IN PARTNERSHIP WITH:
Université de Strasbourg

Activity Report 2019

Project-Team CAMUS

Compiling Architectures of Multicores

IN COLLABORATION WITH: ICube

RESEARCH CENTER
Nancy - Grand Est

THEME
Architecture, Languages and Compilation

Table of contents

Project-Team CAMUS

Creation of the Team: 2009 July 01, updated into Project-Team: 2019 March 01

Keywords:

Computer Science and Digital Science:

A1.1.1. - Multicore, Manycore
A1.1.4. - High performance computing
A2.1.1. - Semantics of programming languages
A2.1.6. - Concurrent programming
A2.2.1. - Static analysis
A2.2.4. - Parallel architectures
A2.2.5. - Run-time systems
A2.2.6. - GPGPU, FPGA...
A2.2.7. - Adaptive compilation

Other Research Topics and Application Domains:

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

1. Team, Visitors, External Collaborators

Research Scientists

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

Faculty Members

Philippe Clauss [Team leader, Univ de Strasbourg, Professor, HDR]
Cédric Bastoul [Univ de Strasbourg, Professor, HDR]
Alain Ketterlin [Univ de Strasbourg, Associate Professor]
Vincent Loechner [Univ de Strasbourg, Associate Professor]
Nicolas Magaud [Univ de Strasbourg, Associate Professor, until Mar 2019]
Julien Narboux [Univ de Strasbourg, Associate Professor, until Mar 2019]
Éric Violard [Univ de Strasbourg, Associate Professor, HDR]

Post-Doctoral Fellow

Damien Rouhling [Inria, from Oct 2019]

PhD Students

Paul Godard [Caldera]
Salwa Kobeissi [Inria]
Harenome Ranaivoarivony-Razanajato [Univ de Strasbourg]
Daniel Salas [INSERM, until Sep 2019]
Maxime Schmitt [Univ de Strasbourg, until Sep 2019]

Technical staff

Muthena Abdul-Wahab [Inria, Engineer, until Jan 2019]
Paul Cardosi [Inria, Engineer, from Nov 2019]

Interns and Apprentices

Marek Felsoci [Univ de Strasbourg, from Feb 2019 until Aug 2019]
Garip Kusoglu [Univ de Strasbourg, from Sep 2019]

2. Overall Objectives

2.1. Overall Objectives

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

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

3. Research Program

3.1. Research Directions

The various objectives we are expecting to reach are directly related to the search of adequacy between the sofware and the new multicore processors evolution. They also correspond to the main research directions suggested by Hall, Padua and Pingali in [56]. Performance, correctness and productivity must be the users' perceived effects. They will be the consequences of research works dealing with the following issues:

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

The development of efficient and correct applications for multicore processors requires stepping in every application development phase, from the initial conception to the final run.

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

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

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

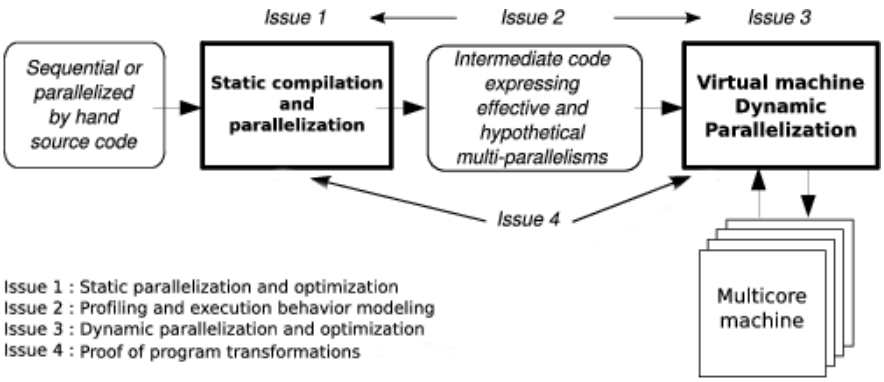

Figure 1. Steps for Automatic parallelization on multicore architectures.

These different parallelization steps are schematized in figure 1.

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

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

3.2. Static Parallelization and Optimization

Participants: Vincent Loechner, Philippe Clauss, Éric Violard, Cédric Bastoul, Arthur Charguéraud, Bérenger Bramas, Harenome Ranaivoarivony-Razanajato.

Static optimizations, from source code at compile time, benefit from two decades of research in automatic parallelization: many works address the parallelization of loop nests accessing multi-dimensional arrays, and these works are now mature enough to generate efficient parallel code [53]. Low-level optimizations, in the assembly code generated by the compiler, have also been extensively dealt with for single-core and require few adaptations to support multicore architectures. Concerning multicore specific parallelization, we propose to explore two research directions to take full advantage of these architectures: adapting parallelization to multicore architectures and expressing many potential parallelisms.

3.3. Profiling and Execution Behavior Modeling

Participants: Alain Ketterlin, Philippe Clauss, Salwa Kobeissi.

The increasing complexity of programs and hardware architectures makes it ever harder to characterize beforehand a given program's run time behavior. The sophistication of current compilers and the variety of transformations they are able to apply cannot hide their intrinsic limitations. As new abstractions like transactional memories appear, the dynamic behavior of a program strongly conditions its observed performance. All these reasons explain why empirical studies of sequential and parallel program executions have been considered increasingly relevant. Such studies aim at characterizing various facets of one or several program runs, *e.g.*, memory behavior, execution phases, etc. In some cases, such studies characterize more the compiler than

the program itself. These works are of tremendous importance to highlight all aspects that escape static analysis, even though their results may have a narrow scope, due to the possible incompleteness of their input data sets.

3.4. Dynamic Parallelization and Optimization, Virtual Machine

Participants: Philippe Clauss, Salwa Kobeissi, Jens Gustedt, Alain Ketterlin, Muthena Abdul-Wahab, Daniel Salas, Bérenger Bramas.

Dynamic parallelization and optimization has become essential with the advent of the new multicore architectures. When using a dynamic scheme, the performed instructions are not only dedicated to the application functionalities, but also to its control and its transformation, and so in its own interest. Behaving like a computer virus, such a scheme should rather be qualified as a "vitamin". It perfectly knows the current characteristics of the execution environment and owns some qualitative information thanks to a behavior modeling process (issue 2). It provides a significant optimization ability compared to a static compiler, while observing the evolution of the availability of live resources.

3.5. Proof of Program Transformations for Multicores

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

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

The optimization process can be separated into two stages: the transformations consisting in optimizing the sequential code and in exhibiting parallelism, and those consisting in optimizing the parallel code itself. The first category of optimizations can be proved within a sequential semantics. For the other optimizations, we need to work within a concurrent semantics. We expect the first stage of optimization to produce data-race free code. For the second stage of optimization we will first assume that the input code is data-race free. We will prove those transformations using Appel's concurrent separation logic [57]. Proving transformations involving programs which are not data-race free will constitute a longer term research goal.

