IN PARTNERSHIP WITH:
Institut Polytechnique de Bordeaux

Université de Bordeaux

Activity Report 2018

Project-Team STORM

STatic Optimizations, Runtime Methods

IN COLLABORATION WITH: Laboratoire Bordelais de Recherche en Informatique (LaBRI)

RESEARCH CENTER
Bordeaux - Sud-Ouest

THEME
Distributed and High Performance Computing

Table of contents

Project-Team STORM

Creation of the Team: 2015 January 01, updated into Project-Team: 2017 July 01

Keywords:

Computer Science and Digital Science:

A1.1.1. - Multicore, Manycore
A1.1.2. - Hardware accelerators (GPGPU, FPGA, etc.)
A1.1.3. - Memory models
A1.1.4. - High performance computing
A1.1.5. - Exascale
A2.1.7. - Distributed programming
A2.2.1. - Static analysis
A2.2.2. - Memory models
A2.2.4. - Parallel architectures
A2.2.5. - Run-time systems
A2.2.6. - GPGPU, FPGA...

Other Research Topics and Application Domains:

B2.2.1. - Cardiovascular and respiratory diseases
B3.2. - Climate and meteorology
B3.3.1. - Earth and subsoil
B3.4.1. - Natural risks
B4.2. - Nuclear Energy Production
B5.2.3. - Aviation
B5.2.4. - Aerospace
B6.2.2. - Radio technology
B6.2.3. - Satellite technology
B6.2.4. - Optic technology

1. Team, Visitors, External Collaborators

Research Scientists

Olivier Aumage [Inria, Researcher]
Emmanuelle Saillard [Inria, Researcher]

Faculty Members

Denis Barthou [Institut National Polytechnique de Bordeaux, Professor, Team leader, HDR]
Marie-Christine Counilh [Univ de Bordeaux, Associate Professor]
Raymond Namyst [Univ de Bordeaux, Professor, HDR]
Samuel Thibault [Univ de Bordeaux, Associate Professor, HDR]
Pierre Andre Wacrenier [Univ de Bordeaux, Associate Professor]

Post-Doctoral Fellow

Philippe Virouleau [Inria, from Jul 2018]

PhD Students

Hugo Brunie [CEA]

Adrien Cassagne [Inria]
Idriss Daoudi [Inria, from Oct 2018]
Pierre Huchant [Univ de Bordeaux]
Romain Lion [Inria, from Oct 2018]
Arthur Loussert [CEA]
Raphaël Prat [CEA]
Leo Villeveygoux [Univ de Bordeaux]
Philippe Virouleau [Inria, from Apr 2018 until Jun 2018]

Technical staff

Nathalie Furmento [CNRS]
Yanis Khorsi [Inria, until Sep 2018]
Chiheb Sakka [Inria, until Mar 2018]
Corentin Salingue [Inria]

Interns

François Audoy [Inria, from May 2018 until Jul 2018]
Loïc Jouans [Ecole Normale Supérieure Lyon, from Jun 2018 until Aug 2018]
Alexis Juven [Inria, from Jun 2018 until Sep 2018]
Maël Keryell [Inria, from Oct 2018]
Antoine Tirel [Inria, from Jun 2018 until Sep 2018]

Administrative Assistant

Sabrina Blondel-Duthil [Inria]

Visiting Scientists

Dana Akhmetova [KTH, Sweden, from Feb 2018 until Mar 2018]
Costin Iancu [Lawrence Berkeley National Lab, Oct 2018]

External Collaborators

Terry Cojean [Univ de Bordeaux, until Aug 2018]
Jean-Marie Couteyen [Airbus]

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.

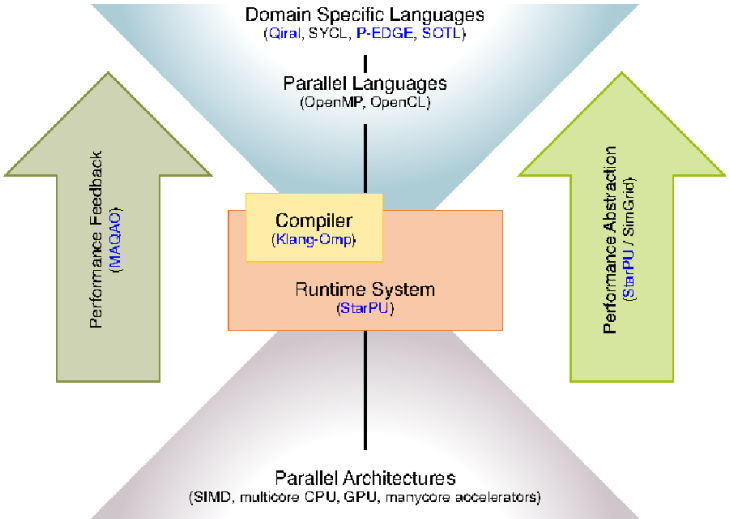

Figure 1. STORM Big Picture

- 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.

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. The results of the team on-going research in 2018 reflect this focus. Results presented in Sections 7.1, 7.2, 7.3, 7.4, 7.5 correspond to efforts for higher abstractions through DSL or libraries, and decouple algorithmics from parallel optimizations. The work described in Sections 7.6, 7.7, 7.8 focus in particular on StarPU and its development in order to better abstract architecture and optimizations. The results described in Sections 7.9, 7.10, 7.11 provide feed-back information, through visualization, deadlock detection and memory allocation optimization for parallel executions. The results in Sections 7.13, 7.14 correspond to efforts on parallelism expression and again abstraction, starting from standard parallel programming languages. Finally, Section 7.15 presents an on-going effort, applying state-of-the-art parallelizing/vectorizing methods on a real molecular dynamics code.

4. Application Domains

4.1. Application domains benefiting from HPC

The application domains of this research are the following:

- Molecular dynamics (see ExaStamp 7.15),
- Bioinformatics (see ADT Gordon 8.2.2)
- Environment, in particular CO_2 capture (see Exa2PRO, 8.3.1)
- Health and hearth disease analysis (see EXACARD, 8.2.1)
- Software infrastructures for Telecommunications (see AFF3CT, 7.4, 8.2.2)
- Aéronautique (collaboration avec Airbus, J.-M. Couteyen)

4.2. Application in High performance computing/Big Data

Most of the research of the team has application in the domain of software infrastructure for HPC and BigData (see HPC Cloud Computing project 8.1, ANR SOLHAR 8.2.1, Inria ADT SwLoc, Gordon8.2.2, IPL HAC-SPECIS and BigData 8.2.3, PIA project ELCI 8.2, FP7 projects INTERTWinE and Exa2Pro 8.3.1 and PRACE project PRACE5IP 8.3.2).

