RESEARCH CENTRE

Inria Centre at Université de Lorraine

IN PARTNERSHIP WITH:

Université de Strasbourg

2023 ACTIVITY REPORT

Project-Team CAMUS

Compilation for multi-processor and multi-core architectures

IN COLLABORATION WITH: ICube

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Inria

Contents

Project-Team CAMUS

Creation of the Project-Team: 2023 October 01

Keywords

Computer sciences and digital sciences

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

A2.4. – Formal method for verification, reliability, certification

Other research topics and application domains

B4.5.1. – Green computing

B6.1.1. – Software engineering

B6.6. – Embedded systems

1 Team members, visitors, external collaborators

Research Scientists

- Bérenger Bramas [INRIA, Researcher]
- Arthur Charguéraud [INRIA, Researcher, HDR]
- Jens Gustedt [INRIA, Senior Researcher, HDR]

Faculty Members

- Philippe Clauss [Team leader, UNIV STRASBOURG, Professor, HDR]
- Cédric Bastoul [UNIV STRASBOURG, Professor, HDR]
- Stephane Genaud [UNIV STRASBOURG, Professor, HDR]
- Alain Ketterlin [UNIV STRASBOURG, Associate Professor]
- Vincent Loechner [UNIV STRASBOURG, Associate Professor]
- Eric Violard [UNIV STRASBOURG, Associate Professor, HDR]

Post-Doctoral Fellows

- Marek Felsoci [INRIA, Post-Doctoral Fellow, from Apr 2023]
- Thomas Koehler [INRIA, Post-Doctoral Fellow]
- Jean Etienne Ndamlabin Mboula [INRIA, Post-Doctoral Fellow]

PhD Students

- Ugo Battiston [INRIA, from Oct 2023]
- Guillaume Bertholon [ENS Paris]
- Raphael Colin [UNIV STRASBOURG, from Oct 2023]
- Clément Flint [UNIV STRASBOURG, ATER]
- Clément Rossetti [UNIV STRASBOURG]
- Anastasios Souris [INRIA]
- Hayfa Tayeb [INRIA, until Sep 2023]
- Arun Thangamani [UNIV STRASBOURG]

Technical Staff

- Raphael Colin [UNIV STRASBOURG, Engineer, until Sep 2023]
- Thai Hoa Trinh [ORANGE, Engineer, from Mar 2023]

Interns and Apprentices

- Julien Gaupp [INRIA, Intern, from Oct 2023]
- Tom Hammer [UNIV STRASBOURG, Intern, until Aug 2023]
- Alexis Hamon [INRIA, Intern, from May 2023 until Jul 2023]
- Atoli Huppé [UNIV STRASBOURG, Intern, from Jun 2023 until Nov 2023]
- Rémy Kimbrough [ENS PARIS, Intern, from Jun 2023 until Jul 2023]
- Mathis Pernias [INRIA, Intern, from Jun 2023 until Aug 2023]

Administrative Assistant

- Ouiza Herbi [INRIA]

2 Overall objectives

The CAMUS team is focusing on developing, adapting and extending automatic and semi-automatic parallelization and optimization techniques, as well as proof and certification methods, for accelerating applications with the efficient use of current and future multi-processor and multicore hardware platforms.

The team's research activities are organized into three main axes which are: (1) semi-automatic and assisted code optimization, (2) fully-automatic code optimization, and (3) fundamental algorithms and mathematical tools. Axes (1) and (2) include two sub-axes each: (1.1) interactive program transformation, (1.2) new language constructs, (2.1) runtime systems and dynamic analysis & optimization, and (2.2) static analysis & optimization. Every axis may include some activities related to interdisciplinary collaborations focusing on high performance computing.

3 Research program

While trusted and fully automatic code optimizations are generally the most convenient solutions for developers, the growing complexity of software and hardware obviously impacts their scope and effectiveness. Although fully automatic techniques can be successfully applied in restricted contexts, it is often beneficial to let expert developers make some decisions on their own. Moreover, some expert knowledge, contextual requirements, and hardware novelties cannot be immediately integrated into automatic tools.

Thus, beside automatic optimizers that play undoubtedly an important role, semi-automatic optimizers providing helpful assistance to expert developers are also essential for reaching high performance. Note that such semi-automatic tools must ideally invoke fully automatic sub-parts, including dependence analyzers, code generators, correctness checkers or performance evaluators, in order to save the user from the burden of these tasks and expand the scope of the tools. Fully automatic tools may either be used as standalone solutions, when targeting the corresponding restricted codes, or used as satellite tools for semi-automatic environments. Fully automatic mechanisms are the elementary pieces of any more ambitious semi-automatic optimizing tool.

CAMUS' main research axes are depicted in Figure 1. Semi-automatic methods for code optimization will be implemented either as interactive transformation tools, or as language extensions allowing users to control the way programs are transformed. Both approaches will be supported by fully automatic processes devoted to baseline code analysis and transformation schemes. Such schemes may be either static, i.e. applied at compile-time, or dynamic, i.e. applied while the target code runs. Note that these characteristics are not mutually exclusive: one optimization process may include simultaneously a static and a dynamic part. Note also that the invoked fully automatic processes may be very ambitious frameworks on their own, as for instance implementing advanced speculative optimization strategies.

Strong advances in code analysis and transformation are often due to fundamental algorithms and mathematical tools, that enable the extraction of important properties of programs, through a

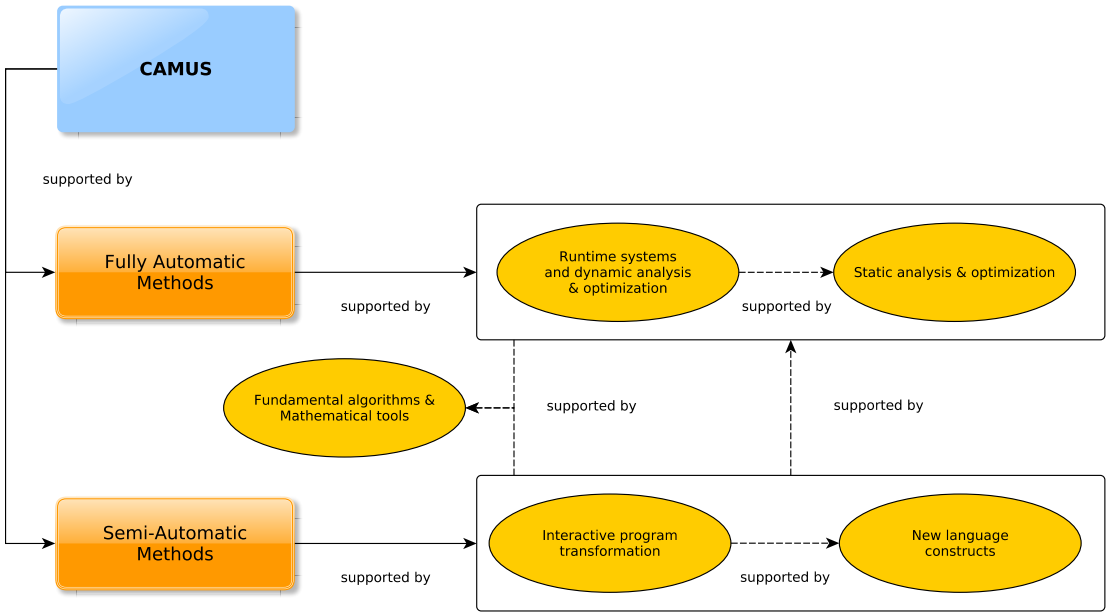

Figure 1: General view of CAMUS' research objectives.

constructive conceptual modeling. We believe that the investment in core mathematics and computer science research must be permanent in the following directions:

- Mathematics are obviously a great pool of modeling and computing methods that may have a high impact in the field of program analysis and transformation. Additionally, mathematical results must be adapted and transformed into algorithms which are usable for our purpose. This task may require some mathematical extensions and the creation of fast and reliable algorithms and implementations.