4. Application Domains

4.1. Application Domains

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

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

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

5. Highlights of the Year

5.1. Highlights of the Year

One of the main challenges of parallelization is the selection of the appropriate granularity to balance between the ideal degree of parallelism and the mitigation of the runtime system's overhead. We have worked on the granularity control for parallel applications focusing on two different paradigms. In the first one, which is the tasks with spawn/sync mechanism, we combined the use of asymptotic complexity functions provided by the programmer, with runtime measurements to predict the execution time of tasks with reasonable accuracy. This estimation can then be used to select the proper task granularity, while making sure to put enough work inside each task. In the second one, which is related to the tasks with dependencies paradigm, we have improved an existing algorithm to cluster a graph of tasks to obtain a meta-graph with larger tasks. This approach was used in an application in collaboration with the TONUS team, and we have demonstrated that it allows for a significant speedup.

6. New Software and Platforms

6.1. CLooG

Code Generator in the Polyhedral Model

KEYWORDS: Polyhedral compilation - Optimizing compiler - Code generator

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

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

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

6.2. OpenScop

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

KEYWORDS: Polyhedral compilation - Optimizing compiler

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

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

6.3. ORWL

Ordered Read-Write Lock

KEYWORDS: Task scheduling - Deadlock detection

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

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

6.4. musl

KEYWORDS: Standards - Library

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

musl's efficiency is unparalleled in Linux libc implementations. Designed from the ground up for static linking, musl carefully avoids pulling in large amounts of code or data that the application will not use. Dynamic linking is also efficient, by integrating the entire standard library implementation, including threads, math, and even the dynamic linker itself into a single shared object, most of the startup time and memory overhead of dynamic linking have been eliminated.

FUNCTIONAL DESCRIPTION: We participate in the development of musl, a re-implementation of the C library as it is described by the C and POSIX standards. It is lightweight, fast, simple, free, and strives to be correct in the sense of standards-conformance and safety. Musl is production quality code that is mainly used in the area of embedded devices. It gains more market share also in other areas, e.g. there are now Linux distributions that are based on musl instead of Gnu LibC.

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

6.5. Modular C

KEYWORDS: Programming language - Modularity

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

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

6.6. arbogast

KEYWORD: Automatic differentiation

SCIENTIFIC DESCRIPTION: This high-level toolbox for the calculus with Taylor polynomials is named after L.F.A. Arbogast (1759-1803), a French mathematician from Strasbourg (Alsace), for his pioneering work in derivation calculus. Its modular structure ensures unmatched efficiency for computing higher order Taylor polynomials. In particular it permits compilers to apply sophisticated vector parallelization to the derivation of nearly unmodified application code.

FUNCTIONAL DESCRIPTION: Arbogast is based on a well-defined extension of the C programming language, Modular C, and places itself between tools that proceed by operator overloading on one side and by rewriting, on the other. The approach is best described as contextualization of C code because it permits the programmer to place his code in different contexts – usual math or AD – to reinterpret it as a usual C function or as a differential operator. Because of the type generic features of modern C, all specializations can be delegated to the compiler.

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

6.7. CFML

Interactive program verification using characteristic formulae

KEYWORDS: Coq - Software Verification - Deductive program verification - Separation Logic

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

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

6.8. SPETABARU

SPEculative TAsk-BAsed RUntime system

KEYWORDS: HPC - Parallel computing - Task-based algorithm

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

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

6.9. APAC

KEYWORDS: Source-to-source compiler - Automatic parallelization - Parallelisation - 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 or SPETABARU as runtime system. Internally, it is based on Clang-LLVM.

FUNCTIONAL DESCRIPTION: Automatic task-based parallelization compiler

- Participants: Bérenger Bramas, Stéphane Genaud and Garip Kusoglu
- Contact: Bérenger Bramas
- URL: https://gitlab.inria.fr/bramas/apac

6.10. Dagpar

KEYWORDS: Graph algorithmics - Clustering - Partitioning

SCIENTIFIC DESCRIPTION: This library is a clustering algorithm to create macro-tasks in a DAG of tasks. It extends a clustering/partitioning strategy proposed by Rossignon et al. to speed up the parallel execution of a task-based application. In this package, we provide two additional heuristics to this algorithm, which have been validated on a large graph set. The objective of clustering the nodes of task graphs is to increase the granularity of the tasks and thus obtain faster execution by mitigating the overhead from the management of the dependencies. An important asset of this approach is that working at the graph level allows us to create a generic method independent of the application and of what is done at the user level, but also independent of the task-based runtime system that could be used underneath.

FUNCTIONAL DESCRIPTION: Acyclic Dag Partitioning.

- Participants: Bérenger Bramas and Alain Ketterlin
- Contact: Bérenger Bramas
- URL: https://gitlab.inria.fr/bramas/dagpar

6.11. LetItBench

Lenient to Errors, Transformations, Irregularities and Turbulence Benchmarks

KEYWORDS: Approximate computing - Benchmarking

FUNCTIONAL DESCRIPTION: LetItBench is a benchmark set to help evaluating works on approximate compilation techniques. We propose a set of meaningful applications with an iterative kernel, that is not too complex for automatic analysis and can be analyzed by polyhedral tools. The benchmark set called LetItBench (Lenient to Errors, Transformations, Irregularities and Turbulence Benchmarks) is composed of standalone applications written in C, and a benchmark runner based on CMake. The benchmark set includes fluid simulation, FDTD, heat equations, game of life or K-means clustering. It spans various kind of applications that are resilient to approximation.

- Contact: Cédric Bastoul
- URL: https://github.com/Syllo/LetItBench

6.12. ACR

Adaptive Code Refinement

KEYWORDS: Approximate computing - Optimizing compiler

FUNCTIONAL DESCRIPTION: ACR is to approximate programming what OpenMP is to parallel programming. It is an API including a set of language extensions to provide the compiler with pertinent information about how to approximate a code block, a high-level compiler to automatically generate the approximated code, and a runtime library to exploit the approximation information at runtime according to the dataset properties. ACR is designed to provide approximate computing to non experts. The programmer may write a trivial code without approximation, provide approximation information thanks to pragmas, and let the compiler generate an optimized code based on approximation.

- Contact: Cédric Bastoul
- URL: https://github.com/Syllo/acr

6.13. APOLLO

Automatic speculative POLyhedral Loop Optimizer

KEYWORD: Automatic parallelization

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

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

7. New Results

7.1. The Polyhedral Model Beyond Loops

Participants: Salwa Kobeissi, Philippe Clauss.

There may be a huge gap between the statements outlined by programmers in a program source code and instructions that are actually performed by a given processor architecture when running the executable code. This gap is due to the way the input code has been interpreted, translated and transformed by the compiler and the final processor hardware. Thus, there is an opportunity for efficient optimization strategies, that are dedicated to specific control structures and memory access patterns, to be applied as soon as the actual runtime behavior has been discovered, even if they could not have been applied on the original source code.

