RESEARCH CENTRE

Inria Center
at Université Grenoble Alpes

IN PARTNERSHIP WITH:

Université de Grenoble Alpes

2022

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. – Compilation

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

A2.2.9. – Security by compilation

A2.3.2. – Cyber-physical systems

A4.4. – Security of equipment and software

A7.1. – Algorithms

Other research topics and application domains

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, HDR]
- Guillaume Iooss [INRIA, Researcher]

Faculty Members

- Florent Bouchez-Tichadou [UGA, Associate Professor]
- François Broquedis [GRENOBLE INP, Associate Professor]
- Ylies Falcone [UGA, Associate Professor]
- Manuel Selva [GRENOBLE INP, Associate Professor]

Post-Doctoral Fellows

- Hugo Brunie [INRIA, until Sep 2022]
- Victor Roussanaly [INRIA, until Jun 2022]
- Mariana Vargas [INRIA, until Sep 2022]

PhD Students

- Theo Barollet [INRIA]
- Theophile Bastian [UGA]
- Nicolas Derumigny [UGA, International joint supervision with the USA]
- Florian Gallay [UGA, from Oct 2022]
- Nihel Kaboubi [ORANGE LABS]
- Marius Monnier [UGA, until Feb 2022, abdication of responsibility]
- Auguste Olivry [UGA, until Aug 2022]
- Chukri Soueidi [INRIA]
- Nicolas Tollenaere [UGA, ATER, until Mar 2022]

Technical Staff

- Christophe Guillon [INRIA]
- Valentin Trophime-Gilotte [INRIA, Engineer, from Nov 2022]

Administrative Assistant

- Maria Immaculada Presseguer [INRIA]

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, *computational 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. Thus *diversity, heterogeneity* (even on a single chip) and thus also *hardware virtualization* is 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 to base our researches on diverse advanced mathematical tools. Compiler optimization requires the usage of the 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 specificity 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 for 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 of programming
- Tools and education of debugging
- Problem based learning. Generation. Recommandation

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 over numerical kernels (linear algebra, FFT, convolution...) that can be adapted on a large 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 STMICROELECTRONICS, ATOS/BULL, ORANGE, KALRAY.

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 have 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 takeback 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) and better usage (consume less) of our compute resources. For this purpose, we organize panels with the objective of sensitize our community to this important problem. 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 instead.

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 (need for 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 it accessible to none-experts.

2. *Increase the lifetime of hardware* (that is, Reuse, Repair, Re...): The need for supporting the development of simple, open-source, 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 repare or repurpose and device.

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

6 Highlights of the year

After humongous efforts, the SSA book is finally available. Twelve years were necessary to give birth to this book, composed of 24 chapters and written by 31 authors.

It provides readers with a single-source reference to static-single assignment (SSA)-based compiler design. It is the first (and up to now only) book that covers in a deep and comprhensive way how an optimizing compiler can be designed using the SSA form. After introducing vanilla SSA and its main properties, the authors describe several compiler analyses and optimizations under this form. They illustrate how compiler design can be made simpler and more efficient, thanks to the SSA form. This book also serves as a valuable text/reference for lecturers, making the teaching of compilers simpler and more effective. Coverage also includes advanced topics, such as code generation, aliasing, predication and more, making this book a valuable reference for advanced students and practicing engineers.

- Provides the first, single-source reference to the widely adopted, static-single assignment (SSA) form of compiler design;
- Offers readers state-of-the-art, advanced compiler optimization techniques
- Includes contributions by subject experts from globally recognized compiler research centers and engineering practitioners at companies such as Google, Facebook, IBM, and Amazon;
- Employs a textbook style of presentation throughout, with coherent and uniform structure, sequence, terminology, and notations;
- Offers valuable content both for lecturers (such as vanilla SSA, construction, destruction, propagation, liveness) and advanced compiler developers (including if-conversion, code-selection, hardware compilation, scalar evolution, register allocation, Gated-SSA, Psi-SSA, Hashed-SSA, Array-SSA, SSI).

7 New software and platforms

7.1 New software

7.1.1 Pipedream

Name: Pipedream

Keywords: Performance analysis, CPU, Reverse engineering

Scientific Description: Pipedream reverse engineers the following performance characteristics: (1) Instruction latency – The number of cycles an instruction requires to execute. (2) Peak micro-op retirement rate – How many fused micro-ops the CPU can retire per cycle. (3) Micro-fusion – The number of fused micro-ops an instruction decomposes into. (4) Micro-op decomposition and micro-op port usage – The list of unfused micro-ops every instruction decomposes into and the list of execution ports every one of these micro-ops can execute on.

The first step of the reverse engineering process consists of generating a number of microbenchmarks. Pipedream then runs these benchmark, measuring their performance using hardware counters. The latency, throughput, and micro-fusion of different instructions can then be read directly from these measurements.

The process of finding port mappings, i.e. micro-op decompositions and micro-op port usage, however, is more involved. For this purpose, we have defined a variation of the maximum flow problem which we call the "instruction flow problem". We have developed a linear program (LP) formulation of the instruction flow problem which can be used to calculate the peak IPC and micro-operations per cycle (MPC) a benchmark kernel can theoretically achieve with a given port mapping. The actual port mapping of the underlying hardware is then determined by finding the mapping for which the throughput predicted by instruction flow best matches the actual measured IPC and MPC.

Functional Description: Pipedream is a tool for measuring specific performance characteristics of CPUs It is used to build the performance model of another tool called Gus (https://gitlab.inria.fr/nderumig/gus). Pipedream finds measured performance characteristics such as the throughput and latency of instructions by running a large set of automatically generated microbenchmarks. The tool can also find port mappings, a model of part of the CPU instruction scheduler, by analysing performance measurements of specially crafted microkernels using a LP solver. We have used it to produce a port mapping for the Intel Skylake CPU architecture. Pipedream is able to find the port mappings for some instructions for which existing approaches fall back to manual analysis.

