de Strasbourg, until Aug 2018]
Paul Godard [Caldera]
Salwa Kobeissi [Inria]
Harenome Ranaivoarivony-Razanajato [Univ de Strasbourg]
Mariem Saied [Univ de Strasbourg, until Sep 2018]
Daniel Salas [INSERM]
Maxime Schmitt [Univ de Strasbourg]

**Technical staff**

Muthena Abdul Wahab [Inria, from Aug 2018]

Maxime Mogé [Inria, until Aug 2018]

**Intern**

Ramon Fernandez [Inria, from Jun 2018 until Sep 2018]

**Administrative Assistant**

Ouiza Herbi [Inria, from Jun 2018]

# 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, correctness 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 correctness 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 [27]. Performance, correctness 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 actual 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, the complexity of current and future architectures 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.

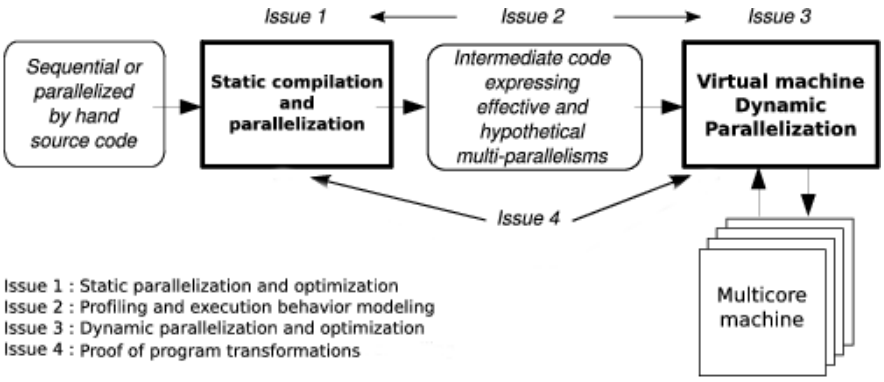


*Figure 1. Automatic parallelizing steps for multicore architectures*

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 correctness 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 [23]. Low-level optimizations, in the assembly code generated by the compiler, have also been extensively dealt with 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, Salwa Kobeissi.

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:** Philippe Clauss, Salwa Kobeissi, Jens Gustedt, Alain Ketterlin, Muthena Abdul Wahab, Mariem Saied, Daniel Salas, Maxime Mogé.

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 controlling the complexity of those new mechanisms. 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 the evolution of the availability of live resources.

## 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 [28]. 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 main 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

Bérenger Bramas, Inria Research Scientist, has joined the team in September 2018.
Matthew Wahab, Inria Research Engineer, has joined the team in August 2018.

# 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. 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/

## 6.6. 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

## 6.7. 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 devices. It gains more market share also in other areas, 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/

## 6.8. 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 an 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 automatic differentiation for special functions with Modular C - Futex based locks for C11's generic atomics
- URL: http://cmod.gforge.inria.fr/

## 6.9. 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 automatic differentiation for special functions with Modular C - Arbogast – Origine d'un outil de dérivation automatique
- URL: https://gforge.inria.fr/projects/arbo

## 6.10. 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 notations and tactics for manipulating characteristic formulae interactively in Coq.

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

## 6.11. SPETABARU

*SPEculative TAsk-BAsed RUntime system*
KEYWORDS: HPC - Parallel computing - Task-based algorithm
FUNCTIONAL DESCRIPTION: SPETABARU is a task-based runtime system for multi-core architectures that includes speculative execution models. It is a pure C++11 product without external dependency. It uses advanced meta-programming and allows for an easy customization of the scheduler. It is also capable to generate execution traces in SVG to better understand the behavior of the applications.

- Contact: Bérenger Bramas
- URL: https://gitlab.inria.fr/bramas/spetabaru

# 7. New Results

## 7.1. AutoParallel: A Python module for automatic parallelization and distributed execution of affine loop nests

**Participant:** Philippe Clauss.

The last improvements in programming languages, programming models, and frameworks have focused on abstracting the users from many programming issues. Among others, recent programming frameworks include simpler syntax, automatic memory management and garbage collection, which simplifies code reusage through library packages, and easily configurable tools for deployment. For instance, Python has risen to the top of the list of the programming languages due to the simplicity of its syntax, while still achieving a good performance even being an interpreted language. Moreover, the community has helped to develop a large number of libraries and modules, tuning the most commonly used to obtain great performance.

