Activity Report 2016

Team STORM

STatic Optimizations, Runtime Methods

Inria teams are typically groups of researchers working on the definition of a common project, and objectives, with the goal to arrive at the creation of a project-team. Such project-teams may include other partners (universities or research institutions).

RESEARCH CENTER
Bordeaux - Sud-Ouest

THEME
Distributed and High Performance Computing

Table of contents

Team STORM

Creation of the Team: 2015 January 01

Keywords:

Computer Science and Digital Science:

2.1.6. - Concurrent programming
2.1.10. - Domain-specific languages
2.2.1. - Static analysis
2.2.3. - Run-time systems
2.2.4. - Parallel architectures
2.2.5. - GPGPU, FPGA, etc.
6.2.6. - Optimization
6.2.7. - High performance computing

Other Research Topics and Application Domains:

3.3.1. - Earth and subsoil
5.2.3. - Aviation

1. Members

Research Scientist

Olivier Aumage [Inria, Researcher]

Faculty Members

Denis Barthou [Team leader, Bordeaux INP, Professor, HDR]
Marie Christine Counilh [Univ. Bordeaux, Associate Professor]
Raymond Namyst [Univ. Bordeaux, Professor, HDR]
Samuel Thibault [Univ. Bordeaux, Associate Professor]
Pierre Andre Wacrenier [Univ. Bordeaux, Associate Professor]

Engineers

Adrien Cassagne [Inria]
Jerome Clet-Ortega [Inria]
Nathalie Furmento [CNRS]
Samuel Pitoiset [Inria]
Chiheb Sakka [Inriafrom Jul 2016]
Corentin Salingue [Inria]

PhD Students

Hugo Brunie [CEA]
Terry Cojean [Inria]
Christopher Haine [Inria]
Pierre Huchant [Univ. Bordeaux]
Suraj Kumar [Inria]
Raphaël Prat [CEA, from Oct 2016]
Arthur Loussert [CEA, from Oct 2016]
Marc Sergent [Univ. Bordeaux, until Dec 2016]
Gregory Vaumourin [CEA, until Oct 2016]

Post-Doctoral Fellow

Luka Stanisic [Inria]

Administrative Assistant

Sylvie Embolla [Inria]

Others

Thomas Caroff [Inria, Intern, from Jun 2016 until Jul 2016]
Arthur Chevalier [Inria, Intern, from May 2016 until Jul 2016]
Berenice Faltrept [Inria, Intern, from May 2016 until Jul 2016]
Loris Lucido [Inria, Intern, from Jun 2016 until Aug 2016]
Berangere Subervie [Inria, Intern, from Jun 2016 until Jul 2016]

2. Overall Objectives

2.1. Overall Objectives

A successful approach to deal with the complexity of modern architectures is centered around the use of runtime systems, to manage tasks dynamically, these runtime systems being either generic or specific to an application. Similarly, on the compiler side, optimizations and analyses are more aggressive in iterative compilation frameworks, fit for library generations, or DSL, in particular for linear algebra methods. To go beyond this state of the art and alleviate the difficulties for programming these machines, we believe it is necessary to provide inputs with richer semantics to runtime and compiler alike, and in particular by combining both approaches.

This general objective is declined into two sub-objectives, the first concerning the expression of parallelism itself, the second the optimization and adaptation of this parallelism by compilers and runtimes.

- Expressing parallelism: As shown in the following figure, we propose to work on parallelism expression through Domain Specific Languages, able to capture the essence of the algorithms used through usual parallel languages such as OpenCL, OpenMP and through high performance libraries. The DSLs will be driven by applications, with the idea to capture at the algorithmic level the parallelism of the problem and perform dynamic data layout adaptation, parallel and algorithmic optimizations. The principle here is to capture a higher level of semantics, enabling users to express not only parallelism but also different algorithms.
- Optimizing and adapting parallelism: The goal here is to leverage the necessary adaptation to evolving hardware, by providing mechanisms allowing users to run the same code on different architectures. This implies to adapt parallelism, in particular the granularity of the work, to the architecture. This relies on the use of existing parallel libraries and their composition, and more generally the separation of concern between the description of tasks, that represent semantic units of work, and the tasks to be executed by the different processing units. Splitting or coarsening moldable tasks, generating code for these tasks and scheduling them is part of this work.

  Finally, the abstraction we advocate for requires to propose a feed back loop. This feed back has two objectives: To make users better understand their application and how to change the expression of parallelism if necessary, but also to propose an abstracted model for the machine. This allows to develop and formalize the compiling, scheduling techniques on a model, not too far from the real machine. Here, simulation techniques are a way to abstract the complexity of the architecture while preserving essential metrics.

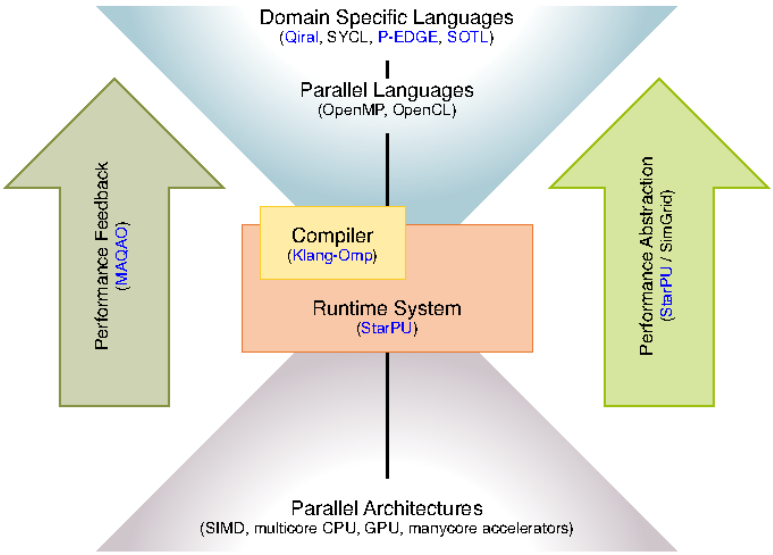

Figure 1. STORM Big Picture

3. Research Program

3.1. Parallel Computing and Architectures

Following the current trends of the evolution of HPC systems architectures, it is expected that future Exascale systems (i.e. Sustaining 10^{18} flops) will have millions of cores. Although the exact architectural details and trade-offs of such systems are still unclear, it is anticipated that an overall concurrency level of $O(10^9)$ threads/tasks will probably be required to feed all computing units while hiding memory latencies. It will obviously be a challenge for many applications to scale to that level, making the underlying system sound like "embarrassingly parallel hardware."

From the programming point of view, it becomes a matter of being able to expose extreme parallelism within applications to feed the underlying computing units. However, this increase in the number of cores also comes with architectural constraints that actual hardware evolution prefigures: computing units will feature extra-wide SIMD and SIMT units that will require aggressive code vectorization or "SIMDization", systems will become hybrid by mixing traditional CPUs and accelerators units, possibly on the same chip as the AMD APU solution, the amount of memory per computing unit is constantly decreasing, new levels of memory will appear, with explicit or implicit consistency management, etc. As a result, upcoming extreme-scale system will not only require unprecedented amount of parallelism to be efficiently exploited, but they will also require that applications generate adaptive parallelism capable to map tasks over heterogeneous computing units.