URL: https://gitlab.inria.fr/fgruber/pipedream

Contact: Nicolas Derumigny

7.1.2 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: Auguste Olivry

7.1.3 IOOpt

Keywords: I/O, Polyhedral compilation

Functional Description: IOOpt takes as input an abstract representation of a tileable program. The tool generates a tractable set of relevant permutations of the tiling loops, and a symbolic I/O cost expression for each of them. It then uses a non linear problem optimizer to find the best permutations and corresponding tile sizes for a given value of machine parameters (cache sizes and bandwidths at each level). IOOop can also be used to find an upper bound on the I/O cost of a program, for a given tiling scheme.

Publication: hal-03200539

Contact: Auguste Olivry

7.1.4 PALMED

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

Functional Description: PALMED computes a bipartite graph assembly instructions <-> 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 use 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

Contact: Nicolas Derumigny

7.1.5 BISM

Name: BISM: Bytecode-level Instrumentation for Software Monitoring

Keywords: Java, Bytecode, Instrumentation, Control Flow

Functional Description: BISM (Bytecode-level Instrumentation for Software Monitoring) is a lightweight Java bytecode instrumentation tool which features an expressive high-level control-flow-aware instrumentation language. The language follows the aspect-oriented programming paradigm by adopting the joinpoint model, advice inlining, and separate instrumentation mechanisms. BISM provides joinpoints ranging from bytecode instruction to method execution, access to comprehensive context information, and instrumentation methods. BISM runs in two modes: build-time and load-time.

URL: https://gitlab.inria.fr/bism/bism-public

Publication: hal-03081265

Contact: Ylies Falcone

Participants: Chukri Soueidi, Ylies Falcone, Ali Kassem

7.1.6 TTiLE

Keywords: Deep learning, Optimization, HPC

Functional Description: TTiLE is a code generation tool for tensor operators present in CNNs such as tensor contraction and convolution. It takes as input: 1. a kernel specification, that is, a fonctionnal description of the operator (iteration space, tensors, single assignment description of the computation), 2. an optimization scheme that describes the layered structure of the generated code. The scheme "language" allows to express loop permutation, loop tiling, vectorization, unrolling, packing (which consists in using temporary buffers making cache access better behaved) and branching.

Indeed, as opposed to existing schemes that rely on partial tiles, TTile can create non-perfectly nested loops to combine several "micro-kernels" (micro-kernel stands for the innermost part of the loop nest that is fully unrolled, register promoted and vectorized here). Using a specialized search strategy that combines operational research on analytical performance model and native execution time measurement, TTile outperforms current highly optimized libraries such as Intel oneDNN and Intel MKL.

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

Publication: hal-03149553

Contact: Guillaume Iooss

7.1.7 EasyTracker

Keywords: Monitoring, Debug, Visualization, Teaching of programming

Scientific Description: Learning to program involves building a mental representation of how a machine executes instructions and stores information in memory. To help students, teachers often use visual representations to illustrate executions of programs or particular concepts in their lectures. EasyTracker is a library that assists teachers of programming courses in building tools that generate representations tuned to their needs from actual programs. At its core, EasyTracker provides ways of controlling the execution and inspecting the state of programs. The control and inspection are driven and customized through a Python interface. The controlled program itself can be written either in Python or in any GDB supported language like C.

Functional Description: EasyTracker is intended for computer science teaching. It encapsulates the execution control and monitoring of another program written in Python or any compiled language supported by GDB. It can pause execution at interest points described in a high level language (variable modified or return of recursive functions for example).

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

Contact: Theo Barollet

Participants: Theo Barollet, Manuel Selva, François Broquedis, Florent Bouchez, Fabrice Rastello

8 New results

8.1 Performance Debugging and Compiler Optimization

Participants: Fabrice Rastello, Guillaume Iooss, Christophe Guillon, Nicolas Derumigny, Théophile Bastian, Nicolas Tollenaere *(Inria CORSE)*, Fabian Gruber *(ARM)*, Albert Cohen *(Google, France)*, P. Sadayappan *(OSU, USA)*, Louis-Noël Pouchet *(CSU, USA)*, Atanas Rountev *(OSU, USA)*.

Our current efforts with regard to code optimization follows two directions.

1. The first consists in improving compiler optimization techniques by considering pattern specific applications such as those that fit into the polyhedral framework or more restrictively those related to machine learning. In this context we developed a new compiler scheme for optimizing computational kernels of DNNs [1] (Section 8.1.1).
2. The second consists in generating dynamic analysis based performance debugging tools. For that purpose we: 1. developed a tool for automatically characterizing CPU resources [4] and, 2. extended a binary translator (QEMU) to perform an abstract simulation based sensitivity analysis (Section 8.1.2).

8.1.1 Hybrid Performance Modeling and Schedule Optimization for Tensor Computations

Tensor computation such as Sparse Matrix Multi-vector multiplication, Sampled Dense Dense Matrix Multiplication, Dense Matrix Multiplication, Tensor Contraction, Convolution are important kernels used in many domains like Fluid Dynamics, Data Analytics, Economic Modelling, and Machine Learning. Developing highly optimized code for such kernels requires the combination of highly tuned register/instruction level micro-kernels and appropriate multi-level tiling. In this context we developed an hybrid (analytical/statistical) performance-based optimization scheme along with a code generator for DNNs.