However, there is still room for improvement when preventing users from dealing directly with distributed and parallel computing issues. This work proposes AutoParallel, a Python module to automatically find an appropriate task-based parallelization of affine loop nests to execute them in parallel in a distributed computing infrastructure. This parallelization can also include the building of data blocks to increase task granularity in order to achieve a good execution performance. Moreover, AutoParallel is based on sequential programming and only contains a small annotation in the form of a Python decorator so that anyone with little programming skills can scale up an application to hundreds of cores.

This work has been published in [18] and is the result of a collaboration between Philippe Clauss, Cristian Ramon-Cortes, PhD student, and Rosa M. Badia, his PhD advisor, both from the Barcelona Supercomputing Center, Spain.

## 7.2. Optimization of recursive functions by transformation into loops

**Participants:** Salwa Kobeissi, Philippe Clauss.

Recursion is a fundamental computing concept that offers the opportunity to elegantly solve various kinds of problems, particularly those whose solutions depend on solutions of smaller instances of their own. Nevertheless, today in imperative languages, recursive functions are still not considered sufficiently time-efficient in comparison with the alternative equivalent iterative code. Although many advanced and aggressive optimizers have been developed to enhance the performance of iterative control structures, there are still no such sophisticated and advanced techniques built for the sake of optimizing recursions.

We propose an approach that makes possible applying powerful optimizations on recursive functions through transforming them into loops. We are particularly interested in applying polyhedral optimization techniques which usually tackle affine loops. Therefore, the scope of our study is restricted to recursive functions whose control flow and memory accesses exhibit an affine behavior, which means that there exists a semantically equivalent affine loop nest, candidate for polyhedral optimizations. Accordingly, our approach is based on analyzing early executions of a recursive program using a Nested Loop Recognition algorithm, performing the convenient recursion-to-iteration transformation of the original program and, finally, applying further loop optimizations using the polyhedral compiler Polly. This approach brings recursion optimization techniques into a higher level in addition to widening the scope of the polyhedral model to include originally non-loop programs.

This work is the topic of Salwa Kobeissi's PhD. A first paper has been submitted to an international workshop.

## 7.3. Impact Study of Data Locality on Task-Based Applications Through the Heteroprio Scheduler

**Participant:** Bérenger Bramas.

Task-based parallelization is massively used in high-performance computing on heterogeneous hardware because it allows programmers to finely describe the intrinsic parallelism of the algorithms while ignoring the hardware details. However, this approach delegates the main decisions to the scheduler, making it a critical component responsible for the distribution of the tasks on the different types of processing unit. In a former work, Bérenger Bramas has proposed the Heteroprio scheduler, which has demonstrated to be extremely efficient in the computation of the fast multipole method or linear algebra factorizations/decompositions. However, the original version was not taking into account data locality leading to loss of execution efficiency from important data movements between the memory nodes.

The current work aimed at improving the Heteroprio scheduler by making it locality sensitive. The idea is to divide the task-lists to have as many lists as there are memory nodes. Then, the two main issues are to find where to store the new ready tasks and to decide how to iterate over all the task-lists. For the first problem, we have studied different locality scores to find the best memory node for each task, and we have demonstrated that taking into account the type of data access - read or write - allows for significant improvement. Concerning the iteration order, we have proposed to use a priority distance and a memory distance such that the tasks are stolen from memory nodes that are close but also that have opposite priorities.

All these ideas were implemented in a scheduler inside StarPU and have been validated on two applications: QrMumps from Alfredo Buttari (IRIT) and SpLDLT from Florent Lopez (Rutherford Appleton Laboratory, UK.). The performance study demonstrated the benefit of our approach with a significant improvement in terms of execution time and data movement. The executions were accelerated by 30% for QrMumps and 80% for SpLDLT. The results will now be written into a dedicated paper for publication.

## 7.4. Combining Locking and Data Management Interfaces

**Participants:** Jens Gustedt, Maxime Mogé, Mariem Saied, Daniel Salas.

### *7.4.1. Ordered Read-Write Locks*

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 [2] 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 6.6. In previous work it has demonstrated its efficiency for a large variety of platforms.

In the context of the thesis of Mariem Saied, a new domain specific language (DSL) has been completed that largely eases the implementation of applications with ORWL. In its first version it provides an interface for stencil codes. The approach allows to describe stencil codes quickly and efficiently, and leads to substantial speedups.

In the framework of the ASNAP project (see 9.1.2) we have used ordered read-write locks (ORWL) as a model to dynamically schedule a pipeline of parallel tasks that realize a parallel control flow of two nested loops; an outer *iteration* loop and an inner *data traversal* loop. Other than dataflow programming we emphasize on upholding the sequential modification order of each data object. As a consequence the visible side effects on any object can be guaranteed to be identical to a sequential execution. Thus the set of optimizations that are performed are compatible with C's abstract state machine and compilers could perform them, in principle, automatically and unobserved. See [19] for first results.

