RESEARCH CENTRE

Inria Centre at Université de Lorraine

IN PARTNERSHIP WITH:

Université de Strasbourg

2024

ACTIVITY REPORT

Project-Team

CAMUS

Compilation for multi-processor and multi-core architectures

IN COLLABORATION WITH: Laboratoire des sciences de l'ingénieur, de l'informatique et de l'imagerie

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Inria

Contents

Project-Team CAMUS 1

1 Team members, visitors, external collaborators 2

2 Overall objectives 3

3 Research program 3
3.1 Semi-automatic and assisted code optimization . . . 4
3.2 Fully-automatic code optimization . . . 5
3.3 Fundamental algorithms & mathematical tools . . . 5

4 Application domains 5

5 Social and environmental responsibility 6
5.1 Footprint of research activities . . . 6
5.2 Impact of research results . . . 6

6 Highlights of the year 6

7 New software, platforms, open data 6
7.1 New software . . . 6
7.1.1 TRAHRHE . . . 6
7.1.2 CFML . . . 7
7.1.3 openCARP . . . 7
7.1.4 SPECX . . . 8
7.1.5 Autovesk . . . 8
7.1.6 SPC5 . . . 8
7.1.7 PolyLib . . . 8
7.1.8 TLC . . . 9
7.1.9 FormalMetaCoq . . . 9
7.1.10 OptiTrust . . . 10
7.1.11 APOLLO . . . 10
7.1.12 APAC . . . 10
7.1.13 Rise & Shine . . . 11
7.1.14 egg-sketches . . . 11
7.1.15 slotted-egraphs . . . 11
7.2 Open data . . . 12

8 New results 12
8.1 Semi-automatic and assisted code optimization . . . 12
8.1.1 OptiTrust: Producing Trustworthy High-Performance Code via Source-to-Source Transformations . . . 12
8.1.2 Specx: A C++ Task-Based Runtime System for Heterogeneous Distributed Architectures 12
8.1.3 Efficient GPU Implementation of Particle Interactions with Cutoff Radius and Few Particles per Cell . . . 12
8.1.4 SPC5: An Efficient SpMV Framework Vectorized Using ARM SVE and x86 AVX-512 . 13
8.1.5 Using the Discrete Wavelet Transform for Lossy On-the-Fly Compression of GPU Fluid Simulations . . . 13
8.1.6 Exploiting Ray Tracing Technology Through OptiX to Compute Particle Interactions with Cutoff in a 3D Environment on GPUs . . . 13
8.1.7 Recognizing Polynomial Loops in Traces . . . 13
8.2 Program Verification . . . 14
8.2.1 Formal Proof of Space Bounds for Concurrent, Garbage-Collected Programs . . . 14
8.3 Fully-automatic code optimization . . . 14
8.3.1 Algebraic tiling . . . 14

8.3.2 Dynamic Task Scheduling with Multiple Priorities on Heterogeneous Computing Systems . . . 15
8.3.3 Scheduling multiple task-based applications on distributed heterogeneous computing nodes . . . 15
8.3.4 Automatic Task-Based Parallelization using Source to Source Transformations . . . 15
8.3.5 Ionic Models Code Generation for Heterogeneous Architectures . . . 16
8.3.6 Rewriting for Specialized CGRAs. . . . 16
8.4 Fundamental algorithms & mathematical tools . . . 17
8.4.1 Trahrhe expressions . . . 17
8.4.2 Polyhedral Scheduling . . . 17
8.4.3 Improvements of the C programming language . . . 17
8.4.4 Fast Counting, Ranking and Rank Inversion . . . 18
8.4.5 Typechecking of Overloading . . . 19
8.4.6 Guided Equality Saturation . . . 19

9 Partnerships and cooperations **19**
9.1 International research visitors . . . 19
9.1.1 Visits of international scientists . . . 19
9.2 European initiatives . . . 20
9.2.1 Horizon Europe . . . 20
9.2.2 H2020 projects . . . 21
9.3 National initiatives . . . 21
9.3.1 ANR OptiTrust . . . 21
9.3.2 ANR AUTOSPEC . . . 22
9.3.3 Exa-SofT project, PEPR NumPEx . . . 22

10 Dissemination **23**
10.1 Promoting scientific activities . . . 23
10.1.1 Scientific events: selection . . . 23
10.1.2 Journal . . . 23
10.1.3 Invited talks . . . 23
10.1.4 Scientific expertise . . . 24
10.1.5 Research administration . . . 24
10.2 Teaching - Supervision - Juries . . . 24
10.2.1 Teaching . . . 24
10.2.2 Supervision . . . 25
10.2.3 Juries . . . 26
10.3 Popularization . . . 26
10.3.1 Specific official responsibilities in science outreach structures . . . 26

11 Scientific production **26**
11.1 Major publications . . . 26
11.2 Publications of the year . . . 27
11.3 Cited publications . . . 30

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, Senior Researcher, from Oct 2024, HDR]
- Arthur Charguéraud [INRIA, Researcher, until Sep 2024, HDR]
- Jens Gustedt [INRIA, Senior Researcher, HDR]
- Thomas Koehler [CNRS, Researcher, from Oct 2024]

Faculty Members

- Philippe Clauss [Team leader, UNIV STRASBOURG, Professor, HDR]
- Stephane Genaud [ENSIIE, Associate Professor, Note: Professor Stéphane Genaud has a convention ("mise à disposition") with University of Strasbourg for both teaching and research, renewed every year until this convention is turned into a permanent professor position at Unistra., HDR]
- Alain Ketterlin [UNIV STRASBOURG, Associate Professor]
- Vincent Loechner [UNIV STRASBOURG, Associate Professor]

Post-Doctoral Fellows

- Marek Felsoci [INRIA, Post-Doctoral Fellow, until Nov 2024]
- Thomas Koehler [INRIA, Post-Doctoral Fellow, until Sep 2024]

PhD Students

- Ugo Battiston [INRIA]
- Guillaume Bertholon [UNIV STRASBOURG]
- Raphael Colin [INRIA]
- Tom Hammer [UNIV STRASBOURG]
- Atoli Huppe [INRIA, from Oct 2024]
- Clément Rossetti [UNIV STRASBOURG]
- Anastasios Souris [INRIA, until Sep 2024]
- Arun Thangamani [UNIV STRASBOURG, until Sep 2024]

Technical Staff

- Erwan Auer [INRIA, Engineer]
- Thai Hoa Trinh [ORANGE, Engineer, until Feb 2024]

Interns and Apprentices

- Julien Gaupp [INRIA, Intern, from Sep 2024]
- Julien Gaupp [INRIA, Intern, until Aug 2024]
- Marceau Noury [INRIA, Intern, from Sep 2024]

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, besides 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 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