5. Highlights of the Year

5.1. Highlights of the Year

- “Habilitation à diriger les recherches” (HDR) of Samuel Thibault, Dec.2018.

6. New Software and Platforms

6.1. Chameleon

KEYWORDS: Runtime system - Task-based algorithm - Dense linear algebra - HPC - 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.

Overall end point. 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).

RELEASE FUNCTIONAL DESCRIPTION: Chameleon includes the following features:

- BLAS 3, LAPACK one-sided and LAPACK norms tile algorithms - Support QUARK and StarPU runtime systems and PaRSEC since 2018 - Exploitation of homogeneous and heterogeneous platforms through the use of BLAS/LAPACK CPU kernels and cuBLAS/MAGMA CUDA kernels - Exploitation of clusters of interconnected nodes with distributed memory (using OpenMPI)

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

6.2. hwloc

Hardware Locality
KEYWORDS: NUMA - Multicore - GPU - Affinities - Open MPI - Topology - HPC - 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: Open MPI consortium - Intel - AMD
- Contact: Brice Goglin
- Publications: hwloc: a Generic Framework for Managing Hardware Affinities in HPC Applications - Managing the Topology of Heterogeneous Cluster Nodes with Hardware Locality (hwloc) - A Topology-Aware Performance Monitoring Tool for Shared Resource Management in Multicore Systems - Exposing the Locality of Heterogeneous Memory Architectures to HPC Applications - Towards the Structural Modeling of the Topology of next-generation heterogeneous cluster Nodes with hwloc - On the Overhead of Topology Discovery for Locality-aware Scheduling in HPC
- URL: http://www.open-mpi.org/projects/hwloc/

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

6.4. KStar

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

FUNCTIONAL DESCRIPTION: The KStar software 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: Nathalie Furmento, Olivier Aumage, Philippe Virouleau and Samuel Thibault
- Contact: Olivier Aumage
- Publications: Bridging the gap between OpenMP and task-based runtime systems for the fast multipole method - Bridging the gap between OpenMP 4.0 and native runtime systems for the fast multipole method - Evaluation of OpenMP Dependent Tasks with the KASTORS Benchmark Suite
- URL: http://kstar.gforge.inria.fr/#!index.md

6.5. MAQAO

SCIENTIFIC DESCRIPTION: MAQAO relies on binary codes for Intel x86 and ARM architectures. For x86 architecture, it can insert probes for instrumention directly inside the binary. There is no need to recompile. The static/dynamic approach of MAQAO analysis is the main originality of the tool, combining performance model with values collected through instrumentation.

MAQAO has a static performance model for x86 and ARM architectures. This model analyzes performance of the codes on the architectures and provides some feed-back hints on how to improve these codes, in particular for vector instructions.

The dynamic collection of data in MAQAO enables the analysis of thread interactions, such as false sharing, amount of data reuse, runtime scheduling policy, ...
FUNCTIONAL DESCRIPTION: MAQAO is a performance tuning tool for OpenMP parallel applications. It relies on the static analysis of binary codes and the collection of dynamic information (such as memory traces). It provides hints to the user about performance bottlenecks and possible workarounds.

- Participants: Christopher Haine, Denis Barthou, James Tombi A Mba and Olivier Aumage
- Contact: Denis Barthou

6.6. StarPU

The StarPU Runtime System
KEYWORDS: Multicore - GPU - Scheduling - HPC - 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: Corentin Salingue, Andra Hugo, Benoît Lize, Cédric Augonnet, Cyril Roelandt, François Tessier, Jérôme Clet-Ortega, Ludovic Courtes, Ludovic Stordeur, Marc Sergent, Mehdi Juhoor, Nathalie Furmento, Nicolas Collin, Olivier Aumage, Pierre-André Wacrenier, Raymond Namyst, Samuel Thibault, Simon Archipoff, Xavier Lacoste, Terry Cojean, Yanis Khorsi, Philippe Virouleau, LOïC JOUANS and Leo Villeveygoux
- Contact: Olivier Aumage
- Publications: Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model - The StarPU Runtime System at Exascale ? - Analyzing Dynamic Task-Based Applications on Hybrid Platforms: An Agile Scripting Approach - Resource aggregation for task-based Cholesky Factorization on top of heterogeneous machines - Resource aggregation in task-based applications over accelerator-based multicore machines - Controlling the Memory Subscription of Distributed Applications with a Task-Based Runtime System - Exploiting Two-Level Parallelism by Aggregating Computing Resources in Task-Based Applications Over Accelerator-Based Machines - Exploiting Two-Level Parallelism by Aggregating Computing Resources in Task-Based Applications Over Accelerator-Based Machines - Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model - Bridging the gap between OpenMP 4.0 and native runtime systems for the fast multipole method - Scalability of a task-based runtime system for dense linear algebra applications - Faithful Performance Prediction of a Dynamic Task-Based Runtime System for Heterogeneous Multi-Core Architectures - Towards seismic wave modeling on heterogeneous many-core architectures using task-based runtime system - Bridging the Gap between Performance and Bounds of Cholesky Factorization on Heterogeneous Platforms - Composing multiple StarPU applications over heterogeneous machines: A supervised approach - Evaluation of OpenMP Dependent Tasks with the KASTORS Benchmark Suite - A runtime approach to dynamic resource allocation for sparse direct solvers - Modeling and Simulation of a Dynamic Task-Based Runtime System for Heterogeneous Multi-Core Architectures - Toward OpenCL Automatic Multi-Device Support - Harnessing clusters of hybrid nodes with a sequential task-based programming model - Taking advantage of hybrid systems for sparse direct solvers via task-based runtimes - Modulariser les ordonnanceurs de tâches : une approche structurelle - Overview of Distributed Linear Algebra on Hybrid Nodes over the StarPU Runtime - StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators - Modeling and Simulation of a Dynamic Task-Based Runtime System for Heterogeneous Multi-Core Architectures - Taking advantage of hybrid systems for sparse direct solvers via task-based runtimes - Adaptive Task Size Control on High Level Programming for GPU/CPU Work Sharing - Composing multiple StarPU applications over heterogeneous machines: a supervised approach - Implementation of FEM Application on GPU with StarPU - Le problème de la composition parallèle : une approche supervisée - Support exécutif scalable pour les architectures hybrides distribuées - SOCL: An OpenCL Implementation with Automatic Multi-Device Adaptation Support - C Language Extensions for Hybrid CPU/GPU Programming with StarPU