- Some new contexts of use require the conception of new algorithms dedicated to well-known fundamental and essential tasks. For instance, many standard code analysis and transformation algorithms, originally developed to be exclusively used at compile-time, need to be revised to be used at runtime. Indeed, their respective execution times may not be acceptable when analyzing and optimizing code on-the-fly. The time-overhead must be dramatically lowered, while the ambitions may be adjusted to the new context. Typically, "optimal" solutions resulting from time-consuming computations may not be the final goal of runtime optimization strategies. Sub-optimal solutions may suffice, since the performance of a dynamically optimized code includes the time overhead of the runtime optimization process.

- It is always useful to identify a restricted class of programs where very efficient optimizations may be applied. Such a restricted class usually takes advantage of an accurate model. Conversely, it may also be fruitful to target the removing of some restrictions regarding the class of programs that are candidate for efficient optimizations.

- Other scientific disciplines may also provide fundamental strategies to track code optimization issues. However, they may also require some prior adaptation. For instance, machine learning techniques are more and more considered in the area of code optimization.

Collaborations with researchers whose applications require high performance will be developed. Beside offering our expertise, we will especially use their applications as an inspiration for new developments of optimization techniques. Those colleagues from other teams will also play the role of beta testers for our semi-automatic code optimizers. Most research axes of CAMUS will include such collaborations. The local scientific environment is particularly favorable to the setting of interactions. For example, we

participate in the inter-disciplinary institute IRMIA++ of the University of Strasbourg, that facilitates collaborations with mathematicians developing high performance numerical simulations.

3.1 Semi-automatic and assisted code optimization

Programming languages as they are used in modern compute intensive software are relatively poor in their possibilities to describe all known properties of a particular code. On the one hand, a language construct may *over-specify* the semantics of the program, for example imposing a specific execution order for the iterations of a loop whereas any order would have been correct. On the other hand, a language construct may *under-specify* the semantics of the program, for example lacking the ability to describe the fact that two pointers must be distinct, or that a given integer value is always less than a small constant.

Modern tools that rewrite code for optimization, be it internally as optimizing compiler passes or externally as source-to-source transformations, miss a lot of opportunities for the programmer to annotate and integrate their knowledge of the code. As a consequence fully-automatic tools, are not easily brought to their full capacity and one-shot platform-specific programmer intervention is required.

To advance this field, we will develop re-usable and traceable features that provide the ability for programmers to specify and control code transformations and to annotate functional interfaces and code blocks with all the meta-knowledge they have.

3.2 Fully-automatic code optimization

We will focus on two main code optimization and parallelization approaches: the polyhedral model based on a geometrical representation and transformation of loops, and task-based model based on a runtime resolution of the dependencies between the tasks. Note that these two approaches can potentially be mixed.

The polyhedral model is a great source of new developments regarding fundamental mathematical tools dedicated to code analysis and transformation. This model was originally exclusively based on linear algebra. We have proposed in the past some extensions to polynomials, and we are currently investigating extensions to algebraic expressions. In the meantime, we also focus on runtime approaches that allow polyhedral-related techniques to be applied to codes that are not usually well-suited candidates. The motivation of such extensions is obviously to propose new compilation techniques with enlarged scope and better efficiency, that are either static, i.e, applied at compile-time, or dynamic, i.e., applied at runtime.

We will also keep studying the task-based method which is complementary to the polyhedral model, and beneficial in scenarios that are not adapted to the polyhedral model. For example, this method can work when the description of the parallelism is entirely performed at runtime, and it is able to parallelize sections with arbitrary structures (i.e., not necessarily loop nests).

In our project, we attempt to bridge the gap between the task-based method and the compiler by designing a novel automatic parallelization mechanism with static source-to-source transformations. We also work on improving the scheduling strategies or the description of the parallelism by designing speculative execution models that operate at runtime.

3.3 Fundamental algorithms & mathematical tools

Regarding our fundamental and theoretical studies, we plan to focus on three main topics: (1) Trahrhe expressions [12], (2) mechanized metatheory and interactive program verification and (3) programmable polyhedral scheduling.

4 Application domains

High performance computing plays a crucial role in the resolution of important problems of science and industry. Additionally, software development companies, and software developers in general, are strongly constrained by the time-to-market issue, while facing growing complexities related to hardware and correctness of the developed programs. Computers become more and more powerful by integrating

numerous and specialized processor cores, and programs taking advantage of such hardware are more and more exposed to correctness issues.

Our goal is to provide automatic and semi-automatic tools that will significantly lower the burden on developers. By ensuring a secured production of correct and well-performing software, developers can mostly concentrate on the implemented functionalities, and produce quality software in reasonable time.

Our scientific proposals are most of the time supported by a related developed software, or an extension of an existing software. Its role is to highlight the automation of the proposed analysis and optimization techniques, to highlight their effectiveness by exhibiting performance improvements on baseline benchmark programs, and to facilitate their application on any program that would be targeted by some potential users. Thus, our software tools must be made as accessible as possible for users of science and industry, for experimenting the implemented optimization procedures with their specific programs. As such, we usually propose a free non-commercial use, through an open-source software licence. While the software is made available in a shape that allows for its use in full autonomy, we expect interested users to contact us for some deeper exchanges related to their specific goals. Such exchanges may be the start of some fruitful collaborations. Publishing our proposals in top rated conferences and journals may obviously also result in a effective impact for their adoption and the use of the related software.

Our proposals in analysis and optimization techniques of programs may find interested users in many international companies, from semi-conductor industry actors, like ARM, SiPearl or STMicroelectronics, to big companies developing high performance or deep learning applications. At a national or local level, any company whose innovative developments require compute or data intensive applications, like Nyx, or dedicated support tools, like Atos, may be interested in our work, and potentially collaborate with us for more specific and dedicated research. Since the project-team is hosted by the University of Strasbourg, contacts with many local companies are made easier thanks to the hiring of former students, and to their involvement in teaching duties and supervision of internship students.

5 Social and environmental responsibility

5.1 Footprint of research activities

We have largely decreased the number of physical meetings, opting for video-conference meetings when possible. We have also favored travel by train rather than by plane in the past year.

5.2 Impact of research results

Regarding the significant impact on energy consumption and related carbon emission of numerical applications, and particularly of high performance computing, every research project of the team will include from now on the important goal of energy efficiency for the generated codes. The optimizing mechanisms that we propose in our research will be evaluated with energy and time performance, both considered at the same priority level.

6 Highlights of the year

The Camus team takes part of the French national research program dedicated to exascale computing, named PEPR NumPEx, and more precisely of its project Exa-SofT focusing on HPC software and tools. Philippe Clauss is co-leader of the work package devoted to just-in-time code optimization with continuous feedback loop.

Arthur Charguéraud released the second edition of his all-in-Coq book entitled *Foundations Separation Logic*. This book is edited as Volume 6 of the Software Foundations series. This second edition features simplified proofs, leveraging a novel construction of Separation Logic for nondeterministic sequential languages based on the technique of *omni-big-step semantics* [8]. It also benefits from numerous presentation improvements and additional exercises. Furthermore, it is accompanied with companion course notes.

7 New software, platforms, open data

7.1 New software

7.1.1 TRAHRHE

Name: Trahrhe expressions and applications in loop optimization

Keywords: Polyhedral compilation, Code optimisation, Source-to-source compiler

Functional Description: This software includes a mathematic kernel for computing Trahthe expressions related to iteration domains, as well as extensions implementing source-to-source transformations of loops for applying optimizations based on Trahrhe expressions.

News of the Year: Useless computations when dichotomy search is selected have been removed for the trahrhe functions inside the generated header file.

URL: https://webpages.gitlabpages.inria.fr/trahrhe

Publications: hal-04379037, hal-03944790, hal-02425752, hal-01581081

Contact: Philippe Clauss

Participants: Philippe Clauss, Maxime Drouhin

7.1.2 CFML

Name: 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.

News of the Year: In 2023, CFML has been migrated to github and equipped with a new compilation infrastructure.

URL: http://www.charguer aud.org/softs/cfml/

Contact: Arthur Charguéraud

Participants: Arthur Charguéraud, Armaël Guéneau, François Pottier

7.1.3 openCARP

Name: Cardiac Electrophysiology Simulator

Keyword: Cardiac Electrophysiology

Functional Description: openCARP is an open cardiac electrophysiology simulator for in-silico experiments. Its source code is public and the software is freely available for academic purposes. openCARP is easy to use and offers single cell as well as multiscale simulations from ion channel to organ level. Additionally, openCARP includes a wide variety of functions for pre- and post-processing of data as well as visualization.

