IN PARTNERSHIP WITH:
Université Rennes 1

Activity Report 2013

Project-Team ALF

Amdahl's Law is Forever

IN COLLABORATION WITH: Institut de recherche en informatique et systèmes aléatoires (IRISA)

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Architecture, Languages and Compilation

Table of contents

Project-Team ALF

Keywords: Processors, Compilation, Real-time, Complexity, Multicore

Creation of the Team: 2009 January 01*, updated into Project-Team:* 2011 January 01.

1. Members

Research Scientists

André Seznec [Team leader, Inria, Senior Researcher, HdR]
Sylvain Collange [Inria, Researcher]
Pierre Michaud [Inria, Researcher]
Erven Rohou [Inria, Senior Researcher]

Faculty Members

François Bodin [Univ. Rennes I, Professor, HdR]
Damien Hardy [Univ. Rennes I, Associate Professor]
Alain Ketterlin [On secondment from Univ. Strasbourg, Associate Professor, from Sep 2013]
Benjamin Lesage [Univ. Rennes I, until Aug 2013]
Isabelle Puaut [Univ. Rennes I, Professor, HdR]

Engineers

Kamil Kedzierski [Inria, granted by FP7 ERC DAL project, until Nov 2013]
Thibault Person [Inria, granted through ADT "Inria Parallel Benchmark Suite", from Nov 2013]
Nathanael Prémillieu [Inria, granted by the ALF project-team, from Oct 2013 until Dec 2013]
Emmanuel Riou [Inria, granted through ADT PADRONE]

PhD Students

Luis German Garcia Morales [Inria, granted by FP7 ERC DAL project, until Feb 2013]
Nabil Hallou [Inria, granted through the Inria Project Lab Multicore, from Feb 2013]
Sajith Kalathingal [Inria, granted by FP7 ERC DAL project]
Surya Natarajan [Inria, granted by FP7 ERC DAL project]
Junjie Lai [Inria, granted by the ALF project-team, until Mar 2013]
Hanbing Li [Inria, granted by ANR W-SEPT project]
Andrea Mondelli [Inria, granted by FP7 ERC DAL project, from Oct 2013]
Bharath Narasimha Swamy [Inria, granted by FP7 ERC DAL project]
Arthur Perais [Inria, granted by FP7 ERC DAL project]
Aswinkumar Sridharan [Inria, granted by FP7 ERC DAL project, from Sep 2013]
Arjun Suresh [Inria, granted by FP7 ERC DAL project]
Ricardo Andres Velasquez Velez [Inria, granted by the ALF project team, until Apr 2013]
Nathanael Prémillieu [Université Rennes I,MESR Allocation, until Aug 2013]

Post-Doctoral Fellow

Tao Sun [Inria, granted by FP7 ERC DAL project, from Sep 2013]

Administrative Assistant

Virginie Desroches [Inria]

2. Overall Objectives

2.1. Panorama

Multicore processors have now become mainstream for both general-purpose and embedded computing. In the near future, every hardware platform will feature thread level parallelism. Therefore, the overall computer science research community, but also industry, is facing new challenges; parallel architectures will have to be exploited by every application from HPC computing, web and entreprise servers, but also PCs, smartphones and ubiquitous embedded systems.

Within a decade, it will become technologically feasible to implement 1000s of cores on a single chip. However, several challenges must be addressed to allow the end-user to benefit from these 1000's cores chips. At that time, most applications will not be fully parallelized, therefore the effective performance of most computer systems will strongly depend on their performance on sequential sections and sequential control threads: Amdahl's law is forever. Parallel applications will not become mainstream if they have to be adapted to each new platform, therefore a simple performance scalability/portability path is needed for these applications. In many application domains, particularly in real-time systems, the effective use of multicore chips will depend on the ability of the software and hardware vendors to accurately assess the performance of applications.

The ALF team regroups researchers in computer architecture, software/compiler optimization, and real-time systems. The long-term goal of the ALF project-team is to allow the end-user to benefit from the 2020's many-core platform. We address this issue through architecture, i.e. we try to influence the definition of the 2020's many-core architecture, compiler, i.e. we intend to provide new code generation techniques for efficient execution on many-core architectures and performance prediction/guarantee, i.e. we try to propose new software and architecture techniques to predict/guarantee the response time of many-core architectures.

High performance on single thread process and sequential code is a key issue for enabling overall high performance on a 1000's cores system. Therefore, we anticipate that future manycore architectures will implement heterogeneous design featuring many simple cores and a few complex cores. Hence the research in the ALF project focuses on refining the microarchitecture to achieve high performance on single thread process and/or sequential code sections. We focus our architecture research in two main directions 1) enhancing the microarchitecture of high-end superscalar processors, 2) exploiting/modifying heterogeneous multicore architecture on a single thread. We also tackle a technological/architecture issue, the temperature wall.

Compilers are keystone solutions for any approach that deals with high performance on 100+ core systems. But general-purpose compilers try to embrace so many domains and try to serve so many constraints that they frequently fail to achieve very high performance. They need to be deeply revisited. We identify four main compiler/software related issues that must be addressed in order to allow efficient use of multi- and many-cores: 1) programming 2) resource management 3) application deployment 4) portable performance. Addressing these challenges requires to revisit parallel programming and code generation extensively.

While compiler and architecture research efforts often focus on maximizing average case performance, applications with real-time constraints do not only need high performance but also performance guarantees in all situations, including the worst-case situation. Worst-Case Execution Time estimates (WCET) need to be upper bounds of any possible execution time. The amount of safety required depends on the criticality of applications. Within the ALF team, our objective is to study performance guarantees for both (i) sequential codes running on complex cores ; (ii) parallel codes running on multicores.

Our research is partially supported by industry (Intel), the Brittany region, the ANR W-SEPT project, and the European Union (NoE HiPEAC3, ERC grant DAL, COST action TACLe).

3. Research Program

3.1. Motivations

Multicores have become mainstream in general-purpose as well as embedded computing in the last few years. The integration technology trend allows to anticipate that a 1000-core chip will become feasible before 2020. On the other hand, while traditional parallel application domains, e.g. supercomputing and transaction servers, are benefiting from the introduction of multicores, there are very few new parallel applications that have emerged during the last few years.

In order to allow the end-user to benefit from the technological breakthrough, new architectures have to be defined for the 2020's many-cores, new compiler and code generation techniques as well as new performance prediction/guarantee techniques have to be proposed .

3.2. The context

3.2.1. Technological context: The advent of multi- and many- core architecture

For almost 30 years since the introduction of the first microprocessor, the processor industry was driven by the Moore's law till 2002, delivering performance that doubled every 18-24 months on a uniprocessor. However since 2002 , and despite new progress in integration technology, the efforts to design very aggressive and very complex wide issue superscalar processors have essentially been stopped due to poor performance returns, as well as power consumption and temperature walls.

Since 2002-2003, the microprocessor industry has followed a new path for performance: the so-called multicore approach, i.e., integrating several processors on a single chip. This direction has been followed by the whole processor industry. At the same time, most of the computer architecture research community has taken the same path, focusing on issues such as scalability in multicores, power consumption, temperature management and new execution models, e.g. hardware transactional memory.

In terms of integration technology, the current trend will allow to continue to integrate more and more processors on a single die. Doubling the number of cores every two years will soon lead to up to a thousand processor cores on a single chip. The computer architecture community has coined these future processor chips as many-cores.

3.2.2. The application context: multicores, but few parallel applications

For the past five years, small scale parallel processor chips (hyperthreading, dual and quad-core) have become mainstream in general-purpose systems. They are also entering the high-end embedded system market. At the same time, very few (scalable) mainstream parallel applications have been developed. Such development of scalable parallel applications is still limited to niche market segments (scientific applications, transaction servers).

3.2.3. The overall picture

Till now, the end-user of multicores is experiencing improved usage comfort because he/she is able to run several applications at the same time. Eventually, in the near future with the 8-core or the 16-core generation, the end-user will realize that he/she is not experiencing any functionality improvement or performance improvement on current applications. The end-user will then realize that he/she needs more effective performance rather than more cores. The end-user will then ask either for parallel applications or for more effective performance on sequential applications.

3.3. Technology induced challenges

3.3.1. The power and temperatures walls

The power and the temperature walls largely contributed to the emergence of the small-scale multicores. For the past five years, mainstream general-purpose multicores have been built by assembling identical superscalar cores on a chip (e.g. IBM Power series). No new complex power hungry mechanisms were introduced in the core architectures, while power saving techniques such as power gating, dynamic voltage and frequency scaling were introduced. Therefore, since 2002, the designers have been able to keep the power consumption budget and the temperature of the chip within reasonable envelopes while scaling the number of cores with the technology.

Unfortunately, simple and efficient power saving techniques have already caught most of the low hanging fruits on energy consumption. Complex power and thermal management mechanisms are now becoming mainstream; e.g. the Intel Montecito (IA64) featured an adjunct (simple) core whose unique mission is to manage the power and temperature on two cores. Processor industry will require more and more heroic efforts on this power and temperature management policy to maintain its current performance scaling path. Hence the power and temperature walls might slow the race towards 100's and 1000's cores unless the processor industry takes a new paradigm shift from the current "replicating complex cores" (e.g. Intel Nehalem) towards many simple cores (e.g. Intel Larrabee) or heterogeneous manycores (e.g. new GPUs, IBM Cell).

3.3.2. The memory wall

For the past 20 years, the memory access time has been one of the main bottlenecks for performance in computer systems. This was already true for uniprocessors. Complex memory hierarchies have been defined and implemented in order to limit the visible memory access time as well as the memory traffic demands. Up to three cache levels are implemented for uniprocessors. For multi- and many-cores the problems are even worse. The memory hierarchy must be replicated for each core, memory bandwidth must be shared among the distinct cores, data coherency must be maintained. Maintaining cache coherency for up to 8 cores can be handled through relatively simple bus protocols. Unfortunately, these protocols do not scale for large numbers of cores, and there is no consensus on coherency mechanism for manycore systems. Moreover there is no consensus on core organization (flat ring? flat grid? hierarchical ring or grid?).

Therefore, organizing and dimensioning the memory hierarchy will be a major challenge for the computer architects. The successful architecture will also be determined by the abilitty of the applications (i.e., the programmers or the compilers or the run-time) to efficiently place data in the memory hierarchy and achieve high performance.

Finally new technology opportunities may demand to revisit the memory hierarchy. As an example, 3D memory stacking enables a huge last-level cache (maybe several gigabytes) with huge bandwidth (several Kbits/ processor cycle). This dwarfs the main memory bandwidth and may lead to other architectural tradeoffs.

3.4. Need for efficient execution of parallel applications

Achieving high performance on future multicores will require the development of parallel applications, but also an efficient compiler/runtime tool chain to adapt codes to the execution platform.

3.4.1. The diversity of parallelisms

Many potential execution parallelism patterns may coexist in an application. For instance, one can express some parallelism with different tasks achieving different functionalities. Within a task, one can expose different granularities of parallelism; for instance a first layer message passing parallelism (processes executing the same functionality on different parts of the data set), then a shared memory thread level parallelism and fine grain loop parallelism (a.k.a vector parallelism).

Current multicores already feature hardware mechanisms to address these different parallelisms: physically distributed memory — e.g. the new Intel Nehalem already features 6 different memory channels — to address task parallelism, thread level parallelism — e.g. on conventional multicores, but also on GPUs or on Cell-based machines —, vector/SIMD parallelism — e.g. multimedia instructions. Moreover they also attack finer instruction level parallelism and memory latency issues. Compilers have to efficiently discover and manage all these forms to achieve effective performance.

3.4.2. Portability is the new challenge

Up to now, most parallel applications were developed for specific application domains in high end computing. They were used on a limited set of very expensive hardware platforms by a limited number of expert users. Moreover, they were executed in batch mode.

In contrast, the expectation of most end-users of the future mainstream parallel applications running on multicores will be very different. The mainstream applications will be used by thousands, maybe millions of non-expert users. These users consider functional portability of codes as granted. They will expect their codes to run faster on new platforms featuring more cores. They will not be able to tune the application environment to optimize performance. Finally, multiple parallel applications may have to be executed concurrently.

