

# 2025 Activity Report

RESEARCH CENTRE: Inria Centre at Université Grenoble Alpes  
IN PARTNERSHIP WITH: Université de Grenoble Alpes

---

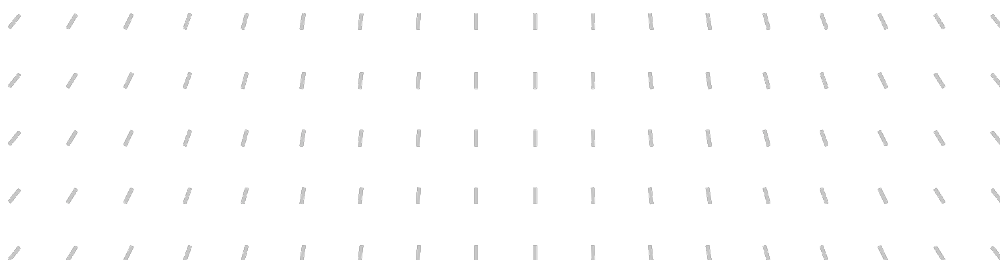
Project-Team

## CORSE

Compiler Optimization and Run-time Systems

---

*In collaboration with* Laboratoire d'Informatique de Grenoble (LIG)



## **Project-Team CORSE**

*Creation of the Project-Team: 2016 July 01*

Each year, Inria research teams publish an Activity Report presenting their work and results over the reporting period. These reports follow a common structure, with some optional sections depending on the specific team. They typically begin by outlining the overall objectives and research programme, including the main research themes, goals, and methodological approaches. They also describe the application domains targeted by the team, highlighting the scientific or societal contexts in which their work is situated. The reports then present the highlights of the year, covering major scientific achievements, software developments, or teaching contributions. When relevant, they include sections on software, platforms, and open data, detailing the tools developed and how they are shared. A substantial part is dedicated to new results, where scientific contributions are described in detail, often with subsections specifying participants and associated keywords. Finally, the Activity Report addresses funding, contracts, partnerships, and collaborations at various levels, from industrial agreements to international cooperations. It also covers dissemination and teaching activities, such as participation in scientific events, outreach, and supervision. The document concludes with a presentation of scientific production, including major publications and those produced during the year.

## Keywords

### Computer sciences and digital sciences

- A1. – Architectures, systems and networks
  - A1.1. – Architectures
    - A1.1.1. – Multicore, Manycore
    - A1.1.2. – Hardware accelerators (GPGPU, FPGA, etc.)
    - A1.1.3. – Memory models
    - A1.1.4. – High performance computing
    - A1.1.12. – Non-conventional architectures
  - A1.6. – Green Computing
    - A2.1.6. – Concurrent programming
    - A2.1.7. – Distributed programming
    - A2.1.10. – Domain-specific languages
  - A2.2. – Compilation
    - A2.2.1. – Static analysis
    - A2.2.2. – Memory models
    - A2.2.3. – Memory management
    - A2.2.4. – Parallel architectures
    - A2.2.6. – GPGPU, FPGA...
    - A2.2.8. – Code generation
  - A2.3.1. – Embedded systems
  - A7.1. – Algorithms
  - A8.2. – Optimization
    - A8.2.1. – Operations research
  - A9.2.1. – Supervised learning
  - A9.2.3. – Reinforcement learning
  - A9.2.5. – Bayesian methods
  - A9.2.8. – Deep learning

### Other research topics and application domains

- B3.1. – Sustainable development
- B4.5. – Energy consumption
- B5.3. – Nanotechnology
- B6.1.2. – Software evolution, maintenance
- B6.6. – Embedded systems
- B6.7. – Computer Industry (hardware, equipments...)
- B9.1. – Education

## Contents

<b>Project-Team CORSE</b>	<b>1</b>
<b>1 Team members, visitors, external collaborators</b>	<b>5</b>
<b>2 Overall objectives</b>	<b>6</b>
<b>3 Research program</b>	<b>6</b>
<b>4 Application domains</b>	<b>7</b>
4.1 Transfer . . . . .	7
<b>5 Social and environmental responsibility</b>	<b>7</b>
5.1 Footprint of research activities . . . . .	7
5.2 Impacting research directions for environment . . . . .	7
5.3 Impacting usage . . . . .	8
<b>6 Latest software developments, platforms, open data</b>	<b>8</b>
6.1 Latest software developments . . . . .	8
6.1.1 GUS . . . . .	8
6.1.2 SEATOP . . . . .	8
6.1.3 PALMED . . . . .	9
6.1.4 Diesect . . . . .	9
6.1.5 sarcasm . . . . .	9
6.1.6 XTC . . . . .	10
6.1.7 JIR . . . . .	10
6.1.8 SDist . . . . .	11
6.1.9 MLOpBench . . . . .	11
6.1.10 Agdbentures . . . . .	11
<b>7 New results</b>	<b>12</b>
7.1 Automatic Derivation of Parametric Data Movement Complexity . . . . .	12
7.2 Automatic Resource Characterization and Performance Feedback . . . . .	13
7.2.1 GUS . . . . .	13
7.2.2 SEATOP . . . . .	13
7.2.3 DIESECT . . . . .	14
7.3 Performance Modeling, Schedule Optimization, and Code Generation for Tensor Computations	14
7.3.1 XTC: A Research Platform for Optimizing AI Workload Operators . . . . .	15
7.3.2 Descript++ . . . . .	15
7.3.3 Sarcasm . . . . .	15
7.3.4 MIOpBench: Benchmarking of Machine learning Operators . . . . .	16
7.3.5 First-Class Verification Dialects for MLIR . . . . .	16
7.3.6 Multi-level x86 Backend for xDSL . . . . .	16
7.3.7 SDist: Abstract distribution primitives and generate code for Kalray MPPA and Amd AIE . . . . .	16
7.4 Teaching of Algorithms, Programming, and Debugging . . . . .	17
7.4.1 Easytracker : A generic library for controlling and inspecting program execution and state . . . . .	17
7.4.2 Agdbentures: A game to learn how to debug in autonomy . . . . .	17
<b>8 Bilateral contracts and grants with industry</b>	<b>18</b>
8.1 Bilateral contracts with industry . . . . .	18

<b>9 Partnerships and cooperations</b>	<b>18</b>
9.1 International research visitors	18
9.1.1 Visits to international teams	18
9.2 National initiatives	19
<b>10 Dissemination</b>	<b>20</b>
10.1 Promoting scientific activities	20
10.1.1 Scientific events: organisation	20
10.1.2 Scientific events: selection	20
10.1.3 Invited talks	20
10.1.4 Leadership within the scientific community	20
10.1.5 Scientific expertise	21
10.1.6 Research administration	21
10.2 Teaching - Supervision - Juries - Educational and pedagogical outreach	21
10.2.1 Teaching	21
10.2.2 Supervision	21
10.2.3 Juries	21
10.3 Popularization	22
10.3.1 Book	22
<b>11 Scientific production</b>	<b>22</b>
11.1 Publications of the year	22
11.2 Cited publications	23

# 1 Team members, visitors, external collaborators

## Research Scientists

- Fabrice Rastello [Team leader, INRIA, Senior Researcher, HDR]
- Guillaume Iooss [INRIA, Researcher]
- Hugo Pompougnac [INRIA, Starting Research Position, from Mar 2025]

