RESEARCH CENTRE

Inria Centre at Université Grenoble Alpes

IN PARTNERSHIP WITH:

Université de Grenoble Alpes

2024

ACTIVITY REPORT

Project-Team

CORSE

Compiler Optimization and Run-time SystEms

IN COLLABORATION WITH: Laboratoire d'Informatique de Grenoble (LIG)

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Inria

Contents

Project-Team CORSE

Creation of the Project-Team: 2016 July 01

Keywords

Computer sciences and digital sciences

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
A2.1.8. – Aspect-oriented programming
A2.1.10. – Domain-specific languages
A2.2.1. – Static analysis
A2.2.2. – Memory models
A2.2.3. – Memory management
A2.2.4. – Parallel architectures
A2.2.5. – Run-time systems
A2.2.6. – GPGPU, FPGA...
A2.2.8. – Code generation
A2.3.2. – Cyber-physical systems
A3.4.1. – Supervised learning
A3.4.3. – Reinforcement learning
A3.4.5. – Bayesian methods
A3.4.8. – Deep learning
A4.4. – Security of equipment and software
A7.1. – Algorithms
A9.6. – Decision support

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 (harware, equipments...)
B9.1. – Education

1 Team members, visitors, external collaborators

Research Scientists

- Fabrice Rastello [Team leader, INRIA, Senior Researcher]
- Cesar Fuguet Tortolero [INRIA, Researcher, from Nov 2024]
- Guillaume Iooss [INRIA, Researcher]

Faculty Members

- Florent Bouchez [UGA, Associate Professor]
- Ylies Falcone [UGA, Associate Professor]
- Guillaume Huard [UGA, Associate Professor, from Sep 2024]
- Manuel Selva [GRENOBLE INP, Associate Professor]

Post-Doctoral Fellow

- Hugo Pompougnac [INRIA, Post-Doctoral Fellow]

PhD Students

- Theophile Bastian [UGA, from Sep 2024]
- Theophile Bastian [UGA, until Aug 2024]
- Sylvain Noiry [KALRAY, from Apr 2024]

Technical Staff

- Zikang Liu [INRIA, Engineer, from Oct 2024]
- Jevgenijs Protopopovs [INRIA, Engineer, from Feb 2024 until Aug 2024]
- Chukri Soueidi [INRIA, Engineer, until Mar 2024]
- Valentin Trophime-Gilotte [INRIA, Engineer, until Sep 2024]

Interns and Apprentices

- Thomas Civade [INRIA, Intern, from Jun 2024 until Aug 2024]
- Noe Genevois [UGA, Intern, from May 2024 until Jul 2024]
- Nikolaos Mavrogeorgis [INRIA, Intern, from Nov 2024]
- Tommy Prats [UGA, Intern, from May 2024 until Jul 2024]

Administrative Assistant

- Maria Immaculada Presseguer [INRIA]

External Collaborators

- Sebastian Hack [UNIV MANNHEIM]
- Gabriel Rodriguez Alvarez [Universidade da Coruña]

2 Overall objectives

Languages, compilers, and run-time systems are some of the most important components to bridge the gap between applications and hardware. With the continuously increasing power of computers, expectations are evolving, with more and more ambitious, *computationally intensive and complex applications*. As desktop PCs are becoming a niche and servers mainstream, three categories of computing impose themselves for the next decade: mobile, cloud, and super-computing. Hence *diversity, heterogeneity* (even on a single chip) and thus also *hardware virtualization* are putting more and more pressure both on compilers and run-time systems. However, because of the energy wall, *architectures* are becoming more and more *complex* and *parallelism ubiquitous* at every level. Unfortunately, the memory-CPU gap continues to increase and energy consumption remains an important issue for future platforms. To address the challenge of *performance and energy consumption* raised by silicon companies, compilers and run-time systems must *evolve* and, in particular, interact, *taking into account the complexity of the target architecture*.

The overall objective of CORSE is to address this challenge by *combining static and dynamic compilation* techniques, with more interactive *embedding of programs and compiler environment in the run-time system*.

3 Research program

3.1 Scientific Foundations

One of the characteristics of CORSE is the use of diverse advanced mathematical tools as a basis to our research. Compiler optimization requires the usage of several tools around discrete mathematics: combinatorial optimization, algorithmic, and graph theory. The aim of CORSE is to tackle optimization not only for general purpose but also for domain specific applications. In addition to run-time and compiler techniques for program instrumentation, hybrid analysis and compilation advances will be mainly based on polynomial and linear algebra.

The other specifics of CORSE is to address technical challenges related to compiler technology, run-time systems, and hardware characteristics. This implies mastering the details of each. This is especially important as any optimization is based on a reasonably accurate model. Compiler expertise will be used in modeling applications (e.g. through automatic analysis of memory and computational complexity); Run-time expertise will be used in modeling the concurrent activities and overhead due to contention (including memory management); Hardware expertise will be extensively used in modeling physical resources and hardware mechanisms (including synchronization, pipelines, etc.).