Addressing the problem of automatic generation of optimized operators raises two challenges: The first is associated to the design of a domain specific code generation framework able to output high-quality binary code. The second is to carefully bound the search space and choose an optimizing objective function that neither leads to yet another combinatorial optimizing problem, nor leads to a too approximate performance objective. This work tackles those two challenges by: 1. revisiting the usual belief that packing should enable stride-1 accesses at every level allowing to make packing optional; 2. highlighting the importance of considering the packing decision and shape as being part of the optimization problem; 3. revisiting the usual belief that register spilling should be avoided if possible allowing to consider other (more packing friendly) micro-kernels as good candidates; 4. revisiting the misleading intuition that convolution dimensions should be brought at the innermost level allowing more freedom for memory reuse at outer-dimensions; 5. showing that the optimization problem can be decoupled into: finding a small set of good micro-kernels candidates using an exhaustive search; finding a good schedule (loop tiling/permutation) and associated packing using operational research; finding the best tiles sizes using auto-tuning; 6. designing a single-pass micro-kernel generation algorithm, to emit code for any choice of register blocking dimensions, unrolling factor, and packing decisions; 7. designing a lowering scheme for abstract iterators, compatible with diverse packing and tiling strategies thrifty with integer arithmetic and loop control usage; 8. designing a packing algorithm compatible with various choices of transposition and subviews; 9. implementing a code generator based on these algorithms, driven by a simple and modular configuration language.

Part of this work lead to a paper that has been accepted for publication at ACM TACO [1] and that will be presented at HiPEAC 2023.

8.1.2 Automatic Resource Characterization and Performance Feedback

Performance modeling is a critical component for program optimizations, assisting compilers as well as developers in predicting the performance of code variations ahead of time. Performance models can be obtained through different approaches that span from precise and complex simulation of a hardware description (Zesto, GEM5, PTLSim) to application level analytical formulations. An interesting approach for modeling the CPU of modern pipelined, super-scalar, out-of-order processors trades simulation time with accuracy by separately characterizing both latency and throughput of instructions. This approach is suitable both for optimizing compilers, but also for hand-tuning critical kernels written in assembler (see Section 8.1.1). It is used by performance-analysis tools such as CQA, Intel IACA, OSACA, MIAMI or llvm-mca. Cycle-approximate simulators such as ZSim or MCsimA can also take advantage of such an instruction characterization. In this context, we developed two tools: PALMED and GUS (see new software section).

This work has been done in the context of the European project CPS4EU (see Section 9.3.1).

8.2 Runtime Monitoring, Verification, and Enforcement

Participants: Yliès Falcone, Florian Gallay, Irman Faqrizal, Chukri Soueidi, Hamzah Al-Qadasi, Ahang Zuo, Victor Roussanaly, Gwen Salaün, Saddek Bensalem, Marius Bozga, Hosein Nazarpour, Camilo Rocha, Francisco Durán, Antoine Rollet, Saumya Shankar, Srinivas Pinisetty, Denis Furian, Shaun Azzopardi, Gerardo Schneider, Angel Contreras, Antoine El-Hokayem.

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

- runtime verification, where we define applied work on the decentralized verification of smart homes and for the relatively new programming language Kotlin as well as theoretical work on the decentralized verification of timed properties;
- runtime enforcement, where we apply enforcement to IEC 61499 applications, we define enforcement mechanism with bounded memory and a benchmark tool for decentralized runtime enforcement.
- instrumentation, where we define an instrumentation approaches to capture conservative models of Java programs for runtime verification;
- analysis of business process, where we apply modeling, probabilistic model checking as well as combined static and dynamic approaches for such processes.

8.2.1 Runtime Verification

Bringing runtime verification home: a case study on the hierarchical monitoring of smart homes using decentralized specifications.
We use runtime verification (RV) to check various specifications in a smart apartment. The specifications can be broken down into three types: behavioral correctness of the apartment sensors, detection of specific user activities (known as activities of daily living), and composition of specifications of the previous types. The context of the smart apartment provides us with a complex system with a large number of components with two different hierarchies to group specifications and sensors: geographically within the same room, floor or globally in the apartment, and logically following the different types of specifications. We leverage a recent approach to decentralized RV of decentralized specifications, where monitors have their own specifications and communicate together to verify more general specifications. We leverage the hierarchies, modularity and re-use afforded by decentralized specifications to: (1) scale beyond existing centralized RV techniques, and (2) greatly reduce computation and communication costs.

Runtime Verification of Kotlin Coroutines.
Kotlin was introduced to Android as the recommended language for development. One of the unique functionalities of Kotlin is that of co-routines, which are lightweight tasks that can run concurrently inside threads. Programming using co-routines is difficult, among other things, because they can move between threads and behave unexpectedly. We introduce runtime verification in Kotlin. We provide a language to write properties and produce runtime monitors tailored to verify Kotlin co-routines. We identify, formalize and runtime verify seven properties about common runtime errors that are not easily identifiable by static analysis. To demonstrate the acceptability of the technique in real applications, we apply our framework to an in-house Android app and micro-benchmarks and measure the execution time and memory overheads.

This work has been published in RV [10].

Decentralized Runtime Verification of Timed Regular Expressions.
Ensuring the correctness of distributed cyber-physical systems can be done at runtime by monitoring properties over their behavior. In a decentralized setting, such behavior consists of multiple local traces, each offering an incomplete view of the system events to the local monitors, as opposed to the standard centralized setting with a unique global trace. We introduce the first monitoring framework for timed properties described by timed regular expressions over a distributed network of monitors. First, we define functions to rewrite expressions according to partial knowledge for both the centralized and decentralized cases. Then, we define decentralized algorithms for monitors to evaluate properties using these functions, as well as proofs of soundness and eventual completeness of said algorithms. Finally, we implement and evaluate our framework on synthetic timed regular expressions, giving insights on the cost of the centralized and decentralized settings and when to best use each of them.

Residual Runtime Verification via Reachability Analysis.
In this work, we leverage static verification to reduce monitoring overhead when runtime verifying a property. We present a sound and efficient analysis to statically find safe execution paths in the control flow at the intra-procedural level of programs. Such paths are guaranteed to preserve the monitored property and thus can be ignored at runtime. Our analysis guides an instrumentation tool to select program points that should be observed at runtime. The monitor is left to perform residual runtime verification for parts of the program that the analysis could not statically prove safe. Our approach does not depend on dataflow analysis, thus separating the task of residual analysis from static analysis; allowing for seamless integration with many RV frameworks and development pipelines. We implement our approach within BISM, which is a recent tool for bytecode-level instrumentation of Java programs. Our experiments on the DaCapo benchmark show a reduction in instrumentation points by a factor of 2.5 on average (reaching 9), and accordingly, a reduction in the number of runtime events by a factor of 1.8 on average (reaching 6).