We develop this idea by identifying code excerpts that behave as polyhedral-compliant loops at runtime, while not having been outlined at all as loops in the original source code. In particular, we are interested in recursive functions whose runtime behavior can be modeled as polyhedral loops. Therefore, the scope of this study exclusively includes recursive functions whose control flow and memory accesses exhibit an affine behavior, which means that there exists a semantically equivalent affine loop nest, candidate for polyhedral optimizations. Accordingly, our approach is based on analyzing early executions of a recursive program using a Nested Loop Recognition (NLR) algorithm [3], performing the affine loop modeling of the original program runtime behavior, which is then used to generate an equivalent iterative program, finally optimized using the polyhedral compiler Polly. We present some preliminary results showing that this approach brings recursion optimization techniques into a higher level in addition to widening the scope of the polyhedral model to include originally non-loop programs.

This work is the topic of Salwa Kobeissi's PhD. A first paper has been published at the 9th International Workshop on Polyhedral Compilation Techniques [22].

7.2. New release of Apollo

Participants: Muthena Abdul-Wahab, Philippe Clauss.

Apollo has been updated to use LLVM/Clang version 6.0.1. The unmodified sources are now included, as tar-files, in the APOLLO distribution.

Regarding the build system:

- All components of APOLLO are now installed into the installation directory. Once installed, APOLLO does not need the build directory to be kept.
- The RPATH on APOLLO libraries has been set to the installation directory. This allows APOLLO to be run without having to set up library paths.
- APOLLO_BUILD_JOBS has been introduced to specify the maximum number of build jobs to use. The replaces NB_JOBS which is still supported but deprecated.
- The sources for external dependencies are now included in the APOLLO distribution. They are no longer downloaded during a build.
- A new build target 'check' has been added to run the testsuite. This is supported by Makefiles ('make check') and Ninja ('ninja check').
- The build type (Debug/Release) for LLVM/Clang is now the same as the rest of APOLLO. New build variable APOLLO_LLVM_BUILD_TYPE can be used to specify a separate build type for LLVM/Clang.

Regarding bug fixes:

- Valid code using floating point types (float or double) could make APOLLO stop with an message about unsupported scalars. This has been fixed by removing the Loop Invariant Code Motion (LICM) pass in such cases, preventing floating-point scalars to be generated.
- Code containing try-catch blocks could make APOLLO crash. This has been fixed.
- Dynamic loop bounds were no more instrumented and interpolated. This has been fixed.

7.3. Uniform Random Sampling in Polyhedra

Participant: Philippe Clauss.

We propose a method for generating uniform samples among a domain of integer points defined by a polyhedron in a multi-dimensional space. The method extends to domains defined by parametric polyhedra, in which a subset of the variables are symbolic. We motivate this work by a list of applications for the method in computer science. The proposed method relies on polyhedral ranking functions, as well as a recent inversion method for them, named *trahrhe* expressions. This work has been accomplished in collaboration with Benoît Meister from Reservoir Labs, New York, USA, and has been published at the 10th International Workshop on Polyhedral Compilation Techniques, January 2020.

7.4. Runtime Multi-Versioning and Specialization

Participant: Philippe Clauss.

We have developped an extension of APOLLO that implements code multi-versioning and specialization to optimize and parallelize loop kernels that are invoked many times with varying parameters. These parameters may influence the code structure, the touched memory locations, the workload, and the runtime performance. They may also impact the validity of the parallelizing and optimizing polyhedral transformations that are applied on-the-fly.

For a target loop kernel and its associated parameters, a different optimizing and parallelizing transformation is evaluated at each invocation, among a finite set of transformations (multi-versioning and specialization). The best performing transformed code version is stored and indexed using its associated parameters. When every optimizing transformation has been evaluated, the best performing code version regarding the current parameters, which has been stored, is relaunched at next invocations (memoization).

This work has been accomplished in collaboration with Raquel Lazcano and Eduardo Juarez of the Universidad Politécnica de Madrid, Spain, and has been published at the ACM SIGPLAN 2020 International Conference on Compiler Construction (CC 2020).

7.5. AutoParallel: Automatic parallelization and distributed execution of affine loop nests in Python

Participant: Philippe Clauss.

The last improvements in programming languages and models have focused on simplicity and abstraction; leading Python to the top of the list of the programming languages. However, there is still room for improvement when preventing users from dealing directly with distributed and parallel computing issues. We propose AutoParallel, a Python module to automatically find an appropriate task-based parallelisation of affine loop nests to execute them in parallel in a distributed computing infrastructure. This parallelization can also include the building of data blocks to increase tasks' granularity in order to achieve a good execution performance. Moreover, AutoParallel is based on sequential programming and only contains a small annotation in the form of a Python decorator so that anyone with intermediate-level programming skills can scale up an application to hundreds of cores.

This work has been accomplished in collaboration with Cristian Ramon-Cortes, Ramon Amela, Jorge Ejarque and Rosa M. Badia of the Barcelona Supercomputing Center (BSC), Spain. A journal paper is in preparation.

7.6. Combining Locking and Data Management Interfaces

Participants: Jens Gustedt, Daniel Salas.

Handling data consistency in parallel and distributed settings is a challenging task, in particular if we want to allow for an easy to handle asynchronism between tasks. Our publication [2] shows how to produce deadlock-free iterative programs that implement strong overlapping between communication, IO and computation.

An implementation (ORWL) of our ideas of combining control and data management in C has been undertaken, see Section 6.3. In previous work it has demonstrated its efficiency for a large variety of platforms.

In the framework of the ASNAP project we have used ordered read-write locks (ORWL) as a model to dynamically schedule a pipeline of parallel tasks that realize a parallel control flow of two nested loops; an outer *iteration* loop and an inner *data traversal* loop. Other than dataflow programming, for each individual data object we conserve the same modification order as the sequential algorithm. As a consequence the visible side effects on any object can be guaranteed to be identical to a sequential execution. Thus the set of optimizations that are performed are compatible with C's abstract state machine and compilers could perform them, in principle, automatically and unobserved. See [16] for first results.

In the context of the Prim'Eau project (see 9.1.2) we use ORWL to integrate parallelism into an already existing `Fortran` application that computes floods in the region that is subject to the study. A first step of such a parallelization has been started by using ORWL on a process level. Our final goal will be to extend it to the thread level and to use the application structure for automatic placement on compute nodes. A first step to this goal has been a specific decomposition of geological data, see [21].

Within the framework of the thesis of Daniel Salas we have successfully applied ORWL to process large histopathology images. We are now able to treat such images distributed on several machines or shared in an accelerator (Xeon Phi) transparently for the user. This year, Daniel has successfully defended his thesis, see [7].

7.7. Granularity Control for Parallel Programs

Participant: Arthur Charguéraud.

Arthur Charguéraud studied the development of techniques for controlling granularity in parallel programs. Granularity control is an essential problem because creating too many tasks may induce overwhelming overheads, while creating too few tasks may harm the ability to process tasks in parallel. Granularity control turns out to be especially challenging for nested parallel programs, i.e., programs in which parallel constructs such as fork-join or parallel-loops can be nested arbitrarily.