The variety of possible hardware platforms, the lack of expertise of the end-users and the varying run-time execution environments will represent major difficulties for applications in the multicore era.

First of all, the end user considers functional portability without recompilation as granted, this is a major challenge on parallel machines. Performance portability/scaling is even more challenging. It will become inconceivable to rewrite/retune each application for each new parallel hardware platform generation to exploit them. Therefore, apart from the initial development of parallel applications, the major challenge for the next decade will be to *efficiently* run parallel applications on hardware architectures radically different from their original hardware target.

3.4.3. The need for performance on sequential code sections

3.4.3.1. Most software will exhibit substantial sequential code sections

For the foreseeable future, the majority of applications will feature important sequential code sections.

First, many legacy codes were developed for uniprocessors. Most of these codes will not be completely redeveloped as parallel applications, but will evolve to applications using parallel sections for the most compute-intensive parts. Second, the overwhelming majority of the programmers have been educated to program in a sequential programming style. Parallel programming is much more difficult, time consuming and error prone than sequential programming. Debugging and maintaining a parallel code is a major issue. Investing in the development of a parallel application will not be cost-effective for the vast majority of software developments. Therefore, sequential programming style will continue to be dominant in the foreseeable future. Most developers will rely on the compiler to parallelize their application and/or use some software components from parallel libraries.

3.4.3.2. Future parallel applications will require high performance sequential processing on 1000's cores chip

With the advent of universal parallel hardware in multicores, large diffusion parallel applications will have to run on a broad spectrum of parallel hardware platforms. They will be used by non-expert users who will not be able to tune the application environment to optimize performance. They will be executed concurrently with other processes which may be interactive.

The variety of possible hardware platforms, the lack of expertise of the end-user and the varying run-time execution environments are major difficulties for parallel applications. This tends to constrain the programming style and therefore reinforces the sequential structure of the control of the application.

Therefore, *most future parallel applications will rely on a single main thread or a few main threads in charge of distinct functionalities of the application. Each main thread will have a general sequential control and can initiate and control the parallel execution of parallel tasks.*

In 1967, Amdahl [39] pointed out that, if only a portion of an application is accelerated, the execution time cannot be reduced below the execution time of the residual part of the application. Unfortunately, even highly parallelized applications exhibit some residual sequential part. For parallel applications, this indicates that the effective performance of the future 1000's cores chip will significantly depend on their ability to be efficient on the execution of the control portions of the main thread as well as on the execution of sequential portions of the application.

3.4.3.3. The success of 1000's cores architecture will depend on single thread performance

While the current emphasis of computer architecture research is on the definition of scalable multi- many- core architectures for highly parallel applications, we believe that the success of the future 1000-core architecture will depend not only on their performance on parallel applications including sequential sections, but also on their performance on single thread workloads.

3.5. Performance evaluation/guarantee

Predicting/evaluating the performance of an application on a system without explicitly executing the application on the system is required for several usages. Two of these usages are central to the research of the ALF project-team: microarchitecture research (the system to be be evaluated does not exist) and Worst Case Execution Time estimation for real-time systems (the numbers of initial states or possible data inputs is too large).

When proposing a micro-architecture mechanism, its impact on the overall processor architecture has to be evaluated in order to assess its potential performance advantages. For microarchitecture research, this evaluation is generally done through the use of cycle-accurate simulation. Developing such simulators is quite complex and microarchitecture research was helped but also biased by some popular public domain research simulators (e.g. Simplescalar [40]). Such simulations are CPU consuming and simulations cannot be run on a complete application. Sampling representative slices of the application was proposed [4] and popularized by the Simpoint [48] framework.

Real-time systems need a different use of performance prediction; on hard real-time systems, timing constraints must be respected independently from the data inputs and from the initial execution conditions. For such a usage, the Worst Case Execution Time (WCET) of an application must be evaluated and then checked against the timing constraints. While safe and tight WCET estimation techniques and tools exist for reasonably simple embedded processors (e.g. techniques based on abstract interpretation such as [42]), accurate evaluation of the WCET of an algorithm on a complex uniprocessor system is a difficult problem. Accurately modelling data cache behavior [3] and complex superscalar pipelines are still research questions as illustrated by the presence of so-called *timing anomalies* in dynamically scheduled processors, resulting from complex interactions between processor elements (among others, interactions between caching and instruction scheduling) [46].

With the advance of multicores, evaluating / guaranteeing a computer system response time is becoming much more difficult. Interactions between processes occurs at different levels. The execution time on each core depends on the behavior of the other cores. Simulations of 1000's cores micro-architecture will be needed in order to evaluate future many-core proposals. While a few multiprocessor simulators are available for the community, these simulators cannot handle realistic 1000's cores micro-architecture. New techniques have to be invented to achieve such simulations. WCET estimations on multicore platforms will also necessitate radically new techniques, in particular, there are predictability issues on a multicore where many resources are shared; those resources include the memory hierarchy, but also the processor execution units and all the hardware resources if SMT is implemented [52].

3.6. General research directions

The overall performance of a 1000's core system will depend on many parameters including architecture, operating system, runtime environment, compiler technology and application development. In the ALF project, we will essentially focus on architecture, compiler/execution environment as well as performance predictability, and in particular WCET estimation. Moreover, architecture research, and to a smaller extent, compiler and WCET estimation researches rely on processor simulation. A significant part of the effort in ALF will be devoted to define new processor simulation techniques.

3.6.1. Microarchitecture research directions

The overall performance of a multicore system depends on many parameters including architecture, operating system, runtime environment, compiler technology and application development. Even the architecture dimension of a 1000's core system cannot be explored by a single research project. Many research groups are exploring the parallel dimension of the multicores essentially targeting issues such as coherency and scalability.

We have identified that high performance on single threads and sequential codes is one of the key issues for enabling overall high performance on a 1000's core system and we anticipate that the general architecture of such 1000's core chip will feature many simple cores and a few very complex cores.

Therefore our research in the ALF project will focus on refining the microarchitecture to achieve high performance on single process and/or sequential code sections within the general framework of such an heteregeneous architecture. This leads to two main research directions 1) enhancing the microarchitecture of high-end superscalar processors, 2) exploiting/modifying heterogeneous multicore architecture on a single process. The temperature wall is also a major technological/architectural issue for the design of future processor chips.

3.6.1.1. Enhancing complex core microarchitecture

Research on wide issue superscalar processors was merely stopped around 2002 due to limited performance returns and the power consumption wall.

When considering a heterogeneous architecture featuring hundreds of simple cores and a few complex cores, these two obstacles will partially vanish: 1) the complex cores will represent only a fraction of the chip and a fraction of its power consumption. 2) any performance gain on (critical) sequential threads will result in a performance gain of the whole system

On the complex core, the performance of a sequential code is limited by several factors. At first, on current architectures, it is limited by the peak performance of the processor. To push back this first limitation, we will explore new microarchitecture mechanisms to increase the potential peak performance of a complex core enabling larger instruction issue width. The processor performance is also limited by control dependencies. To push back this limitation, we will explore new branch prediction mechanisms as well as new directions for reducing branch misprediction penalties [8], [10]. As data dependencies may strongly limit performance, we will revisit data prediction. Processor performance is also often highly dependent on the presence or absence of data in a particular level of the memory hierarchy. For the ALF multicore, we will focus on sharing the access to the memory hierarchy in order to adapt the performance of the main thread to the performance of the other cores. All these topics should be studied with the new perspective of quasi unlimited silicon budget.

3.6.1.2. Exploiting heterogeneous multicores on single process

When executing a sequential section on the complex core, the simple cores will be free. Two main research directions to exploit thread level parallelism on a sequential thread have been initiated in late 90's within the context of simultaneous multithreading and early chip multiprocessor proposals: helper threads and speculative multithreading.

Helper threads were initially proposed to improve the performance of the main threads on simultaneous multithreaded architectures [41]. The main idea of helper threads is to execute codes that will accelerate the main thread without modifying its semantic.

In many cases, the compiler cannot determine if two code sections are independent due to some unresolved memory dependency. When no dependency occurs at execution time, the code sections can be executed in parallel. Thread-Level Speculation has been proposed to exploit coarse grain speculative parallelism. Several hardware-only proposals were presented [47], but the most promising solutions integrate hardware support for software thread-level speculation [50].

In the context of future manycores, thread-level speculation and helper threads should be revisited. Many simple cores will be available for executing helper threads or speculative thread execution during the execution of sequential programs or sequential code sections. The availability of these many cores is an opportunity as well as a challenge. For example, one can try to use the simple cores to execute many different helper threads that could not be implemented within a simultaneous multithreaded processor. For thread level speculation, the new challenge is the use of less powerful cores for speculative threads. Moreover the availability of many simple cores may lead to the use of helper threads and thread level speculation at the same time.

3.6.1.3. Temperature issues

Temperature is one of the constraints that have prevented the processor clock frequency to be increased in recent years. Besides techniques to decrease the power consumption, the temperature issue can be tackled with *dynamic thermal management* [7] through techniques such as clock gating or throttling and *activity migration* [49][5].

Dynamic thermal management (DTM) is now implemented on existing processors. For high performance, processors are dimensioned according to the average situation rather than to the worst case situation. Temperature sensors are used on the chip to trigger dynamic thermal management actions, for instance thermal throttling whenever necessary. On multicores, it is possible to migrate the activity from one core to another in order to limit temperature.

A possible way to increase sequential performance is to take advantage of the smaller gate delay that comes with miniaturization, which permits in theory to increase the clock frequency. However increasing the clock frequency generally requires to increase the instantaneous power density. This is why DTM and activity migration will be key techniques to deal with Amdahl's law in future many-core processors.

3.6.2. Processor simulation research

Architecture studies, and in particular microarchitecture studies, require extensive validations through detailed simulations. Cycle accurate simulators are needed to validate the microarchitectural mechanisms.

Within the ALF project, we can distinguish two major requirements on the simulation: 1) single process and sequential code simulations 2) parallel code sections simulations.

For simulating parallel code sections, a cycle-accurate microarchitecture simulator of a 1000-core architecture will be unacceptably slow. In [6], we showed that mixing analytical modeling of the global behavior of a processor with detailed simulation of a microarchitecture mechanism allows to evaluate this mechanism. Karkhanis and Smith [43] further developed a detailed analytical simulation model of a superscalar processor. Building on top of these preliminary researches, simulation methodology mixing analytical modeling of the simple cores with a more detailed simulation of the complex cores is appealing. The analytical model of the simple cores will aim at approximately modeling the impact of the simple core execution on the shared resources (e.g. data bandwidth, memory hierarchy) that are also used by the complex cores.

Other techniques such as regression modeling [44] can also be used for decreasing the time required to explore the large space of microarchitecture parameter values. We will explore these techniques in the context of many-core simulation.

In particular, research on temperature issues will require the definition and development of new simulation tools able to simulate several minutes or even hours of processor execution, which is necessary for modeling thermal effects faithfully.

3.6.3. Compiler research directions

3.6.3.1. General directions

Compilers are keystone solutions for any approach that deals with high performance on 100+ processors systems. But general-purpose compilers try to embrace so many domains and try to serve so many constraints that they frequently fail to achieve very high performance. They need to be deeply revisited. We identify four main compiler/software related issues that must be addressed in order to allow efficient use of multi- and many-cores: 1) programming 2) resource management 3) application deployment 4) portable performance. Addressing these challenges will require to revisit parallel programming and code generation extensively.

The past of parallel programming is scattered with hundreds of parallel languages. Most of these languages were designed to program homogeneous architectures and were targeting a small and well-trained community of HPC programmers. With the new diversity of parallel hardware platforms and the new community of non-expert developers, expressing parallelism is not sufficient anymore. Resource management, application deployment and portable performance are intermingled issues that require to be addressed holistically.

As many decisions should be taken according to the available hardware, resource management cannot be separated from parallel programming. Deploying applications on various systems without having to deal with thousands of hardware configurations (different numbers of cores, accelerators, ...) will become a major concern for software distribution. The grail of parallel computing is to be able to provide portable performance on a large set of parallel machines and varying execution contexts.