The current situation is already alarming, since European HPC end-users are forced to invest in a difficult and time-consuming process of tuning and optimizing their applications to reach most of current supercomputers' performance. It will go even worse at horizon 2020 with the emergence of new parallel architectures (tightly integrated accelerators and cores, high vectorization capabilities, etc.) featuring unprecedented degree of parallelism that only too few experts will be able to exploit efficiently. As highlighted by the ETP4HPC initiative, existing programming models and tools won't be able to cope with such a level of heterogeneity, complexity and number of computing units, which may prevent many new application opportunities and new science advances to emerge.

The same conclusion arises from a non-HPC perspective, for single node embedded parallel architectures, combining heterogeneous multicores, such as the ARM big.LITTLE processor and accelerators such as GPUs or DSPs. The need and difficulty to write programs able to run on various parallel heterogeneous architectures has led to initiatives such as HSA, focusing on making it easier to program heterogeneous computing devices. The growing complexity of hardware is a limiting factor to the emergence of new usages relying on new technology.

3.2. Scientific and Societal Stakes

In the HPC context, simulation is already considered as a third pillar of science with experiments and theory. Additional computing power means more scientific results, and the possibility to open new fields of simulation requiring more performance, such as multi-scale, multi-physics simulations. Many scientific domains able to take advantage of Exascale computers, these "Grand Challenges" cover large panels of science, from seismic, climate, molecular dynamics, theoretical and astrophysics physics... Besides, embedded applications are also able to take advantage of these performance increase. There is still an on-going trend where dedicated hardware is progressively replaced by off-the-shelf components, adding more adaptability and lowering the cost of devices. For instance, Error Correcting Codes in cell phones are still hardware chips, but with the forthcoming 5G protocol, new software and adaptative solutions relying on low power multicores are also explored. New usages are also appearing, relying on the fact that large computing capacities are becoming more affordable and widespread. This is the case for instance with Deep Neural Networks where the training phase can be done on supercomputers and then used in embedded mobile systems. The same consideration applies for big data problems, of internet of things, where small sensors provide large amount of data that need to be processed in short amount of time. Even though the computing capacities required for such applications are in general a different scale from HPC infrastructures, there is still a need in the future for high performance computing applications.

However, the outcome of new scientific results and the development of new usages for mobile, embedded systems will be hindered by the complexity and high level of expertise required to tap the performance offered by future parallel heterogeneous architectures.

3.3. Towards More Abstraction

As emphasized by initiatives such as the European Exascale Software Initiative (EESI), the European Technology Platform for High Performance Computing (ETP4HPC), or the International Exascale Software Initiative (IESP), the HPC community needs new programming APIs and languages for expressing heterogeneous massive parallelism in a way that provides an abstraction of the system architecture and promotes high performance and efficiency. The same conclusion holds for mobile, embedded applications that require performance on heterogeneous systems.

This crucial challenge given by the evolution of parallel architectures therefore comes from this need to make high performance accessible to the largest number of developpers, abstracting away architectural details providing some kind of performance portability. Disruptive uses of the new technology and groundbreaking new scientific results will not come from code optimization or task scheduling, but they require the design of new algorithms that require the technology to be tamed in order to reach unprecedented levels of performance.

Runtime systems and numerical libraries are part of the answer, since they may be seen as building blocks optimized by experts and used as-is by application developers. The first purpose of runtime systems is indeed to provide *abstraction*. Runtime systems offer a uniform programming interface for a specific subset of hardware (e.g., OpenGL or DirectX are well-established examples of runtime systems dedicated to hardware-accelerated graphics) or low-level software entities (e.g., POSIX-thread implementations). They are designed as thin user-level software layers that complement the basic, general purpose functions provided by the operating system calls. Applications then target these uniform programming interfaces in a portable manner. Low-level, hardware dependent details are hidden inside runtime systems. The adaptation of runtime systems is commonly handled through drivers. The abstraction provided by runtime systems thus enables portability. Abstraction alone is however not enough to provide portability of performance, as it does nothing to leverage low-level-specific features to get increased performance. Consequently, the second role of runtime systems is to *optimize* abstract application requests by dynamically mapping them onto low-level requests and resources as efficiently as possible. This mapping process makes use of scheduling algorithms and heuristics to decide the best actions to take for a given metric and the application state at a given point in its execution time. This allows applications to readily benefit from available underlying low-level capabilities to their full extent without breaking their portability. Thus, optimization together with abstraction allows runtime systems to offer portability of performance. Numerical libraries provide sets of highly optimized kernels for a given field (dense or sparse linear algebra, FFT, etc.) either in an autonomous fashion or using an underlying runtime system.

Application domains cannot resort to libraries for all codes however, computation patterns such as stencils are a representative example of such difficulty. The compiler technology plays here a central role, in managing high level semantics, either through templates, domain specific languages or annotations. Compiler optimizations, and the same applies for runtime optimizations, are limited by the level of semantics they manage. Providing part of the algorithmic knowledge of an application, for instance knowing that it computes a 5-point stencil and then performs a dot product, would lead to more opportunities to adapt parallelism, memory structures, and is a way to leverage the evolving hardware.

Compilers and runtime play a crucial role in the future of high performance applications, by defining the input language for users, and optimizing/transforming it into high performance code. The objective of STORM is to propose better interactions between compiler and runtime and more semantics for both approaches. We recall in the following section the expertise of the team.

4. Application Domains

4.1. Application Fields

The application of our work concerns linear algebra, solvers and fast-multipole methods, in collaboration with other Inria teams and with industry. This allows a wide range of scientific and industrial applications possibly interested in the techniques we propose, in the domain of high performance computing but also in order to compute intensive embedded applications. In terms of direct application, the software developed in the team are used in applications in various fields, ranging from seismic, mechanic of fluids, molecular dynamics, high energy physics or material simulations. Similarly, the domains of image processing and signal processing can take advantage of the expertise and software of the team.

5. New Software and Platforms

5.1. Chameleon

KEYWORDS: HPC - Dense linear algebra - Task-based algorithm - Runtime system - Task scheduling
SCIENTIFIC DESCRIPTION

Chameleon is part of the MORSE (Matrices Over Runtime Systems @ Exascale) project. The overall objective is to develop robust linear algebra libraries relying on innovative runtime systems that can fully benefit from the potential of those future large-scale complex machines.

We expect advances in three directions based first on strong and closed interactions between the runtime and numerical linear algebra communities. This initial activity will then naturally expand to more focused but still joint research in both fields.

1. Fine interaction between linear algebra and runtime systems. On parallel machines, HPC applications need to take care of data movement and consistency, which can be either explicitly managed at the level of the application itself or delegated to a runtime system. We adopt the latter approach in order to better keep up with hardware trends whose complexity is growing exponentially. One major task in this project is to define a proper interface between HPC applications and runtime systems in order to maximize productivity and expressivity. As mentioned in the next section, a widely used approach consists in abstracting the application as a DAG that the runtime system is in charge of scheduling. Scheduling such a DAG over a set of heterogeneous processing units introduces a lot of new challenges, such as predicting accurately the execution time of each type of task over each kind of unit, minimizing data transfers between memory banks, performing data prefetching, etc. Expected advances: In a nutshell, a new runtime system API will be designed to allow applications to provide scheduling hints to the runtime system and to get real-time feedback about the consequences of scheduling decisions.