## Faculty Members

- Florent Bouchez [UGA, Associate Professor]
- Ylies Falcone [UGA, Associate Professor, until Jul 2025]
- Guillaume Huard [UGA, Associate Professor]
- Manuel Selva [GRENOBLE INP, Associate Professor]

## Post-Doctoral Fellow

- Hugo Pompougnac [INRIA, Post-Doctoral Fellow, until Feb 2025]

## PhD Students

- Alban Dutilleul [INRIA, from Oct 2025]
- Leon Frenot [UGA, from Sep 2025]
- Sylvain Noiry [KALRAY]

## Technical Staff

- Rui Cesista [INRIA, Engineer, from Dec 2025]
- Gallas Gaye [ GROUPE LR TECHNOLOGIES, Engineer, from Aug 2025]
- Gallas Gaye [GROUPE LR TECHNOLOGIES, Engineer, from Feb 2025 until Jun 2025]
- Christophe Guillon [INRIA, Engineer]
- Zikang Liu [INRIA, Engineer, until Aug 2025]
- Liam Semeria [INRIA, Engineer, from Nov 2025]
- Lucas Sube [INRIA, Engineer, from Oct 2025]

## Interns and Apprentices

- Arthur Debaud [INRIA, Intern, from Jun 2025 until Aug 2025]
- Emile Dechenaud [INRIA, Intern, from Apr 2025 until Jul 2025]
- Alban Dutilleul [INRIA, Intern, from Mar 2025 until Aug 2025]
- Nikolaos Mavrogeorgis [INRIA, Intern, until Mar 2025]
- Oussama Ouaghlissi [UGA, Intern, from May 2025 until Jul 2025]
- Ryan Rouibah [INRIA, Intern, from May 2025 until Jul 2025]

- Ryan Rouibah [INRIA, Intern, from Mar 2025 until Apr 2025]
- Leopold Saint Denis [UGA, Intern, from May 2025 until Jul 2025]
- Lucas Sube [INRIA, Intern, from Mar 2025 until Apr 2025]
- Lucas Sube [INRIA, Intern, until Mar 2025]
- Anh Khoi Truong [INRIA, Intern, from Jun 2025 until Jul 2025]
- Chloe Viennot [INRIA, Intern, from Jun 2025 until Aug 2025]

### Administrative Assistants

- Marie-Anne Dauphin-Rizzi [INRIA]
- Maria Immaculada Presseguer [INRIA]

## 2 Overall objectives

Our research team investigates the foundations and tools required to fully exploit modern computing systems, with a particular focus on the tight interaction between compilers, software stacks, and microarchitectural design. Our overarching objective is to bridge the growing gap between increasingly complex hardware platforms and the software that must efficiently and reliably use them.

As computing enters a new era dominated by hardware accelerators (especially for artificial intelligence and data-intensive workloads) traditional approaches must be revisited. Performance, energy efficiency, and predictability increasingly depend on a deep understanding of architectural and microarchitectural details such as memory hierarchies, dispatch queues, or specialized compute units. Our work recognizes that these details must be first-class citizens in the software toolchain rather than hidden behind opaque layers.

Compilers and compiler-related tools play a central role in this vision. We develop techniques that combine high-level programmability with low-level awareness, enabling automated transformations, optimizations, and code generation that are explicitly informed by hardware characteristics. By integrating microarchitectural knowledge into compilation, scheduling, and analysis tools, we aim to unlock the potential of systems composed of CPUs, GPUs, and domain-specific accelerators.

Ultimately, our goal is to contribute to a co-designed hardware–software ecosystem in which compiler technology serves as the key enabler for performance portability.

## 3 Research program

Today, the team’s research is structured around four interconnected axes, each targeting critical aspects of computational efficiency and education:

**Automatic Derivation of Parametric Data Movement Complexity** This research axis focuses on understanding and optimizing data movement in algorithms to improve computational performance. The team has developed methods to derive both lower and upper bounds for data movement. We use polyhedral compilation tools to automate the derivation of lower bounds, addressing challenges like small problem sizes and specific patterns that hinder tiling. To determine upper bounds, we leveraged our expertise in the polyhedral model, data movement complexity estimation, and mathematical proof techniques to create an algorithm that calculates symbolic over-approximations of data movement for tiled affine codes. This algorithm helps in determining optimal tile sizes and loop orderings. One of the objectives of this axis is to gain insights that help the development of more efficient algorithms and tools in the third axis.

**Automatic Resource Characterization and Performance Feedback** This axis focuses on developing tools for characterizing CPU architectures and debugging performance issues. The team created tools for automatic resource characterization, which can be used for modeling the performance of computing kernels. These tools enhance CPU performance understanding and optimization, requiring expertise in CPU architecture, performance modeling, and binary instrumentation. The insights gained are crucial for optimizing computational kernels, as explored in the next axis.

**Performance Modeling, Schedule Optimization, and Code Generation for Tensor Computations** This research axis focuses on optimizing tensor computations, which are crucial for various applications including machine learning. The team develops optimization schemes, code generators, as well as autotuning methods, evaluates cache models, and explores reinforcement learning for complex architectures. We also optimize Sparse Matrix Multi-vector multiplication for graph neural networks and pruned convolutional neural networks. This work requires expertise in performance modeling, schedule optimization, code generation, cache modeling, and reinforcement learning to create efficient tensor computation solutions. The last two axes are decoupled from the first three but share common expertise in program instrumentation.

**Teaching of Algorithms, Programming, and Debugging** This research axis focuses on enhancing computer science education by developing tools for visualizing program execution, gamifying debugging practice, implementing active learning strategies, and generating automated exercises for Intelligent Tutoring Systems. It requires expertise in educational pedagogy, visualization tools, game design, AI, machine learning, software development, and user interface design. These efforts leverage techniques from program instrumentation and visualization to improve the teaching and learning experience.

## 4 Application domains

### 4.1 Transfer

The main industrial sector related to the research activities of CORSE is the one of semi-conductor (programmable architectures spanning from embedded systems to servers). Obviously any computing application which has the objective of exploiting as much as possible the resources (in terms of high-performance but also low energy consumption) of the host architecture is intended to take advantage of advances in compiler and run-time technology. These applications are based on numerical kernels (linear algebra, FFT, convolution, etc.) that can be adapted to a wide spectrum of architectures. More specifically, an important activity concerns the optimization of machine learning applications for some high-performance accelerators. Members of CORSE already maintain fruitful and strong collaborations with several companies such as GOOGLE, STMICROELECTRONICS, AMD, or ARM.

## 5 Social and environmental responsibility

### 5.1 Footprint of research activities

As expected, after the COVID pandemia, team members kept travel activities quite low compared to before the pandemia. Whenever long distance meetings (such as conference PC) could be done virtually, travel has been avoided. Also, team members try to better use existing hardware instead of replacing them (buying new ones).

### 5.2 Impacting research directions for environment

Because of rebound effect, improving efficiency does not necessarily improve environmental impact. It is thus crucial to think how our community can have actual impact on sustainable computing, that is, influence better design ("R" friendly – Reuse, Reduce, Repair...) and better usage (consume less) of our compute resources. To achieve this goal, we arrange panels with the aim of raising awareness within our community about this significant issue. We expect some of our future research projects to address the challenge of

sustainable computing without just focusing on energy efficiency but by considering the global systemic impact as much as possible.