The proposed approach combines the use of asymptotic complexity functions provided by the programmer, with runtime measurements to estimate the constant factors that apply. Exploiting these two sources of information makes it possible to predict with reasonable accuracy the execution time of tasks. Such predictions may be used to guide the generation of tasks, by sequentializing computations of sufficiently small size. An analysis is developed, establishing that task creation overheads are indeed bounded to a small fraction of the total runtime. These results extend prior work by the same authors [52], extending them with a carefully-designed algorithm for ensuring convergence of the estimation of the constant factors deduced from the measures, even in the face of noise and cache effects, which are taken into account in the analysis. The approach is demonstrated on a range of benchmarks taken from the state-of-the-art PBBS benchmark suite. These results have been accepted for publication at PPoPP'19 [14].

7.8. Program Verification and Formal Languages

Participant: Arthur Charguéraud.

- Armaël Guéneau, a PhD student advised by A. Charguéraud and F. Pottier (Cambium), has developed a formal proof of the functional correctness and the asymptotic complexity of a state-of-the-art incremental cycle detection algorithm due to Bender, Fineman, Gilbert, and Tarjan. This work moreover proposes a simple change that allows the algorithm to be regarded as genuinely online. The verification proof is carried out by exploiting Separation Logic with Time Credits, in the CFML tool, to simultaneously verify the correctness and the worst-case amortized asymptotic complexity of the modified algorithm. This work was published at ITP'19 [17]. It leverages previous work on the extension of the CFML verification tool to allow the specification of the asymptotic complexity of higher-order, imperative programs [55], and shows that this framework scales up to larger, more complex programs.
- Arthur Charguéraud, together with Jean-Christophe Filliâtre and Cláudio Lourenço (CNRS, Inria and Université Paris Saclay), and Mário Pereira (NOVA LINCS & DI, Universidade Nova de Lisboa), developed a behavioral specification language for OCaml, called GOSPEL. It is designed to enable modular verification of data structures and algorithms. Compared with writing specifications directly in Separation Logic, it provides a high-level syntax that greatly improves conciseness and makes it accessible to programmers with no familiarity with Separation Logic. GOSPEL is applied to the development of a formally verified library of general-purpose OCaml data structures. This work was published at the World Congress on Formal Methods (FM) 2019 [15].

7.9. Improvement of Schnaps on multi-GPU nodes using the LAHeteroprio Scheduler

Participant: Bérenger Bramas.

The TONUS team has developed Schnaps, a discontinuous finite element solver with OpenCL and StarPU. The team members have been facing challenges in the scalability of their application when using more than one GPU. This has been the starting point of a collaboration in which Bérenger Bramas has participated in the development of Schnaps and plugged its StarPU scheduler called LAHeteroprio [9]. The improvements obtained were significant and included in a paper [50] (currently under revision).

The potential of LAHeteroprio is now demonstrated. However, setting up this scheduler remains a complicated task. Therefore, we plan to work on its automatic configuration, which will require us to perform on the fly analysis of the graph of tasks.

7.10. Improving Parallel Executions by Increasing Task Granularity in Task-based Runtime Systems using Acyclic DAG Clustering

Participants: Bérenger Bramas, Alain Ketterlin.

Bérenger Bramas and Alain Ketterlin collaborate with the TONUS team in the development of a parallel solver for the resolution of conservative hyperbolic upwind kinetic of unstructured tokamaks [49]. In their methods, they must solve the transport equation on an unstructured mesh, which can be seen as having a wave propagating from neighbor-to-neighbor. The resulting computation can be represented using a direct acyclic graph (DAG) of operations, where each operation is a tiny task. Therefore, Bérenger Bramas and Alain Ketterlin contributed mainly on two aspects. First, they have proposed a highly optimized lock-free parallel implementation of the solution based on atomic instructions. Second, they have improved an existing algorithm from the literature to cluster a DAG of tasks with the aim of increasing the granularity of the tasks and to reduce the overhead of the parallelization consequently. This new approach has been accepted in a dedicated paper (accepted but not yet published).

7.11. FMM Kernel for the Integral Equation Formulation of the N-body Dielectric Spheres Problem

Participant: Bérenger Bramas.

Bérenger Bramas worked with Benjamin Stamm and Muhammad Hassan (RWTH) to create a kernel for the fast multipole method (FMM). The kernel relies on the previously developed kernel with spherical harmonics and accelerated by rotations. It has been extended to accept spherical harmonics (with orders different from the ones used in the kernel) instead of points as input. The kernel allowed us to accelerate the computation and was used for a complexity analysis that has been submitted [54].

7.12. Automatic Task-Based Parallelization using Source to Source Transformations

Participants: Bérenger Bramas, Garip Kusolgu.

Bérenger Bramas and Garip Kusolgu worked on a new approach to parallelize automatically 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. With this aim, they 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 accesses and thus generating the correct dependencies. An important and challenging part of the work consists in managing the granularity, which requires to work both statically on the code but also by delegating decisions at runtime.

7.13. Large Scale Particle Fusion Algorithm for Tracing Systems in Fluid Mechanics Applications

Participant: Bérenger Bramas.

Bérenger Bramas worked with Michael Wilczek and Cristian Lalescu (Max Planck Institute for Dynamics and Self-Organization) in designing a new method to merge particles in a large scale application (*i.e.,* designed to run on thousands of computing nodes). In this context, the particles are originally used in a tracing system to extract information from a vector field in fluid mechanics. However, the physicists are now interested having the particles interacting and even fusioning. Due to the constraints of large scale computing, the system tries to reduce the number and amount of communications. This development has been done in the TurTLE application (not publicly available) and is currently under evaluation.

7.14. Pipelined Multithreaded Code Generation

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

State-of-the-art automatic polyhedral parallelizers extract and express parallelism as isolated parallel loops. For example, the Pluto high-level compiler generates and annotates loops with `#pragma omp parallel for` directives. In this work, we took advantage of pipelined multithreading, a parallelization strategy that can address a wider class of codes, currently not handled by automatic parallelizers. Pipelined multithreading requires interlacing iterations of some loops in a controlled way that enables the parallel execution of these iterations.

This work has been accepted for presentation at the International Workshop on Polyhedral Compilation Techniques (IMPACT 2020), in conjunction with HiPEAC '20 (Jan. 2020, Bologna, Italy).

7.15. Raster Image Processing (RIP) Optimization

Participants: Cédric Bastoul, Paul Godard, Vincent Loechner.

In the context of our collaboration with the Caldera company, we are interested in original challenges for the computer systems in charge of driving very wide printer farms and very fast digital presses.