Recent techniques are showing promises. Iterative compilation techniques, exploiting the huge CPU cycle count now available, can be used to explore the optimization space at compile-time. Second, machine-learning techniques can be used to automatically improve compilers and code generation strategies. Speculation can be used to deal with necessary but missing information at compile-time. Finally, dynamic techniques can select or generate at run-time the most efficient code adapted to the execution context and available hardware resources.

Future compilers will benefit from past research, but they will also need to combine static and dynamic techniques. Moreover, domain specific approaches might be needed to ensure success. The ALF research effort will focus on these static and dynamic techniques to address the multicore application development challenges.

3.6.3.2. Portability of applications and performance through virtualization

The life cycle is much longer for applications than for hardware. Unfortunately the multicore era jeopardizes the old binary compatibility recipe. Binaries cannot automatically exploit additional computing cores or new accelerators available on the silicon. Moreover maintaining backward binary compatibility on future parallel architectures will rapidly become a nightmare, applications will not run at all unless some kind of dynamic binary translation is at work.

Processor virtualization addresses the problem of portability of functionalities. Applications are not compiled to the final native code but to a target independent format. This is the purpose of languages such as Java and .NET. Bytecode formats are often *a priori* perceived as inappropriate for performance intensive applications and for embedded systems. However, it was shown that compiling a C or C++ program to a bytecode format produces a code size similar to dense instruction sets [2]. Moreover, this bytecode representation can be compiled to native code with performance similar to static compilation [1]. Therefore processor virtualization for high performance, i.e., for languages like C or C++, provides significant advantages: 1) it simplifies software engineering with fewer tools to maintain and upgrade; 2) it allows better code readability and easier code maintenance since it avoids code specialization for specific targets using compile time macros such as #ifdef ; 3) the *execution code* deployed on the system is the execution code that has been debugged and validated, as opposed to the same *source code* has been recompiled for another platform; 4) new architectures will come with their JIT compiler. The JIT will (should) automatically take advantage of new architecture features such as SIMD/vector instructions or extra processors.

Our objective is to enrich processor virtualization to allow both functional portability and high performance using JIT at runtime, or bytecode-to-native code offline compiler. Split compilation can be used to annotate the bytecode with relevant information that can be helpful to the JIT at runtime or to the bytecode to native code offline compiler. Because the first compilation pass occurs offline, aggressive analyses can be run and their outcomes encoded in the bytecode. For example, such information include vectorizability, memory references (in)dependencies, suggestions derived from iterative compilation, polyhedral analysis, or integer linear programming. Virtualization allows to postpone some optimizations to run time, either because they increase the code size and would increase the cost of an embedded system or because the actual hardware platform characteristics are unknown.

3.6.4. Performance predictability for real-time systems

While compiler and architecture research efforts often focus on maximizing average case performance, applications with real-time constraints do not need only high performance but also performance guarantees in all situations, including the worst-case situation. Worst-Case Execution Time estimates (WCET) need to be upper bounds of any possible execution time. The safety level required depends on the criticality of applications: missing a frame on a video in the airplane for passenger in seat 20B is less critical than a safety critical decision in the control of the airplane.

Within the ALF project, our objective is to study performance guarantees for both (i) sequential codes running on complex cores ; (ii) parallel codes running on the multicores. This results in two quite distinct problems.

For sequential code executing on a single core, one can expect that, in order to provide real-time possibility, the architecture will feature an execution mode where a given processor will be guaranteed to access a fixed portion of the shared resources (caches, memory bandwidth). Moreover, this guaranteed share could be optimized at compile time to enforce the respect of the time constraints. However, estimating the WCET of an application on a complex micro-architecture is still a research challenge. This is due to the complex interaction of micro-architectural elements (superscalar pipelines, caches, branch prediction, out-of-order execution) [46]. We will continue to explore pure analytical and static methods. However when accurate static hardware modeling methods cannot handle the hardware complexity, new probabilistic methods [45] might be needed to explore to obtain as safe as possible WCET estimates.

Providing performance guarantees for parallel applications executed on a multicore is a new and challenging issue. Entirely new WCET estimation methods have to be defined for these architectures to cope with dynamic resource sharing between cores, in particular on-chip memory (either local memory or caches) are shared, but also buses, network-on-chip and the access to the main memory. Current pure analytical methods are too pessimistic at capturing interferences between cores [53], therefore hardware-based or compiler methods such as [51] have to be defined to provide some degree of isolation between cores. Finally, similarly to simulation methods, new techniques to reduce the complexity of WCET estimation will be explored to cope with manycore architectures.

4. Application Domains

4.1. Any computer usage

The ALF team is working on the fundamental technologies for computer science: processor architecture and performance-oriented compilation. The research results have impacts on any application domain that requires high performance executions (telecommunication, multimedia, biology, health, engineering, environment ...), but also on many embedded applications that exhibit other constraints such as power consumption, code size and guaranteed response time. Our research activity implies the development of software prototypes.

5. Software and Platforms

5.1. Panorama

The ALF team is developing several software prototypes for research purposes: compilers, architectural simulators, programming environments, ...

Among the many prototypes developed in the project, we describe here **ATMI**, a microarchitecture temperature model for processor simulation, **STiMuL**, a temperature model for steady state studies, **ATC**, an address trace compressor, **HAVEGE**, an unpredictable random number generator, **tiptop**, a user-level Linux utility that collects data from hardware performance counters for running tasks, and **Padrone**, a platform for dynamic binary analysis and optimization.

5.2. ATMI

Participant: Pierre Michaud.

Microarchitecture temperature model

Status : Registered with APP Number IDDN.FR.001.250021.000.S.P.2006.000.10600, Available under GNU General Public License

Research on temperature-aware computer architecture requires a chip temperature model. General purpose models based on classical numerical methods like finite differences or finite elements are not appropriate for such research, because they are generally too slow for modeling the time-varying thermal behavior of a processing chip.

We have developed an ad hoc temperature model, ATMI (Analytical model of Temperature in MIcroprocessors), for studying thermal behaviors over a time scale ranging from microseconds to several minutes. ATMI is based on an explicit solution to the heat equation and on the principle of superposition. ATMI can model any power density map that can be described as a superposition of rectangle sources, which is appropriate for modeling the microarchitectural units of a microprocessor.

Visit http://www.irisa.fr/alf/ATMI or contact Pierre Michaud.

5.3. STiMuL

Participant: Pierre Michaud.

Microarchitecture temperature modeling

Status: Registered with APP Number IDDN.FR.001.220013.000.S.P.2010.000.31235, Available under GNU General Public License

Some recent research has started investigating the microarchitectural implications of 3D circuits, for which the thermal constraint is stronger than for conventional 2D circuits.

STiMuL can be used to model steady-state temperature in 3D circuits consisting of several layers of different materials. STiMuL is based on a rigorous solution to the Laplace equation.The number and characteristics of layers can be defined by the user. The boundary conditions can also be defined by the user. In particular, STiMuL can be used along with thermal imaging to obtain the power density inside an integrated circuit. This power density could be used for instance in a dynamic simulation oriented temperature modeling such as ATMI.

STiMuL is written in C and uses the FFTW library for discrete Fourier transforms computations.

Visit http://www.irisa.fr/alf/stimul or contact Pierre Michaud.

5.4. ATC

Participant: Pierre Michaud.

Address trace compression

Status: registered with APP number IDDN.FR.001.160031.000.S.P.2009.000.10800, available under GNU LGPL License.

Trace-driven simulation is an important tool in the computer architect's toolbox. However, one drawback of trace-driven simulation is the large amount of storage that may be necessary to store traces. Trace compression techniques are useful for decreasing the storage space requirement. But general-purpose compression techniques are generally not optimal for compressing traces because they do not take advantage of certain characteristics of traces. By specializing the compression method and taking advantages of known trace characterics, it is possible to obtain a better tradeoff between the compression ratio, the memory consumption and the compression and decompression speed.

ATC is a utility and a C library for compressing/decompressing address traces. It implements a new lossless transformation, Bytesort, that exploits spatial locality in address traces. ATC leverages existing general-purpose compressors such as gzip and bzip2. ATC also provides a lossy compression mode that yields higher compression ratios while preserving certain important characteristics of the original trace.

Visit http://www.irisa.fr/alf/atc or contact Pierre Michaud.

5.5. HAVEGE

Participant: André Seznec.

Unpredictable random number generator

Contact : André Seznec

Status : Registered with APP Number IDDN.FR.001.500017.001.S.P.2001.000.10000. Available under the LGPL license.

An unpredictable random number generator is a practical approximation of a truly random number generator. Such unpredictable random number generators are needed for cryptography. HAVEGE (HArdware Volatile Entropy Gathering and Expansion) is a user-level software unpredictable random number generator for general-purpose computers that exploits the continuous modifications of the internal volatile hardware states in the processor as a source of uncertainty [9]. HAVEGE combines on-the-fly hardware volatile entropy gathering with pseudo-random number generation.

The internal state of HAVEGE includes thousands of internal volatile hardware states and is merely unmonitorable. HAVEGE can reach an unprecedented throughput for a software unpredictable random number generator: several hundreds of megabits per second on current workstations and PCs.

The throughput of HAVEGE favorably competes with usual pseudo-random number generators such as `rand()` or `random()`. While HAVEGE was initially designed for cryptology-like applications, this high throughput makes HAVEGE usable for all application domains demanding high performance and high quality random number generators, e.g., Monte Carlo simulations.

Visit http://www.irisa.fr/alf/index.php?option=com_content&view=article&id=5/havege&catid=3/projects&Itemid=3&lang=fr or contact André Seznec.

5.6. Tiptop

Participant: Erven Rohou.

Performance, hardware counters, analysis tool.

Status: Registered with APP (Agence de Protection des Programmes). Available under GNU General Public License v2.

Tiptop is a new simple and flexible user-level tool that collects hardware counter data on Linux platforms (version 2.6.31+). The goal is to make the collection of performance and bottleneck data as simple as possible, including simple installation and usage. In particular, we stress the following points.

- Installation is only a matter of compiling the source code. No patching of the Linux kernel is needed, and no special-purpose module needs to be loaded.
- No privilege is required, any user can run *tiptop* — non-privileged users can only watch processes they own, ability to monitor anybody's process opens the door to side-channel attacks.
- The usage is similar to *top*. There is no need for the source code of the applications of interest, making it possible to monitor proprietary applications or libraries. And since there is no probe to insert in the application, understanding of the structure and implementation of complex algorithms and code bases is not required.
- Applications do not need to be restarted, and monitoring can start at any time (obviously, only events that occur after the start of *tiptop* are observed).
- Events can be counted per thread, or per process.
- Any expression can be computed, using the basic arithmetic operators, constants, and counter values.
- A configuration file lets users define their prefered setup, as well as custom expressions.

Tiptop is written in C. It can take advantage of libncurses when available for pseudo-graphic display.

Tiptop version 2.2 was released in March 2013.

For more information, please contact Erven Rohou and/or visit http://tiptop.gforge.inria.fr.

5.7. Padrone

Participants: Erven Rohou, Emmanuel Riou.

Performance, profiling, dynamic optimization

Status: Ongoing development, early prototype.

Padrone is new platform for dynamic binary analysis and optimization. It provides an API to help clients design and develop analysis and optimization tools for binary executables. Padrone attaches to running applications, only needing the executable binary in memory. No source code or debug information is needed. No application restart is needed either. This is specially interesting for legacy or commercial applications, but also in the context of cloud deployment, where actual hardware is unknown, and other applications competing for hardware resources can vary. The profiling overhead is minimum.

Padrone is written in C.

For more information, please contact Erven Rohou.

6. New Results

6.1. Processor Architecture within the ERC DAL project

Participants: Pierre Michaud, Nathanaël Prémillieu, Luis Germán Garcia Morales, Bharath Narasimha Swamy, Sylvain Collange, André Seznec, Arthur Perais, Surya Natarajan, Sajith Kalathingal, Tao Sun, Andrea Mondelli, Aswinkumar Sridharan, Alain Ketterlin, Kamil Kedzierski.