In the context of the Prim'Eau project (see 9.1.1) we use ORWL to integrate parallelism into an already existing `Fortran` application that computes floods in the region that is subject to the study. A first step of such a parallelization has been started by using ORWL on a process level. Our final goal will be to extend it to the thread level and to use the application structure for automatic placement on compute nodes.

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.

### *7.4.2. Low level locks*

Our low level locks algorithm that is based on atomics and Linux' futexes [25] [26] has been integrated into the `musl` C library (see Section 6.7) and is thus deployed in several Linux distributions that use musl as their base.

## 7.5. 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. On recent multi-core hardware, performance of this code is often limited by memory bandwidth. We describe a multi-core PIC algorithm that achieves close-to-minimal number of memory transfers with the main memory, while at the same time exploiting SIMD instructions for numerical computations and exhibiting a high degree of OpenMP-level parallelism. Our algorithm keeps particles sorted by cell at every time step, and represents particles from the same cell using a linked list of fixed-capacity arrays, called chunks. Chunks support either sequential or atomic insertions, the latter being used to handle fast-moving particles. To validate our code, called Pic-Vert, we consider a 3d electrostatic Landau-damping simulation as well as a 2d3v transverse instability of magnetized electron holes. Performance results on a 24-core Intel Skylake hardware confirm the effectiveness of our algorithm, in particular its high throughput and its ability to cope with fast moving particles. A paper describing this work was published at Euro-par [13] and is described in more details in Yann Barsamian's PhD thesis [6].

## 7.6. 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 recently 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 [22], 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 have been accepted for publication at PPoPP'19.

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 published at PLDI'18 [12].

## 7.7. Program verification and formal languages

**Participant:** Arthur Charguéraud.

- 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 describing this work was published at ESOP'18 [15].

- A. Charguéraud, together with Ralf Jung and Jan-Oliver Kaiser and Derek Dreyer (MPI-SWS), Robbert Krebbers (Delft University of Technology), Jacques-Henri Jourdan (Inria), Joseph Tassarotti (Carnegie Mellon University), and Amin Timany (KU Leuven), developed MoSel, a general and extensible Coq framework for carrying out separation-logic proofs mechanically using an interactive proof assistant. This tool extends the Iris Proof Mode (IPM) to make it applicable to both affine and linear separation logics (and combinations thereof), and to provide generic tactics that can be easily extended to account for the bespoke connectives of the logics with which it is instantiated. To demonstrate the effectiveness of MoSeL, the tool has been instantiated to provide effective tactical support for interactive and semi-automated proofs in six very different separation logics. This work was published at ICFP'18 [17].
- A. Charguéraud advised Ramon Fernandez for a 4-month internship. The aim of that internship was to formalize, using the Coq proof assistant, several data layout transformations such as the transformation from an array of structures to a structure of arrays (AoS-to-SoA). Such transformations are routinely employed to develop high-performance code. Ramon investigated the literature on data layout transformations, listed the most useful transformations exploited in practice, and identified several core transformations from which almost all others can be derived. He then successfully carried out proofs of semantic preservation for the three most important transformations: field grouping, tiling, and AoS-to-SoA.
- 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* [24]. 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 appeared at the international conference Web Programming [14].

## 7.8. Flexible Runtime System with High Throughput for Many-to-Many Data Stream Problems

**Participants:** Paul Godard, Vincent Loechner, Cédric Bastoul.

In the context of our collaboration with the Caldera company, we are interested in high throughput data stream problems, that require low latency, maximal bandwidth usage, and that avoid starvations. We suppose that we receive jobs from an external system through a queue, each job including a description of its computation needs, output data requirements and output locations.

The computations are distributed on a cluster organized in a many-to-many logical topology where one or many computing tasks (producers) send data to one or many consumer tasks (consumers). The runtime system is orchestrated by a centralized scheduler, which decomposes jobs into tasks and dynamically assigns them to producers. The producers perform the computations and send their output data to the consumers. The consumers collect and order output data to make them available to the final user.

We implemented our framework, and performed some experiments on a real-world use case: real time professional digital printing, that may require tens of Gbit/s sustained output rates. We show in our measurements that our system scales and reaches data rates that are close to the maximum throughput of our experimental hardware. The architecture as a cluster and using the standard TCP/IP network protocol allow our system to be highly adaptive to the user's requirements. We are in the process of writing a paper describing our framework architecture for many-to-many data stream problems and results.