We explored new approaches inspired by the high performance computing field to speedup the graphics processing (RIP) necessary to digital printing. To achieve this goal, we developed a distributed system which provides the adequate flexibility and performance by exploiting and optimizing both processing and synchronization techniques. Our architecture meets the specific constraints on generating streams for printing purpose. We performed an evaluation of our solution and provided experimental evidence of its great performance and viability. This work has been presented at the 2019 IEEE International Parallel and Distributed Processing Symposium Workshop (IPDPSW): PDSEC '19, in May 2019, Rio de Janeiro.

The second topic we worked on during this collaboration is an out-of-core and out-of-place rectangular matrix transposition and rotation algorithm. An originality of our processing algorithm is to rely on an optimized use of the page cache mechanism. It is parallel, optimized by several levels of tiling and independent of any disk block size. We evaluated our approach on four common storage configurations: HDD, hybrid HDD-SSD, SSD and software RAID 0 of several SSDs. We showed that it brings significant performance improvement over a hand-tuned optimized reference implementation developed by the Caldera company and we confront it against the roofline speed of a straight file copy. This work is under submission in the IEEE Transaction on Computers.

Paul Godard has defended his PhD thesis on Dec. 16th, 2019.

7.16. Static Versus Dynamic Memory Allocation

Participant: Vincent Loechner.

Vincent Loechner and Toufik Baroudi (PhD student, Univ. Batna, Algeria) compared the performance of linear algebra kernels using different array allocation modes: as static declared arrays or as dynamically allocated arrays of pointers. They studied the possible reasons of the difference in performance of parallelized or sequential linear algebra kernels on two different architectures: an AMD (Magny-Cours) and an Intel Xeon (Haswell-EP). Static or dynamic memory allocation has an impact on performance in many cases. Both the processor architecture and the compiler can provoke significant and sometimes surprising variations in the number of cache misses and vectorization opportunities taken by the compiler.

This work has been accepted for presentation at the International Workshop on Polyhedral Compilation Techniques (IMPACT 2020), in conjunction with HiPEAC '20 (Jan. 2020, Bologna, Italy).

7.17. Automatic Adaptive Approximation for Stencil Computations

Participants: Maxime Schmitt, Cédric Bastoul.

This work has been done in collaboration with Philippe Helluy (TONUS).

Approximate computing is necessary to meet deadlines in some compute-intensive applications like simulation. Building them requires a high level of expertise from the application designers as well as a significant development effort. Some application programming interfaces greatly facilitate their conception but they still heavily rely on the developer's domain-specific knowledge and require many modifications to successfully generate an approximate version of the program. In this work we designed new techniques to semi-automatically discover relevant approximate computing parameters. We believe that superior compiler-user interaction is the key to improved productivity. After pinpointing the region of interest to optimize, the developer is guided by the compiler in making the best implementation choices. Static analysis and runtime monitoring are used to infer approximation parameter values for the application. We evaluated these techniques on multiple application kernels that support approximation and show that with the help of our method, we achieve similar performance as non-assisted, hand-tuned version while requiring minimal intervention from the user.

These techiques and the underlying compiler infrastructure are a significant output of collaboration with the Inria Nancy - Grand Est team TONUS, specialized on applied mathematics (contact: Philippe Helluy), to bring models and techniques from this field to compilers. A paper presenting these extensions has been accepted to the CC international conference [18].

Maxime Schmitt has defended his PhD thesis on Sep. 30th, 2019 [8].

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. Caldera

Participants: Cédric Bastoul, Vincent Loechner.

Duration : 2016 - 2019

Caldera (www.caldera.com) is a company specialized in software development for wide image processing. The goal of this collaboration is the development of a parallel and scalable image processing pipeline for industrial printing. The project started in September 2016 and it includes the industrial thesis (CIFRE) of Paul Godard, defended in Dec. 2019.

9. Partnerships and Cooperations

9.1. Regional Initiatives

9.1.1. ADT SPETABARU-H

Participants: Bérenger Bramas, Vincent Loechner, Paul Cardosi.

Duration: 2019 - 2021

The SPETABARU task-based runtime system is now being developed in CAMUS. This tool is the first runtime system build on the tasks and dependencies paradigm that supports speculative execution. It is at the same time a robust runtime system that could be used for high-performance applications, and the central component to perform research in parallelization, speculation and scheduling.

The SPETABARU-H project started in November 2019 for 2 years aims in improving SPETABARU on several aspects:

- Implement a generic speculative execution model based on the team's research;
- Implement the mechanisms to make SPETABARU supporting GPUs (and heterogeneous computing nodes in general);
- Split the management of the workers and the management of the graph of tasks to allow multiple independent graphs to be used on a single node;
- Use SPETABARU in the Complexes++ application, which is a bio-physic software for protein simulation;
- Maintain and update the code to keep it modern and up to date.

9.1.2. Idex Prim'Eau

Participant: Jens Gustedt [contact].

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

9.2. National Initiatives

9.2.1. ANR AJACS

Participant: Arthur Charguéraud.

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

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

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

9.2.2. ANR Vocal

Participant: Arthur Charguéraud.

The Vocal research project is funded by the programme "Société de l'information et de la communication" of the ANR, from October 2015 until October 2020 https://vocal.lri.fr/.

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

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

9.3. European Initiatives

9.3.1. Collaborations with Major European Organizations

Benjamin Stamm and Muhammad Hassan: Université d'Aix-la-Chapelle RWTH, MATHCCES (Germany). An integral equation formulation of the N-body dielectricspheres problem.

Michael Wilczek and Cristian Lalescu: Max Planck Institute for Dynamics and Self-Organization (Germany). Pseudospectral direct numerical simulations (DNS) of the incompressible Navier-Stokes equations.

Juergen Koefinger: Max Planck Institute of Biophysics, Theoretical Biophysics (Germany). Monte-Carlo simulation for coarse grained protein models.

Pavel Kus: Czech Academy of Sciences, Institute of Mathematics (Tchequia). Direct solver for several matrices at a time.

9.4. International Initiatives

9.4.1. Informal International Partners

The CAMUS team has collaborated with the following entities in 2019:

- Reservoir Labs, New York, NY, USA (See subsection 7.3)
- University of Batna, Algeria (See subsection 7.16)
- Universidad Politécnica de Madrid, Spain (See subsection 7.4)
- Barcelona Supercomputing Center, Barcelona, Spain (See subsection 7.5)

9.5. International Research Visitors

9.5.1. Visits of International Scientists

9.5.1.1. Internships

Toufik Baroudi is a PhD student under the supervision of Rachid Seghir at the University of Batna (Algeria). He is co-advised by Vincent Loechner, and has been visiting our team as an intern for one year from Nov. 2018 to Nov. 2019, founded by the Algerian *Programme National Exceptionnel (PNE)*. His PhD defense is planned at the beginning of 2020.
Raquel Lazcano is a PhD student under the supervision of Eduardo Juárez Martínez at the University of Madrid. She is also co-advised by Philippe Clauss and has been visiting our team as an intern for three months, from February to April 2019. Her PhD defense is planned at the beginning of 2020.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Organisation

10.1.1.1. Member of the Organizing Committees

Philippe Clauss organized the Special Session on Compiler Architecture, Design and Optimization (CADO) of the 17th International Conference on High Performance Computing & Simulation (HPCS 2019), July 2019, Dublin, Ireland.
Philippe Clauss will organize the 10th edition of the International Workshop on Polyhedral Compilation Techniques, held in conjunction with HiPEAC 2020, January 22, 2020, Bologna, Italy.
Cédric Bastoul co-organized HIP3ES 2019 (International Workshop on High Performance Energy Efficient Embedded Systems), in conjunction with the international conference HiPEAC 2019.
Arthur Charguéraud co-organized the summer school École des Jeunes Chercheurs en Programmation (EJCP), June 2019, Strasbourg, France.