This work has been published in VSTT [6].

8.2.2 Runtime Enforcement

DECENT: A Benchmark for Decentralized Enforcement.
DECENT is a benchmark for evaluating decentralized enforcement. It implements two enforcement algorithms that differ in their strategy for correcting the execution: the first one explores all alternatives to perform a globally optimal correction, while the second follows an incremental strategy based on locally optimal choices. Decent allows comparing these algorithms with a centralized enforcement algorithm in terms of computational metrics and metrics for decentralized monitoring such as the number and size of messages or the required computation on each component. Our experiments show that (i) the number of messages sent and the internal memory usage is much smaller with decentralized algorithms (ii) the locally optimal algorithm performs closely to the globally optimal one.

This work has been published in RV [11].

Bounded-Memory Runtime Enforcement.
Runtime Enforcement (RE) is a monitoring technique to ensure that a system obeys a set of formal requirements (properties). RE employs an enforcer (a safety wrapper for the system) which modifies the (untrustworthy) output by performing actions such as delaying (by storing/buffering) and suppressing events, when needed. In this work, to handle practical applications with memory constraints, we propose a new RE paradigm where the memory of the enforcer is bounded/finite. Besides the property to be enforced, the user specifies a bound on the enforcer memory. Bounding the memory poses various challenges such as how to handle the situation when the memory is full, how to optimally discard events from the buffer to accommodate new events and let the enforcer continue operating. We define the bounded-memory RE problem and develop a framework for any regular property. The proposed framework is implemented and its performance evaluated via some examples from application scenarios indicates that the enforcer has reasonable execution time overhead.

Runtime Enforcement for IEC 61499 Applications.
Industrial automation is a complex process involving various stakeholders. The international standard IEC 61499 helps to specify distributed automation using a generic architectural model, targeting the technical development of the automation. However, analyzing the correctness of IEC 61499 models remains a challenge because of their informal semantics and distributed logic. We propose new verification techniques for IEC 61499 applications. These techniques rely on the concept of runtime enforcement, which can be applied to systems for preventing bad behaviors from happening. The main idea of our approach is to integrate an enforcer in the application for allowing it to respect specific properties when executing. The techniques begin with the definition of a property. The language of this property supports features such as discarding and replacing events. Next, this property is used to synthesize an enforcer in the form of a function block. Finally, the synthesized enforcer is integrated into the application. Our approach is illustrated on a realistic example and fully automated.

This work has been published in SEFM [8].

8.2.3 Capturing program models with BISM

In this work, we present an extension of the Java bytecode instrumentation tool BISM that captures and prepares a model that abstracts the program behavior at the intra-procedural level. We analyze program methods we are interested in monitoring and construct a control-flow graph automaton where the states represent actions of the program that produce events. Directed towards monitoring general behavioral properties at runtime, the resulting model is presented for the users to write static analyzers and combine both static and runtime verification.

8.2.4 Monitoring and Verification of BPMN Processes

WEASY: A Tool for Modeling Optimized BPMN Processes.
Business Process Model and Notation (BPMN) is a standard modeling language for workflow-based processes. Building an optimized process with this language is not easy for non-expert users due to the lack of support at design time. This work presents a lightweight modeling tool to support such users in building optimized processes. First, the user defines the tasks involved in the process and possibly gives a partial order between tasks. The tool then generates an abstract graph, which serves as a simplified version of the process being specified. Next, the user can refine this graph using the minimum and maximum execution time of the whole graph computed by the tool. Once the user is satisfied with a specific abstract graph, the tool synthesizes a BPMN process corresponding to that graph. Our tool is called WEASY and is available as an open-source web application.

Probabilistic Model Checking of BPMN Processes at Runtime.
Business Process Model and Notation (BPMN) is a standard business process modeling language that allows users to describe a set of structured tasks, which results in a service or product. Before running a BPMN process, the user often has no clear idea of the probability of executing some task or specific combination of tasks. This is, however, of prime importance for adjusting resources associated with tasks and thus optimizing costs. In this work, we define an approach to perform probabilistic model checking of BPMN models at runtime. To do so, we first transform the BPMN model into a Labeled Transition System (LTS). Then, by analyzing the execution traces obtained when running multiple instances of the process, we can compute the probability of executing each transition in the LTS model, and thus generate a Probabilistic Transition System (PTS). Finally, we perform probabilistic model checking for verifying that the PTS model satisfies a given probabilistic property. This verification loop is applied periodically to update the results according to the execution of the process instances. All these steps are implemented in a tool chain, which was applied successfully to several realistic BPMN processes.

This work has been published in iFM [9].

From Static to Dynamic Analysis and Allocation of Resources for BPMN Processes.
Business process optimization is a strategic activity in organizations because of its potential to increase profit margins and reduce operational costs. One of the main challenges in this context is concerned with the problem of optimizing the allocation and sharing of resources. In this work, processes are described using the BPMN notation extended with an explicit description of execution time and resources associated with tasks, and can be concurrently executed multiple times. First, a simulation-based approach for computing certain metrics of interest, such as average execution time or resource usage, is presented. This approach applies off-line and is static in the sense that the number of resources does not evolve over the time of the simulation. In a second step, an alternative approach is presented, which works online, thus requiring the instrumentation of an existing platform for retrieving information of interest during the processes' execution. This second approach is dynamic because the number of resource replicas is updated over the time of the execution. This work aims at stressing pros and cons of both approaches, and at showing how they complement each other.

This work has been published in WRLA [5].