The core foundation of the team is related to the combination of static and dynamic techniques, of compilation, and run-time systems. We believe this to be essential in addressing high-performance and low energy challenges in the context of new important changes shown by current application, software, and architecture trends.

3.2 Main Research Directions

Our project is structured along three main directions. The first direction belongs to the area of **program analysis and optimization**. This direction breaks down into:

- Performance debugging, binary instrumentation, automatic characterization and simulation of architectures
- Loop scheduling, data locality, I/O complexity
- Compiler design, hybrid compilation, domain-specific intermediate representations

The second direction belongs to the area of **runtime monitoring, verification, and enforcement**. This direction breaks into:

- Instrumentation of Java programs for performance and security

- Monitoring of learning-enabled components using geometrical shape abstraction
- Decentralization of the monitoring process for multi-threaded and distributed systems
- Predictive monitoring of business processes

The third direction belongs to the area of **teaching and tutoring of programming**. This direction breaks into:

- Visualisation tools for teaching programming
- Tools and education of debugging
- Problem based learning, generation, recommendation

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 KALRAY, GOOGLE, STMICROELECTRONICS, 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 New software, platforms, open data

6.1 New software

6.1.1 IOLB

Keywords: Complexity, Polyhedral compilation, Performance analysis

Functional Description: IOLB computes a symbolic lower bound on the I/O, or data movement, complexity of a computer program, that is the amount of data that needs to be moved between cache and main memory to perform its computation. The input is a C program, and the output is a mathematical formula that depends on program parameters (array sizes...) and cache size.

URL: https://gitlab.inria.fr/CORSE/iolb

Publications: hal-02421026, hal-02910961

Contact: Guillaume Iooss

6.1.2 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

Participants: Nicolas Derumigny, Fabrice Rastello, Theophile Bastian

6.1.3 GUS

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

Functional Description: GUS' goal is to detect performance bottlenecks at the very low level on monothread 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: Nicolas Derumigny

6.1.4 CesASMe

Keywords: CPU, Microarchitecture simulation, Performance analysis

Functional Description: CesASMe fulfills 2 goals: 1- It automatically generates a large amount of microbenchmarks (i.e. benchmarks whose computation is L1-resident) from a benchmark suite, each implementing a different loop optimisation : tiling, loop fusion, etc. 2- For each compiled microbenchmark, it collects execution time estimates provided by the studied code analyzers (uica, iaca, Gus...), lifts them to common metrics and compare them with each other and with a measure.

Contact: Theophile Bastian

6.1.5 staticdeps

Keywords: CPU, Microarchitecture simulation, Performance analysis

Functional Description: Given an executable, staticdep focuses on the loops within. It computes statically a triplet (source, dest, k) for each dependency, where source is the instruction producing the data carrying the dependency, dest is the instruction consuming the data carrying the dependency, and k is the number of iterations required for the dependency to appear.

URL: https://gitlab.inria.fr/CORSE/uica-staticdeps

Contact: Theophile Bastian

6.1.6 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

6.1.7 sarcasm

Name: Set-Associative Rotating Cache Analytical/Simulating Model

Keywords: Cache, Operational intensity

Functional Description: This repository includes several elements: (i) an experiment script based on Dinero, a cache simulator, that extracts its cache miss prediction, (ii) a fully-associative cache model, based on the IOOpt hypothesis, (iii) a set-associative cache model, based on the notion of detailed footprint, (iv) a collection of experimental data and experiment script, whose objective is to estimate the ability of cache models (and of the operational intensity metric) to select highly performant configurations. These implementations focus on the configurations from a pertinent search space for tensor operators (cf the Ttile tool), and more precisely on matrix multiplication and convolutions.

URL: https://gitlab.inria.fr/CORSE/sarcasm

Contact: Guillaume Iooss

6.1.8 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: - Concision & clarity for interactive transformation - Ease of manipulation for scheduling transformations - Limited scope focusing exclusively on scheduling of linear algebra operations found in ML frameworks - Integration with existing compiler infrastructure (MLIR, Polygeist, Z3)

Core Features: - Code transformation and optimization framework - Schedule-based optimization system - Constraint-based autotuning capabilities - Support for benchmarking and performance analysis - Interactive transformation tool

Technical Details: - AST based J IR for loops and buffer descriptions - Transformations directly done at the J IR level - Backend codegenration for MLIR/LLVM infrastructure - Leverages Z3 theorem prover for constraint solving - Implements Gibbs sampling for schedule exploration

Specific Features: - Unlike Halide, TACO, or TVM, JIR deliberately excludes computation description - Relies on externally supplied computation primitives - Concentrates purely on scheduling problems

High-Level MLIR Integration: - Uses MLIR affine dialect as target - Preserves loop structure and iteration space information - Enables additional backend-level optimizations

Unified Intermediate Language: - Maintains consistent representation throughout transformations - No lowering until final MLIR translation - Enables better reasoning about transformations

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

URL: https://gitlab.inria.fr/CORSE/jir

Contact: Christophe Guillon

6.1.9 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 projet provides high-level syntax for applying transformations like: - Tiling - Loop interchange - Vectorization - Unrolling