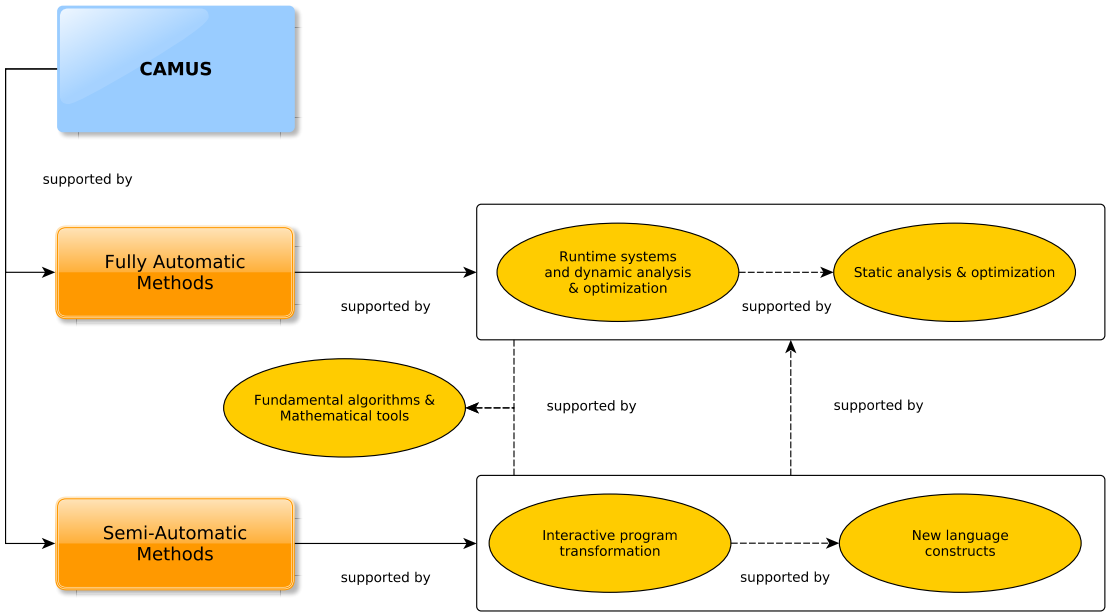

Figure 1: General view of CAMUS' research objectives.

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 to which 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 removal of some restrictions regarding the class of programs that are candidates 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. Besides 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, (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

- Thomas Koehler, CNRS Research Scientist, has joined the team in October 2024.
- Arthur Charguéraud, became senior Inria researcher (Directeur de Recherche) in October 2024.
- The new C standard, C23, was published by ISO [51].
- The paper on *Guided Equality Saturation* [21] was selected as 1 of 9 MIT PL review highlights.

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: A more robust way of computing the ranking polynomials has been implemented. A quite new version of the software written in C/C++ has been be published.

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

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

Contact: Philippe Clauss

Participants: Philippe Clauss, Marceau Noury

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 2024, CFML has been upgraded to a more recent version of Coq.

URL: http://www.chargueraud.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.

News of the Year: Improvements of the code generation of ionic models (limpetMLIR) : generation of CUDA and AMD kernels. Building of all targets embedded in functions for the runtime interface. StarPU interface to the kernels. Benchmarks (execution time, energy consumption).

URL: https://opencarp.org/

Publications: hal-04206195, hal-03977688

Contact: Vincent Loechner

Participants: Vincent Loechner, 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

Participant: Bérenger Bramas

7.1.6 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

Participant: Bérenger Bramas

7.1.7 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.

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

News of the Year: Maintenance, upgrade of the user interface, upgrade of the build process.

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

Contact: Vincent Loechner

Participant: Vincent Loechner

7.1.8 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.charguéraud.org/softs/tlc/

Contact: Arthur Charguéraud

Participant: Arthur Charguéraud

7.1.9 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

Participant: Arthur Charguéraud

7.1.10 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 reached a milestone, with the completion of 3 major case studies.

URL: http://optitrust.inria.fr

Contact: Arthur Charguéraud

Participants: Arthur Charguéraud, Thomas Koehler, Guillaume Bertholon

7.1.11 APOLLO

Name: Automatic speculative POLyhedral Loop Optimizer

Keyword: Automatic parallelization

Scientific Description: APOLLO - Automatic speculative POLyhedral Loop Optimizer is a compiler framework dedicated to automatic, dynamic and speculative parallelization and optimization of programs' loop nests. This framework allows a user to mark in a C/C++ source code some nested loops of any kind (for, while or do-while loops) in order to be handled by a speculative parallelization process, to take advantage of the underlying multi-core processor architecture. The framework is composed of two main parts: extensions to the CLANG-LLVM compiler and a runtime system.

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. Some recent extensions enabling dynamic multi-versioning have been implemented in 2020.

News of the Year: Apollo has been upgraded to LLVM 17 and to pluto 0.12.

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

Publications: hal-01244464, hal-01533692, hal-01155172, hal-02457425, hal-01377656

Contact: Philippe Clauss

Participants: Aravind Sukumaran Rajam, Erwan Auer, Raphael Colin, Juan Manuel Martinez Caamano, Manuel Selva, Philippe Clauss

7.1.12 APAC

Keywords: Source-to-source compiler, Automatic parallelization, Parallel programming

Scientific Description: APAC is a compiler for automatic parallelization that transforms C++ source code to make it parallel by inserting tasks. It uses the tasks+dependencies paradigm and relies on OpenMP as runtime system. Internally, it is based on Optitrust (and Clang-LLVM).

Functional Description: Automatic task-based parallelization compiler

News of the Year: Additional case studies have been developed.

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

Contact: Bérenger Bramas

Participants: Marek Felsoci, Bérenger Bramas, Stephane Genaud

7.1.13 Rise & Shine

Keywords: Programming language, Compilation

Functional Description: Programming language and compiler for array computing. Programs are expressed at a high level in the RISE language. Programs are transformed using a set of rewrite rules that encode implementation and optimization choices. The Shine compiler generates high-performance parallel C or OpenCL code while preserving the optimization choices made during rewriting.

News of the Year: A prototype Rise to C compiler executable that preserves floating-point semantics was added by Thomas Koehler, as part of his collaboration with Eva Darulova (Uppsala University).

URL: http://rise-lang.org

Contact: Thomas Koehler

Participant: Thomas Koehler

Partner: Technische Universität Berlin

7.1.14 egg-sketches

Keyword: Program rewriting techniques

Functional Description: egg-sketches is a library adding support for program sketches on top of the egg (e-graphs good) library, an e-graph library optimized for equality saturation. Sketches are program patterns that are satisfied by a family of programs. They can also be seen as incomplete or partial programs as they can leave details unspecified. With egg-sketches, it is possible to perform Guided Equality Saturation: a semi-automatic technique that allows programmers to guide rewriting via program sketches.

News of the Year: Upgraded to latest egg dependency, added an extra sketch construct, fixed one bug.

URL: https://github.com/Bastacyclop/egg-sketches

Contact: Thomas Koehler

Participant: Thomas Koehler

Partner: University of Amsterdam

7.1.15 slotted-egraphs

Keyword: Term Rewriting Systems

Functional Description: Implementation of the slotted e-graph data structure, an extension of e-graphs representing terms that differ only in the names of their variables uniquely. With slotted-egraphs, users of languages with variables can perform equality saturation by: (1) defining the term language, representing variables and binders via slots, (2) defining rewrite rules, without having to worry about naming collisions, and leveraging built-in mechanisms for freshness predicates and substitutions, (3) performing equality saturation by initializing a slotted e-graph, growing it by applying rewrites, and extracting from it.