Processor, cache, locality, memory hierarchy, branch prediction, multicore, power, temperature

Multicore processors have now become mainstream for both general-purpose and embedded computing. Instead of working on improving the architecture of the next generation multicore, with the DAL project, we deliberately anticipate the next few generations of multicores. While multicores featuring 1000s of cores might become feasible around 2020, there are strong indications that sequential programming style will continue to be dominant. Even future mainstream parallel applications will exhibit large sequential sections. Amdahl's law indicates that high performance on these sequential sections is needed to enable overall high performance on the whole application. On many (most) applications, the effective performance of future computer systems using a 1000-core processor chip will significantly depend on their performance on both sequential code sections and single threads.

We envision that, around 2020, the processor chips will feature a few complex cores and many (may be 1000's) simpler, more silicon and power effective cores.

In the DAL research project, http://www.irisa.fr/alf/index.php?option=com_content&view=article&id=55&Itemid=3&lang=en, we explore the microarchitecture techniques that will be needed to enable high performance on such heterogeneous processor chips. Very high performance will be required on both sequential sections, -legacy sequential codes, sequential sections of parallel applications-, and critical threads on parallel applications, -e.g. the main thread controlling the application. Our research focuses essentially on enhancing single process performance.

6.1.1. Microarchitecture exploration of control flow reconvergence

Participants: Nathanaël Prémillieu, André Seznec.

After continuous progress over the past 15 years [8], [10], the accuracy of branch predictors seems to be reaching a plateau. Other techniques to limit control dependency impact are needed. Control flow reconvergence is an interesting property of programs. After a multi-option control-flow instruction (i.e. either a conditional branch or an indirect jump including returns), all the possible paths merge at a given program point: the reconvergence point.

Superscalar processors rely on aggressive branch prediction, out-of-order execution and instruction level parallelism for achieving high performance. Therefore, on a superscalar core , the overall speculative execution after the mispredicted branch is cancelled, leading to a substantial waste of potential performance. However, deep pipelines and out-of-order execution induce that, when a branch misprediction is resolved, instructions following the reconvergence point have already been fetched, decoded and sometimes executed. While some of this executed work has to be cancelled since data dependencies exist, canceling the control independent work is a waste of resources and performance. We have proposed a new hardware mechanism called SYRANT, SYmmetric Resource Allocation on Not-taken and Taken paths, addressing control flow reconvergence at a reasonable cost. Moreover, as a side contribution of this research we have shown that, for a modest hardware cost, the outcomes of the branches executed on the wrong paths can be used to guide branch prediction on the correct path [13].

6.1.2. *Efficient Execution on Guarded Instruction Sets*

Participants: Nathanaël Prémillieu, André Seznec.

ARM ISA based processors are no longer low complexity processors. Nowadays, ARM ISA based processor manufacturers are struggling to implement medium-end to high-end processor cores which implies implementing a state-of-the-art out-of-order execution engine. Unfortunately providing efficient out-of-order execution on legacy ARM codes may be quite challenging due to guarded instructions.

Predicting the guarded instructions addresses the main serialization impact associated with guarded instructions execution and the multiple definition problem. Moreover, guard prediction allows to use a global branch-and-guard history predictor to predict both branches and guards, often improving branch prediction accuracy. Unfortunately such a global branch-and-guard history predictor requires the systematic use of guard predictions. In that case, poor guard prediction accuracy would lead to poor overall performance on some applications.

Building on top of recent advances in branch prediction and confidence estimation, we propose a hybrid branch and guard predictor, combining a global branch history component and global branch-and-guard history component. The potential gain or loss due to the systematic use of guard prediction is dynamically evaluated at run-time. Two computing modes are enabled: systematic guard prediction use and high confidence only guard prediction use. Our experiments show that on most applications, an overwhelming majority of guarded instructions are predicted. Therefore a relatively inefficient but simple hardware solution can be used to execute the few unpredicted guarded instructions. Significant performance benefits are observed on most applications while applications with poorly predictable guards do not suffer from performance loss [35], [34], [13].

6.1.3. *Revisiting Value Prediction*

Participants: Arthur Perais, André Seznec.

Value prediction was proposed in the mid 90's to enhance the performance of high-end microprocessors. The research on Value Prediction techniques almost vanished in the early 2000's as it was more effective to increase the number of cores than to dedicate some silicon area to Value Prediction. However high end processor chips currently feature 8-16 high-end cores and the technology will allow to implement 50-100 of such cores on a single die in a foreseeable future. Amdahl's law suggests that the performance of most workloads will not scale to that level. Therefore, dedicating more silicon area to value prediction in high-end cores might be considered as worthwhile for future multicores.

First, we introduce a new value predictor VTAGE harnessing the global branch history [32]. VTAGE directly inherits the structure of the indirect jump predictor ITTAGE [8]. VTAGE is able to predict with a very high accuracy many values that were not correctly predicted by previously proposed predictors, such as the FCM predictor and the stride predictor. Three sources of information can be harnessed by these predictors: the global branch history, the differences of successive values and the local history of values. Moreover, VTAGE does not suffer from short critical prediction loops and can seamlessly handle back-to-back predictions, contrarily to previously proposed, hard to implement FCM predictors.

Second, we show that all predictors are amenable to very high accuracy at the cost of some loss on prediction coverage [32]. This greatly diminishes the number of value mispredictions and allows to delay validation until commit-time. As such, no complexity is added in the out-of-order engine because of VP (save for ports on the register file) and pipeline squashing at commit-time can be used to recover. This is crucial as adding *selective replay* in the OoO core would tremendously increase complexity.

Third, we leverage the possibility of validating predictions at commit to introduce a new microarchitecture, EOLE [31]. EOLE features *Early Execution* to execute simple instructions whose operands are ready in parallel with Rename and *Late Execution* to execute simple predicted instructions and high confidence branches just before Commit. EOLE depends on Value Prediction to provide operands for *Early Execution* and predicted instructions for *Late Execution*. However, Value Prediction requires EOLE to become truly practical. That is, EOLE allows to reduce the out-of-order issue-width by 33% without impeding performance. As such, the number of ports on the register file diminishes. Furthermore, optimizations of the register file such as *banking* further reduce the number of required ports. Overall EOLE possesses a register file whose complexity is on-par with that of a regular wider-issue superscalar while the out-of-order components (scheduler , bypass) are greatly simplified. Moreover, thanks to Value Prediction, speedup is obtained on many benchmarks of the SPEC'00/'06 suite.

6.1.4. Helper threads

Participants: Bharath Narasimha Swamy, Alain Ketterlin, André Seznec.

As the number of cores on die increases with the improvements in silicon process technology, the strategy of replicating identical cores does not scale to meet the performance needs of mixed workloads. Heterogeneous Many Cores (HMC) that mix many simple cores with a few complex cores are emerging as a design alternative that can provide both high performance and power-efficient execution. The availability of many simple cores in a HMC presents an opportunity to utilize low power cores to accelerate sequential execution on the complex core. For example simple cores can execute pre-computational (or helper) code and generate prefetch requests for the main thread.

We explore the design of a lightweight architectural framework that provides instruction set support and a low-latency interface to simple-cores for efficient helper code execution. We utilize static analyses and profile data to generate helper codelets that target delinquent loads in the main thread. The main thread is instrumented to initiate helper execution ahead of time, and utilizes instruction set support to signal helper execution on the simple core, and to pass live-in values for the helper codelet. Pre-computational code executes on the simple core and generates prefetch requests that install data into a shared last-level cache. Initial experiments with a trace based simulation framework show that helper execution has the potential to cover cache-missing loads on the main thread.

The restriction of prefetching to a lower level shared cache in a loosely coupled system limits the benefits of helper execution. The main thread should have a low latency access mechanism to data prefetched by helper execution. We plan to explore direct, yet light weight, mechanisms for data communication between the helper core and the main core.

6.1.5. Adaptive Intelligent Memory Systems

Participants: André Seznec, Aswinkumar Sridharan.

On multicores, the processors are sharing the memory hierarchy, buses, caches, and memory. The performance of any single application is impacted by its environment and the behavior of the other applications co-running on the multicore. Different strategies have been proposed to isolate the behavior of the different co-running applications, for example performance isolation cache partitioning, while several studies have addressed the global issue of optimizing throughput through the cache management.

However these studies are limited to a few cores (2-4-8) and generally features mechanisms that cannot scale to 50-100 cores. Moreover so far the academic propositions have generally taken into account a single parameter, the cache replacement policy or the cache partitioning. Other parameters such as cache prefetching and its aggressiveness already impact the behavior of a single thread application on a uniprocessor. Cache prefetching policy of each thread will also impact the behavior of all the co-running threads.

Our objective is to define an Adaptive and Intelligent Memory System management hardware, AIMS. The goal of AIMS will be to dynamically adapt the different parameters of the memory hierarchy access for each individual co-running process in order to achieve a global objective such as optimized throughput, thread fairness or respecting quality of services for some privileged threads.

6.1.6. Modeling multi-threaded programs execution time in the many-core era

Participants: Surya Natarajan, Bharath Narasimha Swamy, André Seznec.

Multi-core have become ubiquitous and industry is already moving towards the many-core era. Many open-ended questions remain unanswered for the upcoming many-core era. From the software perspective, it is unclear which applications will be able to benefit from many cores. From the hardware perspective, the tradeoff between implementing many simple cores, fewer medium aggressive cores or even only a moderate number of aggressive cores is still in debate. Estimating the potential performance of future parallel applications on the yet-to-be-designed future many cores is very speculative. The simple models proposed by Amdahl's law or Gustafson's law are not sufficient and may lead to erroneous conclusions. In this paper, we propose a still simple execution time model for parallel applications, the SNAS model. As previous models, the SNAS model evaluates the execution time of both the serial part and the parallel part of the application, but takes into account the scaling of both these execution times with the input problem size and the number of processors. For a given application, a few parameters are collected on the effective execution of the application with a few threads and small input sets. The SNAS model allows to extrapolate the behavior of a future application exhibiting similar scaling characteristics on a many core and/or a large input set. Our study shows that the execution time of the serial part of many parallel applications tends to increase along with the problem size, and in some cases with the number of processors. It also shows that the efficiency of the execution of the parallel part decreases dramatically with the number of processors for some applications. Our model also indicates that since several different applications scaling will be encountered, hybrid architectures featuring a few aggressive cores and many simple cores should be privileged.

6.1.6.1. Augmenting superscalar architecture for efficient many-thread parallel execution

Participants: Sylvain Collange, André Seznec, Sajith Kalathingal.

We aim at exploring the design of a unique core that efficiently run both sequential and massively parallel sections. We explore how the architecture of a complex superscalar core has to be modified or enhanced to be able to support the parallel execution of many threads from the same application (10's or even 100's a la GPGPU on a single core).

SIMD execution is the preferred way to increase energy efficiency on data-parallel workloads. However, explicit SIMD instructions demand challenging auto-vectorization or manual coding, and any change in SIMD width requires at least a recompile, and typically manual code changes. Rather than vectorize at compile-time, our approach is to dynamically vectorize SPMD programs at the micro-architectural level. The SMT-SIMD hybrid core we propose extracts data parallelism from thread parallelism by scheduling groups of threads in lockstep, in a way inspired by the execution model of GPUs. As in GPUs, conditional branches whose outcome differ between threads are handled with conditionally masked execution. However, while GPUs rely on explicit re-convergence instructions to restore lockstep execution, we target existing general-purpose instruction sets, in order to run legacy binary programs. Thus, the main challenge consists in detecting re-convergence points dynamically.

We proposed instruction fetch policies that apply heuristics to maximize the cycles spent in lockstep execution. We evaluated their efficiency and performance impact on an out-of-order superscalar core simulator. Results validate the viability of our approach, by showing that existing compiled SPMD programs are amenable to lockstep execution without modification nor recompilation.

6.2. Other Architecture Studies

Participants: Damien Hardy, Pierre Michaud, Ricardo Andrés Velásquez, Sylvain Collange, André Seznec, Sajith Kalathingal, Junjie Lai.