- Programming Models and Runtime Systems for Heterogeneous Architectures - Programmation unifiée multi-accélérateur OpenCL - StarPU-MPI: Task Programming over Clusters of Machines Enhanced with Accelerators - Parallelization on Heterogeneous Multicore and Multi-GPU Systems of the Fast Multipole Method for the Helmholtz Equation Using a Runtime System - High-Level Support for Pipeline Parallelism on Many-Core Architectures - Programmability and Performance Portability Aspects of Heterogeneous Multi-/Manycore Systems - Programmation des architectures hétérogènes à l'aide de tâches divisibles - StarPU: a unified platform for task scheduling on heterogeneous multicore architectures - PEPPHER: Efficient and Productive Usage of Hybrid Computing Systems - The PEPPHER Approach to Programmability and Performance Portability for Heterogeneous many-core Architectures - Flexible runtime support for efficient skeleton programming on hybrid systems - LU Factorization for Accelerator-based Systems - QR Factorization on a Multicore Node Enhanced with Multiple GPU Accelerators - Programmation multi-accélérateurs unifiée en OpenCL - Détection optimale des coins et contours dans des bases d'images volumineuses sur architectures multicœurs hétérogènes - Association de modèles de programmation pour l'exploitation de clusters de GPUs dans le calcul intensif - Programming heterogeneous, accelerator-based multicore machines:current situation and main challenges - Scheduling Tasks over Multicore machines enhanced with acelerators: a Runtime System's Perspective - Composabilité de codes parallèles sur architectures hétérogènes - Data-Aware Task Scheduling on Multi-Accelerator based Platforms - Dynamically scheduled Cholesky factorization on multicore architectures with GPU accelerators. - StarPU: a Runtime System for Scheduling Tasks over Accelerator-Based Multicore Machines - StarPU : un support exécutif unifié pour les architectures multicoeurs hétérogènes - Automatic Calibration of Performance Models on Heterogeneous Multicore Architectures - StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures - Exploiting the Cell/BE architecture with the StarPU unified runtime system - Bridging the gap between OpenMP and task-based runtime systems for the fast multipole method

- URL: http://starpu.gforge.inria.fr/

6.7. PARCOACH

PARallel Control flow Anomaly CHecker

KEYWORDS: High-Performance Computing - Program verification - Debug - MPI - OpenMP - Compilation

SCIENTIFIC DESCRIPTION: PARCOACH verifies programs in two steps. First, it statically verifies applications with a data- and control-flow analysis and outlines execution paths leading to potential deadlocks. The code is then instrumented, displaying an error and synchronously interrupting all processes if the actual scheduling leads to a deadlock situation.

FUNCTIONAL DESCRIPTION: Supercomputing plays an important role in several innovative fields, speeding up prototyping or validating scientific theories. However, supercomputers are evolving rapidly with now millions of processing units, posing the questions of their programmability. Despite the emergence of more widespread and functional parallel programming models, developing correct and effective parallel applications still remains a complex task. As current scientific applications mainly rely on the Message Passing Interface (MPI) parallel programming model, new hardwares designed for Exascale with higher node-level parallelism clearly advocate for an MPI+X solutions with X a thread-based model such as OpenMP. But integrating two different programming models inside the same application can be error-prone leading to complex bugs - mostly detected unfortunately at runtime. PARallel COntrol flow Anomaly CHecker aims at helping developers in their debugging phase.

- Participants: Emmanuelle Saillard, Denis Barthou and Pierre Huchant
- Partner: CEA
- Contact: Emmanuelle Saillard
- URL: https://esaillar.github.io/PARCOACH/

6.8. AFF3CT

A Fast Forward Error Correction Toolbox
KEYWORDS: High-Performance Computing - Signal processing - Error Correction Code
FUNCTIONAL DESCRIPTION: AFF3CT proposes high performance Error Correction algorithms for Polar, Turbo, LDPC, RSC (Recursive Systematic Convolutional), Repetition and RA (Repeat and Accumulate) codes. These signal processing codes can be parameterized in order to optimize some given metrics, such as Bit Error Rate, Bandwidth, Latency, ...using simulation. For the designers of such signal processing chain, AFF3CT proposes also high performance building blocks so to develop new algorithms. AFF3CT compiles with many compilers and runs on Windows, Mac OS X, Linux environments and has been optimized for x86 (SSE, AVX instruction sets) and ARM architectures (NEON instruction set).

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

6.9. MORSE

KEYWORDS: High performance computing - Matrix calculation - Fast multipole method - Runtime system
FUNCTIONAL DESCRIPTION: MORSE (Matrices Over Runtime Systems @ Exascale) is a scientific project, its objectives are to solve matrix problems on complex architectures, using runtime systems. More specifically, the goal is to write codes that reach a high level of performance for all architectures. The algorithms are written independently of the architecture, and the runtime system dispatches the different computational parts to the different computing units. This methodology has been validated on three classes of problems: dense linear algebra, sparse and dense, and fast multipole methods. The corresponding codes have been incorporated into several softwares, MAGMA, Pastix and ScalFMM.

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

6.10. SwLoc

Software Contexts for Locality
KEYWORDS: HPC - Locality - Contexts - Multicore - GPU
FUNCTIONAL DESCRIPTION: SwLoc is a library for flexible and generic partitioning of computing resources (processors, accelerators) to be able to co-excute confined parallel regions which can rely on different runtime systems (e.g. OpenMP, Intel TBB, StarPU, etc.). With all different hypervisor strategies, It is possible to adapt dynamically the computing resources of each context, in order to match each parallel region's need as closely as possible.