It allow the integration of multiple backend and implements:

- MLIR backend (using MLIR linalg and transform dialects) - TVM backend (using TVM tensor IR and schedule APIs) - JIR backend (using JIR intermediate representation)

Core components: - Abstract interfaces for tensors, graphs, nodes, and operators - Implementers for different backends - Schedulers for transformation management - Compilers for code generation - Evaluators for performance measurement

Main tools: ‘mlir-loop‘: Compilation driver ‘loop-explore‘: Automatic exploration of different scheduling strategies ‘loop-display‘: Visualization of exploration results

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

Contact: Hugo Pompougnac

7 New results

7.1 Performance Debugging and Compiler Optimization

Participants: Fabrice Rastello, Guillaume Iooss, Christophe Guillon, Hugo Pompougnac, Alban Dutilleul, Nicolas Derumigny, Théophile Bastian, Albert Cohen *(Google, France)*.

Our current efforts with regard to code optimization follow fives directions.

1. The first direction focuses on **performance modeling**, especially data movement across cache hierarchies in pattern-specific applications. These include polyhedral frameworks and machine learning. Predicting cache misses quickly and accurately helps optimize loop transformations. We developed a new approach and tool (SARCASM) that estimates cache misses either more precisely, or much faster than existing methods.

2. The second direction also focuses on data movement. However, unlike the first, which models real architectures, this one uses abstract models. The goal is to automatically derive (prooved) **data movement complexity** bounds for polyhedral computing kernels. Last year, we developed a new proof technique called the *hourglass*. We then implemented it in our tool, IOLB. Later, we generalized it to cover a broader pattern, allowing for tighter bounds in more applications.

3. The third direction concerns **compiler infrastructure** driven by the need to optimize code for deep learning. In this context, we improved the previously developped code generator for sparse tensor computation. Most of our work here is associated to our contribution to the development of the AIDGE infrastructure (a sovereign infrastructure for deploying deep learning applications in embedded systems) which involves substantial engineering efforts. We developped two MLIR-based compiler prototypes that provide the ability for an expert to express scheduling constraints in an interactive or programative way. Thanks to a new back-end interface of AIDGE that we developped, we integrated different optimizers into the platform.

4. The fourth direction concerns the use of **autotuning** for compiler optimization. We refactored our code of Bayesian optimizer and performed some experimental evaluations allowing to benchmark and compare different representations such as the random forest and the Gaussian process. We also extensively benchmarked the use of SMT+Gibbs sampling in our constrained discrete search space.

5. The last direction concerns our efforts for promoting our **performance debugging** tool GUS. We believe that the tool improves over the state of the art both by the unique design of the simulator it is based on, but also by the software-based technique it implements to pinpoint fine grain performance bottlenecks. Our difficulties to publish this work lead us to evaluate its precision against state of the arts simulators (e.g. showing that while being faster than Gem5, GUS is also

more precise) and performance debugging tools such as TMA used in VTune and perf (showing the inaccuracy and limitations of such tools). With the objective of publishing the work in a top conference, we continue our efforts for improving GUS both in terms of speed, precision of cycle prediction and accuracy of bottleneck diagnostic, but also for evaluating those elements.

7.1.1 Performance Modeling: Data Movements for Tensor Computations

Dense Tensor computations, such as Matrix Multiplication, Tensor Contraction, and Convolution, are crucial operations used across various fields like Fluid Dynamics, Data Analytics, Economic Modeling, and Machine Learning. Creating highly optimized code for these operations requires a combination of finely tuned micro-kernels at the register/instruction level and appropriate multi-level tiling strategies.

To enhance the performance of tensor operations on CPUs, we attempted to combine two of our previous works (Ttile and Ioopt). However, during the evaluation, we found that the effectiveness of this combination was inconsistent. This led us to investigate the reasons behind these performance instabilities.

We focused on specific kernels like convolution and matrix multiplication, which are conceptually straightforward yet highly relevant across many scientific domains. These kernels consist of perfectly nested loops with rectangular domains, and we assumed that the array dimensions align with cache lines. This assumption introduces periodicity in memory accesses, enabling efficient computation of cache line distribution across cache sets. Based on this, we developed a new analytical set-associative cache model tailored for these computations, which is also very fast to evaluate. The implementation of this cache model is available in the Sarcasm tool (see Section 6.1.7).

To assess the effectiveness of cache models in identifying high-performing computations, we compared four methods for estimating cache misses: (i) direct measurement, (ii) cache simulation (Dinero), (iii) our set-associative analytical cache model (Sarcasm), and (iv) a fully-associative analytical cache model (based on the Ioopt hypothesis). We used an L3-resident benchmark to ensure that the L2 cache was the performance bottleneck and evaluated these models on 1,000 randomly selected configurations. Our study concluded that the fully-associative hypothesis in any cache model is unsuitable for selecting or predicting configurations with high computational intensity, which was the source of the instability issues we encountered with our Ioopt tool. Our findings are detailed in a preprint available on HAL [13], and we are preparing an improved version for journal submission.