URL: https://opencarp.org/

Publications: hal-04206195, hal-03977688

Contact: Vincent Loechner

Participants: Arun Thangamani, Stephane Genaud, Bérenger Bramas, Tiago Trevisan Jost, Raphael Colin

Partner: Karlsruhe Institute of Technology

7.1.4 SPECX

Name: SPEculative eXecution task-based runtime system

Keywords: HPC, Parallelization, Task-based algorithm

Functional Description: Specx (previously 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.

News of the Year: Since 2023, Specx supports GPUs (CUDA/Hip).

URL: https://gitlab.inria.fr/bramas/specx

Contact: Bérenger Bramas

7.1.5 Autovesk

Keywords: HPC, Vectorization, Source-to-source compiler

Functional Description: Autovesk is a tool to produce vectorized implementation from static kernels.

News of the Year: The tool is described in "Autovesk: Automatic vectorization of unstructured static kernels by graph transformations" ACM TACO, 2023.

URL: https://gitlab.inria.fr/bramas/autovesk

Contact: Bérenger Bramas

7.1.6 Farm-SVE

Keywords: Vectorization, ARM

Functional Description: Naive/scalar implementation of the ARM C language extensions (ACLE) for the ARM Scalable Vector Extension (SVE) in standard C++.

News of the Year: Several minor improvements have been added in 2023.

URL: https://gitlab.inria.fr/bramas/farm-sve

Contact: Bérenger Bramas

7.1.7 TBFMM

Keywords: FMM, OpenMP, C++

Functional Description: TBFMM is a Fast Multipole Method (FMM) library parallelized with the task-based method. It is designed to be easy to customize by creating new FMM kernels or new parallelization strategies. It uses the block-tree hierarchical data structure (also known as the group-tree), which is well-designed for the task-based parallelization, and now supports heterogeneous architectures . Users can implement new FMM kernels, new types of interacting elements or even new parallelization strategies. As such, it can be used as a simulation toolbox for scientists in physics or applied mathematics. It enables users to perform simulations while delegating the data structure, the algorithm and the parallelization to the library. Besides, TBFMM can also provide an interesting use case for the HPC research community regarding parallelization, optimization and scheduling of applications handling irregular data structures.

News of the Year: Since 2023, TBFMM has been ported to StarPU (and works with CPU/GPU).

URL: https://gitlab.inria.fr/bramas/tbfmm

Contact: Bérenger Bramas

7.1.8 SPC5

Name: SPC5

Keywords: Spmv, SIMD, OpenMP

Functional Description: SPC5 is a tool that helps with specific calculations involving large, sparse matrices (matrices with a lot of zero values) on two types of computer systems: ARM-SVE-based architectures (like the a64fx) and X86 AVX-512 architectures. It provides different methods and formats to store and process these matrices, making it easier to handle them depending on their structure and layout.

News of the Year: SPC5 has been ported to ARM SVE.

URL: https://gitlab.inria.fr/bramas/spc5

Contact: Bérenger Bramas

7.1.9 PolyLib

Name: 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. It 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.

Release Contributions: Continuous Integration process has been added to the latest version. The license has been moved from GPL to MIT.

URL: http://icps.u-strasbg.fr/PolyLib/

Contact: Vincent Loechner

Participant: Vincent Loechner

7.1.10 TLC

Name: 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.

News of the Year: This year, TLC has been upgraded to reflect on important changes in the Coq proof assistant, including the handling of hints and of decision procedures.

URL: http://www.chargueraud.org/softs/tlc/

Contact: Arthur Charguéraud

7.1.11 FormalMetaCoq

Keyword: Formal semantics

Functional Description: FormalMetaCoq consists of a library of Coq formalizations of programming language semantics and type soundness proofs.

News of the Year: FormalMetaCoq has been extended with formalization of omni-semantics.

URL: https://github.com/charguer/formalmetacoq

Contact: Arthur Charguéraud

7.1.12 OptiTrust

Name: OptiTrust

Keyword: Code optimisation

Functional Description: The OptiTrust framework provides programmers with means of optimizing their programs via user-guided source-to-source transformations.

News of the Year: OptiTrust has been extended with many features, including: a Separation Logic based type system, the validation of transformations, and two major case studies.

URL: http://optitrust.inria.fr

Contact: Arthur Charguéraud

8 New results

8.1 Static Parallelization and Optimization

8.1.1 Improvements of the C programming language

Participants: Jens Gustedt.

For the upcoming version of the C standard, C23, we contributed with a large number of proposals that are now integrated in the draft that has been presented by ISO JTC1/SC22 to the national bodies (NB). The final adjustments of the content according to more than 500 NB comments have been made in plenary online meetings Jan and Oct, 2023. For these integrations we coordinated the response for France' NB, AFNOR.

The final vote on the new C23 standard is due in the plenary meeting that will be hosted by us in Strasbourg in Jan 2024. We wrote or contributed to 22 of the about 50 major changes in that new revision. Our contributions include:

- adding new keywords such as `bool`, `static_assert`, `true`, `false`, `thread_local` that replace archaic spellings (such as `_Bool`) or had only been provided as macros;
- removing integer width constraints and obsolete sign representations (so-called "1's complement" and "sign-magnitude");
- removing support for function definitions with identifier lists, so-called K&R definitions;
- including the attributes `[[reproducible]]` and `[[unsequenced]]` for marking "pure" function types;
- adding the `constexpr` specifier for object definitions;
- adding a `nullptr` constant and a `nullptr_t` type;
- allowing Unicode identifiers following Unicode Standard Annex, UAX #31;
- adding an `unreachable` feature to mark dead code that can safely be removed;
- type inference via the `auto` type specifier

In our January 2024 we will also start to discuss additions to the next version of the C standard, coined C2y at the moment. We have published the following work which will be discussed, there.

- Extensions to the C preprocessor [23]
- Simple translation unit initialization and cleanup handling with dependencies, [24].
- Rephrasing and reassesment of the terms constant and literal, [25].
- A discussion on the future of imaginary types, in particular to remove them from the C standard, [26].
- A clarification for array length specifications and sizeof expressions, removing the permission to have side effects, [27].
- Initialization, allocation and effective type, [29].
- Adding syntax and terminology to allow forward declarations of array lengths in different contexts, [30, 28].

In addition to the C standard we continued our work for the technical specification TS 6010 (see for example [22]) for a sound and verifiable memory model that is based on provenance, that is on the unique attribution of a storage instance to any valid pointer value. Unfortunately, this TS has met a severe setback because of changed ISO policies that make it more and more difficult to work on programming languages in the context of their insufficient normalization framework.

To promote the new C standard, we also prepared a C23 edtion of the book Modern C, [5, 32]

8.1.2 Ionic Models Code Generation for Heterogeneous Architectures

Participants: Arun Thangamani, Raphaël Colin, Atoli Huppé, Vincent Loechner, Stéphane Genaud, Bérenger Bramas.

We participate in the research and development of a cardiac electrophysiology simulator in the context of the MICROCARD European project. The CAMUS team provides their optimizing compiler expertise to build a bridge from a high-level language convenient for ionic model experts (called EasyML) to a code that will run on future exascale supercomputers. We aim to generate multiple parallel versions of the ionic simulation to exploit the various parallel computing units that are available in the target architecture nodes (SIMD, multicore, GPU, etc.).

The frontend that we contributed to in 2023 is based on the MLIR extensible compiler. We developed a Python script generating MLIR code from an EasyML ionic model description and integrated it in the openCARP project (7.1.3). We have achieved the automatic vectorization of the ionic code and this work has been published and presented at the CGO '23 conference [13]. The MLIR code generation has been further extended for GPUs for both CUDA or ROCm targets. This work has been published and presented at the EuroPar'23 conference [14]. Each of these publications have an artifact that was validated by the community as being able to reproduce the published results.