News of the Year: Developement started in February 2024, led by Rudi Schneider from TU Berlin. Instigated and supervised by Thomas Koehler and Michel Steuwer. A paper was submitted to PLDI 2025.

URL: https://github.com/memoryleak47/slotted-egraphs

Contact: Thomas Koehler

Participant: Thomas Koehler

Partners: Technische Universität Berlin, University of Amsterdam

7.2 Open data

Polyhedral Compilers DockerFile

Contributors: Arun Thangamani, Vincent Loechner, Stephane Genaud

Description: The publicly available docker script sets the docker environment for seven general-purpose polyhedral compilers and builds these compilers in a Linux environment. Those seven polyhedral compilers are: Pluto, PoCC, Polly, Graphite, PolyOpt, PPCG, and Polygeist.

It contains the PolyBench/C benchmarks to test them and measure the execution time of the resulting codes on CPU architectures.

Project link: https://github.com/vincentloechner/PolyhedralCompilers

Publications: [15]

Contact: loechner@unistra.fr

8 New results

8.1 Semi-automatic and assisted code optimization

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

Participants: Arthur Charguéraud, Guillaume Bertholon, Thomas Koehler.

In 2024, we pursued the development of the OptiTrust prototype framework for producing high-performance code via source-to-source transformations. We developed a type system inspired by separation logic for analyzing the resource usage of every instruction and every subexpression. We leverage this type system to guide and to check the correctness of source-to-source program transformations. We demonstrated the applicability of OptiTrust on three major case studies: OpenCV's box-blur, TVM's matrix multiply, and a simplified Particle-In-Cell simulation. We have presented preliminary results at the French JFLA'24 workshop [24] as well as at the SOAP workshop associated with PLDI'24 [19]. We have presented a description of OptiTrust' bidirectional translation between C and its internal lambda-calculus at JFLA'25 [18]. We have written a 60-page paper that describes OptiTrust, and have recently submitted it for publication at a top-tier journal.

8.1.2 Specx: A C++ Task-Based Runtime System for Heterogeneous Distributed Architectures

Participants: Bérenger Bramas.

In 2024, Bérenger Bramas and Paul Cardosi completed and submitted the paper presenting Specx several years after the end of Paul Cardosi's contract (a preprint is available [34]). Specx is now capable of executing task graphs on heterogeneous distributed architectures. It provides an elegant way to define task graphs and describe objects that the runtime system can move or send. We plan to further enhance Specx in 2025 by integrating new speculative execution models.

8.1.3 Efficient GPU Implementation of Particle Interactions with Cutoff Radius and Few Particles per Cell

Participants: Bérenger Bramas.

As part of David Algis's industrial PhD, co-supervised by Emmanuelle Darles and Lilian Aveneau from XLIM and Bérenger Bramas from the CAMUS team, we investigated optimizations for achieving an efficient particle interaction kernel on GPUs. Specifically, we focused on cases where particles are mostly separated by the cutoff radius, resulting in few particles per cell when using a grid-based approach. Such situations happen in SPH fuild simulation, for example. We explored several strategies and demonstrated that shared memory is rarely useful in most cases. Additionally, we proposed a novel prefix sum kernel that allows reducing the data movement compared to the previous state of the art. The results were published in a conference paper [17].

8.1.4 SPC5: An Efficient SpMV Framework Vectorized Using ARM SVE and x86 AVX-512

Participants: Bérenger Bramas.

SPC5 is a library consisting of a sparse matrix storage format and a vectorized SpMV kernel. It was originally designed for AVX-512. As part of Evann Regnault's Master's research project, we improved SPC5 and ported it to ARM SVE. While the sparse matrix format remained unchanged, the vectorized kernels required significant adaptation. The results were published in a journal [14].

8.1.5 Using the Discrete Wavelet Transform for Lossy On-the-Fly Compression of GPU Fluid Simulations

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

In the context of Clément Flint's PhD, we worked on compressing simulation data for a Lattice Boltzmann application. A preliminary work conducted during a summer school in applied mathematics has laid the foundations for the use of Haar wavelets [12]. This work was significantly advanced afterwards during Atoli Huppé's internship, during which the GPU compression kernel was drastically improved. Thanks to this advancement, we were able to run simulations on GPUs that would otherwise have required extensive CPU-GPU data transfers due to memory limitations. The results were published in a journal [13], and Clément Flint defended his thesis [30] on Oct. 2024. We continue to work on the compression kernel to provide an open source package.

8.1.6 Exploiting Ray Tracing Technology Through OptiX to Compute Particle Interactions with Cutoff in a 3D Environment on GPUs

Participants: Bérenger Bramas.

Bérenger Bramas and David Algis worked on utilizing OptiX for neighbor finding in n-body simulations. Several methods were implemented, including two novel approaches based on new geometric patterns. A preprint demonstrates that these methods can achieve significant speedups when the grid is sparse (i.e., when particles are not uniformly distributed) [33].

8.1.7 Recognizing Polynomial Loops in Traces

Participants: Alain Ketterlin.

This work is the continuation and extension of previous work using profiling data to enable dynamic optimizations. The goal is to analyze traces of program executions and extract potentially useful information. One of our past contribution in this domain was an algorithm that recognizes affine loop nests in a trace (usually containing memory use information), from which data dependencies and parallelization opportunities could be derived [8]. The algorithm was designed to be efficient both in execution time and memory usage.

Our work this year has focused on extending our loop recognition algorithm to polynomial characteristics (like instruction counts and ranks). This was allowed by our previous development of a specific representation of integer polynomials [27] which, among other properties, have extremely efficient interpolation algorithms. The rest of the recognition algorithm needed very little adjustments. The result is an purely empirical tool producing "polynomial loops", i.e., loops where all bounds and values are multivariate polynomials in the surrounding loop counters (a model that is, in its full generality, too expressive for our current analysis and optimization abilities). The long term goal is to use such profiling techniques to enable sophisticated dynamic optimization techniques, similar to the current static techniques developped in our team (see for instance the work of Philippe Clauss). This research has been published in early 2025 [26].

8.2 Program Verification

8.2.1 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. Our recent work has focused on generalizing our previous results to handle concurrent 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. Our journal paper describing the results will appear in the journal TOPLAS in 2025.

8.3 Fully-automatic code optimization

8.3.1 Algebraic tiling

Participants: Philippe Clauss, Clément Rossetti.

We propose a new loop tiling approach based on the volumes of the tiles, *i.e.*, the number of iterations delimited by the tiles, instead of the sizes of standard (hyper-)rectangular tiles, *i.e.*, the sizes of the edges of the tiles. In the proposed approach, tiles are dynamically generated and have almost equal volumes, even if their shape and edge sizes may differ. The iteration domain is well covered by a minimum number of tiles that are all almost full. Since the bounds of the generated tiles are not linear and defined by algebraic mathematical expressions, we call this loop tiling technique *algebraic tiling* [52] [28]. It uses the mathematical engine TRAHRHE also developed in the team.