### 5.3 Impacting usage

The main two challenges of sustainable computing are:

1. *Decrease usage*: While the actual environmental impact of our usage is already not that clear to experts like us (lack of open data), it is even less clear for users and developers. It is thus our responsibility to expose estimations of resource usage (and associated environmental impact) to the developers. Performance debugging tools should evolve to provide meaningful metrics and make them accessible to non experts.
2. *Increase the lifetime of hardware* (that is, Reuse, Repair, Re...): The need for supporting the development of simple, open-source, digital commons, low-impact (not necessarily low-tech) hardware/software solutions is becoming critical but not sufficient. We also need to provide the microscope and the tool-box so that a majority (including sometimes the end-user) can repair or repurpose their device.

Compiler analysis, programming infrastructure, hardware modeling, teaching tools, HIM, etc. are at the heart of those challenges.

## 6 Latest software developments, platforms, open data

### 6.1 Latest software developments

#### 6.1.1 GUS

**Keywords:** CPU, Microarchitecture simulation, Performance analysis, Dynamic Analysis

**Functional Description:** GUS' goal is to detect performance bottlenecks at the very low level on multithread applications by the use of sensitivity analysis. It is coded as a QEMU plug-in in order to collect runtime information that are later treated by the generic CPU model.

**News of the Year:** We have designed a “tainting” mechanism to identify instructions or resources that constrain others.

In addition, we have extended Gus to new architectures and microarchitectures beyond Intel x86 Sky Lake: ARM Maia, Intel x86 Sandy Bridge, Intel x86 Ice Lake, Intel x86 Alder Lake.

**URL:** <https://gitlab.inria.fr/nderumig/gus>

**Publication:** hal-04796942

**Contact:** Fabrice Rastello

#### 6.1.2 SEATOP

**Name:** Semantically enhanced asm tracer for optimised programs

**Keywords:** Software, Debug

**Functional Description:** SEATOP combines binary instrumentation and static debug information to compare an unoptimized execution with an optimized execution of the same program. It produces an assembly trace of the instructions executed by the optimized version, annotated with high-level semantic information: equivalent source code line, variable values.

SEATOP assists in debugging performance or the correctness of the implemented optimization (tiling, interchange, -O3, ...).

**URL:** <https://gitlab.inria.fr/CORSE/seatop>

**Contact:** Lucas Sube

### 6.1.3 PALMED

**Keywords:** CPU, Performance measure, Performance analysis, Reverse engineering

**Scientific Description:** PALMED is a software to generate automatically performance models for superscalar processors. It supports Armv8a and x86 ISA, and outputs a bipartite graph (instructions, resources) that represents the behaviour of the tested CPU. PALMED is based on a convex encoding of the resource mapping problem, solved by a convex solver, Gurobi.

**Functional Description:** PALMED computes a bipartite graph between assembly instructions and abstract resources that may be used for performance prediction, targeting static analysis tools and compilers. Internally, PALMED uses PIPEDREAM as a framework for microbenchmarking code generation, and uses gurobi to find a first small graph. Then, PALMED deduces from the found resources and the microbenchmarks that saturates them a mapping of every supported instruction.

**URL:** <https://gitlab.inria.fr/nderumig/palmed>

**Publications:** [hal-03114933](#), [hal-03531740](#), [tel-04653883](#)

**Contact:** Fabrice Rastello

**Participant:** 3 anonymous participants

### 6.1.4 Diesect

**Name:** Diesect

**Keywords:** CPU, Reverse engineering, Microarchitecture

**Scientific Description:** Diesect computes the decomposition of instructions in a given ISA towards the  $\mu$ OPs into backend structures. At first, it tries to isolate the capacities and relationships of instruction queues (IQs) thanks to a set of basic instructions. This characterization then allow to stress these resources to design contention situations to deduce the mapping between instructions and micro-operations ( $\mu$ OPs) footprints in IQs. This data can then be plugged into a performance model or cross-referenced with other microbenchmarks to have a finer-grain modelization of the pipeline.

**Functional Description:** Diesect is a reverse-engineering tool that reveals the hidden internal logic of modern processors. While hardware manufacturers often keep their chip designs secret, most processors follow a set of universal design principles. Diesect operates as a grey-box tool, which means it uses these shared structural patterns to look inside a processor without needing proprietary documentation or specialized vendor tools. The software automatically creates specialized tests that cause intentional and controlled bottlenecks within the chip. By measuring the precise timing of these delays, the tool can deduce the processor's internal organization and performance limits. This approach is highly portable and works across various manufacturers like Intel, Apple, or ARM because it is independent of vendor-specific details. Ultimately, Diesect provides the analytical insights needed to build accurate performance models and develop more efficient software.

**URL:** <https://gitlab.inria.fr/CORSE/frontendexplorer>

**Contact:** Alban Dutilleul

### 6.1.5 sarcasm

**Name:** Set-Associative Rotating Cache Analytical/Simulating Model

**Keywords:** Cache, Operational intensity

**Functional Description:** This repository includes: (i) A definition and management of Ttile scheme, which describes schedules for tensor operators, (ii) Algorithms to generate such scheme and explore a microkernel-based search space (algorithm from Ttile), (iii) Interfacing with XTC to launch them, using TVM/MLIR as possible backends, (iv) A fully-associative cache model (inspired by IOOpt/IOUB), and (v) the Sarcasm set-associative cache model, that can be applied to any Ttile scheme to predict its number of cache misses.

**URL:** <https://gitlab.inria.fr/CORSE/ttile-xdsl>

**Contact:** Guillaume Iooss

### 6.1.6 XTC

**Name:** Xdsl Transform Compiler

**Keywords:** Compilation, Machine learning, Autotuning

**Functional Description:** The XTC (XDSL Transform Compiler) project, is a compiler framework that provides high-level scheduling specifications for linear algebra operations over a dataflow graph.

The project provides high-level syntax for applying transformations like: (i) Tiling, (ii) Loop interchange, (iii) Vectorization, (iv) Unrolling.

It allows the integration of multiple backend and implements: (i) MLIR backend (using MLIR linalg and transform dialects), (ii) TVM backend (using TVM tensor IR and schedule APIs), and (iii) JIR backend (using JIR intermediate representation).

Its core components are: (i) abstract interfaces for tensors, graphs, nodes, and operators, (ii) implementers for different backends, (iii) schedulers for transformation management, (iv) compilers for code generation, and (v) evaluators for performance measurement

The main tools are: (i) ‘mlir-loop’: Compilation driver, (ii) ‘loop-explore’: Automatic exploration of different scheduling strategies, (iii) ‘loop-display’: Visualization of exploration results.

**URL:** <https://gitlab.inria.fr/hpompoug/xdsl-transform>

**Contact:** Hugo Pompougnac

### 6.1.7 JIR

**Name:** JIR Intermediate representation and Tools

**Keywords:** Compilation, Autotuning, Machine learning

**Functional Description:** JIR (J IR) is an intermediate representation. The JIR project provides in addition a compiler/optimization framework.