- Contact: Corentin Salingue
- URL: http://swloc.gforge.inria.fr/

6.11. VITE

Visual Trace Explorer
KEYWORDS: Visualization - Execution trace
FUNCTIONAL DESCRIPTION: ViTE is a trace explorer. It is a tool made to visualize execution traces of large parallel programs. It supports Pajé, a trace format created by Inria Grenoble, and OTF and OTF2 formats, developed by the University of Dresden and allows the programmer a simpler way to analyse, debug and/or profile large parallel applications.

- Participant: Mathieu Faverge
- Contact: Mathieu Faverge
- URL: http://vite.gforge.inria.fr/

7. New Results

7.1. InKS Programming Model

Existing programming models tend to tightly interleave algorithm and optimization in HPC simulation codes. This requires scientists to become experts in both the simulated domain and the optimization process and makes the code difficult to maintain and port to new architectures. The InKS programming model, developed within the context of the PhD. Thesis of Ksander Ejjaaouani [9], decouples these two concerns with distinct languages for each. The simulation algorithm is expressed in the InKS pia language with no concern for machine-specific optimizations. Optimizations are expressed using both a family of dedicated optimizations DSLs (InKS O) and plain C++. InKS O relies on the InKS pia source to assist developers with common optimizations while C++ is used for less common ones. Our evaluation demonstrates the soundness of the approach by using it on synthetic benchmarks and the Vlasov-Poisson equation. It shows that InKS offers separation of concerns at no performance cost.

7.2. Porting Chameleon on top of OpenMP

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. Algorithms were implemented on top of several task-based runtime systems: QUARK, PaRSEC, and StarPU (for which there is also an optional heterogeneous implementation). In the context of PRACE-5IP, we introduced OpenMP as an alternative backend for these linear algebra kernels.

7.3. StarPU in Julia

Julia is a modern language for parallelism and simulation that aims to ease the effort for developping high performance codes. In this context, we have started to develop a StarPU binding inside Julia. It is now possible to launch StarPU kernels inside Julia, either given as libraries, or described in Julia directly. Julia has the advantage to simplify significantly the syntax required to express the task and data management in StarPU (defining a new scope for StarPU, using automatic desallocation of buffers, ...).

Besides, using the introspection properties of Julia, the kernels written in Julia are automatically translated in both C codes and CUDA codes. Some preliminary experimental results show encouraging speedups on some limited codes. This is a work in progress, developped with A.Juven and M.Keryell.

7.4. Simulation and Validation of Error Correction Code Algorithms

The AFF3CT Error Correction Code (ECC) development and experimentation toolchain reached a major milestone with the release of the 2.x branch. It incorporates a hefty set of new modules and capabilities:

- New code families: Reed-Solomon, Turbo Product Code (TCP);
- New decoders: Maximum Likelihood (ML), Chase, LDPC Approximate Min-Star (AMS), LDPC Vertical Layered, LDPC Peeling, LDPC Bit Flipping;
- New channels: Optical, Binary Erasure Channel (BEC), Binary Symmetric Channel (BSC);
- New modem: On-Off Keying (OOK).

This new branch comes with extensive documentation of all available parameters at any point in the chain (https://aff3ct.readthedocs.io). In the process of the new release, the source code has also be reorganized in a rational and compartimentalized way, in terms of modules, tasks and sockets. This refactoring streamlines the use AFF3CT as a library, building on the concept of tasks, with well defined input and output sockets.

7.5. Speeding-Up Error Correction Code Processing using a Portable SIMD Wrapper

Error correction code (ECC) processing has so far been performed on dedicated hardware for previous generations of mobile communication standards, to meet latency and bandwidth constraints. As the 5G mobile standard, and its associated channel coding algorithms, are now being specified, modern CPUs are progressing to the point where software channel decoders can viably be contemplated. A key aspect in reaching this transition point is to get the most of CPUs SIMD units on the decoding algorithms being pondered for 5G mobile standards. The nature and diversity of such algorithms requires highly versatile programming tools. We proposed the virtues and versatility of our MIPP SIMD wrapper in implementing a high performance portfolio of key ECC decoding algorithms on AFF3CT [8].

7.6. Runtime System Interoperability with StarPU

Parallel HPC applications increasingly build on multiple parallel libraries, which results in interferences if the parallel entities in the application and in the libraries it uses access computing resources in an uncoordinated manner. A set of resource management APIs has therefore been designed within the context of H2020 project INTERTWinE (see http://www.intertwine-project.eu/developer-hub/resource-manager), and implemented in the StarPU task-based runtime system developed by Team STORM, as well as in the OmpSs/Nanos 6 task-based runtime system developed at the Barcelona Supercomputing Center (BSC). It enables StarPU and OmpSs to interoperate within an application, along multiple scenarios such as *nested interoperability*, with a host runtime system executing parallel tasks over a guest runtime system, or *concurrent interoperability*, with several runtime systems dynamically sharing computing resources over the application lifespan.

7.7. Hierarchicals Tasks

The programming model of StarPU, namely the sequential task flow model, was successfuly used in several applicative areas and was able to achieve high performance. However, the submission process needed either to be completely static, in the sense that the whole task graph is submitted at once, or to be stopped from time to time in order to control the execution. To overcome these limitations, we have introduced a new paradigm which we call *hierarchical tasks* where the so-called *control tasks* allow to submit at runtime a task subgraph. By allowing the submission of some parts of the task graph to be delayed until the execution of the corresponding control task, this feature allows to timely and dynamically choose the right version of the computation task subgraph (e.g. OpenMP, StarPU, cuda, or sequential, etc. implementations).

The graph of control tasks provide a high-level description of the computations which allow to use and design sophisticated scheduling algorithms. Furthermore, the cost of managing the control graph being much smaller than the one of the computation task graph, relying on the hierarchical tasks scheme enhances the scalability of the runtime system and allows to parallelize the submission process. Finally, this mechanism represents an elegant way of tackling the granularity issues which represent a key problem for achieving high performance in a heterogeneous context. The specificity of our implementation is to nicely combine hierarchical tasks with data partitioning, without needless synchronisation points.