## 7.9. Visual Program Manipulation in the Polyhedral Model

**Participants:** Cédric Bastoul, Oleksandr Zinenko, Stéphane Huot.

While a plethora of libraries and frameworks focus on expressing parallelism, identifying and extracting it remains a challenging task. Automatic parallelization relies on imprecise heuristics resulting in cumbersome manual code analysis and transformation in case of underperformance. Alternatively, directive-based approaches often require transforming the program from scratch when a slightly modified version of an automatically-computed transformation would suffice. We propose an interactive visual approach building on the polyhedral model that (1) visualizes exact dependences and parallelism, (2) decomposes a complex automatically-computed transformation into simple steps for replay and easier modification, and (3) allows for directly manipulating the visual representation as a means of transforming the program with immediate feedback. User studies suggest that our visualization is understood by experts and non-experts alike, and that it may favor an exploratory approach to transformation. Finally, an eye-tracking study suggests that programmers may resort to visualizations instead of code if visualizations are clearly efficient for a given task.

This is a joint work with PARKAS team at Inria Paris (contact: Oleksandr Zinenko) and MJOLNIR team at Inria Lille (contact: Stéphane Huot), published in TACO [10].

## 7.10. A language extension set to generate adaptive versions automatically

**Participants:** Maxime Schmitt, Cédric Bastoul.

A large part of the development effort of compute-intensive applications is devoted to optimization, i.e., achieving the computation within a finite budget of time, space or energy. Given the complexity of modern architectures, writing simulation applications is often a two-step workflow. Firstly, developers design a sequential program for algorithmic tuning and debugging purposes. Secondly, experts optimize and exploit possible approximations of the original program to scale to the actual problem size. This second step is a tedious, time-consuming and error-prone task. During this year, we investigated language extensions and compiler tools to achieve that task semi-automatically in the context of approximate computing. We identified the semantic and syntactic information necessary for a compiler to automatically handle approximation and adaptive techniques for a particular class of programs. We proposed a set of language extensions generic enough to provide the compiler with the useful semantic information when approximation is beneficial. We implemented the compiler infrastructure to exploit these extensions and to automatically generate the adaptively approximated version of a program. We conducted an experimental study of the impact and expressiveness of our language extension set on various applications.

These language extensions and the underlying compiler infrastructure are a significant output of collaboration with Inria Nancy - Grand Est team TONUS, specialized on applied mathematics (contact: Philippe Helluy), to bring models and techniques from this field to compilers. A paper presenting these extensions has been accepted to the OGST journal, targeting typical end-users.

# 8. Bilateral Contracts and Grants with Industry

## 8.1. Bilateral Contracts with Industry

### *8.1.1. Caldera*

Vincent Loechner and Cédric Bastoul are involved in a collaboration with the Caldera company (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. It also 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. Regional Initiatives

### 9.1.1. Idex Prim'Eau

**Participant:** Jens Gustedt [contact].

In the framework of the Prim'Eau project of the University of Strasbourg, we study surface runoff for hydrological periods of several days. We use an efficient domain decomposition method that we apply to a real world example of Mutterbach (Moselle) with geological and flood data from the years 1920, 1940 and 2017. As the time and memory usage for these computations is important, we aim to parallelize them.

### 9.1.2. ADT ASNAP

**Participants:** Philippe Clauss, Jens Gustedt, Maxime Mogé.

Philippe Clauss, Jens Gustedt and Maxime Mogé have been involved until August 2018 in the ADT Inria project ASNAP (Accélération des Simulations Numériques pour l'Assistance Peropératoire), in collaboration with the Inria team MIMESIS. The goal was to find opportunities in the SOFA simulation platform for applying automatic parallelization techniques developed by Camus. We have investigated two approaches. One approach uses memory behavior memoïzation to generate a parallel code made of independent threads at runtime.

### 9.1.3. ADT ALTO (ApoLlo TakeOff)

**Participants:** Philippe Clauss, Muthena Abdul Wahab.

The Apollo compilation platform [4] that is being developed in Camus, dedicated to speculative and dynamic optimization and parallelization of loop nests, is the achievement of many original advances in compilation algorithms, in extensions of the polyhedral model, in speculative parallelization and in dynamic optimization of programs. It is a library of implemented knowledge and a fertile ground for other advances and extensions : for instance, an extension of the polyhedral model for handling non-linear loops would not have been possible without Apollo. However, this software platform must continuously be maintained, improved and extended.