Its key Design Principles are: (i) Concision & clarity for interactive transformation, (ii) Ease of manipulation for scheduling transformations, (iii) Limited scope focusing exclusively on scheduling of linear algebra operations found in ML frameworks, and (iv) Integration with existing compiler infrastructure (MLIR, Polygeist, Z3)

Its core Features are: (i) Code transformation and optimization framework, (ii) Schedule-based optimization system, (iii) Constraint-based autotuning capabilities, (iv) Support for benchmarking and performance analysis, and (v) Interactive transformation tool

Technical Details: (i) AST based J IR for loops and buffer descriptions, (ii) Transformations directly done at the J IR level, (iii) Backend codegeneration for MLIR/LLVM infrastructure, (iv) Leverages Z3 theorem prover for constraint solving, and (v) Implements Gibbs sampling for schedule exploration

Specific Features: (i) Unlike Halide, TACO, or TVM, JIR deliberately excludes computation description, (ii) Relies on externally supplied computation primitives, (iii) Concentrates purely on scheduling problems

High-Level MLIR Integration: (i) Uses MLIR affine dialect as target, (ii) Preserves loop structure and iteration space information, (iii) Enables additional backend-level optimizations

Unified Intermediate Language: (i) Maintains consistent representation throughout transformations, (ii) No lowering until final MLIR translation, (iii) Enables better reasoning about transformations.

Constraint System: (i) Implements scheduling space through discrete constraints, (ii) Integrates with SMT solver (Z3), (iii) Enables schedule verification and completion, (iv) Supports Gibbs sampling for schedule space exploration

**URL:** <https://gitlab.inria.fr/CORSE/jir>

**Contact:** Christophe Guillon

### 6.1.8 SDist

**Name:** MLIR-SDist

**Keywords:** MLIR, Compilation, Matrix calculation, Kalray, Optimizing compiler

**Functional Description:** The development of a compilation toolchain for tensor operations on a machine with distributed local memories requires to generate complex on-chip horizontal communications between memories. SDist provides a series of abstractions dedicated for the distribution of tensors and computation loops in this context. So the user can reason on higher level concepts like memory meshes. Then, unlike existing tools that produces MPI-style primitives, SDist automates the code generation of Hardware-specific communication primitives, like one sided DMA transfers. The project is designed as an extension to the MLIR infrastructure, by adding dialects and passes. The code generation currently supports two accelerators with distributed local memories, the Kalray MPPA, and the Amd AIE.

**URL:** <https://gitlab.inria.fr/CORSE/mlir-sdist>

**Contact:** Sylvain Noiry

### 6.1.9 MLOpBench

**Name:** Machine Learning Operators Benchmark

**Keywords:** Machine learning, Benchmarking, LLM, Optimizing compiler

**Functional Description:** MLOpBench provides an easy way to extract operators from modern machine learning models (including LLMs), and benchmark them on several state of the art libraries and compilers. The tool has a command line interface. The user chooses the model or set of models (e.g. MLPerf) to analyze and the input parameters (e.g. context size, batch, etc). The extraction of operators, that relies on the Pytorch ecosystem, outputs information like the operator type and shape, data type, and shapes of inputs/outputs. Then, MLOpBench can benchmark some of the extracted operators on one of the following operators libraries: Intel MKL, OpenBLAS, ARM Compute Library, ARM Performance Library, CuBLAS and CuBLAS-It, Google XNNPACK. As well as the following compilers: XTC (<https://bil.inria.fr/fr/software/view/5448/tab>), TVM. The tool reuses the runtime the evaluation harness developed for XTC for fair and reproducible measurements with hardware counters.

**URL:** <https://gitlab.inria.fr/CORSE/ML2Bench>

**Contact:** Sylvain Noiry

### 6.1.10 Agdbentures

**Keywords:** Debug, Teaching of programming, Video Game

**Functional Description:** Agdbentures is a game to teach debugging. It is based on GDB (the Gnu Debugger), uses the Easytracker library, and proposes to students an RPG-like (Role Playing Game) 2D interface, where each level is a program in the C language that contains one or more bugs. To validate a level, one needs to first correct the bugs that block the main character in its goals. Difficulty is gradual, and the code for each level is based (and expands) on the preceding level, which allows players to get familiar with the code base.

**URL:** <https://gitlab.inria.fr/CORSE/agdbentures>

**Contact:** Florent Bouchez

## 7 New results

Our team effort is structured along four axis:

1. *Automatic Derivation of Parametric Data Movement Complexity:* This research axis focuses on understanding and optimizing data movement in algorithms to improve computational performance
2. *Automatic Resource Characterization and Performance Feedback:* This axis focuses on developing tools for characterizing CPU architectures and debugging performance issues
3. *Performance Modeling, Schedule Optimization, and Code Generation for Tensor Computations:* This research axis focuses on optimizing tensor computations, which are crucial for various applications including machine learning.
4. *Teaching of Algorithms, Programming, and Debugging:* This research axis focuses on enhancing computer science education by developing tools for visualizing program execution, gamifying debugging practice, implementing active learning strategies, and generating automated exercises for Intelligent Tutoring Systems.

### 7.1 Automatic Derivation of Parametric Data Movement Complexity

**Participants:** Guillaume Iooss, Fabrice Rastello.

**I/O Complexity** When analyzing an algorithm’s properties, we often focus on its computational complexity, which measures the amount of computation required (ex:  $O(n^2)$  for a quadratic algorithm) and provides a rough information about the evolution of its performance related to the problem sizes. However, there are other important factor that are critical for performance, such as the amount of data movement needed to perform these computations.

We consider a simple generic memory model, composed of (i) a large memory of infinite size which contains the input data of the program; and (ii) a small memory of limited parametric size ( $S$ ) from which the computational unit can interact with its content. The *I/O complexity* of a program, also known as data-movement complexity, refers to the minimum amount of data movement an algorithm requires across all valid execution schedules.

To estimate this I/O complexity, we derive lower and upper parametric bounds using two different approaches. A lower bound can be established by mathematically proving that a certain volume of computation is necessary, using strategies such as the K-partitioning method or the wavefront method. An upper bound can be determined by demonstrating a schedule that matches this data movement quantity.

**Specialization to the Hourglass pattern** Our effort is focused on improving the lower bound derivation techniques, in order to improve their tightness compared to the upper bound. In particular, we consider specializations of the classical derivation techniques, and we leverage the additional properties of the targeted programs to improve asymptotically the derived bound.

We have identified a pattern of data dependencies, called the *hourglass dependency pattern*. An hourglass pattern involves successive reductions and broadcasts over a parametric number of elements, significantly constraining the shape of a valid tiling. Several important linear algebra algorithms, such as Gram-Schmidt, QR Householder, reduction to a bidiagonal matrix (geb2), and Hessenberg matrix factorization (gehd2), exhibit this pattern.

**Recent contribution** This year, we have studied an generalization of the hourglass pattern to cover even more algorithms, in particular from dynamic programming. Instead of focusing on pairs of reduction and broadcast dependencies, we rely on dominance properties. We also consider the possibility of reordering the terms of a reduction (using associativity and commutativity properties of their operators) as part of the scheduling decision, which can lead to a different bound. This is important for some kernels such as nussinov. A journal paper on the topic of this work is currently in preparation.

## 7.2 Automatic Resource Characterization and Performance Feedback

**Participants:** Alban Dutilleul, Lucas Sube, Guillaume Huard, Fabrice Rastello.

### 7.2.1 Gus

Modern Out-of-Order (OoO) CPUs are intricate systems with numerous interconnected components. Identifying performance bottlenecks and understanding the root causes of program performance issues are essential to fully leverage the hardware's capabilities. Current performance debugging methods either measure resource utilization to determine which CPU parts cause performance limitations or analyze code based on capacity/throughput models to derive bottleneck information. These approaches are constrained by instrumental and methodological precision, face portability issues across different microarchitectures, and often provide factual data about resource constraints without offering causal insights on how to resolve them.