Our ongoing work regards the handling of heterogeneous architectures to possibly exploit simultaneously CPU cores and mulitple GPUs. For this purpose, we rely on the StarPU runtime distributed environment. This is a joint work with members of the STORM team (Inria Bordeaux) also implied in the MICROCARD project.

8.1.3 Polyhedral Scheduling

Participants: Vincent Loechner, Stéphane Genaud, Arun Thangamani, Tom Hammer, Alain Ketterlin, Cedric Bastoul.

Scheduling is the central operation in the polyhedral compilation chain, to find the best execution order of loop iterations for parallelizing and optimizing the resulting code. Discovering the best polyhedral schedules remains a grand challenge reinforced by the need to build *affine* scheduling functions.

A first work on this topic focuses on allowing polyhedral compilation experts to produce highly customizable scheduling algorithms in an efficient manner, without having to develop from scratch a new Farkas lemma-based solver and the whole polyhedral compilation toolchain. It will be included in the open-source PeriScop suite (OpenScop, CLooG, PIPLib, etc.). We designed a specific mini-language (DSL) that allows the user to specify and prioritize its objectives. This was Tom Hammer's internship main work, presented at the IMPACT '23 workshop [18] collocated with the HiPEAC '23 conference.

Cédric Bastoul is still on leave, detached from the team and working at Qualcomm. However, the work he initiated on *superloop scheduling* is going on with the rest of the team. The main idea of superloop scheduling is to leverage basic-block-level instruction reordering compilation techniques, and then to agglomerate statement instances into "super" loops offered by modern trace analysis tools. We presented this new scheduling technique at the IMPACT '23 workshop [17]. This is the main topic of Tom Hammer's PhD, starting in September 2023.

Arun Thangamani, Vincent Loechner and Stéphane Genaud contributed to a third work in the field of polyhedral scheduling. We wrote a survey that analyses and compares the performance of various open-source polyhedral compilers on different architectures (vector single-core CPUs, multi-core CPUs, GPUs) using the polybench/C set of benchmarks. Our reproducible, very fine and low-level architecture performance analyses will ease the work of developing new and better scheduling algorithms. The survey has been submitted to a journal.

8.1.4 Automatic Task-Based Parallelization using Source to Source Transformations

Participants: Marek Felsoci, Bérenger Bramas, Stéphane Genaud.

We worked on a new approach to automatically parallelize any application written in an object-oriented language. The main idea is to parallelize a code as an HPC expert would do it using the task-based method. In a previous work, we created a new source-to-source compiler on top of Clang-LLVM called APAC. APAC is able to insert tasks in a source-code by evaluating data access and generating the correct dependencies. In the previous year, we reimplemented APAC from Clang to OptiTrust (developed in the team), where source code transformations can be written in a compact way. We are now evaluating and upgrading this new compiler.

8.1.5 Automatic vectorization of static computational kernels

Participants: Hayfa Tayeb, Bérenger Bramas.

Compilers can automatically vectorize codes if there are zero to few data transformations to perform. However, when memory accesses are not contiguous, non linear or even chaotic, compilers usually fail. This is why we proposed a new method where we transform the dependency graph into a graph made of vectorial instructions. Our different layers aim at minimizing the number of instructions, especially data transformation instructions, that are needed to get a vectorized code. To do so we have developed the Autovesk compiler, which converts scalar C++ code into a vectorized C++ equivalent. We tested the performance of our approach by vectorizing different types of kernels, and we showed that we outperform the GNU C++ compiler by a speedup factor of 4 on average and up to 11. This work is described in a publication [10].

8.1.6 Extending the task dataflow model with speculative data accesses

Participants: Anastasios Souris, Bérenger Bramas, Philippe Clauss.

In the context of the AUTOSPEC project, we created a novel approach for enhancing the task dataflow model. Our method integrates new access types into the model, which allows for speculative data accesses, thereby providing applications with speculation capabilities. We demonstrate the effectiveness of this model through two practical scenarios: the parallelization of a finite-state machine and Monte-Carlo simulations. These examples show significant performance benefits, illustrating the potential of our approach in realistic application settings. This work contributes to the field by offering a data-centric, version-based methodology for improving parallelism and efficiency in computing applications [16].

8.1.7 Algebraic Loop Tiling

Participants: Clément Rossetti, Alexis Hamon, Mathis Pernias, Philippe Clauss.

We are currently developing a new approach for loop tiling, where tiles are no longer characterized by the sizes of their edges, but by their volumes, i.e., the number of embedded iterations. Algebraic tiling is particularly dedicated to parallel loops, where load balancing among the parallel threads is crucial for reaching high runtime performance.

Rectangular tiles of quasi-equal volumes, but of different shapes, are obtained thanks to mathematical computations based on the inversion of Ehrhart polynomials. The Trahrhe software (see Section 7.1.1) is dedicated to such computations, and to the automatic generation of header files in C, that can be used for the runtime computation of the algebraic tile bounds.

Clément Rossetti has started his related PhD work in October 2022. Integration of algebraic loop tiling into the source-to-source polyhedral compiler Pluto has been initiated by Clément. Any Pluto user will get access to automatic algebraic tiling thanks to the new Pluto flag `-atiling`.

A paper presenting for the first time the algebraic loop tiling technique has been presented at IMPACT 2023 [12].

The non-constant size of the algebraic tiles makes the skewing of the tiles statically intractable. In the last months, some first solutions have been proposed when a skewing transformation of the tiles is required for parallelization. A paper discussing this issue will be presented at IMPACT 2024 [19].

8.2 Profiling and Execution Behavior Modeling

8.2.1 Multi-GPU parallelization of the Lattice Boltzmann Method

Participants: Clément Flint, Bérenger Bramas, Stéphane Genaud, Atoli Huppé.

In the context of the ITI IRMIA++, we work on the parallelization of the Lattice Boltzmann Methodology (LBM), a general framework for constructing efficient numerical fluid simulations. This method is well-suited to GPU computations. However, a number of GPU implementations are limited in the simulation scale because the data size that has to be handled is constrained by the GPU memory. One way to cope with the memory limitation is to use compression. In 2023 we have developped a Wawelet-based compression scheme, which has been described in [21] (under submission) and is currently being integrated into the Lattice Boltzmann GPU simulator that has been developped in the first part of Clement's thesis.

8.2.2 Scheduling multiple task-based applications on distributed heterogeneous computing nodes

Participants: Jean Etienne Ndamlabin, Bérenger Bramas.

The size, complexity and cost of supercomputers continue to grow making any waste more critical than in the past. Consequently, we need methods to reduce the waste coming from the users' choices, badly optimized applications or heterogeneous workloads during executions. In this context, we started activities on the scheduling of several task-based applications on given hardware resources. Specifically, we created load balancing heuristics to distribute the task-graph over the processing units. We are creating a new scheduler in StarPU to validate our approach. We will perform performance evaluation in the next stage, first using a simulator (SimdGrid) and then on real hardware.

8.2.3 Multreeprio: a scheduler for task-based applications on heterogeneous hardware

Participants: Hayfa Tayeb, Bérenger Bramas.

In the context of the TEXTAROSSA project, we investigated how the Heteroprio scheduler can be improved. We created a new scheduler called Multreeprio where the tasks can have multiple priorities, instead of having priorities for each type of tasks. This novel scheduler relies on a set of priority queues allowing the assignment of tasks to available resources according to priority scores per task for each type of processing unit. These scores are computed through heuristics giving hints on the acceleration of tasks on different processing units, their criticality, and data locality. We study the performance of several task-based applications using the StarPU runtime system on heterogeneous computing nodes and demonstrated efficient executions (paper under submission).

8.3 Program Optimization

8.3.1 Guided Equality Saturation

Participants: Thomas Koehler.

Thomas Koehler worked with colleagues from Scotland (Andrés Goens, Siddharth Bhat, Tobias Grosser, Phil Trinder and Michel Steuwer) on publishing his prior PhD work on *guided equality saturation*. Guided equality saturation is a semi-automatic term rewriting technique that scales beyond the fully-automatic rewriting technique called equality saturation by allowing human insight at key decision points. When unguided equality saturation fails, the rewriting is split into two simpler equality saturation steps: from the original term to a human-provided intermediate guide, and from the guide to the target. Complex rewriting tasks may require multiple guides, resulting in a sequence of equality saturation steps. A guide can be a complete term, or a more concise sketch containing undefined elements that are instantiated by the equality saturation search.