7.1.2 Automatic derivation of parametric data movement complexity

When analyzing an algorithm's properties, we often focus on its computational complexity, which measures the amount of computation required. However, this metric does not account for other critical factors, such as the amount of data movement needed to perform these computations.

The *I/O complexity*, 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, 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.

We published [4] a new specialized derivation proof strategy to enhance the I/O complexity lower bound for programs exhibiting an *hourglass pattern* in their dependence graph. An hourglass pattern involves successive reductions and broadcasts over a parametric number of elements, significantly constraining the shape of a valid tiling. This pattern's properties can be leveraged to derive tight asymptotic lower bounds. Several important linear algebra algorithms, such as Gram-Schmidt, QR Householder, reduction to a bidiagonal matrix (gebd2), and Hessenberg matrix factorization (gehd2), exhibit this pattern. We integrated this proof technique into the automatic I/O complexity lower bound derivation tool, IOLB (see Section 6.1.1).

From this work, we generalized the hourglass pattern to cover even more algorithms. In particular, studying dominance sets instead of finding dependencies path between statements allows us to cover several dynamic programming algorithms, such as Nussinov, and refine even further the asymptotic bounds of several linear algebraic kernels. We study the interaction of the associativity and commutativity

properties of the reduction operator on the lower bound we derived. We are currently preparing a paper on this generalization.

7.1.3 Compiler infrastructure for optimizing dense and sparse tensor computation

This research axis is linked to two significant projects: Holigrail (PEPR IA) and Deepgreen. The goal of Deepgreen is to develop a comprehensive sovereign software infrastructure for running deep learning applications on embedded systems. The core platform, named AIDGE, enables model transformations (such as quantization) and code generation for various target architectures. CORSE's role is to provide compiler optimization capabilities to the platform.

In this context, substantial engineering efforts have been made to contribute to the core platform and to develop an appropriate API for integrating diverse code generators. This has allowed us to incorporate both TVM and the optimizers and code generators we developed. This setup provides an ideal infrastructure for experimenting with our research products.

For embedded systems, we believe there is a need for a compiler that offers a programmatic and interactive way to apply code transformations. The domain of tensor computation presents a great opportunity to reinvent compiler design. MLIR (Multi-Level Intermediate Representation) provides an excellent framework for developing an optimizing domain-specific compiler infrastructure. To compare possible approaches, we developed two different MLIR-based prototypes with their respective scheduling languages and code generators. Preliminary comparisons with TVM show comparable performance, allowing us to focus now on search strategies.

In parallel, we have been working on improving the performance of the Sparse-Matrix Multiplication (SpMM) kernel, which involves multiplying a sparse matrix A with a dense matrix B. Our initial goal is to understand and explain observed performance based on the sparsity structure of A, its density, its size (along with the size of B), and the optimization strategy and code generation used. This is challenging but crucial for designing scheduling heuristics and code generators.

Alongside this study, we designed a new execution scheme for SpMM on CPUs. State-of-the-art solutions exploit recurring patterns of non-zero values in the sparse matrix to create optimized register-level micro-kernels. Wilkinson et al. [1] aimed to generalize Goto et al.'s scheme [2], used in Intel MKL, for SpMM by generating specialized vectorized code for frequent patterns and scheduling them together, reducing branch misprediction overhead. However, their solution increases the randomness of memory accesses, exacerbating the bandwidth bottleneck.

We developed a code generator that results in more regular memory accesses without introducing branch misprediction. Although our current tiling strategy is too simplistic, our approach outperforms existing solutions when random accesses are the primary bottleneck. Before further refining our code generator, we decided to conduct an in-depth study of performance bottlenecks, as described above.

7.1.4 Autotuning of tiling transformation using Reinforcement learning

Existing static/analytic performance models are insufficient for identifying the best loop transformations, even for a simple kernel like GeMM.

Following the approach used in TVM, the standard method for optimizing the performance of tiled tensor operations involves using reinforcement learning techniques through iterative compilation in the transformation space. We adopt a similar strategy by utilizing several components: tools for generating tiling solutions and AI compiler backends with execution feedback loops; unsupervised surrogate models trained during compilation; performance metrics, as mentioned earlier, as additional inputs to the surrogate model; Bayesian Optimization frameworks managing the reinforcement loop and accumulating statistical knowledge over executions; and expert knowledge through a carefully designed schedule language that allows the expression of high-level constraints.

Our goals are to: (a) Improve the convergence time for finding optimal optimization choices compared to the current state of the art. (b) Discover better solutions for terminating the iterative search and achieving global optimization within a given time budget. (c) Provide an easy interface for incorporating expert features, such as estimated performance metrics.