Algebraic tiles are built by successive hierarchical slicing of the initial iteration domain, from the outermost to the innermost depth dimensions of the target loop nest, in a way ensuring that slices have

all quasi-equal volumes. The bounds of the loop nests that are handled must be constants, or linear functions of the surrounding loop iterators and of unknown parameters – which are typically related to the data input size. Such loops are also called polyhedral loops since they may be handled using the polyhedral model. Quasi-perfect load balancing is achieved when each parallel loop is sliced using as many slices of quasi-equal volumes as parallel threads, and when most of the iterations have close execution times. Thus, such dynamic slicing strategy makes the resulting parallel loop scalable regarding the number of threads. Good data locality is reached by slicing profitably the non-parallelized loops, and by slicing the parallel loops in a number of parts equal to a multiple of the number of parallel threads. The number of generated slices for each dimension may stay as a parameter at compile-time, making algebraic tiling a parameterized loop tiling technique, and allowing the produced code to adapt to the number of parallel threads and data layout.

We are currently implementing algebraic tiling as an extension of the automatic loop optimizer Pluto (Clément Rossetti's PhD work). Our experiments show that algebraic tiling outperforms significantly (hyper-)rectangular tiling when parallelizing loops with OpenMP using static scheduling, and mostly provides similar or lower execution times when compared to traditionally tiled loops parallelized using dynamic scheduling of OpenMP. Thus, algebraic tiling makes dynamic scheduling fairly purposeless for the handled loop nests.

8.3.2 Dynamic Task Scheduling with Multiple Priorities on Heterogeneous Computing Systems

Participants: Hayfa Tayeb, Bérenger Bramas.

In the context of Hayfa Tayeb's PhD, we worked on improving our priority-based scheduler in StarPU. We presented an initial version [22], in which we demonstrated that our automatic strategy was promising. In this version, we were able to automatically assign different priorities to all tasks, using different heuristics, allowing them to be ordered differently for each processing unit. Using this method, tasks could be sorted based on criteria such as criticality, locality, speedup, etc. We continue to work on this scheduler to reduce its overhead, build better heuristics and optimize energy [48].

8.3.3 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 worked 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 [47].

8.3.4 Automatic Task-Based Parallelization using Source to Source Transformations

Participants: Marek Felsoci, Bérenger Bramas, Stéphane Genaud.

We extended our approach to automatically parallelize any application using the task-based method. We reimplemented APAC using OptiTrust (developed in the team), where source code transformations can be written in a compact way. We solved several challenges related to explicit synchronizations, dependency definitions for arrays, code duplication to have both a sequential and a parallel version. A journal paper is in preparation.

8.3.5 Ionic Models Code Generation for Heterogeneous Architectures

Participants: Arun Thangamani, Atoli Huppé, Vincent Loechner, Stephane Genaud, Bérenger Bramas, Thai Hoa Trinh.

We participate in the research and development of a cardiac electrophysiology simulator called openCARP (7.1.3) 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. In 2023 we have extended the capabilities of OpenCARP for generating multiple parallel versions of the ionic simulation, hence enabling the exploitation of the various parallel computing units that are available in the target architecture nodes (SIMD, multicore, GPU, etc.). This work has been published in the proceedings of two international conferences in 2023.

In 2024, we have collaborated with members of the STORM team (Inria Bordeaux), also implied in the MICROCARD project, to further extend the capability of executing simulations simultaneously on CPU cores and mulitple GPUs. For this purpose, we rely on the StarPU runtime distributed environment. This work has been published in an IPDPS Workshop [25] and a journal paper is under preparation.

We have also started to investigate if we could generalize our work on the vectorization of the computations involved in the ionic model. We have first reviewed the existing polyhedral compilers to see if any could be fitted to implement some automatic transformations using MLIR. This work led us to first write a full survey paper published in TACO [15]. Then, at the end of Arun Thangamani's thesis, a preliminary work regarding modifications to the PolyGeist polyhedral compiler has been presented at the PhD symposium of the EuroPar conference [23]. Arun Thangamani defended his thesis on September 2024 [31].

The MICROCARD project ended in November 2024 but the proposal of a continuation project called MICROCARD-2 (see 9.2.1) has been accepted and started on November 2024, for a duration of 30 months. We will recruit an engineer and a young researcher in 2025. The CAMUS members' tasks in this new project will address low-precision optimizations (reduce the floating-point precision when full precision is not necessary), energy consumption optimization and power capping, and experiment with new architectures providing the Scalable Vector Extension (SVE), for example ARM. We are also implied in the development of new higher order time stepping schemes with our collegues from the University of Bordeaux and ZIB (Berlin).

8.3.6 Rewriting for Specialized CGRAs.

Participants: Thomas Koehler.

Coarse-grained reconfigurable arrays (CGRAs) are specialized accelerators promising both the flexibility of Field-Programmable Gate Arrays (FPGAs) and the performance of Application-Specific Integrated Circuits (ASICs). However, with specialization comes a real danger: designing chips that cannot accelerate enough current and future software to justify the hardware cost, wasting chip area by being almost always switched off.

Thomas Koehler worked with Jackson Woodruff (University of Edinburgh) on developing FlexC [53], the first framework designed to support heterogeneous CGRAs, which builds rewrite rules that enable CGRAs to be adapted to programs they do not natively support. FlexC uses a novel algorithm based on equality saturation to generate CGRA-specific rewrite systems from simple, generic rewrite systems. With these rewrites, FlexC uses dataflow rewriting to replace unsupported operations with equivalent operations that are supported by the CGRA, enabling code that would otherwise be un-acceleratable to be accelerated.

FlexC was applied to over 2,000 loop kernels, compiling to three different research CGRAs, achieving a 1.4× increase in the number of loop kernels accelerated leading to a 1.6× geomean speedup compared to an Arm A5 CPU on kernels that would otherwise not be accelerated.

8.4 Fundamental algorithms & mathematical tools

8.4.1 Trahrhe expressions

Participants: Philippe Clauss, Marceau Noury.

In the mid-1990s, Philippe Clauss and Vincent Loechner introduced the mathematical theory of Ehrhart polynomials in computer science for the quantitative analysis of iterative programs [2, 5]. These special mathematical objects give the exact number of integer points contained in a polyhedron depending linearly on parameters. In the context of polyhedral modeling of nested loops, this number can correspond to the total number of iterations, the number of parallel iterations, the number of accessed data, etc.

A particular use of these Ehrhart polynomials are *ranking polynomials*. Such polynomials give the position, or rank, of an iteration of a loop nest, according to the lexicographic order of execution of the iterations. These polynomials are determined by calculating the number of integer points lexicographically inferior to any point in the polyhedral domain of the iterations. Philippe Clauss has shown a first application of such polynomials to data layout transformation for optimal spatial locality in 2000.