7.8. Load Balancing Management in a Distributed Task-Based Programming Model

Distributed task-based programming models such as StarPU optimize the execution of applications based on an initial distribution of data. The resulting computational load on each node may however evolve over the course of the application, to the point where this initial distribution of data leads becomes suboptimal. It becomes necessary to correct the distribution the distribution of data to rebalance the load among nodes. Tools such as Zoltan or ParMetis do exist to perform this rebalancing job. However, they cannot be employed without breaking the application execution flow, and force synchronizing steps in fundamentally asynchronous task parallelism paradigms. Within the context of the internship of Loïc Jouans, we proposed a mechanism to enable the detection of load imbalance as well as the application of corrective measures to rebalance it while preserving the execution asynchrony.

7.9. Task-based Execution Visualization

One of the purpose of task-based programming is to let asynchronous execution achieve extreme pipelining of operations. This however make it a real challenge to determine why an execution performs poorly, since the execution trace shows the mixture of unrelated tasks. With the the University of Grenoble, we have designed a visualization framework which allows to easily visualize different metrics of the execution trace and apply different techniques to reveal the execution behavior. This allowed to determine and fix some erratic behaviors for instance in the StarPU runtime system and OpenMPI communication library [13], [4]

7.10. Interprocedural Collectives Verification

The advent to exascale requires more scalable and efficient techniques to help developers to locate, analyze and correct errors in parallel applications. PARallel COntrol flow Anomaly CHecker (PARCOACH) is a framework that detects the origin of collective errors in applications using MPI and/or OpenMP. In MPI, such errors include collective operations mismatches. In OpenMP, a collective error can be a barrier not called by all tasks in a team. We have developed an extension of PARCOACH which improves its collective errors detection [11]. The new analysis is more precise and accurate than the previous one on different benchmarks and real applications.

7.11. Profile-Guided Scope-Based Data Allocation Method

The complexity of High Performance Computing nodes memory system increases in order to challenge application growing memory usage and increasing gap between computation and memory access speeds. As these technologies are just being introduced in HPC supercomputers no one knows if it is better to manage them with hardware or software solutions. Thus both are being studied in parallel. For both solutions, the problem consists in choosing which data to store on which memory at any time.

In this context, we propose a linear formulation of the data allocation problem. Moreover, we propose a new profile- guided scope-based approach which reduces the data allocation problem complexity, thus enhancing the precision of state of the art analyzes. Finally we have implemented our method in a framework made of GCC plugins, dynamic libraries and python scripts, allowing to test the method on several benchmarks. We have evaluated our method on an INTEL Knight's Landing processor. To this aim we have run LULESH, HydroMM, two hydrodynamic codes, and MiniFE, a finite element mini application. We have compared our framework performance over these codes to several straight- forward solutions: MCDRAM as a cache, in hybrid mode, in flat mode using numactl command and existing AutoHBW dynamic library [7]

7.12. Lightweight Containerization of Computing Resources

SwLoc is a library for flexible and generic partitioning of computing resources (CPU, accelerators). It allows applications to create contexts (i.e. resource partitions) and run parallel codes inside such lightweight containers. Many libraries developed using OpenMP, Pthreads or Intel TBB can ben executed concurrently with little or no modification. SwLoc also features dynamic context resizing capabilities that enables parallel applications to perform resource negotiation.

7.13. Adaptive Partitioning for Iterated Sequences of Irregular OpenCL Kernels

OpenCL defines a common parallel programming language for CPU and GPU devices, although writing tasks adapted to the devices, managing communication and load-balancing issues are left to the programmer. We propose [10] a static/dynamic approach for the execution of an iterated sequence of data-dependent kernels on a multi-device heterogeneous architecture. The method allows to automatically distribute irregular kernels onto multiple devices and tackles, without training, both load balancing and data transfers issues coming from hardware heterogeneity, load imbalance within the application itself and load variations between repeated executions of the sequence. Our evaluation on some benchmarks and a complex N-body application, SOTL, simulating the electromagnetic Coulomb force applied on particles, show the interest of our approach.

7.14. A compiler front-end for OpenMP's variants

OpenMP 5.0 introduced the concept of *variant*: a directive which can be used to indicate that a function is a variant of another existing *base function*, in a specific context (eg: `foo_gpu_nvidia` could be declared as a variant of `foo`, but only when executing on specific NVidia hardware).

In the context of PRACE-5IP, we want to leverage this construct to be able to take advantage of the StarPU heterogeneous scheduler through the interoperability layer between OpenMP and StarPU.

We started this work by implementing the necessary changes in the Clang front-end to support OpenMP's *variant*.

7.15. Combining Task-based Parallelism and Adaptive Mesh Refinement Techniques in Molecular Dynamics Simulations

Modern parallel architectures require applications to generate massive parallelism so as to feed their large number of cores and their wide vector units. We have revisited the extensively studied classical Molecular Dynamics N-body problem in the light of these hardware constraints. We have introduced Adaptive Mesh Refinement techniques to store particles in memory, and to optimize the force computation loop using multi-threading and vectorization-friendly data structures [14]. Our design is guided by the need for load balancing and adaptivity raised by highly dynamic particle sets, as typically observed in simulations of strong shocks resulting in material micro-jetting. We have analyzed performance results on several simulation scenarios, over 512 nodes equipped by Intel Xeon Phi Knights Landing (KNL) processors. Performance obtained with our OpenMP implementation outperforms state-of-the-art implementations (LAMMPS) on both steady and micro-jetting particles simulations. In the latter case, our implementation is 1.38 times faster on KNL.

These results were obtained in the context of joint work between Inria and CEA/DAM.

8. Partnerships and Cooperations

8.1. Regional Initiatives

HPC Cloud Computing

Participants: Olivier Aumage, Nathalie Furmento, Samuel Thibault.

Other participants : David Auber, Olivier Beaumont, Lionel Eyraud-Dubois, Gérald Point

Abstract: The goal of this project is to gather teams from the HPC and Big Data communities to work at the intersection between these domains. We will focus on how StarPU can be adapted to achieve good performances on Big Data platforms.

8.2. National Initiatives

ELCI The ELCI PIA 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.