2. Runtime systems. A runtime environment is an intermediate layer between the system and the application. It provides low-level functionality not provided by the system (such as scheduling or management of the heterogeneity) and high-level features (such as performance portability). In the framework of this proposal, we will work on the scalability of runtime environment. To achieve scalability it is required to avoid all centralization. Here, the main problem is the scheduling of the tasks. In many task-based runtime environments the scheduler is centralized and becomes a bottleneck as soon as too many cores are involved. It is therefore required to distribute the scheduling decision or to compute a data distribution that impose the mapping of task using, for instance the so-called “owner-compute” rule. Expected advances: We will design runtime systems that enable an efficient and scalable use of thousands of distributed multicore nodes enhanced with accelerators.

3. Linear algebra. Because of its central position in HPC and of the well understood structure of its algorithms, dense linear algebra has often pioneered new challenges that HPC had to face. Again, dense linear algebra has been in the vanguard of the new era of petascale computing with the design of new algorithms that can efficiently run on a multicore node with GPU accelerators. These algorithms are called "communication-avoiding" since they have been redesigned to limit the amount of communication between processing units (and between the different levels of memory hierarchy). They are expressed through Direct Acyclic Graphs (DAG) of fine-grained tasks that are dynamically scheduled. Expected advances: First, we plan to investigate the impact of these principles in the case of sparse applications (whose algorithms are slightly more complicated but often rely on dense kernels). Furthermore, both in the dense and sparse cases, the scalability on thousands of nodes is still limited, new numerical approaches need to be found. We will specifically design sparse hybrid direct/iterative methods that represent a promising approach.

The overall goal of the MORSE associate team is to enable advanced numerical algorithms to be executed on a scalable unified runtime system for exploiting the full potential of future exascale machines.
FUNCTIONAL DESCRIPTION

Chameleon is a dense linear algebra software relying on sequential task-based algorithms where sub-tasks of the overall algorithms are submitted to a Runtime system. A Runtime system such as StarPU is able to manage automatically data transfers between not shared memory area (CPUs-GPUs, distributed nodes). This kind of implementation paradigm allows to design high performing linear algebra algorithms on very different type of architecture: laptop, many-core nodes, CPUs-GPUs, multiple nodes. For example, Chameleon is able to perform a Cholesky factorization (double-precision) at 80 TFlop/s on a dense matrix of order 400 000 (e.i. 4 min).

- Participants: Emmanuel Agullo, Mathieu Faverge, Cédric Castagnede and Florent Pruvost
- Partners: Innovative Computing Laboratory (ICL) - King Abdullha University of Science and Technology - University of Colorado Denver
- Contact: Emmanuel Agullo
- URL: https://project.inria.fr/chameleon/

5.2. KLANG-OMP

The KStar OpenMP Compiler
KEYWORDS: Compilers - Task scheduling - OpenMP - Source-to-source compiler - Data parallelism
FUNCTIONAL DESCRIPTION

The Klang-Omp compiler, now renamed KStar following the recommendation of the local experimentation and development service, is a source-to-source OpenMP compiler for languages C and C++. The KStar compiler translates OpenMP directives and constructs into API calls from the StarPU runtime system or the XKaapi runtime system. The KStar compiler is virtually fully compliant with OpenMP 3.0 constructs. The KStar compiler supports OpenMP 4.0 dependent tasks and accelerated targets.

- Participants: Olivier Aumage, Nathalie Furmento, Samuel Pitoiset and Samuel Thibault
- Contact: Olivier Aumage
- URL: http://kstar.gforge.inria.fr/#!index.md

5.3. KaStORS

The KaStORS OpenMP Benchmark Suite
KEYWORDS: Benchmarking - HPC - Task-based algorithm - Task scheduling - OpenMP - Data parallelism
FUNCTIONAL DESCRIPTION

The KaStORS benchmarks suite has been designed to evaluate implementations of the OpenMP dependent task paradigm, introduced as part of the OpenMP 4.0 specification.

- Participants: Olivier Aumage, François Broquedis, Pierrick Brunet, Nathalie Furmento, Thierry Gautier, Samuel Thibault and Philippe Virouleau
- Contact: Thierry Gautier
- URL: http://kastors.gforge.inria.fr/#!index.md

5.4. MORSE

- Contact: Emmanuel Agullo
- URL: http://icl.cs.utk.edu/morse/

5.5. AFF3CT

A Fast Forward Error Correction Tool (previously named P-Edge).

KEYWORDS: Code generation - Error Correction Code
FUNCTIONAL DESCRIPTION

The AFF3CT library joins genericity techniques together with code generation capabilities to enable implementing efficient and portable error correction codes. The genericity offered allows to easily experiment with a large panel of algorithmic variants.

- Previous name: P-Edge
- Authors: Adrien Cassagne, Olivier Aumage, Bertrand Le Gal, Camille Leroux and Denis Barthou
- Partner: IMS
- Contact: Adrien Cassagne
- URL: https://aff3ct.github.io/

5.6. StarPU

The StarPU Runtime System
KEYWORDS: HPC - Scheduling - GPU - Multicore - Performance
SCIENTIFIC DESCRIPTION

Traditional processors have reached architectural limits which heterogeneous multicore designs and hardware specialization (eg. coprocessors, accelerators, ...) intend to address. However, exploiting such machines introduces numerous challenging issues at all levels, ranging from programming models and compilers to the design of scalable hardware solutions. The design of efficient runtime systems for these architectures is a critical issue. StarPU typically makes it much easier for high performance libraries or compiler environments to exploit heterogeneous multicore machines possibly equipped with GPGPUs or Cell processors: rather than handling low-level issues, programmers may concentrate on algorithmic concerns.Portability is obtained by the means of a unified abstraction of the machine. StarPU offers a unified offloadable task abstraction named "codelet". Rather than rewriting the entire code, programmers can encapsulate existing functions within codelets. In case a codelet may run on heterogeneous architectures, it is possible to specify one function for each architectures (eg. one function for CUDA and one function for CPUs). StarPU takes care to schedule and execute those codelets as efficiently as possible over the entire machine. In order to relieve programmers from the burden of explicit data transfers, a high-level data management library enforces memory coherency over the machine: before a codelet starts (eg. on an accelerator), all its data are transparently made available on the compute resource.Given its expressive interface and portable scheduling policies, StarPU obtains portable performances by efficiently (and easily) using all computing resources at the same time. StarPU also takes advantage of the heterogeneous nature of a machine, for instance by using scheduling strategies based on auto-tuned performance models.

StarPU is a task programming library for hybrid architectures

The application provides algorithms and constraints: - CPU/GPU implementations of tasks - A graph of tasks, using either the StarPU's high level GCC plugin pragmas or StarPU's rich C API

StarPU handles run-time concerns - Task dependencies - Optimized heterogeneous scheduling - Optimized data transfers and replication between main memory and discrete memories - Optimized cluster communications

Rather than handling low-level scheduling and optimizing issues, programmers can concentrate on algorithmic concerns!

FUNCTIONAL DESCRIPTION

StarPU is a runtime system that offers support for heterogeneous multicore machines. While many efforts are devoted to design efficient computation kernels for those architectures (e.g. to implement BLAS kernels on GPUs), StarPU not only takes care of offloading such kernels (and implementing data coherency across the machine), but it also makes sure the kernels are executed as efficiently as possible.