Probabilistic Analysis of Industrial IoT Applications.
Business Process Model and Notation (BPMN) is a standard business process modeling language that allows users to describe a set of structured tasks, which results in a service or product. Before running

a BPMN process, the user often has no clear idea of the probability of executing some task or specific combination of tasks. This is, however, of prime importance for adjusting resources associated with tasks and thus optimizing costs. In this work, we define an approach to perform probabilistic model checking of BPMN models at runtime. To do so, we first transform the BPMN model into a Labeled Transition System (LTS). Then, by analyzing the execution traces obtained when running multiple instances of the process, we can compute the probability of executing each transition in the LTS model, and thus generate a Probabilistic Transition System (PTS). Finally, we perform probabilistic model checking for verifying that the PTS model satisfies a given probabilistic property. This verification loop is applied periodically to update the results according to the execution of the process instances. All these steps are implemented in a tool chain, which was applied successfully to several realistic BPMN processes.

This work has been published in [7].

8.3 Teaching of Algorithms, Programming, Debugging, and Automata

Participants: Théo Barollet, Florent Bouchez Tichadou, Manuel Selva, François Broquedis, Fabrice Rastello.

This domain is a new axis of the CORSE team. 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 derived into two projects: Easytracker, which is a library that helps building tools to visualize program execution and data structures; and Agdbentures, a game that helps learners to gain skills in debugging, which is based on EasyTracker.

8.3.1 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 information in memory. To help students, teachers often use visual representations to illustrate executions of programs or particular concepts in their lectures. As a famous example, references/pointers are very often represented with arrows pointing to objects or memory locations. While these visual representations are most of the time hand-drawn, they nowadays tend to be supplemented by tool-generated ones. These tools have the advantage of being usable by learners, empowering them with the ability of validating their own understanding of the concept the tool aims at representing. However, building such a tool from scratch requires a lot of effort and a high level of technical expertize, and the ones that already exist are difficult to adapt to different contexts. In this work we developped EasyTracker, a library that assists teachers of programming courses in building tools that generate representations tuned to their needs from actual programs. At its core, EasyTracker provides ways of controlling the execution and inspecting the state of programs. The control and inspection are driven and customized through a Python interface. The controlled program itself can be written either in Python or in any GDB supported language like C. This work showcases two tools built on EasyTracker which are used in a teaching context to explain the notions of stack and heap, and to visualize recursion as well as Agdbentures, presented in the next section, a game prototype to help students learn debugging.

This work has been submitted for publication at ACM ITICSE 2023.

8.3.2 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 developped Agdbentures, 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 do debugging; 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 has interactive objects, NPCs (non playable characters), level logic (activating levers, collecting items...). This allows the player to get familiar with the codebase over time so we can present more difficult bugs which could arise 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.

Although there is currently only about ten levels, the first experiments we conducted are very encouraging about the engagement of students at the L2 university level. All where eager to participate and declared they would really like to continue playing Agdbentures on their own with more levels.

This work has been done in the context of the AI4HI Inria exploratory project and has been submitted for publication at ACM ITICSE 2023.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Inria associate team not involved in an IIL or an international program

RV4IoT

Title: Runtime Verification for the Internet of Things

Duration: 2022 -> 2025

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.

9.2 International research visitors

9.2.1 Visits to international teams

Research stays abroad

Fabrice Rastello

Visited institution: Colorado State University (CSU)

Country: USA

Dates: July-August 2022

Context of the visit: Collaboration on optimization of sparse computation

Mobility program/type of mobility: research stay

Chukri Soueidi

Visited institution: Université du Québec à Chicoutimi

Country: Canada

Dates: 30 Oct to 1 December

Context of the visit: The main purpose of this visit was to collaborate on the tool support of the RV4IoT team. Mainly, Chukri's objective was to connect the BISM and BeepBeep tool to form a complete monitoring solution.

Mobility program/type of mobility: RV4IoT

9.3 European initiatives

9.3.1 H2020 projects

CPS4EU (CPS4EU project on cordis.europa.eu)

Title: Cyber Physical Systems for Europe

Duration: From July 1, 2019 to September 30, 2022

Partners:

- KALRAY SA (KALRAY), France
- INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (INRIA), France
- UNIVERSITA DEGLI STUDI DI SALERNO, Italy
- UNIVERSITE DE LORRAINE (UL), France
- SCHNEIDER ELECTRIC FRANCE SAS (SEF), France
- ASSOCIATION JESSICA FRANCE (JESSICA FRANCE), France
- M3 SYSTEMS SAS (M3S), France
- TRUMPF WERKZEUGMASCHINEN SE + COKG (TRUMPF), Germany
- GREENWAVES TECHNOLOGIES (GREENWAVES TECHNOLOGIES), France
- ACS PLUS GMBH (ACS), Germany
- COMMISSARIAT A L'ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES (CEA), France
- WIKA MOBILE CONTROL GMBH & CO KG (WIKA), Germany
- VALEO VISION SAS (Valeo Vision), France
- VALEO VISION SAS (Valeo Vision), France
- TECHNISCHE UNIVERSITAT CLAUSTHAL (TUC), Germany
- BUDAPESTI MUSZAKI ES GAZDASAGTUDOMANYI EGYETEM (BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS), Hungary
- EUROTECH SPA (EUROTECH), Italy
- VALEO COMFORT AND DRIVING ASSISTANCE (Valeo Comfort And Driving Assistance), France
- VALEO COMFORT AND DRIVING ASSISTANCE (Valeo Comfort And Driving Assistance), France
- FUNDACION CENTRO DE TECNOLOGIAS DE INTERACCION VISUAL Y COMUNICACIONES VICOMTECH (VICOM), Spain
- ARCURE SA (ARCURE), France