[1] Register Tiling for Unstructured Sparsity in Neural Network Inference. PLDI 2020
[2] Anatomy of High-Performance Matrix Multiplication. TOMS 2008

To achieve these objectives, we have explored various dimensions in the choices available for this type of framework in our specific context. These dimensions include sampling strategy, acquisition function strategy, scalability of Gaussian Processes, categorical versus continuous kernels, a priori assumptions on performance metrics, management of reduced dimensions when using metrics, and variational or closed-form processes. For uniform sampling, we compared different approaches, including SMT+Gibbs. Some experimental results were presented during the keynote presentation at C4ML [15].

7.1.5 Performance Debugging

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, 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. Additionally, GUS features a new technique we developed this year called causality analysis, which propagates constraints to determine how each instruction contributes to the overall execution time. Both sensitivity and causality analyses are easily implemented in our simulator due to its unique design, which makes cycle prediction differentiable.

GUS should be evaluated on two key aspects. The first one is the simulator itself, including its speed, precision (cycle accuracy), and portability (ability to model various microarchitectures with minimal effort). For this evaluation, we used a set of high-performance computing kernels from the Polybench benchmark suite, applying a wide range of transformations and measuring precision on a few Intel CPU and Arm microarchitectures. We compared GUS against Gem5, demonstrating better precision (largely due to superior portability) and a speed improvement of two orders of magnitude. The second aspect is the accuracy of the bottleneck feedback it provides. This is more challenging to evaluate systematically, so we illustrated its effectiveness against the widely used top-down analysis (TMA) implemented in Intel VTune and Linux Perf using a few examples. These examples showed the superiority of our simulator-based approach over the use of extended hardware counters available in recent Intel architectures. We also used one of the benchmarks (correlation) as an illustrative example to show how our tool's bottleneck analysis can optimize code. Further comparisons with existing tools are needed to publish our findings in a top-ranked conference. A description of our design, some comparison results and discussions are provided in a research report [12].

7.2 Runtime Monitoring, Verification, and Enforcement

Participants: Yliès Falcone, Irman Faqrizal, Chukri Soueidi, Ahang Zuo, Gwen Salaün, Antoine Rollet, Srinivas Pinisetty.

This section overviews our ongoing efforts on the topics of runtime monitoring, verification, and testing. More specifically, our work can be categorized into the following topics:

- Runtime verification and enforcement of Business processes [5];
- Runtime enforcement of IEC61499 applications [1, 7];
- Runtime verification of decentralised systems.

7.2.1 Runtime verification and enforcement of business processes

A business process is a collection of structured tasks corresponding to a service or a product. Business processes do not execute once and for all, but are executed multiple times resulting in multiple instances. In this context, it is particularly difficult to ensure correctness and efficiency of the multiple executions of a process. In this work, we propose to rely on Probabilistic Model Checking (PMC) to automatically verify that multiple executions of a process respect some specific probabilistic property. This approach applies at runtime, thus the evaluation of the property is periodically verified and the corresponding results updated. However, we go beyond runtime PMC for BPMN, since we propose runtime enforcement techniques to keep executing the process while avoiding the violation of the property. To do so, our approach combines monitoring techniques, computation of probabilistic models, PMC, and runtime enforcement techniques. The approach has been implemented as a toolchain and has been validated on several realistic BPMN processes.

7.2.2 Runtime enforcement of IEC61499 applications

This work targets IEC 61499 which is a standard for developing industrial automation systems. It is known for its reusability, reconfigurability, interoperability, and portability. However, during their life cycle, industrial systems need to evolve according to requirements, and modifying the applications to satisfy these requirements can be complex and error-prone. This paper proposes techniques to guide the evolution of IEC 61499 applications. Given an initial application and the evolution requirements, we generate guidelines for modifying the application to satisfy the requirements. The application is first translated into a behavioural model describing all possible sequences of events the application can trigger. We then apply algorithms to extract relevant submodels of the application and modify them according to the requirements. Finally, the submodels are analysed to generate guidelines for modifying the application. These guidelines can bridge the gap between the requirements and the target application. Instead of only considering the requirements when exploring possible modifications, the developers can use the guidelines to make necessary changes to the application. A mixing tank system is used as a running example to illustrate the approach. In addition, a prototype to automate the evolution techniques is developed.

In our work, we also envision that industrial control systems can reliably adapt to requirements. The IEC 61499 standard allows downtimeless system evolution such that an application can be modified at runtime to satisfy the requirements. However, an IEC 61499 application consisting of multiple Function Blocks (FBs) can be modified in many different ways, such as inserting or deleting FBs, creating new FBs with their respective internal behaviours and adjusting the connections between FBs. These changes require considerable effort and cost, and there is no guarantee to satisfy the requirements. This article applies runtime enforcement techniques for supporting adaptive IEC 61499 applications. This set of techniques can modify the runtime behaviour of a system according to specific requirements. Our approach begins with specifying the requirements as a state machine-based notation called contract automaton. This automaton is then used to synthesize an enforcer as an FB. Finally, the new FB is integrated into the application to execute according to the requirements. A tool support is developed to automate the approach. Experiments were performed to evaluate the performance of enforcers by measuring the execution time of several applications before and after the integration of enforcers.