More recently, we have been interested in inverting such ranking polynomials, in order to be able to determine, for a given rank, what are the corresponding loop indices. This unranking problem is particularly challenging from a theoretical and practical point of view. Thanks to the specific properties of ranking polynomials, we have developed a method for inverting such polynomials by solving uni-variate polynomial equations and propagating the integer floors of the roots to lower dimensions [3].

Since 2019, the mathematical engine computing Trahrhe expressions has been developed as a software (TRAHRHE) usable for several loop optimization purposes, as non-rectangular loop collapsing [3] or algebraic loop tiling [52]. A completely revised version written in C++ and implementing many improvements is currently being developed by Marceau Noury and will be published soon.

8.4.2 Polyhedral Scheduling

Participants: Tom Hammer, Arun Thangamani, Vincent Loechner, Stephane Genaud, Alain Ketterlin.

Scheduling is the central operation in the polyhedral compilation chain, to find the best execution order of loop iterations for parallelizing and optimizing the code. Discovering the best polyhedral schedules remains a challenge due to the huge search space. Moreover, current classes of polyhedral schedulers proceed from outer to inner loops, making them unpractical for enforcing efficient vectorization in innermost loops. We showed those limitations in our survey on polyhedral compilers [15], that will be presented at the HiPEAC 2025 conference.

Cédric Bastoul is on leave since April 2020. 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 like SLP (superword-level parallelism), and to agglomerate statement instances into ”super” loops discovered by our trace analysis tool NLR. We presented this promising new scheduling technique at the IMPACT’23 workshop. We started to implemented the technique as a set of passes in the LLVM compiler (Tom Hammer’s PhD, since September 2023).

8.4.3 Improvements of the C programming language

Participants: Jens Gustedt.

The final work on the new revision of the C standard, C23, took place in the plenary meeting of ISO PL1/SC22/WG14 that we hosted in Strasbourg in January 2024; about 200 final comments from the

National Bodies (NB) were reviewed and the document was forwarded to the editors for finalization. The standard was then published in autumn 2024 by ISO [51]. Over all, 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.

We now have started to discuss additions to the next version of the C standard, coined C2y at the moment. The discussion on these new features took place in three plenary meetings, one online and two face-to-face (Strasbourg and Minneapolis).

In 2024 we contributed with several papers to the future revision. We contributed to the following subjects:

- the concepts of literals, constants and reproducible expressions [44][41][43],
- improvement array syntax and semantics [36], [46],
- improvement of some problem spots [37][39],
- continued work on function attributes [35],
- the new **defer** feature [38],
- preprocessor recursion [45],
- the removal of imaginary types and the introduction of complex literals [42][40].

In addition to the C standard, we finalized the technical specification TS 6010 (see for example [50]) for a sound and verifiable memory model that is based on provenance. 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. As a result, this TS will only be published in 2025.

To promote the new C standard, we also prepared a C23 edtion of the book Modern C, [29, 49] for which we published the online version with great success, with 40000+ downloads as of January 2025.

8.4.4 Fast Counting, Ranking and Rank Inversion

Participants: Alain Ketterlin.

Some recent advances in loop optimization focus on balancing the load of parallel threads executing the various sub-domains of a loop's iterations [52]. Such optimizations make heavy use of instructions counts for various fragments of the original code, be they sub-loops or sub-intervals of an iteration domain. They also need "random access" to an arbitrary instruction given by its rank in the global execution. Finally they need to be able to map a sequential rank back onto an iteration vector, so as to be able to start an arbitrary chunk of the full execution. Known solutions to these problems use sophisticated algorithms, and unconventional mathematical objects (like quasi-polynomials) to perform their task.

We have developed simpler, lightweight techniques to solve such problems. Taylored to a specific sub-class of loops, our algorithms are fast and do not rely on external libraries at all. Even though some generality is lost, all algorithms are still widely applicable (for instance, they cover all loops allowed by the OpenMP standard). Their core simplicity comes from an original representation of integer polynomials, which proves to be very well fitted for the essential counting (and ranking) requirements. The paper presenting these results includes a complete proof-of-concept implementation of our algorithms, as well as various utilities designed for easy experimentation. It was presented at IMPACT'25 [27].

8.4.5 Typechecking of Overloading

Participants: Arthur Charguéraud.

In joint work with Martin Bodin from Inria Grenoble and Jana Dunfield from Queen's School of Computing (Canada), Arthur Charguéraud has been working on a typechecking algorithm for resolving overloaded symbols. This work applies both to programming languages and mechanized mathematics. A prototype implementation has been developed. We are currently proving theorems and optimizing the implementation, towards a journal article submission. Preliminary results have been presented at the JFLA'25 French workshop [20].

8.4.6 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 published at POPL'2024 [21].

9 Partnerships and cooperations

9.1 International research visitors

9.1.1 Visits of international scientists

Other international visits to the team

Rachid Seghir

Status researcher (professor)

Institution of origin: University of Batna

Country: Algeria

Dates: Dec. 16-20, 2024

Context of the visit: Collaborative research on high performance scientific applications: cardiac simulation and weather forecast

Mobility program/type of mobility: research stay

9.2 European initiatives

9.2.1 Horizon Europe

MICROCARD-2 Centre of Excellence (EuroHPC and ANR)

Participants: Vincent Loechner, Stephane Genaud.

MICROCARD-2 on EuroHPC-Ju

Title: MICROCARD-2: numerical modeling of cardiac electrophysiology at the cellular scale

Duration: from November 1, 2024 to April 30, 2027

Partners: Inria, France; Karlsruher Institut Für Technologie, Germany; Megware, Germany; Simula Research Laboratory (Simula), Norway; Technical University München (TUM), Germany; Università degli Studi di Pavia, Italy; Università di Trento (UTrento), Italy; Université de Bordeaux, France; Université de Strasbourg, France.

Inria contact:

Coordinator: Mark POTSE, Université de Bordeaux

WP4 leader: Vincent Loechner

Summary: The MICROCARD-2 project is coordinated by Université de Bordeaux and involves the Inria teams CARMEN, STORM, and TADAAM in Bordeaux and CAMUS in Strasbourg, among a total of ten partner institutions in France, Germany, Italy, and Norway. This Centre of Excellence for numerical modeling of cardiac electrophysiology at the cellular scale builds on the MICROCARD project (2021–2024) and has the same website.

The modelling of cardiac electrophysiology at the cellular scale requires thousands of model elements per cell, of which there are billions in a human heart. Even for small tissue samples such models require at least exascale supercomputers. In addition the production of meshes of the complex tissue structure is extremely challenging, even more so at this scale. MICROCARD-2 works, in concert, on every aspect of this problem: tailored numerical schemes, linear-system solvers, and preconditioners; dedicated compilers to produce efficient system code for different CPU and GPU architectures (including the EPI and other ARM architectures); mitigation of energy usage; mesh production and partitioning; simulation workflows; and benchmarking.

The contribution of the CAMUS team concerns code optimization of the ionic models, and implies the MLIR compiler frontend and SIMD code generation for CPUs, plus GPU (Nvidia and AMD) code generation. An engineer and a junior researcher will be hired from January 2025.