Gus (Section 6.1.1), a performance debugging and analysis tool we began designing and developing several years ago, uses a resource-centric CPU model driven by dynamic binary instrumentation to detect complex bottlenecks resulting from the interplay of hardware and software factors. Bottlenecks are identified through sensitivity-based analysis, a form of model parameterization that employs differential analysis to reveal constrained resources.

This year efforts on Gus aimed at enhancing the user-friendliness of its output. Key updates include integrating the causality feature and establishing a benchmarking campaign to validate the model against native performance and GEM5. This initiative also provide an opportunity for code refactoring, model fine-tuning, and bug resolution.

### 7.2.2 SEATOP

As Gus provides the developer with information about kernels bottlenecks at the assembly code level, the link between these bottlenecks and the original source code is usually not straightforward at all. Indeed, optimized kernel generally undergo several source-to-source transformations, such as tiling, loop interchange, loop unrolling, vectorization, in addition to the optimization passes performed by the compiler. Thus, the resulting assembly code can be especially hard to understand and correlate with the source code on which the developer is working.

To address this issue, we developed SEATOP (Semantically Enhanced Asm Tracer for Optimized Programs, Section 6.1.2), a tool to help debugging the performance of an optimized kernel. It enriches traces collected from the optimized application with semantic information extracted from the original unoptimized source code. To this indent, it combines binary instrumentation and static debug information to compare two executions of the same program : an unoptimized one an optimized one. It then produces an assembly trace of the instructions executed by the optimized version, annotated with high-level semantic information: equivalent source code line and relevant variables value. Although SEATOP main objective is to assist in debugging performance of the implemented optimizations (tiling, loop interchange, loop unrolling, vectorization, ...), it can also be used to help debugging correctness of some transformations (for instance to

check that the transformed iteration domain remains unchanged). At the time of this report, SEATOP is still under development and should be released in the upcoming year.

### 7.2.3 DIESECT

The rapid proliferation of diverse and opaque processor microarchitectures from heterogeneous cores in Apple Silicon to specialized ARM server CPUs presents a critical scalability challenge for performance modeling.

Conventional characterization faces an unsatisfactory dichotomy: black-box methods are often brittle and susceptible to noise, while white-box approaches rely on non-portable, poorly documented performance counters that require significant human intuition.

To keep pace with the current rate of microarchitectural innovation, there is a clear need for automated, "grey-box" methodologies that can extract deep insights across a broad spectrum of designs.

In this context, we introduce DIESECT (Section 6.1.4), a portable microbenchmarking harness that exploits the fundamental, stable design principles of superscalar, out-of-order pipelines rooted in Tomasulo's algorithm. We utilize timing side-channel attacks on backend structures to infer the decomposition of instructions into micro-operations ( $\mu$ ops). By systematically inducing targeted analytical bottlenecks, we disentangle the intricacies of the throughput function without demanding vendor-specific tooling.

We have already begun applying this technique to extract microarchitectural information across multiple platforms, including Apple Silicon and several ARM designs.

Our objective is to integrate this reverse-engineered structural data with other microbenchmarks, such as throughput measurements, to develop a new, high-fidelity model of the processor pipeline. This combined approach aims to provide deeper, more accurate insights than existing black-box methods while maintaining the portability required for the modern hardware landscape.

## 7.3 Performance Modeling, Schedule Optimization, and Code Generation for Tensor Computations

**Participants:** Hugo Pompougnac, Christophe Guillon, Léon Frenot, Sylvain Noiry, Rui Cesista, Liam Semeria, Guillaume Iooss, Fabrice Rastello, Albert Cohen (*Google, France*), Saday Sadayappan (*University of Utah*).

Tensor computations is a class of program that includes matrix multiplication, convolution and similar operations. These are important for machine learning, since most of the computation of Convolutional Neural Network (CNN) or Large Language Model (LLM) fall into this class.

Our effort focuses on exploring how to optimize these computations from a compilation approach, and is centered around XTC, our new research platform (Section 7.3.1). From this tool, we have several on-going directions:

- With XTC, we have introduced Descript, a scheduling language that allows us to describe a schedule. We are investigating an extension to this language (called Descript++, Section 7.3.2) that allows us to define a search space and to automatically derive a sampler from it.
- In order to find the best schedules on CPU, we have to consider their behavior in respect to caches. Sarcasm (Section 7.3.3) is a new set-associative cache model which is specialized to tensor operator schedules, and fast enough to be applied on many schedules in a reasonable time.
- MIOpBench (Section 7.3.4) is a benchmarking tool that extracts operators from a set of models, and provides benchmarking utilities to estimate the efficiency of their performance.
- SDist (Section 7.3.7) is a backend for the MPPA machine and the *AMD AIE*. It requires extending Descript and the whole XTC framework to support distributed computation, due to the nature of these targets.

### 7.3.1 XTC: A Research Platform for Optimizing AI Workload Operators

We developed XTC, a research platform that decouples scheduling specification from code generation and performance measurement for AI workload optimization. The platform addresses a fundamental limitation in current compiler research: existing scheduling languages such as TVM’s Tensor Expression, Halide, and MLIR’s Transform dialect remain tightly coupled to their respective compilation ecosystems, preventing fair comparison, reuse, and evaluation across frameworks.

XTC provides three main contributions. First, a unified API for scheduling that abstracts core components from multiple scheduling languages, exposing ten scheduling primitives (strip-mine, interchange, split, unroll, vectorize, parallelize, pack, bufferize, and fusion operations) through a consistent interface. This API enables seamless integration with diverse scheduling compilers while supporting a higher-level declarative language for manual experimentation. Second, a unified API for measurement that provides a controlled, cross-platform harness for performance evaluation. This infrastructure accesses hardware performance counters on x86 and ARM CPUs (including, to our knowledge, the first such access on Apple Silicon), as well as NVIDIA GPUs, enabling reproducible and quantitative comparisons across compilation pipelines. Third, integration with state-of-the-art backends including TVM and MLIR, allowing researchers to leverage rapidly evolving compiler ecosystems while maintaining the flexibility to evaluate research prototypes through user-defined backends.

The platform also provides interfaces for automating design space exploration, enabling experts to connect high-level scheduling strategies with custom sampling and predictive models. We demonstrated this capability by reproducing the search spaces of state-of-the-art autotuners such as Anson.

Our evaluation showed that XTC matches hand-tuned C code performance for matrix multiplication using the Goto strategy, validated cross-backend consistency between TVM and MLIR, identified limitations in MLIR’s vectorization pass for convolution operations, and demonstrated integration within the Aidge framework achieving 15-30× speedup on Intel CPUs compared to generic C++ export for neural network inference.

Impact and dissemination. XTC was first presented at the University of Cambridge during the MLIR Summer School, introducing the platform to the compiler research community. We are delivering a tutorial on performance engineering based on XTC at the CGO conference, and presenting the tool at the C4ML workshop. From a research perspective, XTC now serves as the foundation for developing a novel scheduling language based on constraint programming for defining loop optimization search spaces, opening new directions for automated exploration of transformation sequences.

### 7.3.2 Descript++

Optimizing a linear algebra operator implies searching for performing configuration in a large space of configurations, while balancing the various bottlenecks of the considered architecture. An expert can guide this exploration using their insight of the architecture and manually examine a sequence of configurations. On the other extreme, autotuning techniques are fully automated process that require less expert knowledge. However, this knowledge is usually embedded in the search space and cannot be modified.