8.2.1. ANR

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

ANR MONU 2013 Program, 2013 - 2018 (36 months extended)

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 EXACARD

AAPG ANR 2018 (42 months)

Coordinator: Yves Coudière (Carmen) Inria Bordeaux

Abstract: Cardiac arrhythmia affect millions of patients and cause 300,000 deaths each year in Europe. Most of these arrhythmia are due to interaction between structural and electrophysiological changes in the heart muscle. A true understanding of these phenomena requires numerical simulations at a much finer resolution, and larger scale, than currently possible. Next-generation, heterogeneous, high-performance computing (HPC) systems provide the power for this. But the large scale of the computations pushes the limits of current runtime optimization systems, and together with task-based parallelism, prompts for the development of dedicated numerical methods and HPC runtime optimizations. With a consortium including specialists of these domains and cardiac modeling, we will investigate new task-based optimization techniques and numerical methods to utilize these systems for cardiac simulations at an unprecedented scale, and pave the way for future use cases.

8.2.2. ADT - Inria Technological Development Actions

ADT SwLoc (http://swloc.gforge.inria.fr/)

Participants: Raymond Namyst, Pierre-André Wacrenier, Andra Hugo, Brice Goglin, Corentin Salingue.

Inria ADT Campaign 2017, 10/2017 - 9/2019 (24 months)

Coordinator: Raymond Namyst

Abstract: The Inria action ADT SwLoc is aiming at developing a library allowing dynamic flexible partitioning of computing resources in order to execute parallel regions concurrently inside the same processes.

ADT Gordon

Participants: Denis Barthou, Nathalie Furmento, Samuel Thibault, Pierre-André Wacrenier.

Inria ADT Campaign 2018, 11/2018 - 11/2020 (24 months)

Coordinator: Emmanuel Jeannot (Tadaam)

Other partners: HiePACS, PLEIADE, Tadaam (Inria Bordeaux)

Abstract: Teams HiePACS, Storm and Tadaam develop each a brick of an HPC software stack, namely solver, runtime, and communication library. The goal of the Gordon project is to consolidate the HPC stack, to improve interfaces between each brick, and to target a better scalability. The bioinformatics application involved in the project has been selected so as to stress the underlying systems.

ADT AFF3CT Matlab

Participants: Denis Barthou, Olivier Aumage, Adrien Cassagne.

Inria ADT Campaign 2018, 12/2018 - 12/2019 (12 months)

Coordinator: Denis Barthou

Other partners: C.Jego and C.Leroux (IMS lab, U.Bordeaux)

Abstract: AFF3CT is a toolchain for designing, validation and experimentation of new Error Correcting codes. This toolchain is written in C++, and this constitutes a difficulty for many industrial users, who are mostly electronicians. The goal of this ADT is to widen the number of possible users by designing a Matlab and Python interface for AFF3CT, in collaboration with existing users, and proposing a parallel framework in OpenMP.

8.2.3. IPL - Inria Project Lab

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

Participants: Samuel Thibault, Luka Stanisic, Emmanuelle Saillard, Olivier Aumage, Idriss Daoudi.

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.

HPC-BigData (High Performance Computing and Big Data)

Participant: Samuel Thibault.

Inria IPL 2018 - 2022 (48 months)

Coordinator: Bruno Raffin (team DataMove, Inria Rhône Alpes)

Since June 2018, the team is participating to the HPC-BigData https://project.inria.fr/hpcbigdata/ Inria Project Lab (IPL). The goal of this HPC-BigData IPL is to gather teams from the HPC, Big Data and Machine Learning (ML) areas to work at the intersection between these domains. Research is organized along three main axes: high performance analytics for scientific computing applications, high performance analytics for big data applications, infrastructure and resource management.

8.3. European Initiatives

8.3.1. H2020 Projects

INTERTWinE

- Title: Programming Model INTERoperability ToWards Exascale
- Program: H2020
- Duration: October 2015 - October 2018
- Coordinator: EPCC
- Inria contact: Olivier Aumage
- 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)

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.

Exa2PRO

- Title: Enhancing Programmability and boosting Performance Portability for Exascale Computing systems
- Program: H2020-FETHPC
- Duration: May 2018 - April 2021
- Coordinator: ICCS
- Inria contact: Samuel Thibault
- Partners:
  * Institute of Communications and Computer Systems (ICCS) (Greece)
  * Linköpiung University (LIU) (Sweden)
  * Centre for Research and Technology Hellas (CERTH) (Greece)
  * Institut National de Recherche en Informatique et en Automatique (Inria) (France)
  * Maxeler Technologies Limited (MAX) (UK)
  * Forschungszentrum Jülich (JUELICH) (Germany)
  * Centre National de la Rcherche Scientifique (CNRS) (France)

The vision of EXA2PRO is to develop a programming environment that will enable the productive deployment of highly parallel applications in exascale computing systems. EXA2PRO programming environment will integrate tools that will address significant exascale challenges. It will support a wide range of scientific applications, provide tools for improving source code quality, enable efficient exploitation of exascale systems' heterogeneity and integrate tools for data and memory management optimization. Additionally, it will provide various fault-tolerance mechanisms, both user-exposed and at runtime system level and performance monitoring features. EXA2PRO will be evaluated using 4 use cases from 4 different domains, which will be deployed in JUELICH supercomputing center. The use cases will leverage the EXA2PRO tool-chain and we expect:

- Increased applications performance based on EXA2PRO optimization tools (data and memory management)
- Efficient exploitation of heterogeneity by the applications that will allow the evaluation of more complex problems.
- Identification of trade-offs between design qualities (source code maintainability/reusability) and run-time constraints (performance/energy consumption).
- Evaluation of various fault-tolerance mechanisms for applications with different characteristics.

EXA2PRO outcome is expected to have major impact in a) the scientific and industrial community that focuses on application deployment in supercomputing centers: EXA2PRO environment will allow efficient application deployment with reduced effort. b) on application developers of exascale application: EXA2PRO will provide tools for improving source code maintainability/ reusability, which will allow application evaluation with reduced developers' effort. c) on the scientific community and the industry relevant to the EXA2PRO use cases. At least two of the EXA2PRO use cases will have significant impact to the CO2 capture and to the Supercapacitors industry.