- Participants: Cédric Augonnet, Samuel Thibault, Nathalie Furmento, Simon Archipoff, Jérôme Clet-Ortega, Nicolas Collin, Ludovic Courtes, Mehdi Juhoor, Xavier Lacoste, Benoît Lize, Ludovic Stordeur, Cyril Roelandt, Corentin Salingue, Chiheb Sakka, Samuel Pitoiset, François Tessier, Pierre-André Wacrenier, Andra Hugo, Terry Cojean, Raymond Namyst, Olivier Aumage and Marc Sergent
- Contact: Olivier Aumage
- URL: http://starpu.gforge.inria.fr/

5.7. hwloc

Hardware Locality

KEYWORDS: HPC - Topology - Open MPI - Affinities - GPU - Multicore - NUMA - Locality

FUNCTIONAL DESCRIPTION

Hardware Locality (hwloc) is a library and set of tools aiming at discovering and exposing the topology of machines, including processors, cores, threads, shared caches, NUMA memory nodes and I/O devices. It builds a widely-portable abstraction of these resources and exposes it to applications so as to help them adapt their behavior to the hardware characteristics. They may consult the hierarchy of resources, their attributes, and bind task or memory on them.

hwloc targets many types of high-performance computing applications, from thread scheduling to placement of MPI processes. Most existing MPI implementations, several resource managers and task schedulers, and multiple other parallel libraries already use hwloc.

- Participants: Brice Goglin and Samuel Thibault
- Partners: AMD - Intel - Open MPI consortium
- Contact: Brice Goglin
- URL: http://www.open-mpi.org/projects/hwloc/

6. New Results

6.1. Automatic OpenCL Task Adaptation for Heterogeneous Architectures

OpenCL defines a common parallel programming language for all devices, although writing tasks adapted to the devices, managing communication and load-balancing issues are left to the programmer. In this work [11], we propose a novel automatic compiler and runtime technique to execute single OpenCL kernels on heterogeneous multi-device architectures. Our technique splits computation and data automatically across the computing devices. The technique proposed is completely transparent to the user, does not require off-line training or a performance model. It handles communications and load-balancing issues, resulting from hardware heterogeneity, load imbalance within the kernel itself and load variations between repeated executions of the kernel, in an iterative computation. We present our results on benchmarks and on an N-body application over two platforms, a 12-core CPU with two different GPUs and a 16-core CPU with three homogeneous GPUs.

6.2. Fast Forward Error Correction Codes

Erroc Correction Codes are essential for preserving data integrity in communications. These algorithms find errors due to noise in transmissions and correct these errors with a high probability. Several algorithms are used, with different capacities in term of correction and most of them are implemented in cell phones or satellites as ASICS. The need to handle many different usages, different contexts of use pushes towards software solutions. A larger spectrum of algorithms can be explored, in order to meet the expectations in terms of performance, power consumption and error correcting power. These new algorithms, for the 5G for instance, can then be either implemented in software (for large antenna for instance) or in hardware. In both case, software simulation is necessary in order to evaluate the properties of the new algorithms. We developped in collaboration with IMS new versions of algorithms and a new software, AFF3CT http://aff3ct.github.io/index.html, that allows the exploration of many different algorithmic variants and their evaluation. Two conference papers have been published on these new results [7][6].

6.3. Resource aggregation for task-based Cholesky Factorization

Hybrid computing platforms are now commonplace, featuring a large number of CPU cores and accelerators. This trend makes balancing computations between these heterogeneous resources performance critical. In a recent paper [8] we propose aggregating several CPU cores in order to execute larger parallel tasks and thus improve the load balance between CPUs and accelerators. Additionally, we present our approach to exploit internal parallelism within tasks. This is done by combining two runtime systems: one runtime system to handle the task graph and another one to manage the internal parallelism. We demonstrate the relevance of our approach in the context of the dense Cholesky factorization kernel implemented on top of the StarPU task-based runtime system.We present experimental results showing that our solution outperforms state of the art implementations. In addition, we realized an extended version of this paper submitted for review to the Parallel Computing journal special issue for HCW and HeteroPar 2016 workshops. In this new paper [19] we provide additional details on our contribution and propose a brand new study on the recent Intel Xeon Phi Knights Landing (KNL) where we show that we are able to outperform existing state of the art implementations on this platform thanks to our proposed technique.

6.4. Scheduling of Linear Algebra Kernels on Multiple Heterogeneous Resources

In this paper [5], we consider task-based dense linear algebra applications on a single heterogeneous node which contains regular CPU cores and a set of GPU devices. Efficient scheduling strategies are crucial in this context in order to achieve good and portable performance. HeteroPrio, a resource-centric dynamic scheduling strategy has been introduced in a previous work and evaluated for the special case of nodes with exactly two

types of resources. However, this restriction can be limiting, for example on nodes with several types of accelerators, but not only this. Indeed, an interesting approach to increase resource usage is to group several CPU cores together, which allows to use intra-task parallelism. We propose a generalization of HeteroPrio to the case with several classes of heterogeneous workers. We provide extensive evaluation of this algorithm with Cholesky factorization, both through simulation and actual execution, compared with HEFT-based scheduling strategy, the state of the art dynamic scheduling strategy for heterogeneous systems. Experimental evaluation shows that our approach is efficient even for highly heterogeneous configurations and significantly outperforms HEFT-based strategy.

6.5. Analyzing Dynamic Task-Based Applications on Hybrid Platforms: An Agile Scripting Approach

In this paper [10], we present visual analysis techniques to evaluate the performance of HPC task-based applications on hybrid architectures. Our approach is based on composing modern data analysis tools (pjdump, R, ggplot2, plotly), enabling an agile and flexible scripting framework with minor development cost. We validate our proposal by analyzing traces from the full-fledged implementation of the Cholesky decomposition available in the MORSE library running on a hybrid (CPU/GPU) platform. The analysis compares two different workloads and three different task schedulers from the StarPU runtime system. Our analysis based on composite views allows to identify allocation mistakes, priority problems in scheduling decisions, GPU tasks anomalies causing bad performance, and critical path issues.

6.6. Distributed StarPU Scalability on Heterogeneous Platforms

The emergence of accelerators as standard computing resources on supercomput- ers and the subsequent architectural complexity increase revived the need for high-level parallel programming paradigms. Sequential task-based programming model has been shown to efficiently meet this challenge on a single multicore node possibly enhanced with accelerators, which moti- vated its support in the OpenMP 4.0 standard. In this paper, we show that this paradigm can also be employed to achieve high performance on modern supercomputers composed of multiple such nodes, with extremely limited changes in the user code. To prove this claim, we have extended the StarPU runtime system with an advanced inter-node data management layer that supports this model by posting communications automatically [16]. We illustrate our discussion with the task- based tile Cholesky algorithm that we implemented on top of this new runtime system layer. We show that it allows for very high productivity while achieving a performance competitive with both the pure Message Passing Interface (MPI)-based ScaLAPACK Cholesky reference implementation and the DPLASMA Cholesky code, which implements another (non sequential) task-based programming paradigm.