*Descript++* aims at unifying these approaches by introducing a scheduling language, *Descript*, that is extended to obtain an autotuning language. This means that an expert can specify a configuration to examine it, but also build a search space by adding variables and constraints. From this search space specification, we can automatically infer a usable sampling procedure that outputs a list of configuration from this search space. This allows a fine-grain control of the expertise used in an autotuner.

### 7.3.3 Sarcasm

In order to estimate the performance of a schedule at compile time, data movement is a critical part. On CPU, this means that we need to estimate how many cache misses occur at each cache level, since it linked with one of the major potential performance bottleneck. More precisely, given a set of schedules which are already optimized for the register level (vectorized + microkernel), we wish for a cache model that is able to identify the set of configurations that minimize the cache misses.

In the literature, the analytical cache models are either (i) very precise but very slow (for example, by using Presburger sets to accurately model the points in the program in which a cache miss occur); (ii) very

fast but very inaccurate (all fully associative cache model falls into this category, but are unusable on a set-associative cache); or (iii) heuristics that sacrifices precision of the model for speed.

We introduced *Sarcasm* (Set-Associative Rotating Cache Analytical/Simulating Model, Section 6.1.5) as a set-associative analytical cache model that is fast enough to be usable in the context of a sampler, while mostly preserving the ordering of configurations according to their number of cache misses. We exploit the shape of the considered kernels, and in particular the fact that all the footprints at every tile levels are rectangular.

This tool has been integrated inside *XTC*, and extended to support sequential composition of microkernels, which was a feature of one of our previous work (called *Ttile* [7]). A journal paper on this topic has been submitted and is currently being reviewed.

### 7.3.4 MIOpBench: Benchmarking of Machine learning Operators

Having access to relevant benchmark suites is crucial for the evaluation of compiler optimizations that are developed in the team. In the context of machine learning models, we need to compare our work with state of the art compilers and libraries, at the granularity of operators. Thus we have decided to create a dedicated tool to extract operators from models and benchmark them on several implementations.

We have specifically designed the tools for researchers working on compilers or performance optimization for machine learning operators. Starting from existing sets of models (e.g. MLPerf 5.1 inference) or a list of models provided by the user (e.g. LLama3.1 8B), it extracts the datatype, shape and name of each operator. This first step furnishes some data for some early analysis, like operators classification, extraction of tensor sizes, or *I/O complexity* analysis.

The second step is the benchmarking of each operator on a set of libraries that implement it. They can be general (e.g. XNNPack) or hardware specific (e.g ARM performance library). The tool reuses the runtime developed in *XTC* for a fair evaluation of the generated code using performance counters. More advanced analysis like the *Roofline analysis* can be performed at this point.

### 7.3.5 First-Class Verification Dialects for MLIR

A five-month research stay at the University of Cambridge led to a collaboration that resulted in a publication at PLDI [2]. This work addresses a core challenge in the MLIR compilation infrastructure: the lack of a unified framework for expressing and verifying semantic properties directly within the intermediate representation. The paper proposes introducing verification dialects as first-class constructs in MLIR, enabling the encoding of invariants, pre- and post-conditions, and correctness properties at each abstraction level of the compiler. This approach ensures that correctness guarantees are preserved throughout successive transformations of the intermediate representation, whereas existing methods rely on ad hoc or external verification outside the compilation pipeline. The paper demonstrates the applicability of this methodology across several existing MLIR dialects and evaluates its impact on detecting transformation errors.

### 7.3.6 Multi-level x86 Backend for xDSL

This collaboration with Sasha Lopoukhine, building on our joint work with the University of Cambridge, aims to design an x86 backend for the MLIR xDSL compiler. It extends the approach presented in A Multi-level Compiler Backend for Accelerated Micro-kernels Targeting RISC-V ISA Extensions. The original work proposed a multi-level compilation pipeline in xDSL, progressively lowering high-level representations to emit optimized micro-kernels targeting RISC-V ISA extensions, leveraging MLIR's abstractions at each stage. The ongoing work adapts this methodology to the x86 architecture, addressing specific challenges related to the complexity of instruction encoding, register set management, and the exploitation of vector extensions (AVX). The goal is to demonstrate the generality of the multi-level approach beyond a single target architecture and to provide a functional x86 backend within the xDSL ecosystem, paving the way for high-performance code generation on widely deployed platforms.

### 7.3.7 SDist: Abstract distribution primitives and generate code for Kalray MPPA and Amd AIE

Following the development of an end-to-end *MLIR* based toolchain from machine learning models implemented with *Pytorch* to the *Kalray MPPA* accelerator, *SDist* has been created to facilitate the exploration of schedules

for tensor operations, for machines with distributed local memories. It can generate code for the MPPA machine, but also the *AMD AIE* accelerator.

To reason on distributed implementations of machine learning operators, we need to abstract the low-level communication primitives of each Hardware. We decided to create a small set of scheduling primitives for the distribution of tensors and loops. Then we implemented the lowering to the MPPA using the previously developed toolchain, and a lowering to the AIE is in progress. *SDist* is entirely *MLIR* based. A utility for the visualization of distributed tensor is also provided.

## 7.4 Teaching of Algorithms, Programming, and Debugging

**Participants:** Florent Bouchez Tichadou, Guillaume Huard, Christophe Guillon, Manuel Selva, Guillaume Huard, Fabrice Rastello.

Our goal here is to combine our expertise in compilation and teaching to help teachers and learners in computer science fields such as programming, algorithms, data structures, automata, debugging, or more generally computing literacy. This axis is developed into two ongoing projects:

- EasyTracker: a library for controlling and inspecting the execution of a program.
- Agdbentures: a game that helps learners to gain skills in debugging, which is based on EasyTracker. See Section 6.1.10.

### 7.4.1 Easytracker : A generic library for controlling and inspecting program execution and state

Learning to program involves building a mental representation of how a machine executes instructions and stores data in memory. To help students, teachers often use visual representations to illustrate the execution of programs or particular concepts in their lectures. As a famous example, teachers often represent references/pointers with arrows pointing to objects or memory locations. While these visual representations are mostly hand-drawn, there is a tendency to supplement them with tools. However, building such a tool from scratch requires much effort and a high level of debugging technical expertise, while existing tools are difficult to adapt to different contexts.

EasyTracker is a Python library targeting teachers who are not debugging experts. By providing ways of controlling the execution and inspecting the state of programs, EasyTracker simplifies the development of tools that generate tuned visual representations from the controlled execution of a program. The controlled program can be written either in Python, C, or assembly languages.

Currently, the EasyTracker library is sufficiently stable for our usage and only minor improvements were made this year. EasyTracker is now also used in more courses:

- 1st year at Ensimag - Basics of imperative programming with Python
- 1st year at Ensimag - C programming
- L2 at UGA - Algorithms and Imperative Programming
- L3 at UGA - Algorithms and Modelization

### 7.4.2 Agdbentures: A game to learn how to debug in autonomy

Debugging is an important task in software development and can be the source of a lot of frustration and time consumption. However, it is not often taught explicitly in computer science curricula even at university level. For these reasons, we develop Agdbentures (see Section 6.1.10), a debug practicing game where “levels” consist of programs containing bugs that the learner needs to debug to advance in the game.