8.3.2. Collaborations in European Programs, Except FP7 & H2020

PRACE-5IP

- Title: PRACE 5th Implementation Phase
- Program: PRACE
- Duration: 2017 - 2019
- Coordinator: PRACE
- Inria contact for team STORM: Olivier Aumage
- Abstract: The objectives of PRACE-5IP are to build on and seamlessly continue the successes of PRACE and start new innovative and collaborative activities proposed by the consortium. These include:
  * assisting the transition to PRACE2 including analysis of TransNational Access;
  * strengthening the internationally recognised PRACE brand;
  * continuing and extend advanced training which so far provided more than 18 800 persontraining days;
  * preparing strategies and best practices towards Exascale computing;
  * coordinating and enhancing the operation of the multi-tier HPC systems and services;
  * supporting users to exploit massively parallel systems and novel architectures.

  A high level Service Catalogue is provided. The proven project structure will be used to achieve each of the objectives in 6 dedicated work packages. The activities are designed to increase Europe's research and innovation potential especially through:
  * seamless and efficient Tier-0 services and a pan-European HPC ecosystem including national capabilities;

* promoting take-up by industry and new communities and special offers to SMEs;
* implementing a new flexible business model for PRACE 2;
* proposing strategies for deployment of leadership systems;
* collaborating with the ETP4HPC, CoEs and other European and international organisations on future architectures, training, application support and policies.

8.4. International Research Visitors

- Costin Iancu, LBNL (USA), from Oct. 8 to Oct. 12, 2018

8.4.1. Internships

- Dana Akhmetova, KTH (Sweden), PhD Internship, from Feb. 18, 2018 to Mar. 24, 2018, within the context of H2020 INTERTWinE.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

- Olivier Aumage, hosting of the OpenMP Language Committee week at Inria Bordeaux, May 14-18, 30 participants.
- Raymond Namyst, Organization of the "Inria National Scientific Days" in Bordeaux, June 27-29, 250 participants.

9.1.2. Scientific Events Selection

9.1.2.1. Member of the Conference Program Committees

- Olivier Aumage: Cluster 2018, ICPP 2018, SC Asia 2018.
- Emmanuelle Saillard: COMPAS 2018, ISC HPC 2018 (poster committee), SC18 (workshop committee).
- Samuel Thibault: HCW, P3MA'18
- Denis Barthou: CF 2018
- Raymond Namyst: ISC 2018, IPDPS 2018, SC 2018, Euro-MPI 2018, SC Asia 2018.

9.1.2.2. Reviewer

- Olivier Aumage: Cluster, HCW, ICPP, PDP, PDSEC, SC Asia.
- Emmanuelle Saillard: COMPAS, ISC HPC, SC.
- Samuel Thibault: HCW

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

- Olivier Aumage: CCPE special issue.

9.1.3.2. Reviewer - Reviewing Activities

- Olivier Aumage: CCPE, IJPEDS, JPDC
- Samuel Thibault: JPDC, TPDS, IJHPCA

9.1.4. Invited Talks

- Olivier Aumage: SIAMPP (Tokyo), EPCC (Edinburgh), COMPAS (Toulouse), HPCS (Orléans).

- Emmanuelle Saillard: Journées GDR-GPL (Grenoble), Journée LaMHA (Paris).
- Samuel Thibault: ROMA seminar (Lyon), COMPAS (Toulouse), Task-based seminar (Uppsala)

9.1.5. Scientific Expertise

- Olivier Aumage: ANR (1 project in phase 2).
- Samuel Thibault: ANR (1 project in phase 2).

9.1.6. Research Administration

- Olivier Aumage: Permanent Contact for Team STORM.
- Nathalie Furmento: Member of the Commission de Développement Technologique at Inria Bordeaux Sud-Ouest
- Nathalie Furmento: Member of the technical team for PlaFRIM, Federative Platform for Research in Computer Science and Mathematics
- Nathalie Furmento: Member of the HCERES evaluation committee for the IRIF laboratory
- Nathalie Furmento: Member of the selection committee for an engineer position at the Université de Bordeaux

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Engineering School: Olivier Aumage, High Performance Communication Libraries, 20HeTD, M2, ENSEIRB-MATMECA.
- Engineering School: Olivier Aumage, Languages and Supports for Parallelism, 14HeTD, M2, ENSEIRB-MATMECA.
- Engineering School: Emmanuelle Saillard, Introduction to Algorithms, 16HeCI, L3, ENSEIRB-MATMECA.
- Engineering School: Emmanuelle Saillard, Tree Structure, 16HeCI, L3, ENSEIRB-MATMECA.
- Engineering School: Adrien Cassagne, Projet d'algorithmique et de programmation, 30HeTD, L3, ENSEIRB-MATMECA.
- Engineering School: Adrien Cassagne, Introduction aux réseaux, 15HeTD, L3, ENSEIRB-MATMECA.
- Engineering School: Adrien Cassagne, Applications TCP/IP, 15HeTD, L3, ENSEIRB-MATMECA.
- Engineering School: Philippe Virouleau, Programmation Impérative, 18HeTD, L3, ENSEIRB-MATMECA.
- Engineering School: Philippe Virouleau, Projet d'algorithmique et de programmation, 25HeTD, L3, ENSEIRB-MATMECA.
- Licence: Samuel Thibault is responsible for the computer science topic of the first university year.
- Licence: Samuel Thibault is responsible for the new Licence Pro ADSILLH (Administration et Développeur de Systèmes Informatiques à base de Logiciels Libres et Hybrides)
- Licence: Samuel Thibault, Introduction to Computer Science, 32HeTD, L1, University of Bordeaux.
- Licence: Samuel Thibault, Networking, 51HeTD, Licence Pro, University of Bordeaux.
- Engineering School: Denis Barthou is the head of the computer science teaching department of ENSEIRB-MATMECA (300 students, 20 faculty, 120 external teachers)
- Engineering School: Denis Barthou, Architectures (L3), Parallel Architectures (M2), Procedural Generation for 3D Games (M2), C/Algorithm projects (L3)
- Licence: Marie-Christine Counilh, Introduction to Computer Science (64HeTD), Introduction to C programming (52HeTD), L1, University of Bordeaux.

- Master MIAGE: Marie-Christine Counilh, Object oriented programming in Java (30HeTD), M1, University of Bordeaux.