9.2.2 H2020 projects

MICROCARD (EuroHPC and ANR)

Participants: Vincent Loechner, Stephane Genaud, Arun Thangamani, Atoli Huppe.

MICROCARD project on cordis.europa.eu

Title: Numerical modeling of cardiac electrophysiology at the cellular scale

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 (ZIB), 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.

Coordinator: Mark POTSE (University of Bordeaux)

WP6 leader: Vincent Loechner

Summary: Numerical models of cardiac electrophysiology need to move from a continuum approach to a cell-by-cell approach to match observations in aging and diseased hearts. Exascale computers will be needed to run such models. The application is co-designed by HPC experts, numerical scientists, biomedical engineers, and biomedical scientists, from academia and industry. We developed, 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. Our main development platform is the openCARP simulator.

This project made it possible to hire one PhD student and one engineer in the CAMUS team since 2021. It was co-funded by EuroHPC JU (50%) and by ANR (50%).

9.3 National initiatives

9.3.1 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 2027
- Coordinator: Arthur Charguéraud (Inria)
- Partners: Inria team Camus (Strasbourg), Inria team TONUS (Strasbourg), Inria team Cambium (Paris), Inria team CASH (Lyon), CEA team LIST

9.3.2 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

9.3.3 Exa-SofT project, PEPR NumPEx

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

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

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 international conferences ICFP 2024 and CPP 2024, and the workshops CoqPL 2024 and JFLA 2024.
- Marek Felsoci was co-organizer of a three-day workshop on Guix [32].

Reviewer

- Stéphane Genaud was a reviewer for the IEEE International Conference Cluster 2024.
- Thomas Koehler was a judge for the PLDI 2024 Student Research Competition; he also was a reviewer for the EGRAPHS 2024 workshop and ICFP 2024 conference.

10.1.2 Journal

Reviewer - reviewing activities

- Arthur Charguéraud was a reviewer for the Journal of Functional Programming (JFP).
- Philippe Clauss was a reviewer for the Journal of Software: Practice and Experience, ACM Transactions on Architecture and Code Optimization (TACO).
- Bérenger Bramas was a reviewer for IPDPS 2025, Transactions on Architecture and Code Optimization (TACO), Supercomputing (journal of), Mathematics, Cluster Computing, Software: Practice and Experience (SPE), Future Generation Computer Systems, Big Data Journal, Journal of Open Source Software, Parallel Processing Letters (PPL), and Scientific Reports.
- Thomas Koehler was a reviewer for the IEEE Access journal in 2024.

10.1.3 Invited talks

- Vincent Loechner gave a keynote [16] at the C3PO workshop (Workshop on Compiler-assisted Correctness Checking and Performance Optimization for HPC), in conjunction with ISC 2024, Hamburg, Germany.
- Jens Gustedt gave an invited talk at the *séminaire de fiabilité logicielle du CNES*, Paris.

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.

10.1.5 Research administration

- Stéphane Genaud is the head of the ICPS team for the ICube lab. Arthur Charguéraud is vice-head.
- 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.
- Jens Gustedt is (together with Philippe Helluy of the IRMA lab) responsible for the Inria PIQ program for the Strasbourg site.
- 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.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Licence: Philippe Clauss, Computer architecture, 18h, L2, Université de Strasbourg, France
- Master: Philippe Clauss, Compilation, 132h, 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: 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, Parallel programming, 32h, M1, 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
- TODO
- Master: Bérenger Bramas, Compilation and Performance, 24h, M2, Université de Strasbourg, France
- Master: Bérenger Bramas, Compilation, 24h, M1, Université de Strasbourg, France
- Licence: Alain Ketterlin, Culture et pratique de l'Informatique, L1 Math-Info, 48h, Université de Strasbourg, France
- Licence: Alain Ketterlin, Programmation système, L2 Math-Info, 40h, Université de Strasbourg, France

- Licence: Alain Ketterlin, Algorithmique et programmation, L1 Math-Info, 66h, Université de Strasbourg, France
- Licence: Alain Ketterlin, Software Engineering (an Anglais), L2 Math-Info, 64h, Université de Strasbourg, France
- Licence: Stéphane Genaud, Algorithmics and programmation, 82h, L1, Université de Strasbourg, France
- Licence: Stéphane Genaud, Data Structures & Algorithms 2, 25h, L2, UFAZ, France-Azerbadjian
- Licence: Stéphane Genaud, Parallel programming, 30h, M1, 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
- Corps des mines: Arthur Charguéraud, Design and Implementation of Educational Software, Paris, 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 in Computer Science and co-head of Master Data Science and Artificial Intelligence at UFAZ (Baku, Azerbadjian) who delivers Unistra diplomas. Since resp. Aug. 2023 and Aug. 2024.
- 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, advised by Philippe Clauss and Marc Pérache (CEA), since Oct. 2023.
- PhD in progress: Raphaël Colin, Runtime multi-versioning of parallel tasks, advised by Philippe Clauss and Thierry Gautier (Inria project-team Avalon), since Oct. 2023.
- PhD in progress: Guillaume Bertholon, Formal Verification of Source-to-Source Transformations, is advised by Arthur Charguéraud, since Sept 2022.
- 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: 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: Tom Hammer, Synergie entre ordonnancement et optimisation des accès mémoire dans le modèle polyédrique, advised by Stéphane Genaud and Vincent Loechner, since Sept 2023.
- PhD completed: Arun Thangamani, Code generation for heterogeneous architectures, advised by Stéphane Genaud and Vincent Loechner, defended Sept 2024.
- PhD completed: Clément Flint, Efficient data compression for high-performance PDE solvers., advised by Philippe Helluy (Inria project-team Tonus), Stéphane Genaud, and Bérenger Bramas, defended Oct 2024.
- PhD completed: Alexandre Moine, Formal Verification of Space Bounds, is co-advised by Arthur Charguéraud and François Pottier (Inria Paris), defended Sept 2024.

10.2.3 Juries

- Philippe Clauss was a reviewer for:
  - the PhD thesis of Hugo Thievenaz, defended on Dec. 18, 2024, at ENS Lyon.
  - the PhD thesis of Richard Sartori, defended on Dec. 19, 2024, at the University of Bordeaux.

10.3 Popularization

10.3.1 Specific official responsibilities in science outreach structures

- Arthur Charguéraud is co-founder and vice-president of the non-profit organization France-ioi. This organization is in charge of the French participation to international olympiads in informatics. It also organizes numerous contests, such as the Concours Castor, Concours Algorea, concours Alkindi, and the French Olympiads in Informatics.
- 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. 650,000 teenagers played with the interactive exercises in November and December 2024.

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 (cit. on p. 17).

[3] P. Clauss, E. Altintas and M. Kuhn. 'Automatic Collapsing of Non-Rectangular Loops'. In: Parallel and Distributed Processing Symposium (IPDPS), 2017. Orlando, United States: IEEE International, 29th May 2017, pp. 778–787. DOI: 10.1109/IPDPS.2017.34. URL: https://inria.hal.science/hal-01581081 (cit. on p. 17).