6.7. Controlling the Memory Subscription of Distributed Applications with a Task-Based Runtime System

The ever-increasing supercomputer architectural complexity emphasizes the need for high-level parallel programming paradigms. Among such paradigms, task-based programming manages to abstract away much of the archi- tecture complexity while efficiently meeting the performance challenge, even at large scale. Dynamic run-time systems are typically used to execute task-based applications, to schedule computation resource usage and memory allocations. While computation scheduling has been well studied, the dynamic management of memory resource subscription inside such run- times has however been little explored. This paper [12] studies the cooperation between a task-based distributed application code and a run-time system engine to control the memory subscription levels throughout the execution. We show that the task paradigm allows to control the memory footprint of the application by throttling the task submission flow rate, striking a compromise between the performance benefits of anticipative task submission and the resulting memory consumption. We illustrate the benefits of our contribution on a compressed dense linear algebra distributed application.

6.8. StarPU Interfacing with GASPI/GPI2

A version of the distributed dependence support of StarPU has been ported by Corentin Salingue, under the supervision of Olivier Aumage on the high performance GASPI/GPI2 networking layer developed by the Fraunhofer institute in Germany. The GPI2 framework offers a lightweight communication interface specifically designed for thread enabled HPC applications. This work has been conducted as part of the H2020 INTERTWinE european project.

6.9. A Stencil DSEL for Single Code Accelerated Computing with SYCL

Stencil kernels arise in many scientific codes as the result from dis- cretizing natural, continuous phenomenons. Many research works have designed stencil frameworks to help programmer optimize stencil kernels for performance, and to target CPUs or accelerators. However, existing stencil kernels, either library-based or language-based necessitate to write distinct source codes for accelerated ker- nels and for the core application, or to resort to specific keywords, pragmas or language extensions. SYCL is a C++ based approach designed by the Khronos Group to program the core application as well as the application kernels with a single unified, C++ compliant source code. A SYCL application can then be linked with a CPU-only runtime library or processed by a SYCL-enabled compiler to automatically build an OpenCL accelerated application. Our contribution [13] is a stencil domain specific embedded language (DSEL) which leverage SYCL together with expression template techniques to implement statically optimized stencil applications able to run on platforms equipped with OpenCL devices, while preserving the single source benefits from SYCL.

6.10. Bridging the gap between OpenMP 4.0 and native runtime systems for the fast multipole method

With the advent of complex modern architectures, the low-level paradigms long considered sufficient to build High Performance Computing (HPC) numerical codes have met their limits. Achieving efficiency, ensuring portability, while preserving programming tractability on such hardware prompted the HPC community to design new, higher level paradigms. The successful ports of fully-featured numerical libraries on several recent runtime system proposals have shown, indeed, the benefit of task-based parallelism models in terms of performance portability on complex platforms. However, the common weakness of these projects is to deeply tie applications to specific expert-only runtime system APIs. The OpenMP specification, which aims at providing a common parallel programming means for shared-memory platforms, appears as a good candidate to address this issue thanks to the latest task-based constructs introduced as part of its revision 4.0. The goal of this paper [15] is to assess the effectiveness and limits of this support for designing a high-performance numerical library. We illustrate our discussion with the ScalFMM library, which implements state-of-the-art fast multipole method (FMM) algorithms, that we have deeply re-designed with respect to the most advanced features provided by OpenMP 4. We show that OpenMP 4 allows for significant performance improvements over previous OpenMP revisions on recent multicore processors. We furthermore propose extensions to the OpenMP 4 standard and show how they can enhance FMM performance. To assess our statement, we have implemented this support within the Klang-OMP source-to-source compiler that translates OpenMP directives into calls to the StarPU task-based runtime system. This study shows that we can take advantage of the advanced capabilities of a fully-featured runtime system without resorting to a specific, native runtime port, hence bridging the gap between the OpenMP standard and the very high performance that was so far reserved to expert-only runtime system APIs.

6.11. Hierarchical Tasks

Modern computing platforms are heterogeneous and the load balancing is more complex to reach high performance. We decided to deal with the granularity problem in the context of task paralleliism and in a dynamic way through the implementation of hierarchical tasks in StarPU runtime. The idea is to give the runtime the ability to control tasks submission in order to choose the good granularity at the right moment.

The application describes a control graph and the runtime generates the computation tasks graph on-the-fly according to the state of the machine (available computing resources, memory consumption, ...). As a consequence the runtime is able to limit the size of the computation tasks graph without loosing parallelism. Some experiments have been done on a Cholesky application and in the qr-mumps software and show that the work of an application programmer can be alleviated and the granularity choice could be easily delegated to the task based runtime.

6.12. Software-Hardware Exploration for Read-Only Data

We have proposed a new way of managing the cache by exploiting the difference of behavior in the memory system between read-only data and read-write data. A division of the existing cache-based memory hierarchy is proposed in order to create a dedicated data path for read-only data. This proposition is similar to the existing separation at the L1-level between instruction and data caches. In order to justify this approach, an analysis performed on a set of benchmarks shows that read-only data count for significant part of the working set and are less reused than read-write data. A transparent solution is proposed based on specific compilation support to separate automatically the memory accesses of read-only data at L1-level. This organization exploits the properties of the different sub- workloads in order to increase the overall data locality and data reuse. Simulated in a multicore environment, the evaluation of the new memory organization shows reduction of L1 misses up to 28.5%. Moreover, the messages issued on the interconnection network can be reduced up to 14.7% without any penalty on the performance.

Besides the reduced miss-rate allows maintaining performance with smaller cache size on the read-write path while the properties of the read- only part can benefit of a simplified cache implementation despite a shared multicore access [1].

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- HiBOX project, with Airbus and IMACS (2013-2017).
- CEA contracts:
  - Several PhD contracts: for Hugo Brunie, Raphaël Prat, Marc Sergent and Arthur Loussert.
  - Industrial contract with CEA-DAM on particle simulation.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. PIA

ELCI The ELCI project (Software Environment for HPC) aims to develop a new generation of software stack for supercomputers, numerical solvers, runtime and programming development environments for HPC simulation. The ELCI project also aims to validate this software stack by showing its capacity to offer improved scalability, resilience, security, modularity and abstraction on real applications. The coordinator is Bull, and the different partners are CEA, Inria, SAFRAN, CERFACS, CNRS CORIA, CENAERO, ONERA, UVSQ, Kitware and AlgoTech.

8.1.2. ANR

ANR SOLHAR (http://solhar.gforge.inria.fr/doku.php?id=start).

ANR MONU 2013 Program, 2013 - 2016 (36 months)

Identification: ANR-13-MONU-0007

Coordinator: Inria Bordeaux/LaBRI

Other partners: CNRS-IRIT, Inria-LIP Lyon, CEA/CESTA, EADS-IW

Abstract: This project aims at studying and designing algorithms and parallel programming models for implementing direct methods for the solution of sparse linear systems on emerging computers equipped with accelerators. The ultimate aim of this project is to achieve the implementation of a software package providing a solver based on direct methods for sparse linear systems of equations. Several attempts have been made to accomplish the porting of these methods on such architectures; the proposed approaches are mostly based on a simple offloading of some computational tasks (the coarsest grained ones) to the accelerators and rely on fine hand-tuning of the code and accurate performance modeling to achieve efficiency. This project proposes an innovative approach which relies on the efficiency and portability of runtime systems, such as the StarPU tool developed in the runtime team (Bordeaux). Although the SOLHAR project will focus on heterogeneous computers equipped with GPUs due to their wide availability and affordable cost, the research accomplished on algorithms, methods and programming models will be readily applicable to other accelerator devices such as ClearSpeed boards or Cell processors.