The ALTO project, which is a 2.5 years project started in August 2018, is devoted to strengthen Apollo's software implementation in several ways, thanks to the expert engineer who has been recruited for these goals, Matthew Wahab. The main goals are the following:

- making the programming code respecting the standard rules of open-source software;
- making Apollo more robust regarding cases where some inputs may yield extreme behaviors
- implementing required improvements and extensions, as inter-procedural analysis or memory behavior memoïzation.

## 9.2. National Initiatives

### 9.2.1. ANR AJACS

**Participant:** Arthur Charguéraud.

The AJACS research project is funded by the programme "Société de l'information et de la communication" of the ANR, from October 2014, until March 2019. http://ajacs.inria.fr/

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).

### *9.2.2. ANR Vocal*

**Participant:** Arthur Charguéraud.

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.

## 9.3. European Initiatives

### *9.3.1. FP7 & H2020 Projects*

#### *9.3.1.1. ERC Deepsea*

**Participant:** Arthur Charguéraud.

The Deepsea project is funded by ERC from June 2013 to May 2018. It aims at developing 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) 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, Camus team), who works part time on this project. Project website: http://deepsea.inria.fr/.

### *9.3.2. Collaborations with Major European Organizations*

Cristian Ramon-Cortes and Rosa M. Badia: Barcelona Supercomputing Center (Spain)
A Python module for automatic parallelization and distributed execution of affine loop nests

Raquel Lazcano and Eduardo Juárez Martínez: Universidad Politecnica de Madrid (Spain)
Integration of Apollo in the Cerbero dataflow framework for adaptive code generation.

## 9.4. International Initiatives

### *9.4.1. Inria International Partners*

#### *9.4.1.1. 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
- Indian Institute of Science (IIIS) Bangalore, India
- Barcelona Supercomputing Center, Barcelona, Spain

## 9.5. International Research Visitors

### *9.5.1. Visits of International Scientists*

Rachid Seghir (Maître de conférences A, University of Batna, Algeria) visited our team (June 16-23, 2018), to participate to the mid-thesis evaluation of Harenome Ranaivoarivony-Razanajato, and work with Vincent Loechner on our ongoing collaboration co-advising Toufik Baroudi.

### *9.5.2. Internships*

Toufik Baroudi is a PhD student under the supervision of Rachid Seghir at University of Batna (Algeria). He is co-advised by Vincent Loechner, and visiting our team as an intern for one year since November 2018, founded by the Algerian *Programme National Exceptionnel (PNE)*. His PhD defense is planned at the end of 2019.

# 10. Dissemination

## 10.1. Promoting Scientific Activities

### *10.1.1. Scientific Events Organization*

*10.1.1.1. Member of Organizing Committees*

Philippe Clauss organized the Special Session on Compiler Architecture, Design and Optimization (CADO) of the 16th International Conference on High Performance Computing & Simulation (HPCS 2018), June 2018, Orléans, France.

Cédric Bastoul co-organized HIP3ES 2018 (International Workshop on High Performance Energy Efficient Embedded Systems), in conjunction with the international conference HiPEAC 2018.

### *10.1.2. Scientific Events Selection*

*10.1.2.1. Member of Conference Program Committees*

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

Cédric Bastoul and Vincent Loechner are part of the program committee of HIP3ES (International Workshop on High Performance Energy Efficient Embedded Systems) in conjunction with the HiPEAC international conference.

Arthur Charguéraud was a member of the program committee for the Symposium on Implementation and Application of Functional Languages (IFL 2018).

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

*10.1.2.2. Reviewer*

Philippe Clauss has been reviewer for the following conference and workshop: the 2nd International Conference on Computer Science and Application Engineering (CSAE 2018), the International Workshop on Polyhedral Compilation Techniques (IMPACT 2018).
Jens Gustedt has been reviewer for CCGrid 2018.
Arthur Charguéraud has been reviewer for the 30th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2018).
Cédric Bastoul has been reviewer for: international conference on Compiler Construction 2018 (CC'2018), the International Workshop on Polyhedral Compilation Techniques (IMPACT 2018), the International Workshop on High Performance Energy Efficient Embedded Systems (HIP3ES 2018).

### *10.1.3. Journals*

*10.1.3.1. Member of Editorial Boards*

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

*10.1.3.2. Reviewer - Reviewing Activities*

Bérenger Bramas has been reviewer for the following journal: PPL (Parallel Processing Letters).
Philippe Clauss has been reviewer for the following journals: Engineering Computations, IEEE Transactions on Computers, Future Generation Computer Systems.
Jens Gustedt has been a reviewer for Discrete Applied Mathematics.
Arthur Charguéraud has been reviewer for LFMTP (Logical Frameworks and Meta Languages: Theory and Practice).
Cédric Bastoul has been reviewer for IEEE Transactions on Computers.