GPU, performance, simulation, vulnerability

6.2.1. Performance Upperbound Analysis of GPU applications

Participants: Junjie Lai, André Seznec.

In the framework of the ANR Cosinus PetaQCD project (ended Oct 2012), we have been modeling the demands of high performance scientific applications on hardware. GPUs have become popular and cost-effective hardware platforms. In this context, we have been addressing the gap between theoretical peak performance on GPU and the effective performance. There have been many studies on optimizing specific applications on GPU and also a lot of studies on automatic tuning tools. However, the gap between the effective performance and the maximum theoretical performance is often huge. A tighter performance upperbound of an application is needed in order to evaluate whether further optimization is worth the effort. We designed a new approach to compute the CUDA application's performance upperbound through intrinsic algorithm information coupled with low-level hardware benchmarking. Our analysis [11], [22] allows us to understand which parameters are critical to the performance and have more insights of the performance result. As an example, we analyzed the performance upperbound of SGEMM (Single-precision General Matrix Multiply) on Fermi and Kepler GPUs. Through this study, we uncover some undocumented features on Kepler GPU architecture. Based on our analysis, our implementations of SGEMM achieve the best performance on Fermi and Kepler GPUs so far (5 % improvement on average).

6.3. Microarchitecture Performance Analysis

Participants: Ricardo Andrés Velásquez, Pierre Michaud, André Seznec.

6.3.1. Selecting benchmark combinations for the evaluation of multicore throughput

Participants: Ricardo Andrés Velásquez, Pierre Michaud, André Seznec.

In [26], we have shown that fast approximate microarchitecture models such as BADCO [16] can be useful for selecting multiprogrammed workloads for evaluating the throughput of multicore processors. Computer architects usually study multiprogrammed workloads by considering a set of benchmarks and some combinations of these benchmarks. However, there is no standard method for selecting such sample, and different authors have used different methods. The choice of a particular sample impacts the conclusions of a study. Using BADCO, we propose and compare different sampling methods for defining multiprogrammed workloads for computer architecture. We evaluate their effectiveness on a case study, the comparison of several multicore last-level cache replacement policies. We show that random sampling, the simplest method, is robust to define a representative sample of workloads, provided the sample is big enough. We propose a method for estimating the required sample size based on fast approximate simulation. We propose a new method, workload stratification, which is very effective at reducing the sample size in situations where random sampling would require large samples.

6.3.2. A systematic approach for defining multicore throughput metrics

Participant: Pierre Michaud.

This research was done in collaboration with Stijn Eyerman from Ghent University.

Measuring throughput is not as straightforward as measuring execution time. This has led to an ongoing debate on what forms a meaningful throughput metric for multi-program workloads. In [29], we present a method to construct throughput metrics in a systematic way: we start by expressing assumptions on job size, job distribution, scheduling, etc., that together define a theoretical throughput experiment. The throughput metric is then the average throughput of this experiment. Different assumptions lead to different metrics, so one should select the metric whose assumptions are close to the real usage he/she has in mind. We elaborate multiple metrics based on different assumptions. In particular, we identify the assumptions that lead to the commonly used weighted speedup and harmonic mean of speedups. Our study clarifies that they are actual throughput metrics, which was recently questioned. We also propose some new throughput metrics, whose calculation sometimes requires approximation. We use synthetic and real experimental data to characterize

metrics and show how they relate to each other. Our study can also serve as a starting point if one needs to define a new metric based on specific assumptions, other than the ones we consider in this study. Throughput metrics should always be defined from explicit assumptions, because this leads to a better understanding of the implications and limits of the results obtained with that metric.

6.4. Compiler, vectorization, interpretation

Participants: Erven Rohou, Emmanuel Riou, Arjun Suresh, André Seznec, Nabil Hallou, Alain Ketterlin, Sylvain Collange.

6.4.1. Vectorization Technology To Improve Interpreter Performance

Participant: Erven Rohou.

Recent trends in consumer electronics have created a new category of portable, lightweight software applications. Typically, these applications have fast development cycles and short life spans. They run on a wide range of systems and are deployed in a target independent bytecode format over Internet and cellular networks. Their authors are untrusted third-party vendors, and they are executed in secure managed runtimes or virtual machines. Furthermore, due to security policies, these virtual machines are often lacking just-in-time compilers and are reliant on interpreter execution.

The main performance penalty in interpreters arises from instruction dispatch. Each bytecode requires a minimum number of machine instructions to be executed. In this work we introduce a powerful and portable representation that reduces instruction dispatch thanks to vectorization technology. It takes advantage of the vast research in vectorization and its presence in modern compilers. Thanks to a split compilation strategy, our approach exhibits almost no overhead. Complex compiler analyses are performed ahead of time. Their results are encoded on top of the bytecode language, becoming new SIMD IR (i.e., intermediate representation) instructions. The bytecode language remains unmodified, thus this representation is compatible with legacy interpreters.

This approach drastically reduces the number of instructions to interpret and improves execution time. [15]. SIMD IR instructions are mapped to hardware SIMD instructions when available, with a substantial improvement.

6.4.2. Improving sequential performance: the case of floating point computations

Participants: Erven Rohou, André Seznec, Arjun Suresh.

One way to enhance sequential performance is to consider floating point computations. Languages and instruction sets provide support for only a few representations, namely float and double, and programmers are likely to use the most accurate (unless they handle large data structures). Still, in most cases, programmers do not formally specify the precision they require from their applications, and have no guarantee on the precision they actually get. This is an opportunity for a tradeoff between performance and precision: programs could run faster at the expense of a less accurate result (note that existing compilers already embed some unsafe transformations, for example when flags such as -fast or -ffastmath are used).

The first step consisted in applying memoization to the math library libm. In this case, results are still correct. The performance improvement comes from caching results of pure functions, and retrieving them instead of recomputing a result. This shows good results on floating point intensive benchmarks. In a next step, a helper thread will monitor the patterns of parameters and precompute likely values to "prefetch" results ahead of time.

Reduced precision comes into play when no pattern can be identified, but the new value is close enough to already computed values. We plan to apply interpolation to compute the result faster than the standard code. We will also investigate how we can leverage known properties of mathematical functions, as well as programmer hints about useful properties of user-defined functions, and where reduced precision is acceptable.

6.4.3. Identifying divergence in GPU architectures

Participant: Sylvain Collange.

This research is done in collaboration with Fernando M. Q. Pereira, Diogo Sampaio and Rafael Martins de Souza, UFMG, Brazil.

GPU architectures rely on SIMD execution by vectorizing across SPMD threads. They achieve the best performance when consecutive threads take the same paths through conditional branches and access contiguous memory locations. Thus, many GPU code optimizations that target the control flow or memory access patterns necessitate accurate information about which branches and memory accesses are divergent across threads.

To enable such optimizations, we proposed divergence analysis, a compiler pass that identifies similarities in the control flow and data flow of concurrent threads [37]. This static analysis identifies program variables that are affine functions of the thread identifier and propagate this knowledge to conditional branches and memory accesses. Our analysis consistently outperforms other comparable analyses, thanks to the combination of taking into account affine relations between variables and accurately modeling control dependencies.

6.4.4. Code Obfuscation

Participant: Erven Rohou.

This research is done in collaboration with the group of Prof. Ahmed El-Mahdy at E-JUST, Alexandria, Egypt.

We proposed to leverage JIT compilation to make software tamper-proof. The idea is to constantly generate different versions of an application, even while it runs, to make reverse engineering hopeless. More precisely a JIT engine is used to generate new versions of a function each time it is invoked, applying different optimizations, heuristics and parameters to generate diverse binary code. A strong random number generator will guarantee that generated code is not reproducible, though the functionality is the same [38].

On-Stack-Replacement has been previously proposed to recompile functions while they run. However, it relies on compiler-generated switch points. We proposed a new technique to recompile functions at arbitraty points, thus reinforcing the Obfuscating JIT approach. A prototype is being developed [27].

A new obfuscation technique based of decomposition of CFGs into threads has been proposed. We exploit the mainstream multi-core processing in these systems to substantially increase the complexity of programs, making reverse engineering more complicated. The novel method automatically partitions any serial thread into an arbitrary number of parallel threads, at the basic-block level. The method generates new control-flow graphs, preserving the blocks' serial successor relations and guaranteeing that one basic-block is active at a time through using guards. The method generates m^n different combinations for m threads and n basic-blocks, significantly complicating the execution state. We also provide proof of correctness for the method.

6.4.5. Padrone

Participants: Erven Rohou, Alain Ketterlin, Emmanuel Riou.

The objective of the ADT PADRONE is to design and develop a platform for re-optimization of binary executables at run-time. Development is ongoing, and an early prototype is functional. In [24], we described the infrastructure of Padrone, and showed that its profiling overhead is minimum. We illustrated its use through two examples. The first example shows how a user can easily write a tool to identify hotspots in their application, and how well they perform (for example, by computing the number of executed instructions per cycle). In the second example, we illustrate the replacement of a given function (typically a hotspot) by an optimized version, while the program runs.

We believe PADRONE fills an empty design point in the ecosystem of dynamic binary tools.

6.4.6. Dynamic Analysis and Re-Optimization

Participants: Erven Rohou, Emmanuel Riou, Nabil Hallou, Alain Ketterlin.

This work is done in collaboration with Philippe Clauss (Inria CAMUS).

Dynamic binary analysis and re-optimization is specially interesting for legacy or commercial applications, but also in the context of cloud deployment, where actual hardware is unknown, and other applications competing for hardware resources can vary.

Initial results show that we are able to identify function hotspots that contain vectorized code for the Intel SSE extension, analyze them, and reoptimize the loops to target the latest and more powerful AVX ISA extension.

6.4.7. Branch Prediction and Performance of Interpreter

Participants: Erven Rohou, André Seznec, Bharath Narasimha Swamy.

Interpreters have been used in many contexts. They provide portability and ease of development at the expense of performance. The literature of the past decade covers analysis of why interpreters are slow, and many software techniques to improve them. A large proportion of these works focuses on the dispatch loop, and in particular on the implementation of the switch statement: typically an indirect branch instruction. Conventional wisdom attributes a significant penalty to this branch, due to its high misprediction rate. We revisit this assumption [36], considering current interpreters, and modern predictors. Using both hardware counters and simulation, we show that the accuracy of indirect branch prediction is no longer critical for interpreters. We also compare the characteristics of these interpreters and analyze why the indirect branch is less important than before.

6.5. WCET estimation

Participants: Damien Hardy, Benjamin Lesage, Hanbing Li, Isabelle Puaut, Erven Rohou, André Seznec.

Predicting the amount of resources required by embedded software is of prime importance for verifying that the system will fulfill its real-time and resource constraints. A particularly important point in hard real-time embedded systems is to predict the Worst-Case Execution Times (WCETs) of tasks, so that it can be proven that tasks temporal constraints (typically, deadlines) will be met. Our research concerns methods for obtaining automatically upper bounds of the execution times of applications on a given hardware. Our new results this year are on (i) multi-core architectures (ii) WCET estimation for faulty architectures (iii) traceability of flow information in compilers for WCET estimation.

6.5.1. WCET estimation and multi-core systems

6.5.1.1. Predictable shared caches for mixed-criticality real-time systems

Participants: Benjamin Lesage, Isabelle Puaut, André Seznec.

The general adoption of multi-core architectures has raised new opportunities as well as new issues in all application domains. In the context of real-time applications, it has created one major opportunity and one major difficulty. On the one hand, the availability of multiple high performance cores has created the opportunity to mix on the same hardware platform the execution of a complex critical real-time workload and the execution of non-critical applications. On the other hand, for real-time tasks timing deadlines must be met and enforced. Hardware resource sharing inherent to multicores hinders the timing analysis of concurrent tasks. Two different objectives are then pursued: enforcing timing deadlines for real-time tasks and achieving highest possible performance for the non-critical workload.