ANR Songs Simulation of next generation systems (http://infra-songs.gforge.inria.fr/).

ANR INFRA 2011, 01/2012 - 12/2015 (48 months)

Identification: ANR-11INFR01306

Coordinator: Martin Quinson (Inria Nancy)

Other partners: Inria Nancy, Inria Rhône-Alpes, IN2P3, LSIIT, Inria Rennes, I3S.

Abstract: The goal of the SONGS project is to extend the applicability of the SimGrid simulation framework from Grids and Peer-to-Peer systems to Clouds and High Performance Computation systems. Each type of large-scale computing system will be addressed through a set of use cases and lead by researchers recognized as experts in this area.

8.1.3. ADT - Inria Technological Development Actions

ADT K'Star (http://kstar.gforge.inria.fr/#!index.md)

Participants: Olivier Aumage, Nathalie Furmento, Samuel Pitoiset, Samuel Thibault.

Inria ADT Campaign 2013, 10/2013 - 9/2015 (24 months)

Coordinator: Thierry Gautier (team AVALON, Inria Grenoble - Rhône-Alpes) and Olivier Aumage (team RUNTIME, Inria Bordeaux - Sud-Ouest)

Abstract: The Inria action ADT K'Star is a joint effort from Inria teams AVALON and RUNTIME to design the Klang-Omp source-to-source OpenMP compiler to translate OpenMP directives into calls to the API of AVALON and RUNTIME respective runtime systems (XKaapi for AVALON, StarPU for RUNTIME).

8.1.4. IPL - Inria Project Lab

C2S@Exa - Computer and Computational Sciences at Exascale **Participant:** Olivier Aumage.

Inria IPL 2013 - 2017 (48 months)

Coordinator: Stéphane Lantéri (team Nachos, Inria Sophia)

Since January 2013, the team is participating to the C2S@Exa http://www-sop.inria.fr/c2s_at_exa Inria Project Lab (IPL). This national initiative aims at the development of numerical modeling methodologies that fully exploit the processing capabilities of modern massively parallel architectures in the context of a number of selected applications related to important scientific and technological challenges for the quality and the security of life in our society. This collaborative effort involves computer scientists that are experts of programming models, environments and tools for harnessing massively parallel systems, algorithmists that propose algorithms and contribute to generic libraries

and core solvers in order to take benefit from all the parallelism levels with the main goal of optimal scaling on very large numbers of computing entities and, numerical mathematicians that are studying numerical schemes and scalable solvers for systems of partial differential equations in view of the simulation of very large-scale problems.

HAC-SPECIS - High-performance Application and Computers, Studying PErformance and Correctness In Simulation

Participants: Samuel Thibault, Luka Stanisic.

Inria IPL 2016 - 2020 (48 months)

Coordinator: Arnaud Legrand (team Polaris, Inria Rhône Alpes)

Since June 2016, the team is participating to the HAC-SPECIS http://hacspecis.gforge.inria.fr/ Inria Project Lab (IPL). This national initiative aims at answering methodological needs of HPC application and runtime developers and allowing to study real HPC systems both from the correctness and performance point of view. To this end, it gathers experts from the HPC, formal verification and performance evaluation community.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. INTERTWinE

Title: Programming Model INTERoperability ToWards Exascale

Programm: H2020

Duration: October 2015 - October 2018

Coordinator: EPCC

Partners:

Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (Spain)

Deutsches Zentrum für Luft - und Raumfahrt Ev (Germany)

Fraunhofer Gesellschaft Zur Forderung Der Angewandten Forschung Ev (Germany)

Institut National de Recherche en Informatique et en Automatique (France)

Kungliga Tekniska Hoegskolan (Sweden)

T-Systems Solutions for Research (Germany)

The University of Edinburgh (United Kingdom)

Universitat Jaume I de Castellon (Spain)

The University of Manchester (United Kingdom)

Inria contact: Olivier Aumage

This project addresses the problem of programming model design and implementation for the Exascale. The first Exascale computers will be very highly parallel systems, consisting of a hierarchy of architectural levels. To program such systems effectively and portably, programming APIs with efficient and robust implementations must be ready in the appropriate timescale. A single, “silver bullet” API which addresses all the architectural levels does not exist and seems very unlikely to emerge soon enough. We must therefore expect that using combinations of different APIs at different system levels will be the only practical solution in the short to medium term. Although there remains room for improvement in individual programming models and their implementations, the main challenges lie in interoperability between APIs. It is this interoperability, both at the specification level and at the implementation level, which this project seeks to address and to further the state of the art. INTERTWinE brings together the principal European organisations driving the evolution of programming models and their implementations. The project will focus on seven key programming APIs: MPI, GASPI, OpenMP, OmpSs, StarPU, QUARK and PaRSEC, each of which has a project partner with extensive experience in API design and implementation. Interoperability requirements, and evaluation of implementations will be driven by a set of kernels and applications, each of which has a project partner with a major role in their development. The project will implement a co- design cycle, by feeding back advances in API design and implementation into the applications and kernels, thereby driving new requirements and hence further advances.

8.2.1.2. Mont-Blanc 2

Title: Programming Model INTERoperability ToWards Exascale

Programm: FP7

Duration: September 2013 - January 2017

Coordinator: BSC

Partners: Atos/Bull, ARM, Jülich, LRZ, Univ. Stuttgart, CINECA, CNRS, CEA, Univ. Bristol, Allinea Software, Univ. Cantabria

Inria contact: Olivier Aumage

The Mont-Blanc project aims to develop a European Exascale approach leveraging on commodity power-efficient embedded technologies. The project has developed a HPC system software stack on ARM, and will deploy the first integrated ARM-based HPC prototype by 2014, and is also working on a set of 11 scientific applications to be ported and tuned to the prototype system. Team STORM has been involved in porting the MAQAO binary code analyzer and instrumenter on ARM platforms and interfacing it with the kernel autotuning framework BOAST.

8.3. International Initiatives

8.3.1. Inria International Partners

8.3.1.1. Declared Inria International Partners

- Team STORM is supervising the membership of Inria as part of the OpenMP Architecture Review Board (ARB), the international body in charge of the standardisation of the OpenMP parallel programming language. The membership has been supported by an InriaHUB/Standardisation grant.
- Team STORM is member of the Khronos Group Advisory Panel about the standardization of the OpenCL and SYCL programming languages.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Selection

9.1.1.1. Chair of Conference Program Committees

- Samuel Thibault was a Program Committee chair for EuroPar'16.

9.1.1.2. Member of the Conference Program Committees

- Samuel Thibault was a member of the Program Committee for Compas'16, HCW'16, MuCoCos'16, P^3MA'16
- Olivier Aumage was a member of the Program Committee for HUCAA 16'
- Raymond Namyst was a member of the Program Committees for Cluster'16, EuroPar'16 and SAC/MUSEPAT'16
- Denis Barthou was a Program Committee chair for UCHPC'16

9.1.1.3. Reviewer

The members of the team reviewed numerous papers for various international conferences such as IPDPS, Super-Computing, Euro-Par, ICPP.