### *10.1.4. Invited Talks*

Philippe Clauss has been invited at the Dagstuhl Seminar dedicated to Loop Optimization, March 11-16, 2018. The title of his talk was: *The Polyhedral Model Beyond Static Compilation, Affine Functions and Loops*.

Vincent Loechner has been invited as a speaker at the plenary session of the *journées doctorales du laboratoire d'informatique de l'université de Batna* (Algeria), April 25-26 2018, for a talk entitled *Code Optimizations Using the Polyhedral Model*.

### *10.1.5. Scientific Expertise*

*10.1.5.1. 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 clarification report processing, the planning of future versions of the standard and in an subgroup that discusses the improvement of the C memory model.

He was the one of the main forces behind the elaboration of C17, the new version of the C standard that has finally been published by ISO in 2018 [21].

This work on the C programming language also gave rise to the proposal of a language extension, Modular C. It has been used for the implementation of an efficient toolbox for *higher order automatic differenciation*, `arbogast`, see [9], and for the implentation of the work presented in [19].

*10.1.5.2. Expertise*

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

### *10.1.6. Research Administration*

Jens Gustedt is head of the ICPS team for the ICube lab, and in that function a member of the directory committee of the lab. He is also a member of the local recruiting comission for phds and postdocs of the Inria Center Nancy — Grand Est.

Philippe Clauss and Cédric Bastoul are members of the *Collegium Sciences* of the University of Strasbourg, which is a group of representative scientists providing advice regarding the funding of projects.

Philippe Clauss is a member of the *Bureau du Comité des Projets* of the Nancy Grand Est Inria Center since November 2018. This group of scientists provide assistance to the Director of the Center regarding the recruitment of PhD or post-doc students and Engineers, the funding of projects and provide also some scientific expertise regarding actions of the Center.

## 10.2. Teaching - Supervision - Juries

### *10.2.1. Teaching*

Licence : Philippe Clauss, Architecture des ordinateurs, 18h, L2, Université de Strasbourg, France

Licence : Philippe Clauss, Bases de l'architecture informatique, 22h, L1, Université de Strasbourg, France

Master : Philippe Clauss, Compilation, 84h, M1, Université de Strasbourg, France

Master : Philippe Clauss, Système et programmation temps-réel, 37h, M1, Université de Strasbourg, France

Master : Philippe Clauss, Optimisation et transformations de codes, 31h, M1, Université de Strasbourg, France

Master : Bérenger Bramas, Compilation, 40h, M1, Université de Strasbourg, France

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

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

Licence : Vincent Loechner, responsable pédagogique de la licence professionnelle ARS, L3, Université de Strasbourg, France

Licence : Vincent Loechner, accompagnement et jury de VAE licence professionnelle ARS, L3, Université de Strasbourg, France

Licence : Vincent Loechner, administration système et internet, 40h, L3, Université de Strasbourg, France

Master : Vincent Loechner, langages interprétés, 34h, M1, Université de Strasbourg, France

Master : Vincent Loechner, OS embarqués, 30h, M2, Université de Strasbourg, France

Master : Vincent Loechner, calcul parallèle, 20h, , 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, 28h, Université de Strasbourg, France

Licence : Cédric Bastoul, Computer architecture, 92h, L1, Université de Strasbourg, France

Licence : Cédric Bastoul, Parallel programming, 20h, L3, Université de Strasbourg, France

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

Master : Cédric Bastoul, Introduction to Research, 10h, L2+M1, Université de Strasbourg, France

Licence : Éric Violard, Modèles de Calcul, 29h, L1, Université de Strasbourg, France

Licence : Éric Violard, Programmation fonctionnelle, 85h, L2, Université de Strasbourg, France

Licence : Éric Violard, Architecture des ordinateurs, 54h, L2, Université de Strasbourg, France

Licence : Éric Violard, Logique et programmation logique, 27h, L2, Université de Strasbourg, France

Licence : Éric Violard, Systèmes concurrents, 9h, L3, Université de Strasbourg, France

Licence : Éric Violard, Algorithmique et structures de données, 39h, L3, Université de Strasbourg, France

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

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

### *10.2.2. Supervision*

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

PhD: Mariem Saied, *Automatic Code Generation for Multi-Dimensional Stencil Computations on Distributed-Memory Architectures*, Sep. 2018, Jens Gustedt and Gilles Muller.

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

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

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

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

PhD: Yann Barsamian, *Pic-Vert: A Particle-in-Cell Implementation for Multi-Core Architectures*, Université de Strasbourg, 31 Oct. 2018, Éric 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.