We demonstrate the generality and effectiveness of guided equality saturation with case studies in theorem proving and program optimization. First, guided equality saturation allows writing proofs as a series of calculations omitting details and skipping steps, in the Lean 4 proof assistant. Second, guided equality saturation allows performing matrix multiplication optimizations in the RISE array language. In both case studies, guided equality saturation succeeds where unguided equality saturation fails, with orders of magnitude less runtime and memory consumption. This work has been accepted for publication at POPL'2024 [11].

8.3.2 OptiTrust: Producing Trustworthy High-Performance Code via Source-to-Source Transformations

Participants: Arthur Charguéraud, Guillaume Bertholon, Thomas Koehler.

In 2023, we pursued the development of the OptiTrust prototype framework for producing high-performance code via source-to-source transformations. We completed two additional major case studies: an optimization script for matrix multiplication, and one for a standard graphical processing algorithm (Harris' corner-detection). We show that we are able to reproduce the code produced by TVM and Halide, respectively. These two tools are state-of-the-art tools in almost-automated code optimization; they apply to DSL, i.e. specialized programming languages. What our work shows is that equivalent optimizations can be achieved by manipulating standard C code. Thomas Koehler presented these two case studies at the ARRAY workshop [31].

We have also been working on exploiting Separation Logic shape assertions in order to justify the correctness of the code transformations performed.

Besides, Guillaume Bertholon continues his PhD work on the optimization of formally verified code. Our approach consists of carrying Separation Logic derivations through source-to-source transformations. His preliminary work has been presented at the JFLA workshop [15].

8.4 Program Verification

8.4.1 Separation Logic for Sequential Programs

Participants: Arthur Charguéraud.

Arthur Charguéraud released the second edition of his all-in-Coq book entitled *Foundations Separation Logic*, Volume 6 of the Software Foundations series. This second edition is accompanied with companion course notes.

Arthur Charguéraud defended on Febuary 2023 his Habilitation (HDR) [20], entitled *A Modern Eye on Separation Logic for Sequential Programs*. The manuscript gives a complete introduction of the working of the CFML tool (7.1.2), as well as its extensions for reasoning about time bounds (PhD thesis by Armaël Guéneau) and reasoning about space bounds (PhD thesis by Alexandre Moine).

8.4.2 Formal Verification of a Transient Sequence Data Structure

Participants: Arthur Charguéraud, Rémy Kimbrough.

A transient data structure is a package of an ephemeral data structure, a persistent data structure, and fast conversions between them. The OCaml Sek package developed by Arthur Charguéraud and François Pottier provides an optimized implementation of a transient sequence data structures. In particular, it features efficient $O(1)$ push and pop operations at the two ends, as well as $O(\log n)$ concatenation and split operations.

In prior work, Alexandre Moine, co-advised by Charguéraud and Pottier, formally verified a transient stack data structure, using the CFML verification framework (7.1.2). In follow up work, Rémy Kimbrough, advised by Charguéraud, made good progress towards the formal verification of a transient sequence data structure, with similar features as the Sek package. We look forward to complete this formalization and submit it for publication.

8.4.3 Formal Proof of Space Bounds for Concurrent, Garbage-Collected Programs

Participants: Arthur Charguéraud, Alexandre Moine.

Alexandre Moine, co-advised by Arthur Charguéraud and François Pottier have presented a novel, high-level program logic for establishing space bounds in Separation Logic, for programs that execute with a garbage collector. A key challenge is to design sound, modular, lightweight mechanisms for establishing the unreachability of a block. In the setting of a high-level, ML-style language, a key problem is to identify and reason about the memory locations that the garbage collector considers as roots. We demonstrate the expressiveness and tractability of the proposed program logic via a range of examples, including recursive functions on linked lists, objects implemented using closures and mutable internal state, recursive functions in continuation-passing style, and three stack implementations that exhibit different space bounds. These last three examples illustrate reasoning about the reachability of the items stored in a container as well as amortized reasoning about space. All the results are proved in Coq on top of Iris. This work has appeared at POPL'23 [9].

More recently, Alexandre has extended his logic to handle concurrent ML programs. A key challenge is to handle the fact that if an allocation lacks free space, then it is blocked until all other threads exit their critical section. Only at that point may a GC execute, and free the requested space. We are currently writing a journal paper describing this work.

8.4.4 Omnisemantics: Operational Semantics for Nondeterministic Languages

Participants: Arthur Charguéraud.

Arthur Charguéraud worked with collegues from MIT (Adam Chlipala and two of his students, Andres Erbsen and Samuel Gruetter) on a paper describing the technique of *omni-semantics*, a style for describing operational semantics particularly well-suited for nondeterministic languages. This technique introduces judgments that relate starting states to sets of outcomes, rather than to individual outcomes. Thus, a single derivation of these semantics for a particular starting state and program describes all possible nondeterministic executions, whereas in traditional small-step and big-step semantics, each derivation

only talks about one single execution. We demonstrate how this restructuring allows for straightforward modeling of languages featuring both nondeterminism and undefined behavior. Specifically, omni-semantics inherently assert safety, *i.e.*, they guarantee that none of the execution branches gets stuck, while traditional semantics need either a separate judgment or additional error markers to specify safety in the presence of nondeterminism. Applications presented include proofs of type soundness for lambda calculi, mechanical derivation of reasoning rules for program verification, and a forward proof of compiler correctness for terminating but potentially nondeterministic programs. All results are formalized in Coq. This work has been published in the journal ACM Transactions on Programming Languages and Systems (TOPLAS) [8]. The Coq formalization are distributed as part of the FormalMetaCoq library 7.1.11. Moreover, the omni-big-step semantics is used as a key ingredient in the soundness proof of Separation Logic presented in Charguéraud's *Foundations Separation Logic* course.

9 Partnerships and cooperations

9.1 European initiatives

9.1.1 H2020 projects

MICROCARD MICROCARD project on cordis.europa.eu

Title: Numerical modeling of cardiac electrophysiology at the cellular scale

Participants: Vincent Loechner, Arun Thangamani, Stéphane Genaud, Raphaël Colin, Atoli Huppé, Bérenger Bramas.

Duration: From April 1, 2021 to September 30, 2024

Partners:

- INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (INRIA), France
- MEGWARE COMPUTER VERTRIEB UND SERVICE GMBH, Germany
- SIMULA RESEARCH LABORATORY AS, Norway
- UNIVERSITE DE STRASBOURG (UNISTRA), France
- ZUSE-INSTITUT BERLIN (ZUSE INSTITUTE BERLIN), Germany
- UNIVERSITA DELLA SVIZZERA ITALIANA (USI), Switzerland
- KARLSRUHER INSTITUT FUER TECHNOLOGIE (KIT), Germany
- UNIVERSITE DE BORDEAUX (UBx), France
- UNIVERSITA DEGLI STUDI DI PAVIA (UNIPV), Italy
- INSTITUT POLYTECHNIQUE DE BORDEAUX (Bordeaux INP), France
- NUMERICOR GMBH, Austria
- OROBIX SRL (OROBIX), Italy

Inria contact: Mark POTSE

Coordinator: Mark POTSE

Summary: Cardiovascular diseases are the most frequent cause of death worldwide and half of these deaths are due to cardiac arrhythmia, a disorder of the heart's electrical synchronization system. Numerical models of this complex system are highly sophisticated and widely used, but to match observations in aging and diseased hearts they need to move from a continuum approach to a representation of individual cells and their interconnections. This implies a different, harder numerical problem and a 10,000-fold increase in problem size. Exascale computers will be needed to run such models.

We propose to develop an exascale application platform for cardiac electrophysiology simulations that is usable for cell-by-cell simulations. The platform will be co-designed by HPC experts, numerical scientists, biomedical engineers, and biomedical scientists, from academia and industry. We will develop, in concert, numerical schemes suitable for exascale parallelism, problem-tailored linear-system solvers and preconditioners, and a compiler to translate high-level model descriptions into optimized, energy-efficient system code for heterogeneous computing systems. The code will be parallelized with a recently developed runtime system that is resilient to hardware failures and will use an energy-aware task placement strategy.