In this work, we suggest a hybrid hardware-based cache partitioning scheme that aims at achieving these two objectives at the same time. Plainly considering inter-task conflicts on shared cache for real-time tasks yields very pessimistic timing estimates. We remove this pessimism by reserving private cache space for real-time tasks. Upon the creation of a real-time task, our scheme reserves a fixed number of cache lines per set for the task. Therefore uniprocessor worst case execution time (WCET) estimation techniques can be used, resulting in tight WCET estimates. Upon the termination of the real-time task, this private cache space is released and made available for all the executed threads including non-critical ones. That is, apart the private spaces reserved for the real-time tasks currently running, the cache space is shared by all tasks running on the processor, i.e. non-critical tasks but also the real-time tasks for their least recently used blocks. Experiments show that the proposed cache scheme allows to both guarantee the schedulability of a set of real-time tasks with tight timing constraints and enable high performance on the non-critical tasks.

This work is the main contribution of the PhD thesis of Benjamin Lesage [12].

6.5.1.2. WCET estimation for massively parallel processor arrays

Participant: Isabelle Puaut.

This is joint work with Dumitru Potop-Butucaru, Inria, EPI AOSTE.

Classical timing analysis techniques for parallel code isolates micro-architecture analysis from the analysis of synchronizations between cores by performing them in two separate analysis phases (WCET – worst-case execution time – and WCRT – worst-case response time analyses). This isolation has its advantages, such as a reduction of the complexity of each analysis phase, and a separation of concerns that facilitates the development of analysis tools. But isolation also has a major drawback: a loss in precision which can be significant. To consider only one aspect, to be safe the WCET analysis of each synchronization-free sequential code region has to consider an undetermined micro-architecture state. This may result in overestimated WCETs, and consequently on pessimistic execution time bounds for the whole parallel application. The contribution of this work [33], [23] is an *integrated* WCET analysis approach that considers at the same time micro-architectural information and the synchronizations between cores. This is achieved by extending a state-of-the-art WCET estimation technique and tool to manage synchronizations and communications between the sequential threads running on the different cores. The benefits of the proposed method are twofold. On the one hand, the micro-architectural state is not lost between synchronization-free code regions running on the same core, which results in tighter execution time estimates. On the other hand, only one tool is required for the temporal validation of the parallel application, which reduces the complexity of the timing validation toolchain.

Such a holistic approach is made possible by the use of deterministic and composable software and hardware architectures (homogeneous multi-cores without cache sharing, static assignment of the code regions on the cores). We demonstrate the interest of the approach using an adaptive differential pulse-code modulation (*adpcm*) encoder where the integrated WCET approach provides significantly tighter response time estimations than the more classical WCRT approaches, with a gain of 21% on average.

6.5.2. WCET estimation for architectures with faulty caches

Participants: Damien Hardy, Isabelle Puaut.

Semiconductor technology evolution suggests that permanent failure rates will increase dramatically with scaling, in particular for SRAM cells. While well known approaches such as error correcting codes exist to recover from failures and provide fault-free chips, they will not be affordable anymore in the future due to their non-scalable cost. Consequently, other approaches like fine grain disabling and reconfiguration of hardware elements (e.g. individual functional units or cache blocks) will become economically necessary. This fine-grain disabling will lead to degraded performance compared to a fault-free execution.

A common implicit assumption in all static worst-case execution time (WCET) estimation methods is that the hardware is not subject to faults. Their result is not safe anymore when using fine grain disabling of hardware components, which degrades performance.

In [21] a method that statically calculates a probabilistic WCET bound in the presence of permanent faults in instruction caches is provided. The method, from a given program, cache configuration and probability of cell failure, derives a probabilistic WCET bound. The proposed method, because it relies on static analysis, is guaranteed to identify the longest program path, its probabilistic nature only stemming from the presence of faults. The method is computationally tractable because it does not require an exhaustive enumeration of the possible locations of faulty cache blocks. Experimental results show that it provides WCET estimates very close to, but never below, the method that derives probabilistic WCETs by enumerating all possible locations of faulty cache blocks. The proposed method not only allows to quantify the impact of permanent faults on WCET estimates, but also can be used in architectural exploration frameworks to select the most appropriate fault management mechanisms.

6.5.3. Traceability of flow information for WCET estimation

Participants: Hanbing Li, Isabelle Puaut, Erven Rohou.

This research is part of the ANR W-SEPT project.

Control-flow information is mandatory for WCET estimation, to guarantee that programs terminate (e.g. provision of bounds for the number of loop iterations) but also to obtain tight estimates (e.g. identification of infeasible or mutually exclusive paths). Such flow information is expressed though annotations, that may be calculated automatically by program/model analysis, or provided manually.

The objective of this work is to address the challenging issue of the mapping and transformation of the flow information from high level down to machine code. In a first step, we have considered the issue of conveying information through the compilation flow, without any optimization. We have created our own WCET information type and used the annotation files FFX (Flow Fact in XML, provided by IRIT, partner of the W-SEPT project), and applied them to the LLVM compiler framework. We are currently studying the impact of optimizations on the traceability of annotations. We are currently designing a framework for flow fact transformation for a large panel of compiler optimizations.

6.6. HPC and mobile computing

Participant: François Bodin.

We have initiated a research action on the interaction between mobile computing and HPC. We aim at studying data representation linked to parallel programming in heterogeneous systems. In particular, we want to explore energy tradeoffs when changing hardware resources from a light mobile platform to remote execution in a datacenter.

As a test case, we are developing an application for inventorying art pieces in the public domain. This is done in collaboration with University of Rennes 2. This test case is a pluridisplinary collaboration whose goal for University of Rennes 2 is to study how mobile computing can contribute to art studies and dissemination.

6.7. Application-specific number systems

Participant: Sylvain Collange.

This research is done in collaboration with Mark G. Arnold, XLNS Research, USA.

Reconfigurable FPGA platforms let designers build efficient application-specific circuits, when the performance or energy efficiency of general-purpose CPUs is insufficient, and the production volume is not enough to offset the very high cost of building a dedicated integrated circuit (ASIC). One way to take advantage of the flexibility offered by FPGAs is to tailor arithmetic operators for the application. In particular, the Logarithmic Number System (LNS) is suitable for embedded applications dealing with low-precision, high-dynamic range numbers.

Like floating-point, LNS can represent numbers from a wide dynamic range with constant relative accuracy. However, while standard floating-point offer so-called subnormal numbers to represent numbers close to zero with constant absolute accuracy, LNS numbers abruptly overflow to zero, resulting in a gap in representable numbers close to zero that can impact the accuracy of numerical algorithms.

We proposed a generalization of LNS that incorporate features analogous to subnormal floating-point [18], [28]. The Denormal LNS (DLNS) system we introduce defines a class of hybrid number systems that offer quasi-constant absolute accuracy close to zero and quasi-constant relative accuracy on larger numbers. These systems can be configured to range from pure LNS (constant relative accuracy) to fixed-point (constant absolute accuracy across the whole range).

7. Bilateral Contracts and Grants with Industry

7.1. Intel Research Grant

Participant: André Seznec.

Intel is supporting the research of the ALF project-team on "Alternative ways for improving uniprocessor performance".

8. Partnerships and Cooperations

8.1. International Initiatives

8.1.1. Participation In International Programs

8.1.1.1. Imhotep (Egypt)

Program: PHC

Title: Code obfuscation through JIT compilation

Inria principal investigator: Erven ROHOU

International Partner (Institution - Laboratory - Researcher):

Egypt-Japan University for Science and Technology (Egypt)

Duration: Jan 2013 - Dec 2013

This project leverages JIT compilation to make software tamper-proof. The idea is to constantly generate different versions of an application, even while it runs, to make reverse engineering much more complex. A strong random number generator guarantees that generated code is not reproducible – though the semantics is the same. In the course of the project, we also studied new forms of On-Stack-Replacement that let us recompile code even from the middle of a function. Finally, we studied how threads can be exploited to generate new forms of obfuscation, leveraging the fact that parallelism is error-prone, and difficult to debug and reverse-engineer.

8.1.2. Informal International Partners

The ALF team has informal collaborations with several international teams: Carnegie Mellon (Pr Mutlu), Georgia Tech (Pr Qureshi), University of Wisconsin (Pr Wood), University of Cyprus (Pr Sazeides), University of Ghent (Dr Eyerman), XLNS Research (Dr Arnold), UFMG Brazil (Pr Pereira), Barcelona Supercomputing center (Pr Cazorla and Pr Abella),

8.2. National Initiatives

8.2.1. Inria Project Lab: Multicore

Participants: Erven Rohou, Alain Ketterlin, Nabil Hallou.

The Inria Project Lab (formerly *Action d'Envergure*) started in 2013. It is entitled "Large scale multicore virtualization for performance scaling and portability". Partner project-teams include: ALF, ALGORILLE, CAMUS, REGAL, RUNTIME, as well as DALI. This project aims to build collaborative virtualization mechanisms that achieve essential tasks related to parallel execution and data management. We want to unify the analysis and transformation processes of programs and accompanying data into one unique virtual machine.

8.2.2. ADT IPBS 2013-2015

Participants: Sylvain Collange, Erven Rohou, André Seznec, Thibault Person.

As multi-core CPUs and parallel accelerators become pervasive, all execution platforms are now parallel. Research on architecture, compilers and systems now focuses on parallel platforms. New contributions need to be validated against parallel applications that are expected to be representative of current or future workloads. The research community relies today on a few benchmarks sets (SPLASH, PARSEC, ..) Existing parallel benchmarks are scarce, and some of them have issues such as aging workloads or non-representative input sets. The IPBS initiative aims at leveraging the diversity of parallel applications developed within Inria to provide a set of benchmarks, named the Inria Parallel Benchmark Suite, to the research community.

8.2.3. ADT Padrone 2012–2014

Participants: Erven Rohou, Alain Ketterlin, Emmanuel Riou.

Computer science is driven by two major trends: on the one hand, the lifetime of applications is much larger than the lifetime of the hardware for which they are initially designed; on the other hand the diversity of computing hardware keeps increasing. The net result is that many applications are not optimized for their current executing environment. The objective of Padrone is to design and develop a platform for reoptimization of binary executables at run-time. There are many advantages: actual hardware is known, the whole application is visible (including libraries), profiling can be collected, and source code is not necessary (interesting in the case of proprietary applications).

8.2.4. ANR W-SEPT

Participants: Hanbing Li, Isabelle Puaut, Erven Rohou.

Critical embedded systems are generally composed of repetitive tasks that must meet drastic timing constraints, such as termination deadlines. Providing an upper bound of the worst-case execution time (WCET) of such tasks at design time is thus necessary to prove the correctness of the system. Static WCET estimation methods, although safe, may produce largely over-estimated values. The objective of the project is to produce tighter WCET estimates by discovering and transforming flow information at all levels of the software design process, from high level-design models (e.g. Scade, Simulink) down to binary code. The ANR W-SEPT project partners are Verimag Grenoble, IRIT Toulouse, Inria Rennes. A case study is provided by Continental Toulouse.

8.3. European Initiatives

8.3.1. FP7 Projects

8.3.1.1. DAL: ERC AdG 2010- 267175, 04-2011/03-2016

Type: IDEAS

Instrument: ERC Advanced Grant

Duration: April 2011 - March 2016

Coordinator: André Seznec

Inria contact: André Seznec

Abstract: In the DAL, Defying Amdahl's Law project, we envision that, around 2020, the processor chips will feature a few complex cores and many (may be 1000s) simpler, more silicon and power effective cores. In the DAL research project, we will explore the microarchitecture techniques that will be needed to enable high performance on such heterogeneous processor chips. Very high performance will be required on both sequential sections —legacy sequential codes, sequential sections of parallel applications— and critical threads on parallel applications —e.g. the main thread controlling the application. Our research will focus on enhancing single process performance. On the microarchitecture side, we will explore both a radically new approach, the sequential accelerator, and more conventional processor architectures. We will also study how to exploit heterogeneous multicore architectures to enhance sequential thread performance.
For more information, see http://www.irisa.fr/alf/index.php?option=com_content&view=article&id=55&Itemid=3&lang=e

8.3.2. Collaborations in European Programs, except FP7

8.3.2.1. HiPEAC3 NoE

Participants: François Bodin, Pierre Michaud, Erven Rohou, André Seznec.