- UNIVERSITE GRENOBLE ALPES (UGA), France
- EMBEDDED FRANCE (EMBEDDED FRANCE), France
- VSORA (VSORA), France
- YUMAIN (GST), France
- INTERNET OF TRUST, France
- INSTITUTO TECNOLOGICO DE INFORMATICA (ITI), Spain
- LEONARDO - SOCIETA PER AZIONI (LEONARDO), Italy
- RTE RESEAU DE TRANSPORT D'ELECTRICITE, France
- SHERPA ENGINEERING SA (SHERPA), France
- UNIVERSITAET AUGSBURG (UAU), Germany
- THALES (THALES), France
- PROVE&RUN (Prove & Run), France
- EMMTRIX TECHNOLOGIES GMBH (EMMTRIX), Germany
- CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE CNRS (CNRS), France
- ACOEM FRANCE SAS (ACOEM), France
- CENTRALESUPELEC (CentraleSupélec), France
- SPINSPLIT MUSZAKI KUTATO FEJLESZTOKFT (SPINSPLIT TECHNICAL RESEARCH AND DEVELOPMENT LLC), Hungary
- ANSYS FRANCE SAS (ANSYS), France
- AIRLANE TECHNOLOGIES (AIRLANE), France
- SEQUANS COMMUNICATIONS SA (SEQ), France
- ETH LAB SRL (ETH LAB), Italy
- SYSNAV SAS (SYSNAV), France

Inria contact: Eric RUTTEN

Coordinator: Philippe Gougeon, VALEO VISION SAS

Summary: Cyber Physical Systems (CPS) represent key drivers for the innovation capacity of European industries, large and small, generating sustainable economic growth and supporting meaningful jobs for citizens. The ultimate objective of CPS4EU is to strengthen the CPS value chain by creating world class European SMEs and by providing CPS technologies that in turn will sustain the leadership of the large European groups in key economy sectors and, in this way will stimulate innovative products to support the massive digitization increasingly integrated into our everyday environment.

9.4 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 validating e- TAS security. rigorously. 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.

OTPaaS

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

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.

ES3CAP

Title: Embedded Smart Safe Secure Computing Autonomous Platform

CORSE contact: Fabrice Rastello

CORSE participants: Fabrice Rastello, Nicolas Tolenaere

Duration: July 2018 - February 2022

INRIA Partners: AOSTE, PARKAS, CHROMA

Other Partners: Renault-Nissan, EasyMile, Safran E&D, MBDA, ANSYS/ESterel Technologies, Kronno-Safe, Prove & Run, Kalray, Prophesee, CEA

Summary: The objective of ES3CAP is to develop a tool-chain that targets multi- and many-core architectures for critical systems. In particular it should address the different challenges related to making existing critical systems solutions (heterogeneous, decentralized, single-core, single-task) match the industrial constraints targeted by KALRAY's MPPA (MPPA, high-performance, real-time, safety, security). Considered applications are autonomous driving, drones, avionics, and defense. CORSE is involved in the optimization of machine learning algorithms for many-core architectures.

9.5 Regional initiatives

MOAP

Title: Modélisation, optimisation et analyse prédictive des processus métiers

CORSE contact: Yliès Falcone

CORSE participants: Yliès Falcone

Duration: October 2020 - September 2023

INRIA Partners: CONVECS

Other Partners: SOITEC

Summary: La modélisation et l'optimisation des processus dans les entreprises sont un enjeu économique majeur. En effet, minimiser des temps de traitement, ajuster l'utilisation des ressources ou éviter des situations de blocage ou d'attente permettraient de réduire les coûts de fonctionnement de l'entreprise. L'objectif du projet MOAP est de fournir des techniques afin de calculer automatiquement un ensemble de métriques permettant de quantifier l'efficacité des processus déployés dans l'entreprise, et d'ensuite utiliser ces mesures pour les améliorer. La première contribution du projet MOAP consiste à proposer un langage de modélisation suffisamment expressif pour modéliser les processus métiers dans l'industrie du futur, soit par extension de langages existants soit en proposant des langages dédiés. Nous proposerons ensuite des techniques automatisées d'analyse afin de générer plusieurs métriques qui vont permettre dans un second temps d'optimiser les processus suivant différents critères (temps d'exécution, utilisation des ressources, coûts d'infrastructure, etc.). Nous envisageons aussi de mettre en place des techniques de monitoring prédictif qui permettront à l'entreprise d'observer et raisonner sur les traces d'exécutions afin d'améliorer le fonctionnement des-dits processus sans forcément disposer de leur modèles. Ces solutions seront implementées dans un outil qui sera validé sur les processus existants au sein de l'entreprise Soitec.

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events

Member of the Conference Steering Committee

- Fabrice Rastello: Steering Committee Chair of ACM/IEEE CGO till March 2022.
- Fabrice Rastello: Member of the steering Committee of ACM/IEEE CGO.
- Yliès Falcone: Member of the Steering Committee of the Runtime Verification conference.
- Yliès Falcone: Member of the Steering Committee of Software Verification and Testing track at the Symposium on Applied Computing.

Member of the conference program committees

- Guillaume Iooss: IMPACT 2023
- Fabrice Rastello: IEEE IPDPS 2023
- Yliès Falcone: RV 2022
- Yliès Falcone: NFM 2022

Reviewer

- Guillaume Iooss: CFW
- Manuel Selva: ACM ITiCSE
- François Broquedis: ACM ITiCSE

10.1.2 Journal

Reviewer - reviewing activities

- Guillaume Iooss: ACM TOPLAS, ACM TOCS, ACM TECS.