10.1.2. Scientific Events: Selection

10.1.2.1. Member of the Conference Program Committees

Vincent Loechner has been member of the program committees of HIP3ES 2019, PDP 2020, IMPACT 2020.
Philippe Clauss has been part of the program committees of: CC 2020 (ACM SIGPLAN International Conference on Compiler Construction); ICPP 2020 (49th International Conference on Parallel Processing); IPDRM 2019 (Third Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware, held in conjunction with the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 19).
Arthur Charguéraud has been member of the program committees of ITP 2019, POPL 2020, OOPSLA 2019.

Cédric Bastoul has been part of the program committee of HiPC 2019 (IEEE International Conference on High Performance Computing, Data and Analytics), HIP3ES 2019 (International Workshop on High Performance Energy Efficient Embedded Systems), IMPACT 2019 (International Workshop on Polyhedral Compilation Techniques), CADO 2019 (Special Session on Compiler Architecture, Design and Optimization of the 17th International Conference on High Performance Computing & Simulation - HPCS 2019).

10.1.2.2. Reviewer

Bérenger Bramas has been a reviewer for COMPAS 2019 and PDP 2020.
Arthur Charguéraud has been a reviewer for FOSSACS 2020 and ESOP 2020.

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

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

10.1.3.2. Reviewer - Reviewing Activities

Bérenger Bramas has been a reviewer for the following journals: Journal of Parallel Computing (Elsevier), Journal of Parallel and Distributed Computing (Elsevier), Journal of Computer Science and Technology (Springer), Parallel Processing Letters (World Scientific), Software: Practice and Experience (Wiley).
Philippe Clauss has been a reviewer for the Journal of Software: Practice and Experience (Wiley).

10.1.4. Invited Talks

Cédric Bastoul delivered the invited talk "Loop Optimization: A Matter of Art and Science" at the Huawei Symposium on Foundations of Software 2019.

10.1.5. Scientific Expertise

10.1.5.1. Standardization

Since Nov. 2014, Jens Gustedt has been a member of the ISO working group SC22-WG14 for the standardization of the C programming language and serves as a co-editor of the standards document, see [46], [47], [40], [33]. He participates actively in the clarification report processing, the planning of future versions of the standard and in a subgroup that discusses the improvement of the C memory model, see [45], [35], [48].

He was one of the main forces behind the elaboration of C17, the new version of the C standard that has been published by ISO in 2018 [51] and contributes to the future standard "C2x" in various ways. In particular he proposed the removal of the so-called K&R definitions [27], the reform of sign representation [25], [26], maximum width integers [44], keywords [31], [41], [36], [36], null pointer constants [34], timing interfaces [37], [30], [30], atomicity and synchronization [42], [32], [39], and function error conventions [28]. Most of these are either integrated in the latest draft or have been adopted subject to reformulations and adaptations.

10.1.5.2. Expertise

Philippe Clauss has been a reviewer for a promotion case to Full Professor in a US University.
Cédric Bastoul has been an expert for the French research ministry and the French finance ministry for the research tax credit programme.

10.1.6. Research Administration

Cédric Bastoul, Philippe Clauss and Vincent Loechner are members of the *Comité d'Experts (section 27, informatique)* of the Université de Strasbourg, providing their scientific and teaching expertise to the university and to the academy. In particular, this committee is involved in the recruitment of researchers and teachers in computer science. Philippe Clauss has been the Vice President of the committee since April 2019.
Jens Gustedt is the head of the ICPS team for the ICube lab, and in that function a member of the board of directors of the lab. He is also a member of the local recruitment committee for PhD students and postdocs of Inria Center Nancy — Grand Est.
Philippe Clauss and Cédric Bastoul are members of the *Collegium Sciences* of the University of Strasbourg, which is a group of representative scientists providing advice regarding the funding of projects.

Philippe Clauss is a member of the *Bureau du Comité des Projets* of the Inria Center Nancy — Grand Est. This group of scientists provides scientific expertise to the Director of the Center.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Master: Bérenger Bramas, Compilation and Performance, 39h, M2, Université de Strasbourg, France

Master: Bérenger Bramas, Compilation, 30h, M1, Université de Strasbourg, France

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

Licence: Vincent Loechner, algorithmique et programmation, 168h, L1, Université de Strasbourg, France

Licence: Vincent Loechner, administration système et internet, 45h, L3, Université de Strasbourg, France

Licence: Vincent Loechner, programmation parallèle, 23h, L3, Université de Strasbourg, France

Master: Vincent Loechner, programmation temps réel, 10h, M2, Université de Strasbourg, France

Master: Vincent Loechner, calcul parallèle, 20h, 3ième année école d'ingénieur (TPS), 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 (Math-Info): Alain Ketterlin, Algorithmique et programmation, L1, 96h, Université de Strasbourg, France

Licence (Math-Info): Alain Ketterlin, Architecture des systèmes d'exploitation, L3, 38h, Université de Strasbourg, France

Licence (Math-Info): Alain Ketterlin, Programmation système, L2, 40h, Université de Strasbourg, France

Master (Informatique): Alain Ketterlin, Preuves assistées par ordinateur, 18h, Université de Strasbourg, France

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

Licence: Éric Violard, Programmation fonctionnelle, 162h, L1, Université de Strasbourg, France

Licence: Éric Violard, Bases de l'architecture informatique, 62h, L1, Université de Strasbourg, France

Licence: Éric Violard, Architecture des ordinateurs, 45h, L2, Université de Strasbourg, France

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

Licence: Cédric Bastoul, Computer architecture, 78h, L1, Université de Strasbourg, France, and 25h, L1, UFAZ Azerbaijani-French University, Azerbaijan

Licence: Cédric Bastoul, Parallel programming, 20h, L3, Université de Strasbourg, France, and 25h, L3, UFAZ Azerbaijani-French University, Azerbaijan

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

Master: Cédric Bastoul, Project Management, 16h, M1, Université de Strasbourg, France

Master: Cédric Bastoul, Introduction to Research, 3h, L2+M1, Université de Strasbourg, France

10.2.2. Supervision

PhD: Armaël Géneau, *Formal verification of complexity analyses*, co-advised by Arthur Charguéraud and François Pottier, defended on December 16th, 2019.