F. Bodin, P. Michaud, A. Seznec and E. Rohou are members of the European Network of Excellence HiPEAC3. HiPEAC3 addresses the design and implementation of high-performance commodity computing devices in the 10+ year horizon, covering both the processor design, the optimizing compiler infrastructure, and the evaluation of upcoming applications made possible by the increased computing power of future devices.

8.3.2.2. COST Action TACLe - Timing Analysis on Code-Level (http://www.tacle.eu) 10-2012/09-2015
Participants: Damien Hardy, Isabelle Puaut.

Embedded systems increasingly permeate our daily lives. Many of those systems are business- or safety-critical, with strict timing requirements. Code-level timing analysis (used to analyze software running on some given hardware w.r.t. its timing properties) is an indispensable technique for ascertaining whether or not these requirements are met. However, recent developments in hardware, especially multi-core processors, and in software organization render analysis increasingly more difficult, thus challenging the evolution of timing analysis techniques.

New principles for building "timing-composable" embedded systems are needed in order to make timing analysis tractable in the future. This requires improved contacts within the timing analysis community, as well as with related communities dealing with other forms of analysis such as model-checking and type-inference, and with computer architectures and compilers. The goal of this COST Action is to gather these forces in order to develop industrial-strength code-level timing analysis techniques for future-generation embedded systems, through several working groups:

- WG1 Timing models for multi-cores and timing composability
- WG2 Tooling aspects
- WG3 Early-stage timing analysis
- WG4 Resources other than time

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Pr Ahmed El-Mahdy, from the Egyptian-Japanese University of Science and Technology visited the ALF project for 1 week in October 2013.
- Pr Onur Mutlu, from Carnegie Mellon visited the ALF project for 3 weeks June-July 2013.

9. Dissemination

9.1. Scientific Animation

9.1.1. Service to the research community

- Erven Rohou was a member of the program committees of DITAM-PARMA 2014, CC 2014, WPEA 2013.
- Erven Rohou served as an expert for “Région Aquitaine”
- Isabelle Puaut is a member of the program committees of ECRTS 2013, RTNS 2013, RTCSA 2013, RTAS 2014, SIES 2014.
- Isabelle Puaut is member of the Executive Committee (EC) of the IEEE Technical Committee on Real-Time Systems (TCRTS). She is in the steering committee of the ECRTS, RTNS conferences and the WCET workshop.
- Isabelle Puaut is in the management committee of the COST Action TACLe - Timing Analysis on Code-Level (http://www.tacle.eu). She is responsible of Short Term Scientific Missions (STSM) within TACLe. Damien Hardy and Isabelle Puaut participate to TACLe.
- Damien Hardy is a member of the committees of RTNS 2014 and WCET 2014. He was a member of the program committee of WCET 2013, SIES 2013 WIP session and PACT 2013 where he was also the submission chair.
- Pierre Michaud was a member of the program committee of the HPCC 2013 conference.

- André Seznec is a member of the MICRO 2014 top picks committee and a member of SAMOS 2014 program committee.
- André Seznec is a member of the editorial board of the IEEE Micro.
- André Seznec was the Program co-chair of HiPEAC 2013, January 2013
- André Seznec and François Bodin were Program co-chairs of PACT 2013, September 2013.
- François Bodin was a member of ASPLOS 2014, CC 2013, SC 2013 tutorials program committees.
- François Bodin is a member of "Comité de Prospective Scientifique" of the ANR.
- François Bodin is a member of "Conseil Scientifique d'Orap".

9.1.2. Dissemination

- Erven Rohou presented the ANR project W-SEPT at the bi-annual meeting of the "Communauté Française de Compilation".
- Emmanuel Riou and Nabil Hallou presented the Padrone tool at the HiPEAC Computing Systems Week.
- I. Puaut has presented a seminar on "WCET estimation for multi-core architectures" at LIP6, Paris, in September 2013.
- Damien Hardy, has presented a lesson on "Estimation de pires temps d'exècution (WCET - Worst-Case Execution Times)" at the "école d'été temps-réel" Toulouse, August 2013
- André Seznec presented a keynote entitled "Faster unicores are still needed" at SAMOS XIII in Samos, Greece, July 2013.
- André Seznec presented an invited presentation at Intel Braunschweig in January 2013.
- François Bodin presented invited presentations at the EPOPPEA workshop associated with the HIPEAC 2013 conference, to the CSCI (Comité Stratégique pour le Calcul Intensif), at the HTPC workshop at University of Delaware and at Forum ORAP.
- François Bodin presented a keynote at the HPC languages workshop in Lyon, July 2013
- François presented a lesson at EU ComplexHPC Spring School in Uppsala, June 2013.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Master research : A. Seznec, E.Rohou, I. Puaut, F. Bodin, Performance et Microarchitecture, 30 hours, M2, Université de Rennes I, France

Master: A. Seznec, P. Michaud, A. Perais, Architecture des processeurs, 36 hours, M1, Ecole Supérieure d'Ingénieurs de Rennes, France

Master: A. Seznec, P. Michaud, A. Perais, Architecture avancée, 36 hours, M2, Ecole Supérieure d'Ingénieurs de Rennes, France

Master research: I. Puaut, E. Rohou, Rédaction d'articles scientifiques, 28 hours, M2, Université de Rennes I, France

Master research: I. Puaut, Analyse et test formel, 6 hours, M2, Université de Bretagne Occidentale, France

Master: I. Puaut, D. Hardy, Operating systems - process management, 130 hours, M1, Université de Rennes I, France

Master: I. Puaut, Système d'exploitation gestion mémoire, 39 hours, M1, Université de Rennes I, France

Master: I. Puaut, D. Hardy, Systèmes temps-réel, 69 hours, M1, Université de Rennes I, France

Master: F. Bodin, Parallel programming and code optimization, 50 hours, M1, Ecole Supérieure d'Ingénieurs de Rennes, France

Master: F. Bodin, Innovation and technology, 20 hours, M1, Ecole Supérieure d'Ingénieurs de Rennes, France

Master: F. Bodin, Innovation and technology, 40 hours, M1, Université de Rennes I, France

Master: D. Hardy, Systèmes d'exploitation, 44 hours, M1, Université de Rennes I, France

Licence: D. Hardy, Informatique temps-réel, 40 hours, L3, Université de Rennes I, France

9.2.2. Supervision

PhD : J. Lai, Modèle analytique de performance orienté débit d'évaluation de performance des accélérateurs programmables, Université de Rennes I, February 2013. Advisor A. Seznec

PhD : R. Velasquez, Behavioral Application-dependent Superscalar Core Modeling, Université Rennes 1, April 2013. Co-advisors A. Seznec and P. Michaud

PhD : B. Lesage, Architecture multi-coeurs et temps d'exécution au pire cas, Université Rennes 1, May 2013. Co-advisors I. Puaut and A. Seznec

PhD : N. Prémillieu, Améliorer la performance séquentielle à l'ère des processeurs massivement multicoeurs, Université Rennes 1, December 2013. Advisor A. Seznec

PhD in progress: Nabil Hallou, Université Rennes 1, Feb 2013, co-advisors E. Rohou and P. Clauss (EPI Camus Inria Strasbourg)

PhD in progress: Sajith Kalathingal, Université Rennes 1, Dec 2012, co-advisors S. Collange and A. Seznec

PhD in progress: Surya Khizakanchery Natarajan, Université Rennes 1, Jan 2012, advisor A. Seznec

PhD in progress: Hanbing Li, Université Rennes 1, Oct 2012, co-advisors E. Rohou and I. Puaut

PhD in progress: Andrea Mondelli, Université Rennes 1, Oct 2013, co-advisors P. Michaud and A. Seznec

PhD in progress: Bharath Narasimha Swamy, Université Rennes 1, Sept 2011, advisor A. Seznec

PhD in progress: Arthur Perais, Université Rennes 1, Sept 2012, advisor A. Seznec

PhD in progress: Aswinkumar Sridharan, Université Rennes 1, Oct 2013, advisor A. Seznec

PhD in progress: Arjun Suresh , Université Rennes 1, Dec 2012, co-advisors E. Rohou and A. Seznec

9.3. Popularization

- Erven Rohou gave a talk at the SFGP (Société Française du Génie des Procédés): "Stratégies d'augmentation des performances de calcul des logiciels"
- Isabelle Puaut and Erven Rohou gave a lecture at Lycée Descartes: "Les mathématiques au service de la performance des ordinateurs".

9.4. Miscelleanous

- Erven Rohou co-advised a MSc. student at the Egypt-Japan University of Science and Technology.
- Erven Rohou was a member of the working group GTInria2020 whose mission was to produce the next "Plan Stratégique".
- Erven Rohou is a member of the Inria CDT (Commission du Développement Technologique)
- As "correspondant scientifique des relations internationales" for Inria Rennes Bretagne Atlantique, Erven Rohou is a member of the Inria COST GTRI (Groupe de Travail "Relations Internationales" du Comité d'Orientation Scientifique et Technologique).
- Erven Rohou served as an expert for "Région Aquitaine"
- A. Seznec is an elected member of the scientific committee of Inria.

- A. Seznec has been nominated by ACM for 3 years 2011-2013 on the selection committee for the ACM-IEEE Eckert-Mauchly award.
- F. Bodin has participated to the Allistene committee on "Préparation de la Stratégie Nationale de Recherche pour le Numérique".
- F. Bodin is a member of the Advisory board of the LPGPU European Project.

10. Bibliography

Major publications by the team in recent years

[1] M. CORNERO, R. COSTA, R. FERNÁNDEZ PASCUAL, A. ORNSTEIN, E. ROHOU. *An Experimental Environment Validating the Suitability of CLI as an Effective Deployment Format for Embedded Systems*, in "Conference on HiPEAC", Göteborg, Sweden, P. STENSTRÖM, M. DUBOIS, M. KATEVENIS, R. GUPTA, T. UNGERER (editors), Springer, January 2008, pp. 130–144

[2] R. COSTA, E. ROHOU. *Comparing the size of .NET applications with native code*, in "3rd Intl Conference on Hardware/software codesign and system synthesis", Jersey City, NJ, USA, P. ELES, A. JANTSCH, R. A. BERGAMASCHI (editors), ACM, September 2005, pp. 99–104

[3] D. HARDY, I. PUAUT. *WCET analysis of multi-level non-inclusive set-associative instruction caches*, in "Proc. of the 29th IEEE Real-Time Systems Symposium", Barcelona, Spain, December 2008

[4] T. LAFAGE, A. SEZNEC. *Choosing Representative Slices of Program Execution for Microarchitecture Simulations: A Preliminary Application to the Data Stream*, in "Workload Characterization of Emerging Applications", Kluwer Academic Publishers, 2000, pp. 145–163

[5] P. MICHAUD, Y. SAZEIDES, A. SEZNEC, T. CONSTANTINOU, D. FETIS. *A study of thread migration in temperature-constrained multi-cores*, in "ACM Transactions on Architecture and Code Optimization", 2007, vol. 4, n^{o} 2, 9 p.