The platform will be applied in real-life use cases with high impact in the biomedical domain and will showcase HPC in this area where it is painfully underused. It will be made accessible for a wide range of users both as code and through a web interface.

We will further employ our HPC and biomedical expertise to accelerate the development of parallel segmentation and (re)meshing software, necessary to create the extremely large and complex meshes needed from available large volumes of microscopy data.

The platform will be adaptable to similar biological systems such as nerves, and components of the platform will be reusable in a wide range of applications.

TEXTAROSSA TEXTAROSSA project on cordis.europa.eu

Title: Towards EXtreme scale Technologies and Accelerators for euROhpc hw/Sw Supercomputing Applications for exascale

Participants: Bérenger Bramas, Hayfa Tayeb.

Duration: From April 2021 to April 2024

Partners:
- Agenzia Nazionale per l'Energia, le Nuove Tecnologie e lo Sviluppo Economico Sostenibile (ENEA)
- Fraunhofer Gesellschaft Zur Förderung der Angewandten Forschung E.V. (FGH)
- Consorzio Interuniversitario per l'Informatica (CINI)
- Institut National de Recherche en Informatique et en Automatique (Inria)
- Bull SAS
- E4 Computer Engineering SpA
- Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (BSC)
- Instytut Chemii Bioorganicznej Polskiej Akademii Nauk (PSNC)
- Istituto Nazionale di Fisica Nucleare (INFN)
- Consiglio Nazionale delle Ricerche (CNR)
- In Quattro Srl.

Summary: This European project aims at achieving a broad impact on the High Performance Computing (HPC) field both in pre-exascale and exascale scenarios 14 7. The TEXTAROSSA consortium will develop new hardware accelerators, innovative two-phase cooling equipment, advanced algorithms, methods and software products for traditional HPC domains as well as for emerging domains in High Performance Artificial Intelligence (HPC-AI) and High Performance Data Analytics (HPDA). We will focus on the scheduling of task-graphs under energy constraints and on porting scientific codes on heterogeneous computing nodes with FPGAs.

9.2 National initiatives

9.2.1 ANR OptiTrust

Participants: Arthur Charguéraud, Guillaume Bertholon, Jens Gustedt, Thomas Koehler.

Turning a high-level, unoptimized algorithm into a high-performance code can take weeks, if not months, for an expert programmer. The challenge is to take full advantage of vectorized instructions, of all the cores and all the servers available, as well as to optimize the data layout, maximize data locality, and avoid saturating the memory bandwidth. In general, annotating the code with "pragmas" is insufficient, and domain-specific languages are too restrictive. Thus, in most cases, the programmer needs to write, by hand, a low-level code that combines dozens of optimizations. This approach is not only tedious and time-consuming, it also degrades code readibility, harms code maintenance, and can result in the introduction of bugs. A promising approach consists of deriving an HPC code via a series of source-to-source transformations guided by the programmer. This approach has been successfully applied in niche domains, such as image processing and machine learning. We aim to generalize this approach to optimize arbitrary code. Furthermore, the OptiTrust project aims at obtaining formal guarantees on the output code. A number of these transformations are correct only under specific hypotheses. We will formalize these hypotheses, and investigate which of them can be verified by means of static analysis. To handle the more complex hypotheses, we will transform not just code but also formal invariants attached to the code. Doing so will allow exploiting invariants expressed on the original code for justifying transformations performed at the n-th step of the transformation chain.

- Funding: ANR
- Start: October 2022
- End: September 2026
- Coordinator: Arthur Charguéraud (Inria)
- Partners: Inria team Camus (Strasbourg), Inria team TONUS (Strasbourg), Inria team Cambium (Paris), Inria team CASH (Lyon), CEA team LIST

9.2.2 ANR AUTOSPEC

Participants: Bérenger Bramas, Philippe Clauss, Stéphane Genaud, Marek Felosci, Anastasios Souris.

The AUTOSPEC project aims to create methods for automatic task-based parallelization and to improve this paradigm by increasing the degree of parallelism using speculative execution. The project will focus on source-to-source transformations for automatic parallelization, speculative execution models, DAG scheduling, and the activation mechanisms for speculative execution. With this aim, the project will rely on a source-to-source compiler that targets the C++ language, a runtime system with speculative execution capabilities, and an editor (IDE) to enable compiler-guided development. The outcomes from the project will be open-source with the objective of developing a user community. The benefits will be of great interest both for developers who want to use an automatic parallelization method, but also for high-performance programming experts who will benefit from improvements of the task-based programming. The results of this project will be validated in various applications such as a protein complexes simulation software, and widely used open-source software. The aim will be to cover a wide range of applications to demonstrate the potential of the methods derived from this project while trying to establish their limitations to open up new research perspectives.

- Funding: ANR (JCJC)
- Start: October 2021
- End: September 2025
- Coordinator: Bérenger Bramas

9.2.3 Exa-SofT project, PEPR NumPEx

Participants: Bérenger Bramas, Philippe Clauss, Raphael Colin, Ugo Battiston.

Though significant efforts have been devoted to the implementation and optimization of several crucial parts of a typical HPC software stack, most HPC experts agree that exascale supercomputers will raise new challenges, mostly because the trend in exascale compute-node hardware is toward heterogeneity and scalability: Compute nodes of future systems will have a combination of regular CPUs and accelerators (typically GPUs), along with a diversity of GPU architectures. Meeting the needs of complex parallel applications and the requirements of exascale architectures raises numerous challenges which are still left unaddressed. As a result, several parts of the software stack must evolve to better support these architectures. More importantly, the links between these parts must be strengthened to form a coherent, tightly integrated software suite. The Exa-SofT project aims at consolidating the exascale software ecosystem by providing a coherent, exascale-ready software stack featuring breakthrough research advances enabled by multidisciplinary collaborations between researchers. The main scientific challenges we intend to address are: productivity, performance portability, heterogeneity, scalability and resilience, performance and energy efficiency.

- Funding: PEPR NumPEx
- Start: September 2023
- End: August 2028
- Coordinator: Raymond Namyst (Inria STORM)
- WP2 co-leader: Philippe Clauss

9.2.4 TEXAS project, Inria's exploratory actions program

Participants: Bérenger Bramas, Jean Etienne Ndamlabin Mboula.

The TEXAS project aims to optimize the performance of supercomputers, particularly Exascale machines, which are capable of executing a billion billion operations per second. The objectives of the Texas project include: (1) Developing new programming models for parallel computing to enhance the efficiency of supercomputers. (2) Creating dynamic algorithms for task management, allowing supercomputers to adjust their computing power during the execution of high-performance applications (HPC), like numerical simulations. (3) Reducing the high energy consumption of supercomputers, which is a significant scientific and ecological issue.

- Funding: Inria
- Start: January 2022
- End: December 2023
- Coordinator: Bérenger Bramas

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events: selection

Member of the conference program committees

- Arthur Charguéraud was program committee member for the conferences POPL and CPP, and for the workshops CoqPL and JFLA.

Reviewer

- Bérenger Bramas was a reviewer for Supercomputing (Poster) and Compas.

10.1.2 Journal

Member of the editorial boards

- Jens Gustedt has been the Editor-in-Chief of the journal Discrete Mathematics and Theoretical Computer Science (DMTCS), from October 2001 to June 2023.

Reviewer - reviewing activities

- Bérenger Bramas was a reviewer for The Journal of Supercomputing (JOS), IEEE Transactions on Consumer Electronics (TCE), the Journal of Computer Science and Technology (JCST), Parallel Computing (Parco), Scientific reports (SR Nature), PLOS ONE, Software: Practice and Experience (SPE), Journal of Real-Time Image Processing (JRTI), IEEE Transactions on Parallel and Distributed Systems (TPDS), ACM Transactions on Parallel Computing (TPC) and International Journal of High Performance Computing Applications (IJHPCA).
- Arthur Charguéraud wrote a public review for the book *Functional Algorithms, Verified!* for *Formal Methods Europe* and the journal *Formal Aspects of Computing*, and wrote a review for the journal JAR.
- Philippe Clauss was a reviewer for ACM TACO (Transactions on Architecture and Code Optimization).
- Vincent Loechner is a member of the IEEE/ACM International Symposium on Code Generation and Optimization (CGO) Artifact Evaluation Committee.