PhD: Paul Godard, *Parallélisation et passage à l'échelle durable d'une chaîne de traitement graphique pour l'impression professionnelle*, Université de Strasbourg, Dec. 16, 2019. Cédric Bastoul and Vincent Loechner.

PhD: Maxime Schmitt, *Génération automatique de codes adaptatifs*, Université de Strasbourg, Sept. 30, 2019. Cédric Bastoul and Philippe Helluy.

PhD: Daniel Salas, *Parallélisation hybride d'une application de détection de noyaux cellulaires*, Université de Strasbourg, Sept. 10, 2019. Jens Gustedt.

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

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

10.2.3. Juries

Philippe Clauss participated in the following PhD committees in 2019:

Date	**Candidate**	**Place**	**Role**
Jan. 28	Hugo Brunie	Université de Bordeaux	Reviewer
Oct. 25	Ksander EJJAAOUANI	Université de Strasbourg	President
Dec. 9	Arif Ali ANAPPARAKKAL	Université de Rennes	Examiner
Dec. 19	Hang YU	Université Grenoble Alpes	Reviewer

Cédric Bastoul participated in the following PhD committees in 2019:

Date	**Candidate**	**Place**	**Role**
Mar. 29	Pierre Huchant	Université de Bordeaux	Reviewer
Jun. 21	Chandan Reddy	École Normale Supérieure	Reviewer

10.3. Popularization

- A. Charguéraud is a co-organizer of the *Concours Castor informatique*. The purpose of the Concours Castor in to introduce pupils (from *CM1* to *Terminale*) to computer sciences. More than 700,000 teenagers played with the interactive exercises in November 2019. More information on: http://castor-informatique.fr/.

10.3.1. Articles and contents

- Jens Gustedt authored the book *Modern C* [24], which since the first publication of an online draft in 2016 has become one of the major references for the C programming language.
- Jens Gustedt is blogging about efficient programming, in particular about the C programming language. He also is an active member of the stackoverflow community, a technical Q&A site for programming and related subjects.

10.3.2. Education

- Cédric Bastoul participated in the training of high school teachers involved in the forthcoming optional Computer Science course for high school students. Specifically, he produced lectures and materials to teach Computer Architecture to high school students.

10.3.3. Interventions

- Vincent Loechner has been organizing a hub for the Google Hashcode programming contest (online qualification round) at Université de Strasbourg in Feb. 2019. More than 30 students and colleagues were hosted in the university classrooms to participate to this event.
- Cédric Bastoul delivered a presentation on program optimization at "Journée des licences" ("Bachelor Day") in June 2019.

10.3.4. Internal action

- Bérenger Bramas, Jens Gustedt and other members of the scientific computing group (*axe transverse calcul scientifique*) organized two software corners at the ICube laboratory. A software corner is a meeting where researchers exchange about programming best practices, existing and upcoming tools, and their own experiences.

11. Bibliography

Major publications by the team in recent years

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

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

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

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

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

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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[7] D. SALAS. *Hybrid parallellization of a cell nuclei detection application*, Université de Strasbourg, September 2019, https://tel.archives-ouvertes.fr/tel-02384725

[8] M. SCHMITT. *Automatic Generation of Adaptive Codes*, Université de Strasbourg, September 2019, https://hal.inria.fr/tel-02327764

Articles in International Peer-Reviewed Journals

[9] B. BRAMAS. *Impact study of data locality on task-based applications through the Heteroprio scheduler*, in "PeerJ Computer Science", May 2019 [*DOI :* 10.7717/PEERJ-CS.190], https://hal.inria.fr/hal-02120736

[10] B. BRAMAS. *Increasing the degree of parallelism using speculative execution in task-based runtime systems*, in "PeerJ Computer Science", 2019, vol. 5, e183 p. [*DOI :* 10.7717/PEERJ-CS.183], https://hal.inria.fr/hal-02070576

[11] B. BRAMAS, A. KETTERLIN. *Improving parallel executions by increasing task granularity in task-based runtime systems using acyclic DAG clustering*, in "PeerJ Computer Science", January 2020 [*DOI :* 10.7717/PEERJ-CS.247], https://hal.inria.fr/hal-02436826

[12] A. CHARGUÉRAUD, F. POTTIER. *Verifying the Correctness and Amortized Complexity of a Union-Find Implementation in Separation Logic with Time Credits*, in "Journal of Automated Reasoning", March 2019, vol. 62, n° 3, pp. 331–365 [*DOI :* 10.1007/s10817-017-9431-7], https://hal.inria.fr/hal-01652785

Invited Conferences

[13] B. BRAMAS. *SPETABARU: A Task-based Runtime System with Speculative Execution Capability*, in "SIAM CSE 2019 - SIAM Conference on Computational Science and Engineering", Spokane, United States, February 2019, https://hal.inria.fr/hal-02050190

International Conferences with Proceedings

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

[15] A. CHARGUÉRAUD, J.-C. FILLIÂTRE, C. LOURENÇO, M. PEREIRA. *GOSPEL -Providing OCaml with a Formal Specification Language*, in "FM 2019 - 23rd International Symposium on Formal Methods", Porto, Portugal, October 2019, https://hal.inria.fr/hal-02157484

[16] J. GUSTEDT, M. MOGÉ. *Memory access classification for vertical task parallelism*, in "PDP 2019 - 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing", Pavia, Italy, IEEE, February 2019, https://hal.inria.fr/hal-02046105

[17] A. GUÉNEAU, J.-H. JOURDAN, A. CHARGUÉRAUD, F. POTTIER. *Formal Proof and Analysis of an Incremental Cycle Detection Algorithm : (extended version)*, in "Interactive Theorem Proving", Portland, United States, J. HARRISON, J. O'LEARY, A. TOLMACH (editors), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, September 2019, n° 141, https://hal.inria.fr/hal-02167236

[18] M. SCHMITT, P. HELLUY, C. BASTOUL. *Automatic adaptive approximation for stencil computations*, in "CC 2019 - 28th International Conference on Compiler Construction", Washington, United States, ACM Press, February 2019, pp. 170-181 [*DOI :* 10.1145/3302516.3307348], https://hal.inria.fr/hal-02072737

Conferences without Proceedings

[19] P. GODARD. *Échanges non bloquants de données ordonnées entre producteurs multiples et consommateur unique*, in "COMPAS'2019", Anglet, France, June 2019, https://hal.archives-ouvertes.fr/hal-02381769

[20] P. GODARD, V. LOECHNER, C. BASTOUL, F. SOULIER, G. MULLER. *A Flexible and Distributed Runtime System for High-Throughput Constrained Data Streams Generation*, in "IPDPSW 2019 - IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum", Rio de Janeiro, Brazil, IEEE, May 2019, pp. 718-728 [*DOI :* 10.1109/IPDPSW.2019.00120], https://hal.archives-ouvertes.fr/hal-02381750