[6] P. MICHAUD, A. SEZNEC, S. JOURDAN. *An Exploration of Instruction Fetch Requirement in Out-of-Order Superscalar Processors*, in "International Journal of Parallel Programming", 2001, vol. 29, n^{o} 1, pp. 35-58

[7] E. ROHOU, M. SMITH. *Dynamically managing processor temperature and power*, in "Second Workshop on Feedback-Directed Optimizations", 1999

[8] A. SEZNEC, P. MICHAUD. *A case for (partially)-tagged geometric history length predictors*, in "Journal of Instruction Level Parallelism (http://www.jilp.org/vol8)", April 2006, http://www.jilp.org/vol8

[9] A. SEZNEC, N. SENDRIER. *HAVEGE: a user-level software heuristic for generating empirically strong random numbers*, in "ACM Transactions on Modeling and Computer Systems", October 2003

[10] A. SEZNEC. *Analysis of the O-GEHL branch predictor*, in "Proceedings of the 32nd Annual International Symposium on Computer Architecture", June 2005

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] J. LAI. , *Modèle analytique de performance orienté débit d'évaluation de performance des accélérateurs programmables*, Université de Rennes I, February 2013, http://hal.inria.fr/tel-00908579

[12] B. LESAGE. , *Architecture multi-coeurs et temps d'exécution au pire cas*, Université Rennes 1, May 2013, http://hal.inria.fr/tel-00870971

[13] N. PRÉMILLIEU. , *Améliorer la performance séquentielle à l'ère des processeurs massivement multicœurs*, Université Rennes 1, December 2013, http://hal.inria.fr/tel-00916589

[14] R. A. VELÁSQUEZ VÉLEZ. , *Behavioral Application-dependent Superscalar Core Modeling*, Université Rennes 1, April 2013, http://hal.inria.fr/tel-00908544

Articles in International Peer-Reviewed Journals

[15] E. ROHOU, K. WILLIAMS, D. YUSTE. *Vectorization Technology To Improve Interpreter Performance*, in "ACM Transactions on Architecture and Code Optimization", January 2013, vol. 9, n^{o} 4, 26:1 p. [*DOI :* 10.1145/2400682.2400685], http://hal.inria.fr/hal-00747072

[16] R. A. VELASQUEZ, P. MICHAUD, A. SEZNEC. *BADCO: Behavioral Application-Dependent Superscalar Core Models*, in "International Journal of Parallel Programming", October 2013 [*DOI :* 10.1007/s10766-013-0278-1], http://hal.inria.fr/hal-00907659

Articles in National Peer-Reviewed Journals

[17] N. BRUNIE, S. COLLANGE. *Reconvergence de contrôle implicite pour les architectures SIMT*, in "Technique et Science Informatiques (TSI)", February 2013, vol. 32, n^{o} 2, pp. 153-178 [*DOI :* 10.3166/TSI.32.153-178], http://hal.inria.fr/hal-00787749

International Conferences with Proceedings

[18] M. ARNOLD, S. COLLANGE. *The Denormal Logarithmic Number System*, in "ASAP 2013 - 24th IEEE International Conference on Application-specific Systems, Architectures and Processors", Washington D.C., United States, June 2013, pp. 117-124, http://hal.inria.fr/hal-00832505

[19] F. BODIN, D. ROMAIN, G. COLIN DE VERDIERE. *One OpenCL to Rule Them All?*, in "International Workshop on Multi-/Many-core Computing Systems
", Edinburgh, United Kingdom, September 2013, http://hal.inria.fr/hal-00920910

[20] D. HARDY, M. KLEANTHOUS, I. SIDERIS, A. SAIDI, E. OZER, Y. SAZEIDES. *An Analytical Framework for Estimating TCO and Exploring Data Center Design Space*, in "IEEE International Symposium on Performance Analysis of Systems and Software", Austin, United States, April 2013, http://hal.inria.fr/hal-00914593

[21] D. HARDY, I. PUAUT. *Static probabilistic Worst Case Execution Time Estimation for architectures with Faulty Instruction Caches*, in "21st International Conference on Real-Time Networks and Systems", Sophia Antipolis, France, October 2013 [*DOI :* 10.1145/2516821.2516842], http://hal.inria.fr/hal-00862604

[22] J. LAI, A. SEZNEC. *Performance Upper Bound Analysis and Optimization of SGEMM on Fermi and Kepler GPUs*, in "CGO '13 - 2013 International Symposium on Code Generation and Optimization", Shenzhen, China, February 2013, http://hal.inria.fr/hal-00789958

[23] D. POTOP-BUTUCARU, I. PUAUT. *Integrated Worst-Case Execution Time Estimation of Multicore Applications*, in "13th International Workshop on Worst-Case Execution Time Analysis", Paris, France, C. MAIZA (editor), Schloss Dagstuh, July 2013, vol. 30, pp. 21-31 [*DOI :* 10.4230/OASICS.WCET.2013.1], http://hal.inria.fr/hal-00909330

[24] E. RIOU, E. ROHOU, P. CLAUSS, N. HALLOU, A. KETTERLIN. *PADRONE: a Platform for Online Profiling, Analysis, and Optimization*, in "DCE 2014 - International workshop on Dynamic Compilation Everywhere", Vienne, Austria, January 2014, http://hal.inria.fr/hal-00917950

[25] M. SOLINAS, R. BADIA, F. BODIN, A. COHEN, P. EVRIPIDOU, P. FARABOSCHI, B. FECHNER, G. R. GAO, A. GARBADE, S. GIRBAL, D. GOODMAN, B. KHAN, S. KOLIAI, F. LI, M. LUJÁN, L. MORIN, A. MENDELSON, N. NAVARRO, A. POP, P. TRANCOSO, T. UNGERER, M. VALERO, S. WEIS, I. WATSON, S. ZUCKERMAN, R. GIORGI. *The TERAFLUX Project: Exploiting the DataFlow Paradigm in Next Generation Teradevices*, in "DSD", Los Alamitos, United States, 2013, pp. 272-279, http://hal.inria.fr/hal-00920903

[26] R. A. VELASQUEZ, P. MICHAUD, A. SEZNEC. *Selecting Benchmark Combinations for the Evaluation of Multicore Throughput*, in "International Symposium on Performance Analysis of Systems and Software", Austin, United States, February 2013, http://hal.inria.fr/hal-00788824

[27] M. YUSUF, A. EL-MAHDY, E. ROHOU. *On-Stack Replacement to Improve JIT-based Obfuscation - A Preliminary Study*, in "International Japan-Egypt Conference on Electronics, Communications, and Computers", Cairo, Egypt, December 2013, http://hal.inria.fr/hal-00909722

Research Reports

[28] M. ARNOLD, S. COLLANGE. , *Options for Denormal Representation in Logarithmic Arithmetic*, Inria, January 2014, n° RR-8412, 27 p. , http://hal.inria.fr/hal-00909096

[29] S. EYERMAN, P. MICHAUD. , *Defining metrics for multicore throughput on multiprogrammed workloads*, Inria, November 2013, n° RR-8401, http://hal.inria.fr/hal-00908864

[30] S. N. NATARAJAN, B. SWAMY, A. SEZNEC. , *Modeling multi-threaded programs execution time in the many-core era*, Inria, December 2013, n° RR-8453, 23 p. , http://hal.inria.fr/hal-00914335

[31] A. PERAIS, A. SEZNEC. , *EOLE: Paving the Way for an Effective Implementation of Value Prediction*, Inria, November 2013, n° RR-8402, 25 p. , http://hal.inria.fr/hal-00907973

[32] A. PERAIS, A. SEZNEC. , *Practical Data Value Speculation for Future High-end Processors*, Inria, November 2013, n° RR-8395, 21 p. , http://hal.inria.fr/hal-00904743

[33] D. POTOP-BUTUCARU, I. PUAUT. , *Integrated Worst-Case Response Time Evaluation of Multicore Non-Preemptive Applications*, Inria, February 2013, n° RR-8234, http://hal.inria.fr/hal-00787931

[34] N. PRÉMILLIEU, A. SEZNEC. , *Efficient Out-of-Order Execution of Guarded ISAs*, Inria, November 2013, n° RR-8406, 24 p. , http://hal.inria.fr/hal-00910335

[35] N. PRÉMILLIEU, A. SEZNEC. , *SPREPI: Selective Prediction and REplay for predicated Instructions*, Inria, August 2013, n° RR-8351, 25 p. , http://hal.inria.fr/hal-00856160

[36] E. ROHOU, B. NARASIMHA SWAMY, A. SEZNEC. , *Branch Prediction and the Performance of Interpreters - Don't Trust Folklore*, Inria, November 2013, n° RR-8405, 23 p. , http://hal.inria.fr/hal-00911146

[37] D. SAMPAIO, R. DE SOUZA, S. COLLANGE, F. MAGNO QUINTÃO PEREIRA. , *Divergence Analysis*, Inria, November 2013, n° RR-8411, 42 p. , http://hal.inria.fr/hal-00909072

Other Publications

[38] M. HATABA, A. EL-MAHDY, A. SHOUKRY, E. ROHOU. , *OJIT: A Novel Secure Remote Execution Technology By Obfuscated Just-In-Time Compilation*, April 2013, The Third European LLVM Conference, http://hal.inria.fr/hal-00909766

References in notes

[39] G. M. AMDAHL. *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*, in "SJCC.", 1967, pp. 483–485

[40] D. BURGER, T. M. AUSTIN. , *The simplescalar tool set, version 2.0*, 1997

[41] R. S. CHAPPELL, J. STARK, S. P. KIM, S. K. REINHARDT, Y. N. PATT. *Simultaneous subordinate microthreading (SSMT)*, in "ISCA '99: Proceedings of the 26th annual international symposium on Computer architecture", Washington, DC, USA, IEEE Computer Society, 1999, pp. 186–195, http://doi.acm.org/10.1145/300979.300995

[42] C. FERDINAND, R. WILHELM. *Efficient and Precise Cache Behavior Prediction for Real-Time Systems*, in "Real-Time Syst.", 1999, vol. 17, n° 2-3, pp. 131–181, http://dx.doi.org/10.1023/A:1008186323068

[43] T. S. KARKHANIS, J. E. SMITH. *A First-Order Superscalar Processor Model*, in "Proceedings of the International Symposium on Computer Architecture", Los Alamitos, CA, USA, IEEE Computer Society, 2004, 338 p. , http://doi.ieeecomputersociety.org/10.1109/ISCA.2004.1310786

[44] B. LEE, J. COLLINS, H. WANG, D. BROOKS. *CPR : composable performance regression for scalable multiprocessor models*, in "Proceedings of the 41st International Symposium on Microarchitecture", 2008

[45] Y. LIANG, T. MITRA. *Cache modeling in probabilistic execution time analysis*, in "DAC '08: Proceedings of the 45th annual conference on Design automation", New York, NY, USA, ACM, 2008, pp. 319–324, http://doi.acm.org/10.1145/1391469.1391551

[46] T. LUNDQVIST, P. STENSTRÖM. *Timing Anomalies in Dynamically Scheduled Microprocessors*, in "RTSS '99: Proceedings of the 20th IEEE Real-Time Systems Symposium", Washington, DC, USA, IEEE Computer Society, 1999

[47] L. RAUCHWERGER, Y. ZHAN, J. TORRELLAS. *Hardware for Speculative Run-Time Parallelization in Distributed Shared-Memory Multiprocessors*, in "HPCA '98: Proceedings of the 4th International Symposium on High-Performance Computer Architecture", Washington, DC, USA, IEEE Computer Society, 1998, 162 p.

[48] T. SHERWOOD, E. PERELMAN, G. HAMERLY, B. CALDER. *Automatically characterizing large scale program behavior*, in "In Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems", 2002, pp. 45–57

[49] K. SKADRON, M. STAN, W. HUANG, S. VELUSAMY. *Temperature-aware microarchitecture*, in "Proceedings of the International Symposium on Computer Architecture", 2003

[50] J. G. STEFFAN, C. COLOHAN, A. ZHAI, T. C. MOWRY. *The STAMPede approach to thread-level speculation*, in "ACM Trans. Comput. Syst.", 2005, vol. 23, n^{o} 3, pp. 253–300, http://doi.acm.org/10.1145/1082469.1082471

[51] V. SUHENDRA, T. MITRA. *Exploring locking & partitioning for predictable shared caches on multi-cores*, in "DAC '08: Proceedings of the 45th annual conference on Design automation", New York, NY, USA, ACM, 2008, pp. 300–303, http://doi.acm.org/10.1145/1391469.1391545

[52] D. M. TULLSEN, S. EGGERS, H. M. LEVY. *Simultaneous Multithreading: Maximizing On-Chip Parallelism*, in "Proceedings of the 22th Annual International Symposium on Computer Architecture", 1995

[53] J. YAN, W. ZHAN. *WCET Analysis for Multi-Core Processors with Shared L2 Instruction Caches*, in "Proceedings of Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08.", 2008, pp. 80-89