[4] 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.

[5] P. Clauss and V. Loechner. 'Parametric Analysis of Polyhedral Iteration Spaces'. In: *Journal of Signal Processing Systems* 19.2 (July 1998), pp. 179–194. DOI: 10.1023/A:1008069920230. URL: https://inria.hal.science/inria-00534840 (cit. on p. 17).

[6] 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.

[7] A. Jimborean, P. Clauss, J.-F. Dollinger, V. Loechner and M. Juan Manuel. 'Dynamic and Speculative Polyhedral Parallelization Using Compiler-Generated Skeletons'. In: *International Journal of Parallel Programming* 42.4 (Aug. 2014), pp. 529–545. URL: https://hal.inria.fr/hal-01003744.

[8] 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 (cit. on p. 14).

[9] A. Ketterlin and 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, Dec. 2012. URL: https://hal.inria.fr/hal-00780782.

[10] B. Pradelle, A. Ketterlin and P. Clauss. 'Polyhedral parallelization of binary code'. In: *ACM Transactions on Architecture and Code Optimization*. Special issue on high-performance and embedded architectures and compilers 8.4 (Jan. 2012), 39:1–39:21. DOI: 10.1145/2086696.2086718. URL: https://hal.inria.fr/hal-00664370.

[11] 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

[12] C. Flint and P. Helluy. 'Reducing the memory usage of Lattice-Boltzmann schemes with a DWT-based compression'. In: *ESAIM: Proceedings and Surveys* 77 (18th Nov. 2024), pp. 79–99. DOI: 10.48550/arXiv.2302.09883. URL: https://inria.hal.science/hal-03990880 (cit. on p. 13).

[13] C. Flint, A. Huppé, P. Helluy, B. Bramas and S. Genaud. 'Using the Discrete Wavelet Transform for Lossy On-the-Fly Compression of GPU Fluid Simulations'. In: *International Journal for Numerical Methods in Fluids* (27th Oct. 2024). DOI: 10.1002/fld.5344. URL: https://inria.hal.science/hal-04582282 (cit. on p. 13).

[14] E. Regnault and B. Bramas. 'SPC5: an efficient SpMV framework vectorized using ARM SVE and x86 AVX-512'. In: *Computer Science and Information Systems* Vol. 21.No. 1 (1st Jan. 2024), pp. 203–221. DOI: 10.48550/arXiv.2307.14774. URL: https://inria.hal.science/hal-04172241 (cit. on p. 13).

[15] A. Thangamani, V. Loechner and S. Genaud. 'A Survey of General-purpose Polyhedral Compilers'. In: *ACM Transactions on Architecture and Code Optimization* (22nd June 2024). DOI: 10.1145/3674735. URL: https://inria.hal.science/hal-04626373 (cit. on pp. 12, 16, 17).

Invited conferences

[16] V. Loechner. 'Optimized Code Generation of Electrophysiology Kernels using MLIR'. In: Workshop on Compiler-assisted Correctness Checking and Performance Optimization for HPC (C3PO). Hambourg, Germany, 16th May 2024. URL: https://inria.hal.science/hal-04715152 (cit. on p. 23).

International peer-reviewed conferences

[17] D. Algis, B. Bramas, E. Darles and L. Aveneau. 'Efficient GPU Implementation of Particle Interactions with Cutoff Radius and Few Particles per Cell'. In: International Symposium on Parallel Computing and Distributed Systems (PCDS2024). Singapore, Singapore, 23rd June 2024. URL: https://inria.hal.science/hal-04621128 (cit. on p. 13).

[18] G. Bertholon and A. Charguéraud. 'Bidirectional Translation between a C-like Language and an Imperative Lambda-calculus'. In: 36es Journées Francophones des Langages Applicatifs (JFLA 2025). Roiffé, France, 28th Jan. 2025. URL: https://inria.hal.science/hal-04859522 (cit. on p. 12).

[19] G. Bertholon, A. Charguéraud, T. Koehler, B. Bytyqi and D. Rouhling. 'Interactive Source-to-Source Optimizations Validated using Static Resource Analysis'. In: *Proceedings of the 13th ACM SIGPLAN International Workshop on the State Of the Art in Program Analysis (SOAP'24)*. SOAP 2024 - Workshop - PLDI 2024. Copenhagen, Denmark, 24th June 2024. DOI: 10.1145/3652588.3663320. URL: https://inria.hal.science/hal-04604636 (cit. on p. 12).

[20] A. Charguéraud, M. Bodin and L. Riboulet. 'Typechecking of Overloading in Programming Languages and Mechanized Mathematics'. In: 36es Journées Francophones des Langages Applicatifs (JFLA 2025). Roiffé, France, 28th Jan. 2025. URL: https://inria.hal.science/hal-04859446 (cit. on p. 19).

[21] 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 (cit. on pp. 6, 19).

[22] H. Tayeb, B. Bramas, M. Faverge and A. Guermouche. 'Dynamic Tasks Scheduling with Multiple Priorities on Heterogeneous Computing Systems'. In: *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 38th IEEE International Parallel & Distributed Processing Symposium. San franscisco, CA, United States, 2024, pp. 31–40. DOI: 10.1109/IPDPSW63119.2024.00014. URL: https://hal.science/hal-04498634 (cit. on p. 15).

[23] A. Thangamani, V. Loechner and S. Genaud. 'Extending Polygeist to Generate OpenMP SIMD and GPU MLIR Code'. In: *LNCS*. 30th International European Conference on Parallel and Distributed Computing - PhD Symposium. Madrid, Spain, 1st May 2025. URL: https://inria.hal.science/hal-04711965 (cit. on p. 16).

National peer-reviewed Conferences

[24] G. Bertholon, A. Charguéraud and T. Koehler. 'Source-to-Source Optimizations Validated using Separation Logic'. In: 35es Journées Francophones des Langages Applicatifs (JFLA 2024). Saint-Jacut-de-la-Mer, France, Jan. 2024. URL: https://inria.hal.science/hal-04406229 (cit. on p. 12).

Conferences without proceedings

[25] V. Alba, O. Aumage, D. Barthou, R. Colin, M.-C. Counilh, S. Genaud, A. Guermouche, V. Loechner and A. Thangamani. 'Performance portability of generated cardiac simulation kernels through automatic dimensioning and load balancing on heterogeneous nodes'. In: PDSEC 2024. San Francisco (CA, USA), United States, 31st May 2024. DOI: 10.1109/IPDPSW63119.2024.00171. URL: https://hal.science/hal-04606388 (cit. on p. 16).

[26] A. Ketterlin. 'Easy Counting and Ranking for Simple Loops'. In: IMPACT 2024 – 14th International Workshop on Polyhedral Compilation Techniques. Münich, Germany, 17th Jan. 2024. URL: https://inria.hal.science/hal-04922024 (cit. on p. 14).