### *10.2.3. Juries*

Philippe Clauss participated to the following PhD committees in 2018:

| **Date** | **Candidate** | **Place** | **Role** |
|---|---|---|---|
| Apr. 25 | Mohamed Said MOSLI BOUKSIAA | Université de Paris-Saclay | Reviewer |
| Nov. 26 | Adilla SUSUNGI | University de Paris Sciences et Lettres | Reviewer |

Cédric Bastoul participated to the following PhD committees in 2018:

| **Date** | **Candidate** | **Place** | **Role** |
|---|---|---|---|
| Sep. 25 | Mariem Said | Université de Strasbourg | President |
| Dec. 13 | Jie Zhao | École Normale Supérieure | Reviewer |

## 10.3. Popularization

### *10.3.1. Articles and contents*

Jens Gustedt is blogging about efficient programming, in particular about the C programming language. To popularize the development of the future C2x standard he has been interviewed for infoQ. He also is an active member of the stackoverflow community a technical Q&A site for programming and related subjects.

### *10.3.2. Education*

A. Charguéraud is a co-organizer of the *Concours Castor informatique*. The purpose of the Concours Castor in to introduce pupils (from *CM1* to *Terminale*) to computer sciences. More information on: http://castor-informatique.fr/.

### *10.3.3. Interventions*

Cédric Bastoul prepared activities and participated to *Fête de la Science* at University of Strasbourg in October 2018.

# 11. Bibliography

## Major publications by the team in recent years

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

[2] P.-N. CLAUSS, J. GUSTEDT. *Iterative Computations with Ordered Read-Write Locks*, in "Journal of Parallel and Distributed Computing", 2010, vol. 70, n° 5, pp. 496–504 [*DOI :* 10.1016/J.JPDC.2009.09.002], https://hal.inria.fr/inria-00330024

[3] A. KETTERLIN, P. CLAUSS. *Profiling Data-Dependence to Assist Parallelization: Framework, Scope, and Optimization*, in "MICRO-45, The 45th Annual IEEE/ACM International Symposium on Microarchitecture", Vancouver, Canada, December 2012, https://hal.inria.fr/hal-00780782

[4] J. M. MARTINEZ CAAMAÑO, 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° 15 [*DOI :* 10.1002/CPE.4192], https://hal.inria.fr/hal-01581093

[5] A. SUKUMARAN-RAJAM, P. CLAUSS. *The Polyhedral Model of Nonlinear Loops*, in "ACM Transactions on Architecture and Code Optimization", January 2016, vol. 12, n° 4 [*DOI :* 10.1145/2838734], https://hal.inria.fr/hal-01244464

## Publications of the year

### Doctoral Dissertations and Habilitation Theses

[6] Y. A. BARSAMIAN. *Pic-Vert: A Particle-in-Cell Implementation for Multi-Core Architectures*, Université de Strasbourg, October 2018, https://hal.archives-ouvertes.fr/tel-01940700

### Articles in International Peer-Reviewed Journals

[7] Y. A. BARSAMIAN, J. BERNIER, S. A. HIRSTOAGA, M. MEHRENBERGER. *Verification of 2D × 2D and two-species Vlasov-Poisson solvers*, in "ESAIM: Proceedings and Surveys", 2018, vol. 63, pp. 78-108, https://hal.archives-ouvertes.fr/hal-01668744

[8] Y. A. BARSAMIAN, S. A. HIRSTOAGA, E. VIOLARD. *Efficient Data Layouts for a Three-Dimensional Electrostatic Particle-in-Cell Code*, in "Journal of computational science", 2018, vol. 27, pp. 345–356, https://hal.archives-ouvertes.fr/hal-01664207

[9] I. CHARPENTIER, J. GUSTEDT. *Arbogast: Higher order automatic differentiation for special functions with Modular C*, in "Optimization Methods and Software", February 2018, pp. 1-25, https://hal.inria.fr/hal-01307750

[10] O. ZINENKO, S. HUOT, C. BASTOUL. *Visual Program Manipulation in the Polyhedral Model*, in "ACM Transactions on Architecture and Code Optimization", March 2018, vol. 15, n° 1, pp. 1 - 25 [*DOI :* 10.1145/3177961], https://hal.inria.fr/hal-01744426

### International Conferences with Proceedings

[11] U. A. ACAR, V. AKSENOV, A. CHARGUÉRAUD, M. RAINEY. *Provably and Practically Efficient Granularity Control*, in "PPoPP 2019 - Principles and Practice of Parallel Programming", Washington DC, United States, February 2019 [*DOI :* 10.1145/3293883.3295725], https://hal.inria.fr/hal-01973285