[21] S. HARIRI, S. WEILL, J. GUSTEDT, I. CHARPENTIER. *Pairing GIS and distributed hydrological models using Matlab 2*, in "CAJG - 2nd Conference of the Arabian Journal of Geosiences", Sousse, Tunisia, November 2019, https://hal.archives-ouvertes.fr/hal-02333260

[22] S. KOBEISSI, P. CLAUSS. *The Polyhedral Model Beyond Loops Recursion Optimization and Parallelization Through Polyhedral Modeling*, in "IMPACT 2019 - 9th International Workshop on Polyhedral Compilation Techniques, In conjunction with HiPEAC 2019", Valencia, Spain, January 2019, https://hal.inria.fr/hal-02059558

[23] B. MEISTER, P. CLAUSS. *Uniform Random Sampling in Polyhedra*, in "10th International Workshop on Polyhedral Compilation Techniques", Bologna, Italy, January 2020, https://hal.inria.fr/hal-02425752

Scientific Books (or Scientific Book chapters)

[24] J. GUSTEDT. *Modern C*, Manning, November 2019, https://hal.inria.fr/hal-02383654

Research Reports

[25] J.-F. BASTIEN, J. GUSTEDT. *Moving to two's complement sign representation*, ISO JCT1/SC22/WG14, January 2019, n^{o} N2330, https://hal.inria.fr/hal-02046444

[26] J.-F. BASTIEN, J. GUSTEDT. *Two's complement sign representation for C2x*, ISO JCT1/SC22/WG14, August 2019, N2412 p. , https://hal.inria.fr/hal-02311453

[27] L. G. BJØNNES, J. GUSTEDT. *Remove support for function definitions with identifier lists*, ISO JCT1/SC22/WG14, September 2019, n^{o} N2432, https://hal.inria.fr/hal-02311466

[28] N. DOUGLAS, J. GUSTEDT. *Function failure annotation*, ISO JCT1/SC22/WG14, September 2019, n^{o} N2429, https://hal.inria.fr/hal-02311462

[29] J. GUSTEDT. *Add an interface to query resolution of time bases : Proposal for C2x*, ISO JTC1/SC22/WG14, November 2019, n^{o} N2459, https://hal.inria.fr/hal-02378605

[30] J. GUSTEDT. *Add new optional time bases v3 Proposal for C2x*, ISO JCT1/SC22/WG14, November 2019, n^{o} N2460, https://hal.inria.fr/hal-02378645

[31] J. GUSTEDT. *Align spelling of keywords with C++ and make them feature tests proposal for C2x*, ISO JCT1/SC22/WG14, April 2019, n^{o} n2368, https://hal.inria.fr/hal-02089925

[32] J. GUSTEDT. *Clean up atomics, non-normative changes : proposal for integration to C2x*, ISO JTC1/SC22/WG14, June 2019, n^{o} N2389, https://hal.inria.fr/hal-02167823

[33] J. GUSTEDT. *Contain the floating point naming explosion*, ISO JCT1/SC22/WG14, September 2019, n^{o} N2426, https://hal.inria.fr/hal-02311460

[34] J. GUSTEDT. *Introduce the nullptr constant*, ISO JTC1/SC22/WG14, June 2019, n^{o} N2394, https://hal.inria.fr/hal-02167929

[35] J. GUSTEDT. *Introduce the term storage instance*, ISO JTC1/SC22/WG14, June 2019, n^{o} N2388, A previous version of this document was N2328, https://hal.inria.fr/hal-02046329

[36] J. GUSTEDT. *Make false and true first-class language features : proposal for C2x*, ISO JTC1/SC22/WG14, November 2019, n^{o} N2458, https://hal.inria.fr/hal-02167916

[37] J. GUSTEDT. *Modernize time.h functions*, ISO JCT1/SC22/WG14, September 2019, n^{o} N2417, https://hal.inria.fr/hal-02311454

[38] J. GUSTEDT. *Out-of-band bit for exceptional return and errno replacement*, ISO JCT1/SC22/WG14, April 2019, n^{o} N2361, https://hal.inria.fr/hal-02089873

[39] J. GUSTEDT. *Remove ATOMIC VAR INIT*, ISO JTC1/SC22/WG14, June 2019, n^{o} N2390, https://hal.inria.fr/hal-02167838

[40] J. GUSTEDT. *Remove conditional "WANT" macros from numbered clauses proposal for C2x*, ISO JTC1/SC22/WG14, April 2019, n^{o} N2359, https://hal.inria.fr/hal-02089861

[41] J. GUSTEDT. *Revise spelling of keywords : proposal for C2x*, ISO JTC1/SC22/WG14, June 2019, n^{o} N2457, https://hal.inria.fr/hal-02167870

[42] J. GUSTEDT. *Synchronization at thread and execution termination : proposal for integration to C2x*, ISO JTC1/SC22/WG14, September 2019, n^{o} N2461, https://hal.inria.fr/hal-02167850

[43] J. GUSTEDT. *Unify string representation functions*, ISO JCT1/SC22/WG14, April 2019, https://hal.inria.fr/hal-02089868

[44] J. GUSTEDT. *intmax_t, a way out*, ISO JCT1/SC22/WG14, September 2019, n^{o} N2425, https://hal.inria.fr/hal-02311457

[45] J. GUSTEDT, P. SEWELL, K. MEMARIAN, V. B. F. GOMES, M. UECKER. *Moving to a provenance-aware memory object model for C: proposal for C2x*, ISO JCT1/SC22/WG14, April 2019, n^{o} N2362, https://hal.inria.fr/hal-02089889

[46] L. JONES, J. GUSTEDT. *ISO/IEC 9899 editor report March 2019*, ISO JCT1/SC22/WG14, March 2019, n^{o} N2348, https://hal.inria.fr/hal-02089676

[47] L. JONES, J. GUSTEDT. *ISO/IEC 9899 editor report November 2019*, ISO JCT1/SC22/WG14, November 2019, n^{o} N2456, https://hal.inria.fr/hal-02378784

[48] P. SEWELL, K. MEMARIAN, V. B. F. GOMES, J. GUSTEDT, M. UECKER. *C provenance semantics: examples*, ISO JCT1/SC22/WG14, April 2019, n^{o} N2363, https://hal.inria.fr/hal-02089907

Other Publications

[49] M. BOILEAU, B. BRAMAS, E. FRANCK, P. HELLUY, L. NAVORET. *Parallel lattice-boltzmann transport solver in complex geometry*, December 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02404082

[50] B. BRAMAS, P. HELLUY, L. MENDOZA, B. WEBER. *Optimization of a discontinuous finite element solver with OpenCL and StarPU*, July 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01942863

References in notes

[51] JTC1/SC22/WG14 (editor). *Programming languages - C*, ISO, 2018, n^{o} ISO/IEC 9899

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

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

[54] B. BRAMAS, M. HASSAN, B. STAMM. *An Integral Equation Formulation of the N-body Dielectric Spheres Problem. Part II: Complexity Analysis*, 2019

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

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

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