10.1.3 Leadership within the scientific community

- Fabrice Rastello: deputy scientific director of Inria Grenoble Rhône-Alpes (DSA) since Sept 2022
- Fabrice Rastello: scientific council of Inria Grenoble Rhône-Alpes (CoS)

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- License 3: François Broquedis, Imperative programming using Python, 60 hours, Grenoble Institute of Technology (Ensimag)
- License 3: François Broquedis, Introduction to UNIX, 20 hours, Grenoble Institute of Technology (Ensimag)
- License 3: François Broquedis, C programming, 100 hours, Grenoble Institute of Technology (Ensimag)
- Master 1: François Broquedis, Object-Oriented Programming, 40 hours, Grenoble Institute of Technology (Ensimag)
- Master 1: François Broquedis, Operating Systems Development Project, 20 hours, Grenoble Institute of Technology (Phelma)
- François Broquedis is in charge of the first year study at Ensimag
- Master: Florent Bouchez Tichadou, Algorithmic Problem Solving, 41 hours, M1 MoSIG.
- Licence: Florent Bouchez Tichadou, Algorithms languages and programming, 103 hours, L2 UGA.
- Master: Florent Bouchez Tichadou, remise à niveau d'agents de la SNCF en reconversion, 60 hours, UGA.
- DIU EIL, Florent Bouchez Tichadou, formation des enseignants du secondaire suite à la réforme du Baccalauréat. Bloc Algorithmique. 30 hours, UGA.
- Master 1: Yliès Falcone, Programming Language Semantics and Compiler Design, MoSIG and Master informatique, 45 hours
- License: Yliès Falcone, Languages and Automata, Univ. Grenoble Alpes, 45 hours
- Master: Yliès Falcone, is responsible for the two above courses.
- License 3: Manuel Selva, Imperative programming using Python, 118 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva is responsible for the above course.

- License 3: Manuel Selva, Introduction to UNIX, 13 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva, Assembly programming, 15 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva, C programming, 25 hours, Grenoble Institute of Technology (Ensimag)
- Master 1: Manuel Selva, Concurrent programming, 15 hours, Grenoble Institute of Technology (Ensimag)
- License 2: Guillaume Iooss, Algorithms languages and imperative programming (TD/TP), 31.5 hours, DLST, UGA UFR IM2AG
- Licence 2: Marius Monnier, INF302: Languages and automata, 24 hours, DLST (UGA)
- Master 1: Auguste Olivry, Algorithmic Problem Solving, 41 hours, UGA UFR IM2AG / Grenoble INP Ensimag
- License 2: Auguste Olivry, Algorithms languages and imperative programming (TP), 18 hours, DLST, UGA UFR IM2AG
- Licence 2, Nicolas Tollenaere, Algorithms languages and imperative programming (TD/TP), 36 hours, DLST, UGA UFR IM2AG
- Licence 2, Nicolas Derumigny, Algorithms languages and imperative programming (TD/TP), 42 hours, DLST, UGA UFR IM2AG
- License 2: Theo Barollet, Algorithms languages and imperative programming (TD-TP), 33 hours, UGA
- Master 1: Theo Barollet, Compilation Project, 18h, UGA

10.2.2 Supervision

- PhD: Auguste Olivry, Automatic derivation of I/O complexity bounds for affine programs, advised by Fabrice Rastello, defended in June 2022.
- PhD: Nicolas Tollenaere, Decoupling the optimization space of tensor computation for a better understanding of performance on Intel CPU, advised by Fabrice Rastello and Guillaume Iooss, defended in Dec 2022.
- PhD in progress: Florian Gallay, Decentralized Runtime Enforcement, October 2022, advised by Yliès Falcone.
- PhD in progress: Théophile Bastian, Performance study: identifying bottlenecks by means of sensitivity analysis, September 2021, advised by Fabrice Rastello.
- PhD in progress: Nicolas Derumigny, Automatic generation of performance models for heterogeneous architectures, September 2019, advised by Fabrice Rastello.
- PhD in progress: Théo Barollet, Problem-based learning: automatic generation and recommendation of programming exercises, September 2019, advised by Florent Bouchez Tichadou and Fabrice Rastello.
- PhD in progress: Chukri Soueidi, Instrumentation, Runtime Verification and Enforcement for Multithreaded Programs, October 2020, advised by Yliès Falcone.
- Licence 2 internship: Clément Correnoz, 2 months, "Integration of EasyTracker in client/server architecture to have visualisation tools running in a browser", advised by Manuel Selva.
- Licence 3 internship: Benjamin Priour, 2 months, "Add support for multi threaded programs in EasyTracker", advised by Manuel Selva.

10.2.3 Juries

Fabrice Rastello

- Nicolas Tollenaere–Grenoble, Jury, Dec 2022, "Decoupling the optimization space of tensor computation for a better understanding of performance on Intel CPU".
- Auguste Olivry–Grenoble, Jury, June 2022, "Automatic derivation of I/O complexity bounds for affine programs".
- Marcos Horro Varela–Coruna, Reviewer, April 2022, "Manycore Architectures and SIMD Optimizations for High Performance Computing".

Guillaume Iooss

- Maksim Berezov–Paris,Jury, Dec 2022, "L'automatisation des optimisations source-à -source de programmes en utilisant des techniques de Machine Learning".
- Auguste Olivry–Grenoble, Jury, June 2022, "Automatic derivation of I/O complexity bounds for affine programs".

10.3 Popularization

10.3.1 Internal or external Inria responsibilities

- Fabrice Rastello: scientific council of CEA-EDF-Inria summer schools

10.3.2 Interventions

- Guillaume Iooss: Presentation of our work on the automatic lower bound derivation of data movement to the L3 promotion from ENS Lyon, on March 2022.

11 Scientific production

11.1 Publications of the year

International journals

[1] 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* (8th Nov. 2022), pp. 1–23. DOI: 10.1145/3570641. URL: https://hal.inria.fr/hal-03844272.

International peer-reviewed conferences

[2] H. Al Qadasi, C. Wu, Y. Falcone and S. Bensalem. 'DeepAbstraction: 2-Level Prioritization for Unlabeled Test Inputs in Deep Neural Networks'. In: AITest 2022 - IEEE 4th International Conference On Artificial Intelligence Testing. San Francisco, United States: IEEE, 15th Aug. 2022, pp. 1–8. URL: https://hal.inria.fr/hal-03911812.