10.1.3 Invited talks

- Arthur Charguéraud gave an invited talk entited *Comment allier persistance et performance* at the Collège de France, as part of Xavier Leroy's course.

10.1.4 Scientific expertise

- Jens Gustedt is a member of the ISO/IEC working groups ISO/IEC PL1/SC22/WG14 and WG21 for the standardization of the C and C++ programming languages, respectively.
- Philippe Clauss was a reviewer for a project submitted to the French agency ANR (Agence Nationale de la Recherche).

10.1.5 Research administration

- Jens Gustedt is the head of the ICPS team for the ICube lab.
- Jens Gustedt is deputy director of the ICube lab, responsible for the IT and CS policy and for the coordination between the lab and the Inria center. In that function he also represents ICube in the board of the INRIA Nancy - Grand Est research center.
- Jens Gustedt is member of the steering committee of the interdisciplanary institute IRMIA++ of Strasbourg University.
- Philippe Clauss is co-leader of the work package entitled *Just-in-Time code optimization with continuous feedback loop*, in the Exa-SofT project of the PEPR NumPEx.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Licence: Vincent Loechner, Algorithmics and programmation, 82h, L1, Université de Strasbourg, France
- Licence: Vincent Loechner, System administration, 40h, Licence Pro, Université de Strasbourg, France
- Licence: Vincent Loechner, System programming, 20h, L2, Université de Strasbourg, France
- Licence: Vincent Loechner, Parallel programming, 32h, L3, Université de Strasbourg, France
- Master: Vincent Loechner, Real-time systems, 12h, M1, Université de Strasbourg, France
- Eng. School: Vincent Loechner, Parallel programming, 20h, Telecom Physique Strasbourg - 3rd year, Université de Strasbourg, France
- Master: Bérenger Bramas, Compilation and Performance, 24h, M2, Université de Strasbourg, France
- Master: Bérenger Bramas, Compilation, 24h, M1, Université de Strasbourg, France
- Licence: Philippe Clauss, Computer architecture, 18h, L2, Université de Strasbourg, France
- Licence: Philippe Clauss, Bases of computer architecture, 22h, L1, Université de Strasbourg, France
- Master: Philippe Clauss, Compilation, 84h, M1, Université de Strasbourg, France
- Master: Philippe Clauss, Real-time programming and system, 37h, M1, Université de Strasbourg, France
- Master: Philippe Clauss, Code optimization and transformation, 31h, M1, Université de Strasbourg, France
- Licence: Stéphane Genaud, Algorithmics and programmation, 82h, L1, Université de Strasbourg, France
- Licence: Stéphane Genaud, Parallel programming, 30h, L3, Université de Strasbourg, France
- Master: Stéphane Genaud, Cloud and Virtualization, 12h, M1, Université de Strasbourg, France
- Master: Stéphane Genaud, Large-Scale Data Processing, 15h, M1, Université de Strasbourg, France
- Master: Stéphane Genaud, Distributed Storage and Processing, 15h, M2, Université de Strasbourg, France
- Eng. School: Stéphane Genaud, Introduction to Operating Systems, 16h, Telecom Physique Strasbourg - 1st year, Université de Strasbourg, France
- Eng. School: Stéphane Genaud, Object-Oriented Programming, 60h, Telecom Physique Strasbourg - 1st year, Université de Strasbourg, France

10.2.2 Supervision

- Philippe Clauss is in charge of the master's degree in Computer Science of the University of Strasbourg, since Sept. 2020.
- Stéphane Genaud is in charge of the Bachelor and Master in Computer Science curriculae at UFAZ (Baku, Azerbadjian) who delivers Unistra diplomas. Since August 2023.
- PhD in progress: Clément Rossetti, Algebraic loop transformations, advised by Philippe Clauss, since Oct 2022.

- PhD in progress: Ugo Battiston, C++ complexity disambiguation for advanced optimizing and parallelizing code transformations, since Oct. 2023, advised by Philippe Clauss and Marc Pérache (CEA).
- PhD in progress: Raphaël Colin, Runtime multi-versioning of parallel tasks, since Oct. 2023, advised by Philippe Clauss and Thierry Gautier (Inria project-team Avalon).
- PhD in progress: Guillaume Bertholon, *Formal Verification of Source-to-Source Transformations*, since Sept 2022, is advised by Arthur Charguéraud.
- PhD in progress: Alexandre Moine, *Formal Verification of Space Bounds*, is co-advised by Arthur Charguéraud and François Pottier, at Inria Paris, since Oct 2021.
- PhD in progress: Clément Flint, Efficient data compression for high-performance PDE solvers., advised by Philippe Helluy (Inria project-team Tonus), Stéphane Genaud, and Bérenger Bramas, since Nov 2020.
- PhD in progress: Hayfa Tayeb, Efficient scheduling strategies for the task-based parallelization., advised by Bérenger Bramas, Abdou Guermouche (Inria project-team TOPAL), Mathieu Faverge (Inria project-team TOPAL), since Nov 2021.
- PhD in progress: Anastasios Souris, Speculative execution models for the task-based parallelization., advised by Bérenger Bramas, Philippe Clauss, since Jul 2022.
- PhD in progress: Antoine Gicquel, Acceleration of the matrix-vector product using the fast mutlipole method for heterogeneous machine clusters., advised by Bérenger Bramas, Olivier Coulaud, since Oct 2023.
- PhD in progress: David Algis, Hybridization of the Tessendorf method and Smoothed Particle Hydrodynamics for real-time ocean simulation., advised by Bérenger Bramas, Emmanuelle Darles (XLim), Lilian Aveneau (XLim lab), since Oct 2022.
- PhD in progress: Arun Thangamani, *Code generation for heterogeneous architectures*, advised by Stéphane Genaud and Vincent Loechner, since Sept 2021.
- PhD in progress: Tom Hammer, *Synergie entre ordonnancement et optimisation des accès mémoire dans le modèle polyédrique*, advised by Stéphane Genaud and Vincent Loechner, since Sept 2023.

10.3 Popularization

Arthur 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. 690,000 teenagers played with the interactive exercises in November and December 2023.

10.3.1 Internal or external Inria responsibilities

Arthur Charguéraud is a member of the COMIPERS jury for PhD and postdoc grants at Inria Nancy Grand-Est.

Bérenger Bramas is a member of the CDT and IES committee at Inria Nancy Grand-Est.

11 Scientific production

11.1 Major publications

[1] U. A. Acar, V. Aksenov, A. Charguéraud and M. Rainey. 'Provably and Practically Efficient Granularity Control'. In: *PPoPP 2019 - Principles and Practice of Parallel Programming*. Washington DC, United States, Feb. 2019. DOI: 10.1145/3293883.3295725. URL: https://hal.inria.fr/hal-01973285.

[2] P. Clauss. 'Counting Solutions to Linear and Nonlinear Constraints Through Ehrhart Polynomials: Applications to Analyze and Transform Scientific Programs'. In: *ICS, International Conference on Supercomputing*. ACM International Conference on Supercomputing 25th Anniversary Volume. Munich, Germany, 2014. DOI: 10.1145/2591635.2667172. URL: https://hal.inria.fr/hal-01100306.

[3] P. Clauss, F. J. Fernández, D. Garbervetsky and S. Verdoolaege. 'Symbolic polynomial maximization over convex sets and its application to memory requirement estimation'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 17.8 (Aug. 2009), pp. 983–996. DOI: 10.1109/TVLSI.2008.2002049. URL: https://hal.inria.fr/inria-00504617.

[4] P.-N. Clauss and J. Gustedt. 'Iterative Computations with Ordered Read-Write Locks'. In: *Journal of Parallel and Distributed Computing* 70.5 (2010), 496–504. DOI: 10.1016/j.jpdc.2009.09.002. URL: https://hal.inria.fr/inria-00330024.

[5] J. Gustedt. *Modern C*. Manning, 27th Nov. 2019. URL: https://hal.inria.fr/hal-02383654.