7.2.3 Runtime verification of decentralised systems

Prominent challenges in runtime verification of a distributed system are the correct placement, configuration, and coordination of the monitoring nodes. This work considers state-of-the-art decentralized monitoring practices and proposes a framework to recommend efficient configurations of the monitoring system depending on the target specification. Our approach aims to optimize communication over several features (e.g., minimizing the number of messages exchanged, the number of computations happening overall, etc.) in contexts where finding an efficient communication strategy requires slow simulations. We optimize by training multiple machine learning models from simulations combining traces, formulae, and systems of different sizes. The experimental results show that the developed model can reliably suggest the best configuration strategy in a few nanoseconds, contrary to the minutes or possibly hours required by direct simulations that would be impractical at runtime.

7.3 Teaching of Algorithms, Programming and Debugging

Participants: Théo Barollet, Florent Bouchez Tichadou, Christophe Guillon, Fabrice Rastello, Manuel Selva, Valentin Trophime-Gilotte.

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 three 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 6.1.6.
- Usage of active learning techniques in the context of large programming classes

7.3.1 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.

This year the EasyTracker library has been presented at ACM/IEEE CGO 2024 [3].

The stack and heap visualization tool based on EasyTracker has been enriched with several features leading to its usage in three different courses:

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

7.3.2 Agdbentures: A game to learn 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 6.1.6), 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 allows 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 allows us to 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 allows the player to get 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 allows 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.

The first experiments we conducted are very encouraging 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. Again, this year, we proposed intership positions to students to help develop Agdbentures. The core of the project whas redesigned to be more generic, in particular for levels based on the “game engine,” and levels where adapted. This allowed the fixing of old underlying problems as well as finishing levels that where only ideas up to now. The goal is now to finish a few levels to cover a reasonnable spectrum of debug situations.

7.3.3 Active learning method in the context of large programming classes

Manuel Selva has been using for four years an active learning method in the context of large programming classes (called “Cours Magistraux” in France). This method, called scientific debate and initially created in the context of teaching mathematics, focuses on teaching threshold concepts. The scientific debate method involves students by having them defend their position with arguments and scales up with the number of students by leveraging collective intelligence.

From the experience we gathered during the last four years, we wrote a report that presents in detail how we apply scientific debate in a programming class. We also discuss student exchanges during debates and gather feedback after the last debate. Students report they stay more focused and motivated during class with scientific debate compared to traditional transmissive lectures. They also indicate that they understood the goal of this new pedagogical contract.

This work has been presented at ACM SIGCSE TS 2024 [8].

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

With the PhD thesis (CIFRE) of Syvlain Noiri 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 initiatives

9.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

RV4IoT

Title: Runtime Verification for the Internet of Things

Duration: 2020 -> 2024

Coordinator: Sylvain Hallé (shalle@acm.org)

Partners:

- Université du Québec à Chicoutimi Chicoutimi (Canada)

Inria contact: Ylies Falcone

Summary: The goal of the associate team is to develop theories, formal techniques and tools based on runtime verification for the detection of security issues on connected objects, and the mitigation of potential attacks through runtime enforcement mechanisms.

9.1.2 Visits to international teams

Research stays abroad

Hugo Pompougnac

Visited institution: University of Cambridge

Country: UK

Duration: Oct 2024 - Feb 2025

Context of the visit: Collaboration with Tobbias Grosser on MLIR-based interactive compiler optimization

Mobility program/type of mobility: Research stay

Summary: The project is part of the efforts to embed code generation passes (back-end) into MLIR compilers and thus support assembly languages. Specifically, it involves working with the host team on instruction scheduling algorithms and the control of these algorithms by an expert programmer, from an HMI (Human-Machine Interface) perspective. This involves designing meta-programming primitives (in the form of transform dialect instructions) to constrain the scheduling of a basic block. An auto-tuning loop is then used to identify the best candidates within the search space defined by these constraints. This project leverages the host team's expertise in MLIR compilation and the CORSE team's expertise in auto-tuning methods and low-level compilation passes. It also contributes to the host team's work on embedding more meta-programming and assembly languages into MLIR.

9.2 National initiatives

ANR SEVERITAS

Title: Secure and Verifiable Test and Assessment System (SEVERITAS)

Duration: May 2021 – April 2025

Coordinator: Ylies Falcone

Partners:

- Laboratoire d'Informatique de Grenoble (LIG)
- Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)
- University of Luxembourg / Interdisciplianary Center for Security, Reliability and Trust (SnT/UL)
- Laboratoire lorrain de recherche en informatique et ses applications (LORIA)

CORSE contact: Ylies Falcone

Summary: SEVERITAS advances information socio-technical security for Electronic Test and Assessment Systems (e-TAS). These systems measure skills and performances in education and training. They improve management, reduce time-to-assessment, reach larger audiences, but they do not always provide security by design. This project recognizes that the security aspects for e-TAS are still mostly unexplored. We fill these gaps by studying current and other to-be-defined security properties. We develop automated tools to advance the formal verification of security and show how to rigorously validate e-TAS security. We also develop new secure, transparent, verifiable and lawful e-TAS procedures and protocols. We also deploy novel run-time monitoring strategies to reduce frauds and study the user experience about processes to foster e-TAS usable security. And thanks to connections with players in the business of e-TAS, such as OASYS, this project will contribute to the development of secure e-TAS.