[12] U. A. ACAR, A. CHARGUÉRAUD, A. GUATTO, M. RAINEY, F. SIECZKOWSKI. *Heartbeat scheduling: provable efficiency for nested parallelism*, in "PLDI'18 - 39th ACM SIGPLAN Conference on Programming Language Design and Implementation", Philadelphia, United States, ACM Press, June 2018 [*DOI :* 10.1145/3192366.3192391], https://hal.inria.fr/hal-01937946

[13] Y. A. BARSAMIAN, A. CHARGUÉRAUD, S. A. HIRSTOAGA, M. MEHRENBERGER. *Efficient Strict-Binning Particle-in-Cell Algorithm for Multi-Core SIMD Processors*, in "Euro-Par 2018 - 24th International European Conference on Parallel and Distributed Computing", Turin, Italy, August 2018, https://hal.archives-ouvertes.fr/hal-01890318

[14] A. CHARGUÉRAUD, A. SCHMITT, T. WOOD. *JSExplain: A Double Debugger for JavaScript*, in "The Web Conference 2018", Lyon, France, April 2018, pp. 1-9 [*DOI :* 10.1145/3184558.3185969], https://hal.inria.fr/hal-01745792

[15] A. GUÉNEAU, A. CHARGUÉRAUD, F. POTTIER. *A Fistful of Dollars: Formalizing Asymptotic Complexity Claims via Deductive Program Verification*, in "ESOP 2018 - 27th European Symposium on Programming", Thessaloniki, Greece, A. AHMED (editor), LNCS - Lecture Notes in Computer Science, Springer, April 2018, vol. 10801, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018 [*DOI :* 10.1007/978-3-319-89884-1_19], https://hal.inria.fr/hal-01926485

[16] S. KOBEISSI, A. UTAYIM, M. JABER, Y. FALCONE. *Facilitating the Implementation of Distributed Systems with Heterogeneous Interactions*, in "IFM 2018 - 14th International Conference on integrated Formal Methods", Maynooth, Ireland, September 2018, pp. 1-19, https://hal.inria.fr/hal-01868748

### Conferences without Proceedings

[17] R. KREBBERS, J.-H. JOURDAN, R. JUNG, J. TASSAROTTI, J.-O. KAISER, A. TIMANY, A. CHARGUÉRAUD, D. DREYER. *MoSeL: a general, extensible modal framework for interactive proofs in separation logic*, in "International Conference on Functional Programming (ICFP 2018)", St Louis, MO, United States, Proceedings of the ACM on Programming Languages, ACM, September 2018, vol. 2, n$^{o}$ ICFP, 77 p. [*DOI :* 10.1145/3236772], https://hal.archives-ouvertes.fr/hal-01898522

[18] C. RAMON-CORTES, R. AMELA, J. EJARQUE, P. CLAUSS, R. BADIA. *AutoParallel: A Python module for automatic parallelization and distributed execution of affine loop nests*, in "PyHPC 2018 - 8th Workshop on Python for High-Performance and Scientific Computing", Dallas, TX, United States, November 2018, https://hal.inria.fr/hal-01936351

### Research Reports

[19] J. GUSTEDT, M. MOGÉ. *Memory access classification for vertical task parallelism*, Inria Nancy - Grand Est, June 2018, n$^{o}$ RR-9182, pp. 1-20, https://hal.inria.fr/hal-01814740

### Other Publications

[20] T. Coq Development Team. *The Coq Proof Assistant, version 8.8.0*, April 2018, Software [*DOI :* 10.5281/zenodo.1219885], https://hal.inria.fr/hal-01954564

## References in notes

[21] JTC1/SC22/WG14 (editor). *Programming languages - C*, ISO, 2018, n° ISO/IEC 9899

[22] 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

[23] 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

[24] 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

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

[26] J. Gustedt. *Futex based locks for C11's generic atomics (extended abstract)*, in "The 31st Annual ACM Symposium on Applied Computing", Pisa, Italy, April 2016 [*DOI :* 10.1145/2851613.2851956], https://hal.inria.fr/hal-01304108

[27] M. Hall, D. Padua, K. Pingali. *Compiler research: the next 50 years*, in "Commun. ACM", 2009, vol. 52, n° 2, pp. 60–67, http://doi.acm.org/10.1145/1461928.1461946

[28] A. Hobor, A. W. Appel, F. Z. Nardelli. *Oracle Semantics for Concurrent Separation Logic*, in "ESOP", 2008, pp. 353-367