[6] A. Ketterlin and 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*. Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization. Boston, United States: ACM, Apr. 2008, pp. 94–103. DOI: 10.1145/1356058.1356071. URL: https://hal.inria.fr/inria-00504597.

[7] A. Sukumaran-Rajam and P. Clauss. 'The Polyhedral Model of Nonlinear Loops'. In: *ACM Transactions on Architecture and Code Optimization* 12.4 (Jan. 2016). DOI: 10.1145/2838734. URL: https://hal.inria.fr/hal-01244464.

11.2 Publications of the year

International journals

[8] A. Charguéraud, A. Chlipala, A. Erbsen and S. Gruetter. 'Omnisemantics: Smooth Handling of Nondeterminism'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 45.1 (8th Mar. 2023), pp. 1–43. DOI: 10.1145/3579834. URL: https://inria.hal.science/hal-03255472.

[9] A. Moine, A. Charguéraud and F. Pottier. 'A High-Level Separation Logic for Heap Space under Garbage Collection (Extended Version)'. In: *Proceedings of the ACM on Programming Languages* 7.POPL (2023), pp. 718–747. DOI: 10.1145/3571218. URL: https://inria.hal.science/hal-03823056.

[10] H. Tayeb, L. Paillat and B. Bramas. 'Autovesk: Automatic vectorized code generation from unstructured static kernels using graph transformations'. In: *ACM Transactions on Architecture and Code Optimization* (9th Nov. 2023). DOI: 10.1145/3631709. URL: https://inria.hal.science/hal-03914178.

International peer-reviewed conferences

[11] T. Koehler, A. Goens, S. Bhat, T. Grosser, P. Trinder and M. Steuwer. 'Guided Equality Saturation'. In: 51st ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2024). London, United Kingdom, 14th Jan. 2024. DOI: 10.1145/3632900. URL: https://inria.hal.science/hal-04372044.

[12] C. Rossetti and P. Clauss. 'Algebraic Tiling'. In: *IMPACT 2023, 13th International Workshop on Polyhedral Compilation Techniques*. IMPACT 2023, 13th International Workshop on Polyhedral Compilation Techniques. Toulouse, France, 16th Jan. 2023. URL: https://inria.hal.science/hal-03944790.

[13] A. Thangamani, T. Trevisan, V. Loechner, S. Genaud and B. Bramas. 'Lifting Code Generation of Cardiac Physiology Simulation to Novel Compiler Technology'. In: *Proceedings of the 21st ACM/IEEE International Symposium on Code Generation and Optimization (CGO '23)*. 21st ACM/IEEE International Symposium on Code Generation and Optimization (CGO '23). Montréal Québec, Canada: ACM, 27th Feb. 2023, p. 13. DOI: 10.1145/3579990.3580008. URL: https://inria.hal.science/hal-03977688.

[14] T. Trevisan Jost, A. Thangamani, R. Colin, V. Loechner, S. Genaud and B. Bramas. 'GPU Code Generation of Cardiac Electrophysiology Simulation with MLIR'. In: *Lecture Notes in Computer Science*. Euro-Par 2023: Parallel Processing. Vol. LNCS - 14100. Lecture Notes in Computer Science. Limassol, Cyprus: Springer Nature Switzerland, 24th Aug. 2023, pp. 549–563. DOI: 10.1007/978-3-031-39698-4_37. URL: https://inria.hal.science/hal-04206195.

National peer-reviewed Conferences

[15] G. Bertholon and A. Charguéraud. 'An AST for Representing Programs with Invariants and Proofs'. In: *Journées Francophones des Langages Applicatifs*. JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs. Praz-sur-Arly, France, 16th Jan. 2023, pp. 43–58. URL: https://inria.hal.science/hal-03936618.

[16] A. Souris, B. Bramas and P. Clauss. 'Extending the Task Dataflow Model with Speculative Data Accesses'. In: COMPAS 2023 - Conférence francophone d'informatique en Parallélisme, Architecture et Système. Annecy (France), France, 4th July 2023. URL: https://inria.hal.science/hal-04156383.

Conferences without proceedings

[17] C. Bastoul, A. Ketterlin and V. Loechner. 'Superloop Scheduling: Loop Optimization via Direct Statement Instance Reordering'. In: IMPACT 2023. Toulouse, France, 16th Jan. 2023. URL: https://inria.hal.science/hal-04393522.

[18] T. Hammer and V. Loechner. 'PolyLingual: a Programmable Polyhedral Scheduler'. In: IMPACT 2023. Toulouse (31000), France, 16th Jan. 2023. URL: https://inria.hal.science/hal-04393530.

[19] C. Rossetti, A. Hamon and P. Clauss. 'Algebraic Tiling facing Loop Skewing'. In: IMPACT 2024, 14th International Workshop on Polyhedral Compilation Techniques. Munich (Allemagne), Germany, 17th Jan. 2024. URL: https://inria.hal.science/hal-04379037.

Doctoral dissertations and habilitation theses

[20] A. Charguéraud. 'A Modern Eye on Separation Logic for Sequential Programs'. Université de Strasbourg, 27th Feb. 2023. URL: https://inria.hal.science/tel-04076725.

Reports & preprints

[21] C. Flint and P. Helluy. *Reducing the memory usage of Lattice-Boltzmann schemes with a DWT-based compression*. 17th Feb. 2023. URL: https://inria.hal.science/hal-03990880.

[22] J. Gustedt. *A Provenance-aware memory object model for C (slides)*. 10th Feb. 2023. URL: https://inria.hal.science/hal-03984047.

[23] J. Gustedt. *Extensions to the preprocessor for C2Y*. N3190. ISO JCT1/SC22/WG14, 13th Dec. 2023. URL: https://inria.hal.science/hal-04358378.

[24] J. Gustedt. *Simple TU initialization and cleanup handling with dependencies*. N3185. ISO JCT1/SC22/WG14, 13th Dec. 2023. URL: https://inria.hal.science/hal-04358330.

[25] J. Gustedt. *Some constants are literally literals*. N3189. ISO JCT1/SC22/WG14, 13th Dec. 2023. URL: https://inria.hal.science/hal-04358367.

[26] J. Gustedt. *The future of imaginary types*. N3206. ISO JCT1/SC22/WG14, 15th Dec. 2023. URL: https://inria.hal.science/hal-04358385.

[27] J. Gustedt and M. Uecker. *Clarify array length specifications and sizeof expressions*. N3187. ISO JCT1/SC22/WG14, 13th Dec. 2023. URL: https://inria.hal.science/hal-04358353.

[28] J. Gustedt and M. Uecker. *Identifying array length state*. ISO JCT1/SC22/WG14, 13th Dec. 2023. URL: https://inria.hal.science/hal-04358362.

[29] J. Gustedt and M. Uecker. *Initialization, allocation and effective type*. N3186. ISO JCT1/SC22/WG14, 13th Dec. 2023. URL: https://inria.hal.science/hal-04358343.

[30] M. Uecker and J. Gustedt. *Parameter Forward Declarations*. N3140. ISO JCT1/SC22/WG14, 22nd June 2023, p. 3. URL: https://inria.hal.science/hal-04358315.

Other scientific publications

[31] T. Koehler, A. Charguéraud, B. Bytyqi, D. Rouhling and Y. A. Barsamian. *OptiTrust: an Interactive Optimization Framework*. Orlando (Florida), United States, 17th June 2023. URL: https://inria.hal.science/hal-04053772.

11.3 Other

Softwares

[32] [SW] J. Gustedt, *Code examples for the book Modern C* version This is the updated version for the C23 edition of Modern C, 30th Oct. 2023. LIC: MIT License. HAL: ⟨hal-03345464⟩, URL: https://inria.hal.science/hal-03345464, SWHID: ⟨swh:1:dir:ae32375c2fba720ee23e713eed67be162e642d4d;origin=https://hal.archives-ouvertes.fr/hal-03345464;visit=swh:1:snp:11c79030b35adc0b34d957bed7ad2591f8687120;anchor=swh:1:rel:58812d110e28fef8d0a2377552c07a7a00a45ac7;path=/⟩.