In Agdbentures, the level programs are executed using Easytracker, which enables us to present a live visual representation of the program state during execution in the form of a 2D RPG-like world. For instance, the “player\_x” and “player\_y” variables in the level code are inspected at runtime and used to place a character representing the player on a graphical 2D map. The interest is three-fold: First, this makes the

game appealing as the player/learner is plunged into a “real” game; Second, it showcases the importance of having information on the state of the program being executed in order to be able to debug; Third, it separates completely the graphical code, which can be very complex and is hidden from players, from the level code which is given to players: this let us simplify the source code so novice programmers won’t be rebuked. The levels share a common codebase that is increasing in size and complexity as the player advances in the game. It initially only controls the main character position, then more features are added such as interactive objects, NPCs (non playable characters), level logic (activating levers, collecting items...). This helps the player getting familiar with a codebase of increasing size over time so we can present more interesting bugs where locating the problem is similar to what happens in real life development. It also let us to create “fun” levels where bugs have interesting or amusing effects on the visual representation, and where finding the solution (fixing the bugs) is rewarding.

In past experiments, we obtained very encouraging results about the engagement of students at the L2 university level. All were eager to participate and declared they would really like to continue playing Agdbentures on their own with more levels. This year, we continued to propose internship positions to students to help develop Agdbentures and the main current efforts focus on testing the current projet, coverage of interesting debugging situations, and on the whole game scenario. The objective is to complete a release that meet interesting pedagogical goals while maintaining players investment. Future efforts will be focused on making the game accessible to students and evaluating the pedagogical impact on students’ debugging skills.

## 8 Bilateral contracts and grants with industry

### 8.1 Bilateral contracts with industry

With the PhD thesis (CIFRE) of Syvlain Noiry a bilateral contract has been signed with Kalray.

**Title** Neural Network Compilation for a Distributed Memory Acceleration Platform

**Duration** 2024-2027

**Abstract** There are various infrastructures for optimizing deep learning networks, but they often rely on existing libraries, complicating joint optimization of operators. The thesis aims to uniformly manage tensor operator optimization across different hardware levels using a domain-specific intermediate representation (IR) and auto-tuning mechanisms, targeting the Kalray MPPA Coolidge data processing unit.

## 9 Partnerships and cooperations

### 9.1 International research visitors

#### 9.1.1 Visits to international teams

##### Research stays abroad

**Hugo Pompougnac**

**Visited institution:** University of Cambridge

**Country:** United Kingdom

**Dates:** Oct 2024 - Feb 2025

**Context of the visit:** Advancing compiler infrastructure by introducing formal verification dialects in MLIR and extending a multi-level backend approach from RISC-V to x86 architectures.

**Mobility program/type of mobility:** Research stay

## 9.2 National initiatives

### BPI OTPaaS

**Title:** Développement et renforcement de la filière française et européenne du Cloud

**Duration:** October 2021 – Mai 2025

**Coordinator:** P. Betinelli

**CORSE contact:** Fabrice Rastello

**CORSE participants:** Fabrice Rastello, Christophe Guillon

**Partners:** Agileo, Atos, Captronic, Duliprint, IMT, MDM, Prosyst, SE, Soben, Tridimeo, Solem, CEA, Valeo

**INRIA Partners:** DataMove

**Summary:** The OTPaaS project targets massive digitization by offering a suitable cloud for scanning that is compatible with Gaia-X and easy to use by companies including SMEs. The consortium brings together national technology providers and users from major groups and SMEs/ETIs, with strong support from major French research institutes. The platform OTPaaS will be validated by 6 demonstrators and followed by ambitious industrialization programs.

### BPI DeepGreen

**Title:** Plateforme indépendante pour le deep learning embarqué

**Duration:** April 1st 2023 – Mai 2027

**Coordinator:** CEA

**CORSE contact:** Fabrice Rastello

**CORSE participants:** Fabrice Rastello, Christophe Guillon, Hugo Pompougnac, Valentin Trophine, Guillaume Iooss

**Partners:** CEA, Adagos, Pulse Audition, Kalray, Dolphin Design, Thales Research & Technology France, Arcys, MBDA, Arcelormittal, EDF, Sysnav, Hawai.tech, Ezako

**Summary:** The DeepGreen project aims to bring together major industrial players and small and medium-sized enterprises (SMEs) in France for the deployment of Artificial Intelligence on constrained hardware targets through a software platform that meets the requirements and expectations of each stakeholder.

### HOLIGRAIL – PEPR AI

**Title:** HOLlistic approaches to GReener model Architectures for Inference and Learning

**Duration:** Oct 1st 2023 – 2027

**Coordinator:** Olivier Sentieys

**CORSE contact:** Fabrice Rastello

**CORSE participants:** Fabrice Rastello, Christophe Guillon, Hugo Pompougnac

**Partners:** CEA List, TIMA

**INRIA Partners:** Taran, Emeraude

**Summary:** The vision of this action is to create a synergy with the research on the foundations of AI frugality to propose cutting-edge methods that significantly improve the energy efficiency of both inference and training of a model. We will propose (i) more compact and efficient number representations that still maintain a quality of inference or training close to the reference, (ii) hardware-aware training algorithms that enhance certain types of sparsity (e.g., more structured), coding compactness (aggressive quantization, maximum entropy) and tensor transformations. Most state-of-the-art solutions are agnostic of the hardware they run on. By taking advantage of this interplay between the hardware and the algorithms, we can achieve breakthroughs beyond current solutions, in particular by developing (iii) efficient hardware mechanisms, especially optimized to take advantage of sparsity, extreme quantization and ad-hoc number representations, together with (iv) compiler optimizations, to demonstrate the effectiveness of the proposed methods. Our approaches are holistic in the sense that they will jointly optimize the whole computing stack of AI, i.e., at the algorithm, arithmetic, compiler and hardware levels.

## 10 Dissemination

### 10.1 Promoting scientific activities

#### 10.1.1 Scientific events: organisation

##### Member of the organizing committees

- Fabrice Rastello: Member of the steering Committee of ACM/IEEE CGO
- Fabrice Rastello: Member of the steering committee of the MLIR (Un)School September 2025
- Hugo Pompougnac: Member of the steering committee of the MLIR (Un)School September 2025

#### 10.1.2 Scientific events: selection

##### Member of the conference program committees

- Fabrice Rastello: ACM/IEEE CGO 2026 conference
- Fabrice Rastello, [End25](#)

#### 10.1.3 Invited talks

- Guillaume Iooss: Invited talk by the CASH team in Lyon (May 2025).
- Fabrice Rastello: Invited talk at UDC (Universidade da Coruña) "[Compiler design in the age of specialized libraries: bridging performance model with autotuning](#)"

#### 10.1.4 Leadership within the scientific community

- Fabrice Rastello: member of Inria evaluation committee (CE) since Sept 2023
- Fabrice Rastello: deputy scientific director of Inria Grenoble Rhône-Alpes (DSA) since Sept 2022
- Fabrice Rastello: scientific council of Inria Grenoble Rhône-Alpes (CoS)
- Fabrice Rastello: vice-president of the Inria CRCN/IFSP recruiting committees 2025
- Fabrice Rastello: co-PI of the Numeric/ASIC agencies PEPR Camelia (32M€)
- Fabrice Rastello: co-PI of the sub-project PC4 of PEPR Camelia
- Fabrice Rastello: coordinator of UGA Idex REUSE program (500k€)
- Fabrice Rastello: Inria Delegate to CoARA (Coalition for Advancing Research Assessment)