[27] A. Ketterlin. 'Polynomial Loop Recognition in Traces'. In: IMPACT 2025 – 15th International Workshop on Polyhedral Compilation Techniques. Barcelona, Spain, 22nd Jan. 2025. URL: https://inria.hal.science/hal-04922050 (cit. on pp. 14, 19).

[28] 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 (cit. on p. 14).

Scientific books

[29] J. Gustedt. *Modern C*. Manning, 2024. URL: https://inria.hal.science/hal-02383654. In press (cit. on p. 18).

Doctoral dissertations and habilitation theses

[30] C. Flint. 'Efficient data compression for high-performance PDE solvers'. Université de Strasbourg, 2nd Oct. 2024. URL: https://theses.hal.science/tel-04909284 (cit. on p. 13).

[31] A. Thangamani. 'Optimized Code Generation for Parallel and Polyhedral Loop Nests using MLIR'. Strasbourg University, 25th Sept. 2024. URL: https://inria.hal.science/tel-04718259 (cit. on p. 16).

Reports & preprints

[32] C. Acary-Robert, E. Agullo, L. Courtès, M. Felšöci, K. Hinsen, A. Isaac, O. Lünsdorf, P. Prins, S. Tournier, P. Virouleau and R. Wurmus. *Guix-HPC Activity Report 2022–2023*. Inria Bordeaux - Sud Ouest, Feb. 2024, pp. 1–32. URL: https://inria.hal.science/hal-04500140 (cit. on p. 23).

[33] D. Algis and B. Bramas. *Exploiting ray tracing technology through OptiX to compute particle interactions with cutoff in a 3D environment on GPU*. Inria, 26th Aug. 2024. URL: https://inria.hal.science/hal-04677813 (cit. on p. 13).

[34] P. Cardosi and B. Bramas. *Specx: a C++ task-based runtime system for heterogeneous distributed architectures*. inria, 15th Nov. 2024. URL: https://inria.hal.science/hal-04191350 (cit. on p. 12).

[35] J. Gustedt. *Clarify status of non-returning functions with respect to function attributes*. N3424. ISO JCT1/SC22/WG14, 23rd Dec. 2024. URL: https://inria.hal.science/hal-04866903 (cit. on p. 18).

[36] J. Gustedt. *Clarify syntactic terms for array declarators*. N3414. ISO JCT1/SC22/WG14, 19th Nov. 2024. URL: https://inria.hal.science/hal-04794159 (cit. on p. 18).

[37] J. Gustedt. *Clarify the specification of the width macros*. N3413. ISO JCT1/SC22/WG14, 1st Dec. 2024. URL: https://inria.hal.science/hal-04866890 (cit. on p. 18).

[38] J. Gustedt. *Even simpler defer for direct integration*. N3434. ISO JCT1/SC22/WG14, 23rd Dec. 2024. URL: https://inria.hal.science/hal-04866911 (cit. on p. 18).

[39] J. Gustedt. *Inline functions accessing identifiers declared with constexpr*. N3253. ISO JCT1/SC22/WG14, 5th May 2024. URL: https://inria.hal.science/hal-04737111 (cit. on p. 18).

[40] J. Gustedt. *Introduce complex literals*. N3298. ISO JCT1/SC22/WG14, 11th July 2024. URL: https://inria.hal.science/hal-04737134 (cit. on p. 18).

[41] J. Gustedt. *Objects of known constant size*. N3391. ISO JCT1/SC22/WG14, 1st Dec. 2024. URL: https://inria.hal.science/hal-04794146 (cit. on p. 18).

[42] J. Gustedt. *Remove imaginary types*. N3274. ISO JCT1/SC22/WG14, 16th June 2024, p. 12. URL: https://inria.hal.science/hal-04737051 (cit. on p. 18).

[43] J. Gustedt. *Reproducible expressions*. N3392. ISO JCT1/SC22/WG14, 1st Dec. 2024. URL: https://inria.hal.science/hal-04794151 (cit. on p. 18).

[44] J. Gustedt. *Some constants are literally literals*. N3239. ISO JCT1/SC22/WG14, 14th Apr. 2024. URL: https://inria.hal.science/hal-04358367 (cit. on p. 18).

[45] J. Gustedt. *Tail recursion for preprocessor macros*. N3307. ISO JCT1/SC22/WG14, 5th Aug. 2024. URL: https://inria.hal.science/hal-04737180 (cit. on p. 18).

[46] J. A. Múgica and J. Gustedt. *Array subscripting without decay*. N3380. ISO JCT1/SC22/WG14, 14th Oct. 2024. URL: https://inria.hal.science/hal-04737209 (cit. on p. 18).

[47] E. Ndamlabin and B. Bramas. *RSCHED: An effective heterogeneous resources management for simultaneous execution of task-based applications*. 6th May 2024. URL: https://hal.science/hal-04570168 (cit. on p. 15).

[48] A. d'Aviau de Piolant, H. Tayeb, B. Bramas, M. Faverge, A. Guermouche and A. Guermouche. *Improving energy efficiency of HPC applications using unbalanced GPU power capping*. 11th Oct. 2024. URL: https://inria.hal.science/hal-04883872 (cit. on p. 15).

Software

[49] [SW] J. Gustedt, *Code examples for the book Modern C, revised for C23* version This is the updated version for the C23 edition of Modern C, 15th Oct. 2024. LIC: MIT License. HAL: ⟨hal-03345464⟩, URL: https://inria.hal.science/hal-03345464, VCS: https://gitlab.inria.fr/gustedt/modern-c, SWHID: ⟨swh:1:dir:23d45aab396fc1e38ef76b3d456ccb2eefaa6ebe;origin=https://hal.archives-ouvertes.fr/hal-03345464;visit=swh:1:snp:dfb5a473fb881a4e018ac4becde21dd2e47cabf7;anchor=swh:1:rel:93662711ab1afcafb53259cd09896b3d059ffb21;path=/⟩ (cit. on p. 18).

11.3 Cited publications

[50] J. Gustedt, P. Sewell, K. Memarian, V. B. F. Gomes and M. Uecker. *A Provenance-aware Memory Object Model for C*. Draft Technical Specification, proposed to the national bodies for standardization. ISO/IEC TC1/SC22/WG14, June 2022, p. 131. URL: https://inria.hal.science/hal-02957464 (cit. on p. 18).

[51] *ISO/IEC IS 9899:2024: Programming languages - C*. Oct. 2024, p. 758 (cit. on pp. 6, 18).

[52] C. Rossetti and P. Clauss. 'Algebraic Tiling'. In: *IMPACT 2023, 13th International Workshop on Polyhedral Compilation Techniques*. Toulouse, France, Jan. 2023. URL: https://inria.hal.science/hal-03944790 (cit. on pp. 14, 17, 18).

[53] J. Woodruff, A. Brauckmann, S. Ainsworth, T. Koehler, C. Cummins and M. F. P. O'Boyle. 'Rewriting History: Repurposing Domain-Specific CGRAs'. working paper or preprint. Dec. 2024. URL: https://inria.hal.science/hal-04826743 (cit. on p. 16).