[3] A. Contreras, Y. Falcone, G. Salaün and A. Zuo. 'WEASY: A Tool for Modelling Optimised BPMN Processes'. In: FACS 2022 - 18th International Conference on Formal Aspects of Component Software. Oslo / Online, Norway, 10th Nov. 2022. DOI: 10.1007/978-3-031-20872-0_7. URL: https://hal.inria.fr/hal-03848350.

[4] N. Derumigny, T. Bastian, F. Gruber, G. Iooss, C. Guillon, L.-N. Pouchet and F. Rastello. 'PALMED: Throughput Characterization for Superscalar Architectures'. In: CGO 2022 - International Symposium on Code Generation and Optimization. Seoul, South Korea, 2nd Apr. 2022, pp. 1–12. URL: https://hal.inria.fr/hal-03531740.

[5] F. Durán, Y. Falcone, C. Rocha, G. Salaün and A. Zuo. 'From Static to Dynamic Analysis and Allocation of Resources for BPMN Processes'. In: WRLA 2022 - 14th International Workshop on Rewriting Logic and its Applications. Munich, Germany, 2nd Apr. 2022, pp. 1–18. DOI: 10.1007/978-3-031-12441-9_1. URL: https://hal.inria.fr/hal-03766148.

[6] Y. Falcone. 'Residual Runtime Verification via Reachability Analysis'. In: VSTTE 2022 - 14th International Conference on Verified Software: Theories, Tools, and Experiments. Trento, Italy, 17th Oct. 2022, pp. 1–19. URL: https://hal.inria.fr/hal-03911820.

[7] Y. Falcone, I. Faqrizal and G. Salaün. 'Probabilistic Analysis of Industrial IoT Applications'. In: IoT 2022 -The 12th International Conference on the Internet of Things. Delft, Netherlands, 7th Nov. 2022. URL: https://hal.inria.fr/hal-03848674.

[8] Y. Falcone, I. Faqrizal and G. Salaün. 'Runtime Enforcement for IEC 61499 Applications'. In: SEFM 2022 - 20th International Conference on Software Engineering and Formal Methods. Berlin, Germany, 28th Sept. 2022, pp. 1–17. DOI: 10.1007/978-3-031-17108-6_22. URL: https://hal.inria.fr/hal-03766095.

[9] Y. Falcone, G. Salaün and A. Zuo. 'Probabilistic Model Checking of BPMN Processes at Runtime'. In: iFM 2022 - International Conference on integrated Formal Methods. Lugano, Switzerland, 7th June 2022, pp. 1–17. DOI: 10.1007/978-3-031-07727-2_11. URL: https://hal.inria.fr/hal-03665305.

[10] D. Furian, S. Azzopardi, Y. Falcone and G. Schneider. 'Runtime Verification of Kotlin Coroutines'. In: RV 2022 - 22nd International Conference on Runtime Verification. Tbilisi, Georgia, 28th Sept. 2022, pp. 1–19. URL: https://hal.inria.fr/hal-03911794.

[11] F. Gallay and Y. Falcone. 'DECENT: A Benchmark for Decentralized Enforcement'. In: RV 2022 - 22nd International Conference on Runtime Verification. Tblisi, Georgia, 28th Sept. 2022, pp. 1–11. URL: https://hal.inria.fr/hal-03911798.

[12] G. Iooss, A. Cohen, D. Potop-Butucaru, M. Pouzet, V. Bregeon, J. Souyris and P. Baufreton. 'Polyhedral Scheduling and Relaxation of Synchronous Reactive Systems'. In: IMPACT 2022 - 12th International Workshop on Polyhedral Compilation Techniques. Budapest, Hungary, 20th June 2022, pp. 1–12. URL: https://hal.inria.fr/hal-03901645.

[13] V. Roussanaly and Y. Falcone. 'Decentralised Runtime Verification of Timed Regular Expressions'. In: TIME 2022 - 29th International Symposium on Temporal Representation and Reasoning. Online, France, 7th Nov. 2022, pp. 1–18. DOI: 10.4230/LIPIcs...12. URL: https://hal.inria.fr/hal-03911668.

[14] C. Soueidi and Y. Falcone. 'Capturing program models with BISM'. In: SAC 2022 - 37th ACM Symposium on Applied Computing - Software Verification and Testing Track. Brno (Virtuel), Czech Republic, 25th Apr. 2022. DOI: 10.1145/3477314.3507239. URL: https://hal.inria.fr/hal-03911682.

Conferences without proceedings

[15] S. Shankar, A. Rollet, S. Pinisetty and Y. Falcone. 'Bounded-Memory Runtime Enforcement'. In: SPIN 2022 - 28th International Symposium on Model Checking of Software. Vol. 13255. Lecture Notes in Computer Science. Chicago, United States: Springer International Publishing, 23rd Aug. 2022, pp. 114–133. DOI: 10.1007/978-3-031-15077-7_7. URL: https://hal.archives-ouvertes.fr/hal-03758964.

[16] M. Vargas Vieyra. 'Robust Estimation of Laplacian Constrained Gaussian Graphical Models with Trimmed Non-convex Regularization'. In: PODS 2022 - Workshop on Principles of Distribution Shift. Baltimore, United States, 2022, pp. 1–8. URL: https://hal.inria.fr/hal-03697993.

Doctoral dissertations and habilitation theses

[17] A. Olivry. 'Automatic derivation of I/O complexity bounds for affine programs'. Université Grenoble Alpes [2020-....], 8th June 2022. URL: https://theses.hal.science/tel-03877029.

Reports & preprints

[18] N. Derumigny, T. Bastian, F. Gruber, G. Iooss, C. Guillon, L.-N. Pouchet and F. Rastello. *PALMED: Throughput Characterization for Superscalar Architectures - Extended Version*. 18th Jan. 2022. URL: https://hal.inria.fr/hal-03114933.

[19] C. Soueidi, M. Monnier, A. Kassem and Y. Falcone. *Efficient and Expressive Bytecode-Level Instrumentation for Java Programs*. 2nd June 2021. URL: https://hal.inria.fr/hal-03533152.