9.1.2. Journal

The members of the team review papers from many high-level journals such as TPDS, CCPE, TACO, JPDC.

9.1.3. Invited Talks

- Samuel Thibault was invited to present StarPU advances at the "Scalable Task-based Programming Models" workshop of SIAM-PP 2016
- Samuel Thibault was invited to participate to the "What Do You Need to Know About Task-Based Programming for Future Systems?" panel of SIAM-PP 2016
- Samuel Thibault was invited to make a talk on StarPU at an meeting for the H2020 NLAFET project
- Samuel Thibault was then invited to make a talk at the CCDSC-2016 workshop
- Samuel Thibault was invited to make a talk at Jussieu for a APR seminar
- Terry Cojean was invited to present his work at the "Task-based Scientific, High Performance Computing on Top of Runtime Systems" workshop of SIAM-PP 2016
- Terry Cojean was invited to present his work by the research group of Prof. Benkner at the University of Vienna.
- Terry Cojean was invited to give a talk on StarPU at the RESPA workshop of Super-Computing 2016, details available in [2]
- Luka Stanisic was invited to present an effective methodology for reproducible research on dynamic task-based runtime systems at the "Task-based Scientific, High Performance Computing on Top of Runtime Systems" workshop of SIAM-PP 2016
- Luka Stanisic was invited to present advanced usage of Git at the "Reproducible Research" webinars
- Olivier Aumage was invited to present StarPU at the Parallel Programming Frameworks: Technologies, Performance and Applications track of SIAM-PP 2016 in Paris.
- Olivier Aumage was invited to present StarPU at CERFACS in Toulouse.
- Olivier Aumage was invited to present StarPU at the workshop Building a European/American Community for the Development of Dynamic Runtimes in Extreme-Scale Systems, as part of ISC'2016 in Frankfurt.
- Olivier Aumage and Samuel Thibault presented a tutorial session on runtime systems and StarPU as part of the Prace Advanced Training Center (PATC) program in Paris, in partnership with La Maison de la simulation.
- Olivier Aumage was invited to present StarPU by the research group of Prof. Benkner at the university of Vienna.
- Olivier Aumage was invited to give a training session on advanced parallel programming models for HPC platforms as part of the EoCoE European Center of Excellence face-to-face meeting in Rome
- Raymond Namyst was invited to give a talk about Heterogeneous Programming at the RoMoL Workshop, Barcelona, March 2016
- Raymond Namyst was invited to give a talk about StarPU at the CEA 2016 HPC Worshop, Cargèse

9.2. Teaching - Supervision - Juries

9.2.1. Teaching administration

Samuel Thibault is responsible for the computer science topic of the first university semester.

Samuel Thibault is responsible for the creation of the new Licence Pro ADSILLH (Administrateur et Développeur de Systèmes Informatiques sous Licences Libres et Hybrides)

Denis Barthou is responsible for the cyber-security, systems and networks 3rd year of the ENSEIRB-MATMECA engineering school.

Raymond Namyst is Vice-chair of the Computer Science Training Department of University of Bordeaux

9.2.2. Teaching

Licence : Marie-Christine Counilh, Introduction to Computer Science, 64HeTD, L1, University of Bordeaux
Licence : Marie-Christine Counilh, Introduction to C Programming, 52HeTD, L1, University of Bordeaux
Licence : Samuel Thibault, Introduction to Computer Science, 32HeTD, L1, University of Bordeaux
Licence : Samuel Thibault, Networking, 51HeTD, L3, University of Bordeaux
Licence : Samuel Thibault, Computer Architecture, 77HeTD, L2, University of Bordeaux
Licence : Samuel Thibault, Tutored project, 10HeTD, L3, University of Bordeaux
Licence : Pierre-André Wacrenier, Introduction to Computer Science, 64HeTD, L1, University of Bordeaux
Licence : Terry Cojean, Networking, 40HeTD, L1, IUT Bordeaux
Licence : Terry Cojean, Object Oriented Programming, 24HeTD, L3, IUT Bordeaux
Master : Luka Stanisic, Operating Systems, 22HeTD, M1, University of Bordeaux
Master : Samuel Thibault, Operating Systems, 22HeTD, M1, University of Bordeaux
Master : Marie-Christine Counilh, Object Oriented Programming, 30HeTD, M1, University of Bordeaux
Master : Raymond Namyst, Operating Systems, M1, University of Bordeaux
Master : Pierre-André Wacrenier and Raymond Namyst, Parallel Programming, M1, University of Bordeaux
Engineering School: Samuel Thibault, Information System Security, 13HeTD, M1, ENSEIRB-MATMECA/IPB
Engineering School: Olivier Aumage, Languages and Supports for Parallelism, 14HeTD, M2, ENSEIRB-MATMECA/IPB joint with University of Bordeaux
Engineering School: Olivier Aumage, High Performance Communication Libraries, 20HeTD, M2, ENSEIRB-MATMECA/IPB joint with University of Bordeaux
Engineering School: Denis Barthou, Compilation, Architecture, Architecture for HPC, real-time 3D at ENSEIRB-MATMECA (around 200HeTD), from L3 to M2.

9.2.3. Supervision

- PhD: Gregory Vaumourin, Hybrid Memory Hierarchy and Dynamic Data Handling in Embedded Parallel Architectures, University of Bordeaux, defended in Nov 2016, advisors: Denis Barthou, Alexandre Guerre (CEA), Thomas Dombek (CEA)
- PhD: Marc Sergent, Passage à l'échelle d'un support d'exécution à base de tâches pour l'algèbre linéaire dense, University of Bordeaux, defended in Dec 2016, advisors: Raymond Namyst, Olivier Aumage, Samuel Thibault, David Goudin (CEA)
- PhD in progress: Suraj Kumar, Task-based programming paradigms and scheduling, 2013/12, Emmanuel Agullo, Olivier Beaumont, Samuel Thibault
- PhD in progress: Terry cojean, Programming heterogeneous machines using moldable tasks, 2014/09, Pierre-André Wacrenier, Abdou Guermouche, Raymond Namyst
- PhD in progress: Christopher Haine, Estimating efficiency and automatic restructuration of data layout, 2014/01, Olivier Aumage, Denis Barthou
- PhD in progress: Arthur Loussert, Ressource (co)Allocation in HPC systems, 2016/10, Raymond Namyst, Marc Perache (CEA), Benoît Welterlen (ATOS)
- PhD in progress: Raphaël Prat, Load Balancing in Molecular Dynamics, 2016/10, Raymond Namyst, Laurent Colombet (CEA)

9.2.4. Juries

Denis Barthou has participated to the following PhD juries

- Abdul Wahid MEMON, U. Versailles St Quentin, Jun 2016 (reviewer)
- Abderrahmane Nassim HALLI, U. of Grenoble, Sep 2016 (reviewer)
- Milan KABAC, U. Bordeaux, Oct 2016 (president)
- Nans ODRY, U. Aix Marseille, Oct 2016 (reviewer)