BPI OTPaaS

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

Duration: October 2021 – September 2024

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 independante pour le deep learning embarqué

Duration: April 1st 2023 – 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: HOLIistic 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

10.1.2 Scientific events: selection

Member of the conference program committees

- Fabrice Rastello, ENDDay24, https://endday24.inviteo.fr

10.1.3 Journal

Reviewer - reviewing activities

- Guillaume Iooss, Cluster Computing
- Manuel Selva, ACM Technical Symposium on Computer Science Education (SIGCSE TS 2024)
- Guillaume Huard, Europar 2022, TPDS 2024

10.1.4 Invited talks

- Fabrice Rastello, keynote "Debunking ML for Compilers" at the International workshop on Compilers for Machine Learning (C4ML), March 3 2024, Edinburgh, UK
- Fabrice Rastello, panel "Compilers and Machine Learning" at the ACM SIGPLAN 33rd International Conference on Compiler Construction (CC), March 3 2024, Edinburgh, UK
- Fabrice Rastello, roundtable "IA embarquée" at the Kick-off PEPR-IA, March 25 2024, Grenoble, France
- Fabrice Rastello, seminar "Compilation pour le ML ou ML pour la compilation?", Mai 6 2024, Rennes, France
- Florent Bouchez Tichadou, “Chère Alice... : Apprentissage par problèmes en programmation et algorithmique”, journée d’échanges pédagogiques de la faculté des sciences de l’UGA, juillet 2024.

10.1.5 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 2024 and 2025
- Fabrice Rastello: coordinator of PC4 for the program "Composants IA"

10.1.6 Research administration

- Guillaume Iooss: RADAR local correspondant

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- License 3: Manuel Selva, Imperative programming using Python, 80 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva is responsible for the above course.
- License 3: Manuel Selva, Assembly programming, 15 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva, Processor design, 15 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva, C programming, 70 hours, Grenoble Institute of Technology (Ensimag)
- License 2: Guillaume Iooss, Algorithms languages and imperative programming (TD/TP), 37.5 hours, DLST, UGA UFR IM2AG
- Licence 3 : Guillaume Huard, C/Unix programming (C/TD/TP), 50h, UGA UFR IM2AG
- Licence 3 : Guillaume Huard, Object oriented and event based programming (C/TD/TP), 50h, UGA UFR IM2AG
- Master 1 : Guillaume Huard, Object oriented design (C/TD/TP), 37.5h, UGA UFR IM2AG
- Responsible for the Licence 3 year in computer science : Guillaume Huard, Teachers and students management, new students admission, 67h, UGA UFR IM2AG

- Responsible for the Licence diploma in computer science : Guillaume Huard, Coordination of three majors (CS, mathematics and CS, economics and CS), 25h, UGA UFR IM2AG
- Licence 1: Florent Bouchez Tichadou, Operating System and Programming, 40 hours, UGA.
- Licence 2: Florent Bouchez Tichadou, Algorithms languages and programming, 118 hours, UGA.
- Licence 3: Florent Bouchez Tichadou, Algorithms, 17 hours, UGA.
- Licence 3: Florent Bouchez Tichadou, Programming project, 17 hours, UGA.
- License: Yliès Falcone, Languages and Automata, Univ. Grenoble Alpes, 45 hours

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: Théophile Bastian, *Performance study: identifying bottlenecks by means of sensitivity analysis*, advised by Fabrice Rastello, defended in December 2024

10.2.3 Juries

- PhD, Théophile Bastian, Grenoble, Jury/advisor, Dec 2024, *Performance study: identifying bottlenecks by means of sensitivity analysis*
- PhD, Fabian Ritter, Saarland Germany, Reviewer, Oct 2024, *Inferring and Analyzing Microarchitectural Performance Models*
- PhD, Louis Narmour, Fort Collins USA / Rennes, Reviewer, Dec 2024, *Optimizations of Polyhedral Reductions and their use in Algorithm-Based Fault Tolerance*

10.3 Popularization

- Fabrice Rastello: scientific council of CEA-EDF-Inria summer schools
- Guillaume Huard: participant dans l'AMI CMA "La tête dans les nuages" (cloud computing)

11 Scientific production

11.1 Publications of the year

International journals

[1] I. Faqrizal, G. Salaün and Y. Falcone. 'Adaptive Industrial Control Systems via IEC 61499 and Runtime Enforcement'. In: *ACM Transactions on Autonomous and Adaptive Systems* (2024), pp. 1–31. DOI: 10.1145/3691345. URL: https://inria.hal.science/hal-04680168. In press (cit. on p. 11).