### 10.1.5 Scientific expertise

- Fabrice Rastello: Strategic Orientation Committee for ICube/ICPS
- Fabrice Rastello: Inria CRCN-TH recruiting committee 2025
- Fabrice Rastello: ANR review

### 10.1.6 Research administration

- Guillaume Iooss: RADAR Correspondant

## 10.2 Teaching - Supervision - Juries - Educational and pedagogical outreach

### 10.2.1 Teaching

- Licence 1: Florent Bouchez Tichadou, Operating System and Programming, 40 hours, UGA.
- Licence 2: Florent Bouchez Tichadou, Algorithms languages and programming, 101 hours, UGA.
- Licence 3: Florent Bouchez Tichadou, Algorithms, 17 hours, UGA.
- Licence 3: Florent Bouchez Tichadou, Programming project, 20 hours, UGA.
- Licence 3: Guillaume Huard, UNIX & C programming, 50 hours, UGA.
- Licence 3: Guillaume Huard, Object Oriented and Event-Driven Programming, 50 hours, UGA.
- Master 1: Guillaume Huard, Programming for teacher apprentices, 8 hours, UGA.
- Licence 3: Guillaume Huard is responsible of L3 Informatique Générale at UGA.
- Licence: Guillaume Huard is responsible of Licence Informatique diploma at UGA.

### 10.2.2 Supervision

- PhD in progress: Lucas Maisonnave: Maximum Entropy Coding for Deep Neural Networks, march 2023, co-advised by Olivier Bichler (CEA LIAE – 95%) and Fabrice Rastello (5%)
- PhD in progress: Sylvain Noiry: Compilation of Neural Networks for Distributed Memory Architecture, April 2024, advised by Fabrice Rastello
- PhD in progress: Valentin Trophine: Asynchronous Programming Under Memory Constraints, Oct 2024, co-advised by Frédéric Wagner (Inria DATAMOVE – 90%) and Fabrice Rastello (10%)
- PhD in progress: Alban Dutilleul: Beyond Microarchitectural Black Boxes: Grey-Box Throughput Modeling for Superscalar Out-of-Order Processors, Oct 2025, advised by Fabrice Rastello
- PhD in progress: Léon Frénot: Towards an Interactive Compilation Infrastructure Guided by Expertise for the Optimisation of Deep Learning Programs, Oct 2025, advised by Fabrice Rastello

### 10.2.3 Juries

#### Guillaume Iooss

- CSI (Comité de Suivi Individuel de thèse), Arthur Viens.

## Fabrice Rastello

- PhD, Nahuel PALUMBO–Lille, Chair, Nov 28 2025. DRUID - Métacompilation des compilateurs JIT de base.
- PhD, Clément Rosseti–Strasbourg, Chair, Dec 18 2025. Algebraic loop transformations.
- PhD, Omid Asudeh–Univ Utah, Committee, Dec 15 2025. Reordering Sparse Matrices for Performance Gain: Evaluation, Analytical Modeling, and Strategy Design.

## 10.3 Popularization

- Fabrice Rastello: scientific council of CEA-EDF-Inria summer schools
- Hugo Pompougnac: scientific council of the MLIR (Un)School September 2025

### 10.3.1 Book

Hugo Pompougnac co-authored a collective work titled *Que faire de l'IA* (What to Do with AI), produced in collaboration with economists, legal experts, and union representatives. This book aims to bridge the gap in public understanding of the societal challenges posed by artificial intelligence. Positioned at the intersection of technical expertise and social sciences, the book seeks to demystify the underlying mechanisms of AI while examining their tangible impacts on labor, law, and social relations. It critically engages with dominant narratives about AI—whether overly optimistic or alarmist—by grounding the discussion in economic and legal realities. Central to its analysis are the transformation of work, regulatory frameworks, and decision-making power. The multidisciplinary author team brings together complementary perspectives: technical insights into AI systems, analysis of existing and emerging legal frameworks, and firsthand accounts of working conditions in sectors affected by automation.

## 11 Scientific production

### 11.1 Publications of the year

#### International journals

- [1] T. Bastian, H. Pompougnac, A. Dutilleul and F. Rastello. ‘CesASMe and Staticdeps: static detection of memory-carried dependencies for code analyzers’. In: *ACM Transactions on Architecture and Code Optimization* 22.2 (30th June 2025), pp. 1–23. DOI: [10.1145/3715125](https://doi.org/10.1145/3715125). URL: <https://inria.hal.science/hal-05460168>.
- [2] M. Fehr, Y. Fan, H. Pompougnac, J. Regehr and T. Grosser. ‘First-Class Verification Dialects for MLIR - PLDI 2025 Artifact’. In: *Proceedings of the ACM on Programming Languages, PLDI* 9.206 (2025), pp. 1466–1490. DOI: [10.5281/zenodo.15231119](https://doi.org/10.5281/zenodo.15231119). URL: <https://inria.hal.science/hal-05501935> (cit. on p. 16).

#### International peer-reviewed conferences

- [3] O. Asudeh, S. Mahdipour Saravani, F. Rastello, G. Sabin and P. Sadayappan. ‘How effective is matrix reordering for improving performance of sparse matrix-vector multiplication?’ In: *SC Workshops ’25: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*. St Louis, United States: ACM, 15th Nov. 2025, pp. 823–827. DOI: [10.1145/3731599](https://doi.org/10.1145/3731599). URL: <https://inria.hal.science/hal-05460196>.
- [4] L. Maisonnave, C. Moineau, O. Bichler and F. Rastello. ‘Precision Where It Matters: A Novel Spike Aware Mixed-Precision Quantization Strategy for LLaMA-Based Language Models’. In: *ICONIP 2024 - International Conference on Neural Information Processing*. Vol. 2295. Communications in Computer and Information Science. Auckland, New Zealand: Springer Nature Singapore, 24th June 2025, pp. 327–341. DOI: [10.1007/978-981-96-7030-7\\_23](https://doi.org/10.1007/978-981-96-7030-7_23). URL: <https://inria.hal.science/hal-05460303>.

## Reports & preprints

- [5] L. Maisonnave, C. Moineau, O. Bichler and F. Rastello. *Gradual Binary Search and Dimension Expansion : A general method for activation quantization in LLMs*. CORSE - Compiler Optimization and Run-time Systems, 2025. DOI: [10.48550/arXiv.2504.13989](https://doi.org/10.48550/arXiv.2504.13989). URL: <https://inria.hal.science/hal-05460223>.
- [6] H. Pompougnac, C. Guillon, S. Noiry, A. Dutilleul, G. Iooss and F. Rastello. *XTC, A Research Platform for Optimizing AI Workload Operators*. CORSE - Compiler Optimization and Run-time Systems, 2025. DOI: [10.48550/arXiv.2512.16512](https://doi.org/10.48550/arXiv.2512.16512). URL: <https://inria.hal.science/hal-05460129>.

## 11.2 Cited publications

- [7] N. Tollenaere, G. Iooss, S. Pouget, H. Brunie, C. Guillon, A. Cohen, P. Sadayappan and F. Rastello. ‘Autotuning Convolutions is Easier Than You Think’. In: *ACM Transactions on Architecture and Code Optimization* (Nov. 2022), pp. 1–23. DOI: [10.1145/3570641](https://doi.org/10.1145/3570641). URL: <https://inria.hal.science/hal-03844272> (cit. on p. 16).