Raymond Namyst has participated to the following PhD juries

- David Beniamine, U. Grenoble, Dec 2016 (reviewer)
- Alban Rousset, U. Besançon, Oct 2016 (reviewer)
- Naweiluo Zhou, U. Grenoble, Oct 2016 (reviewer)
- Béranger Bramas, U. Bordeaux, Feb 2016 (president)
- Oleg Iegorov, U. Grenoble, Apr 2016 (president)

9.3. Popularization

- Samuel Thibault made a Inria talk about « Building Debian/Ubuntu packages to make it easy for users to install your software »

10. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] G. VAUMOURIN. *Read Only Data Specific Management for an Energy Efficient Memory System*, Université de Bordeaux, October 2016, https://tel.archives-ouvertes.fr/tel-01402354

Invited Conferences

[2] T. COJEAN. *The StarPU Runtime System at Exascale ?: Scheduling and Programming over Upcoming Machines*, in "RESPA workshop at SC16", Salt Lake City, Utah, United States, November 2016, https://hal.inria.fr/hal-01410103

International Conferences with Proceedings

[3] E. AGULLO, O. BEAUMONT, L. EYRAUD-DUBOIS, S. KUMAR. *Are Static Schedules so Bad ? A Case Study on Cholesky Factorization*, in "IEEE International Parallel & Distributed Processing Symposium (IPDPS 2016)", Chicago, IL, United States, IEEE, May 2016, https://hal.inria.fr/hal-01223573

[4] P.-A. ARRAS, D. FUIN, E. JEANNOT, S. THIBAULT. *DKPN: A Composite Dataflow/Kahn Process Networks Execution Model*, in "24th Euromicro International Conference on Parallel, Distributed and Network-based processing", Heraklion Crete, Greece, February 2016, https://hal.inria.fr/hal-01234333

[5] O. BEAUMONT, T. COJEAN, L. EYRAUD-DUBOIS, A. GUERMOUCHE, S. KUMAR. *Scheduling of Linear Algebra Kernels on Multiple Heterogeneous Resources*, in "International Conference on High Performance Computing, Data, and Analytics (HiPC 2016)", Hyderabad, India, Proceedings of the IEEE International Conference on High Performance Computing (HiPC 2016), IEEE, December 2016, https://hal.inria.fr/hal-01361992

[6] A. CASSAGNE, O. AUMAGE, C. LEROUX, D. BARTHOU, B. LE GAL. *Energy Consumption Analysis of Software Polar Decoders on Low Power Processors*, in "The 2016 European Signal Processing Conference (EUSIPCO 2016)", Budapest, Hungary, August 2016, https://hal.archives-ouvertes.fr/hal-01363975

[7] A. CASSAGNE, T. TONNELLIER, C. LEROUX, B. LE GAL, O. AUMAGE, D. BARTHOU. *Beyond Gbps Turbo Decoder on Multi-Core CPUs*, in "International Symposium on Turbo Codes & Iterative Information Processing", Brest, France, Turbo Codes and Iterative Information Processing, September 2016 [*DOI :* 10.1109/ISTC.2016.7593092], https://hal.archives-ouvertes.fr/hal-01363980

[8] T. COJEAN, A. GUERMOUCHE, A. HUGO, R. NAMYST, P.-A. WACRENIER. *Resource aggregation for task-based Cholesky Factorization on top of heterogeneous machines*, in "HeteroPar'2016 worshop of Euro-Par", Grenoble, France, August 2016, https://hal.inria.fr/hal-01181135

[9] T. COJEAN, A. GUERMOUCHE, A.-E. HUGO, R. NAMYST, P.-A. WACRENIER. *Resource aggregation in task-based applications over accelerator-based multicore machines*, in "HeteroPar'2016 worshop of Euro-Par", Grenoble, France, August 2016, https://hal.inria.fr/hal-01355385

[10] V. GARCIA PINTO, L. STANISIC, A. LEGRAND, L. MELLO SCHNORR, S. THIBAULT, V. DANJEAN. *Analyzing Dynamic Task-Based Applications on Hybrid Platforms: An Agile Scripting Approach*, in "3rd Workshop on Visual Performance Analysis (VPA)", Salt Lake City, United States, November 2016, Held in conjunction with SC16, https://hal.inria.fr/hal-01353962

[11] P. HUCHANT, M.-C. COUNILH, D. BARTHOU. *Automatic OpenCL Task Adaptation for Heterogeneous Architectures*, in "Euro-Par", Grenoble, France, Euro-Par 2016: Parallel Processing, August 2016, pp. 684 - 696 [*DOI :* 10.1007/978-3-319-43659-3_50], https://hal.archives-ouvertes.fr/hal-01419366

[12] M. SERGENT, D. GOUDIN, S. THIBAULT, O. AUMAGE. *Controlling the Memory Subscription of Distributed Applications with a Task-Based Runtime System*, in "21st International Workshop on High-Level Parallel Programming Models and Supportive Environments", Chicago, United States, May 2016, https://hal.inria.fr/hal-01284004

Conferences without Proceedings

[13] O. AUMAGE, D. BARTHOU, A. HONORAT. *A Stencil DSEL for Single Code Accelerated Computing with SYCL*, in "SYCL 2016 1st SYCL Programming Workshop during the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming", Barcelone, Spain, March 2016, https://hal.archives-ouvertes.fr/hal-01290099

[14] M. SERGENT, D. GOUDIN, S. THIBAULT, O. AUMAGE. *Controlling the Memory Subscription of Distributed Applications with a Task-Based Runtime System*, in "SIAM Conference on Parallel Processing for Scientific Computing (SIAM PP 2016)", Paris, France, April 2016, pp. 318 - 327, https://hal.inria.fr/hal-01380126

Research Reports

[15] E. AGULLO, O. AUMAGE, B. BRAMAS, O. COULAUD, S. PITOISET. *Bridging the gap between OpenMP 4.0 and native runtime systems for the fast multipole method*, Inria, March 2016, n^{o} RR-8953, 49 p. , https://hal.inria.fr/hal-01372022

[16] E. AGULLO, O. AUMAGE, M. FAVERGE, N. FURMENTO, F. PRUVOST, M. SERGENT, S. THIBAULT. *Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model*, Inria Bordeaux Sud-Ouest ; Bordeaux INP ; CNRS ; Université de Bordeaux ; CEA, June 2016, n^{o} RR-8927, 27 p. , https://hal.inria.fr/hal-01332774

[17] E. AGULLO, B. BRAMAS, O. COULAUD, M. KHANNOUZ, L. STANISIC. *Task-based fast multipole method for clusters of multicore processors*, Inria Bordeaux Sud-Ouest, October 2016, n^{o} RR-8970, 15 p. , https://hal.inria.fr/hal-01387482

Other Publications

[18] O. BEAUMONT, L. EYRAUD-DUBOIS, S. KUMAR. *Approximation Proofs of a Fast and Efficient List Scheduling Algorithm for Task-Based Runtime Systems on Multicores and GPUs*, October 2016, working paper or preprint, https://hal.inria.fr/hal-01386174

[19] T. COJEAN, A. GUERMOUCHE, A. HUGO, R. NAMYST, P.-A. WACRENIER. *Resource aggregation for task-based Cholesky Factorization on top of modern architectures*, November 2016, This paper is submitted for review to the Parallel Computing special issue for HCW and HeteroPar 16 workshops, https://hal.inria.fr/hal-01409965