[2] S. Shankar, A. Pradhan, S. Pinisetty, A. Rollet and Y. Falcone. 'Bounded-memory runtime enforcement with probabilistic and performance analysis'. In: *Formal Methods in System Design* (14th Feb. 2024). DOI: 10.1007/s10703-024-00446-1. URL: https://hal.science/hal-04555219.

International peer-reviewed conferences

[3] T. Barollet, C. Guillon, M. Selva, F. Broquedis, F. Bouchez-Tichadou and F. Rastello. 'EasyTracker: A Python Library for Controlling and Inspecting Program Execution'. In: *International Symposium on Code Generation and Optimization (CGO)*. CGO 2024 - International Symposium on Code Generation and Optimization. Edinburgh, United Kingdom, 2024, pp. 1–14. URL: https://inria.hal.science/hal-04368835 (cit. on p. 13).

[4] L. Eyraud-Dubois, G. Iooss, J. Langou and F. Rastello. 'Tightening I/O Lower Bounds through the Hourglass Dependency Pattern'. In: SPAA 2024 - 36th ACM Symposium on Parallelism in Algorithms and Architectures. Nantes, France, Apr. 2024, pp. 1–34. URL: https://inria.hal.science/hal-04555744 (cit. on p. 9).

[5] Y. Falcone, G. Salaün and A. Zuo. 'Dynamic Resource Allocation for Executable BPMN Processes Leveraging Predictive Analytics'. In: QRS 2024 - 24th International Conference on Software Quality, Reliability, and Security. Cambridge, United Kingdom, 2024, pp. 1–12. URL: https://inria.hal.science/hal-04617808 (cit. on p. 11).

[6] Y. Falcone, G. Salaün and A. Zuo. 'Probabilistic Runtime Enforcement of Executable BPMN Processes'. In: FASE 2024 - 27th International Conference on Fundamental Approaches to Software Engineering. Luxembourg City, Luxembourg, 8th Apr. 2024, pp. 1–21. DOI: 10.1007/978-3-031-57259-3_3. URL: https://inria.hal.science/hal-04533195.

[7] I. Faqrizal, G. Salaün and Y. Falcone. 'Guided Evolution of IEC 61499 Applications'. In: ETFA 2024 - 29th IEEE International Conference on Emerging Technologies and Factory Automation. Padova, Italy: IEEE, 2024, pp. 1–8. URL: https://inria.hal.science/hal-04680109 (cit. on p. 11).

[8] M. Selva and F. Broquedis. 'Mining Jewels Together: Debating about Programming Threshold Concepts in Large Classes'. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024)*. SIGCSE 2024 - 55th ACM Technical Symposium on Computer Science Education. Portland (OR), United States, 2024, pp. 1–7. DOI: 10.1145/3626252.3630893. URL: https://inria.hal.science/hal-04383009 (cit. on p. 14).

[9] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye and F. Desprez. 'Optimizing Cloud Application Scheduling: A Dual-Stage Heuristic Approach'. In: ICCCBDA 2024 - 9th International Conference on Cloud Computing and Big Data Analytics. Chengdu, China: IEEE, 28th June 2024, pp. 134–140. DOI: 10.1109/ICCCBDA61447.2024.10569682. URL: https://hal.science/hal-04886644.

Doctoral dissertations and habilitation theses

[10] C. Soueidi. 'Engineering Instrumentation for Runtime Verification and Monitoring'. Université Grenoble Alpes [2020-....], 13th May 2024. URL: https://theses.hal.science/tel-04771309.

Reports & preprints

[11] T. Bastian, H. Pompougnac, A. Dutilleul and F. Rastello. *CesASMe and Staticdeps: static detection of memory-carried dependencies for code analyzers*. INRIA, 2024, pp. 1–12. DOI: 10.48550/arXiv.2402.14567. URL: https://inria.hal.science/hal-04477227.

[12] A. Dutilleul, H. Pompougnac, N. Derumigny, G. Rodríguez, V. Trophime, C. Guillon and F. Rastello. *Performance debugging through microarchitectural sensitivity and causality analysis*. INRIA, 3rd Dec. 2024, pp. 1–13. DOI: 10.48550/arXiv.2412.13207. URL: https://inria.hal.science/hal-04851704 (cit. on p. 11).

[13] G. Iooss, C. Guillon, F. Rastello, A. Cohen and S. Sadayappan. *SARCASM: Set-Associative Rotating Cache Analytical/Simulating Model*. Dec. 2024. URL: https://inria.hal.science/hal-04814088 (cit. on p. 9).

[14] H. Pompougnac, A. Dutilleul, C. Guillon, N. Derumigny and F. Rastello. *Performance bottlenecks detection through microarchitectural sensitivity*. Institut National de Recherche en Informatique et en Automatique (INRIA), 24th Feb. 2024, pp. 1–15. URL: https://inria.hal.science/hal-04796942.

Other scientific publications

[15] F. Rastello. *Assessment of the Effectiveness of ML-based Performance Models for Compiler Optimization*. 3rd Mar. 2024. URL: https://inria.hal.science/hal-04814005 (cit. on p. 11).