9.2.2. Supervision

- PhD in progress: Ksander Ejjaaouani, Novembre 2016, Olivier Aumage, Michel Mehrenberger, Julien Bigot.
- PhD in progress: Adrien Cassagne, "Parallelization and Code Generation for Error Correcting Codes from Factor Graphs" October 2017, Olivier Aumage, Denis Barthou, Christophe Jego, Camille Leroux.
- PhD in progress: Idriss Daoudi, October 2018, Olivier Aumage, Thierry Gautier.
- PhD in progress: Romain Lion, October 2018, Samuel Thibault
- PhD in progress: Pierre Huchant, "Static analysis and dynamic adaptation of parallelism", oct. 2015, supervised by Marie-Christine Counilh, Denis Barthou.
- PhD in progress: Hugo Brunie, "Optimization of data allocations for high performance applications on heterogeneous memory architectures", oct. 2015, supervised by Julien Jaeger (CEA), Patrick Carribault (CEA) and Denis Barthou.
- Internship: Antoine Tirel, June 2018 - Sept. 2018, Emmanuelle Saillard

9.2.3. Juries

- Olivier Aumage: Ph.D defense of Adrián Castelló Gimeno at the University of Castellon Jaume I, (reviewer).
- Samuel Thibault: Ph.D defense of Germán Ceballos at the University of Uppsala (opponent).
- Denis Barthou: Ph.D defense of Van Long TRAN at Institut Telecom Sud-Paris, University Paris-Saclay (reviewer), Ph.D defense of Arnaud Durocher at the University of Bordeaux (president)

9.3. Popularization

9.3.1. Interventions

- Olivier Aumage, Emmanuelle Saillard, Denis Barthou: Welcoming of the general public for the open days at the Inria research center, October 2018.
- Emmanuelle Saillard, Corentin Salingue: Fête de la Science, Inria, October 2018.
- Emmanuelle Saillard, Corentin Salingue: Semaine des maths, Lycée Saint Genès, March 2018.
- Corentin Salingue: Printemps de la mixité, Inria, April 2018.
- Corentin Salingue: Welcoming of schoolchildren: internship of Matthieu Vigier-Lafosse, January 2018.

9.3.2. Internal action

- Emmanuelle Saillard: DevDays, October 2018
- Denis Barthou: Unithé ou café, June 2018

10. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] S. THIBAULT. *On Runtime Systems for Task-based Programming on Heterogeneous Platforms*, Université de Bordeaux, December 2018, Habilitation à diriger des recherches, https://hal.inria.fr/tel-01959127

Articles in International Peer-Reviewed Journals

[2] O. BEAUMONT, L. EYRAUD-DUBOIS, S. KUMAR. *Fast Approximation Algorithms for Task-Based Runtime Systems*, in "Concurrency and Computation: Practice and Experience", September 2018, vol. 30, n° 17 [*DOI :* 10.1002/CPE.4502], https://hal.inria.fr/hal-01878606

[3] T. COJEAN, A. GUERMOUCHE, A. HUGO, R. NAMYST, P.-A. WACRENIER. *Resource aggregation for task-based Cholesky Factorization on top of modern architectures*, in "Parallel Computing", October 2018, This paper is submitted for review to the Parallel Computing special issue for HCW and HeteroPar 16 workshops, https://hal.inria.fr/hal-01957086

[4] V. GARCIA PINTO, L. M. SCHNORR, L. STANISIC, A. LEGRAND, S. THIBAULT, V. DANJEAN. *A Visual Performance Analysis Framework for Task-based Parallel Applications running on Hybrid Clusters*, in "Concurrency and Computation: Practice and Experience", April 2018, vol. 30, n° 18, pp. 1-31 [*DOI :* 10.1002/CPE.4472], https://hal.inria.fr/hal-01616632

[5] A. GHAFFARI, M. LEONARDON, A. CASSAGNE, C. LEROUX, Y. SAVARIA. *Toward High-Performance Implementation of 5G SCMA Algorithms*, in "IEEE Access", January 2019, vol. 7, pp. 10402-10414 [*DOI :* 10.1109/ACCESS.2019.2891597], https://hal.archives-ouvertes.fr/hal-01977885

[6] M. LEONARDON, A. CASSAGNE, C. LEROUX, C. JEGO, L.-P. HAMELIN, Y. SAVARIA. *Fast and Flexible Software Polar List Decoders*, in "Journal of Signal Processing Systems", January 2019 [*DOI :* 10.1007/S11265-018-1430-3], https://hal.inria.fr/hal-01987848

International Conferences with Proceedings

[7] H. BRUNIE, J. JAEGER, P. CARRIBAULT, D. BARTHOU. *Profile-Guided Scope-Based Data Allocation Method*, in "MEMSYS 2018 - International Symposium on Memory Systems", Alexandria, United States, October 2018, https://hal.inria.fr/hal-01897917

[8] A. CASSAGNE, O. AUMAGE, D. BARTHOU, C. LEROUX, C. JEGO. *MIPP: a Portable C++ SIMD Wrapper and its use for Error Correction Coding in 5G Standard*, in "The 4th Workshop on Programming Models for SIMD/Vector Processing (WPMVP 2018)", Vienna, Austria, ACM Press, February 2018 [*DOI :* 10.1145/3178433.3178435], https://hal.inria.fr/hal-01888010

[9] K. EJJAAOUANI, O. AUMAGE, J. BIGOT, M. MEHRENBERGER, H. MURAI, M. NAKAO, M. SATO. *InKS, a Programming Model to Decouple Performance from Algorithm in HPC Codes*, in "Repara 2018 - 4th International Workshop on Reengineering for Parallelism in Heterogeneous Parallel Platforms", Turin, Italy, August 2018, pp. 1-12, https://hal.archives-ouvertes.fr/hal-01890132

[10] P. HUCHANT, D. BARTHOU, M.-C. COUNILH. *Adaptive Partitioning for Iterated Sequences of Irregular OpenCL Kernels*, in "SBAC-PAD - 30th International Symposium on Computer Architecture and High Performance Computing", Lyon, France, September 2018 [*DOI :* 10.1109/SBAC-PAD.2018.00051], https://hal.archives-ouvertes.fr/hal-01888216

[11] P. HUCHANT, E. SAILLARD, D. BARTHOU, H. BRUNIE, P. CARRIBAULT. *PARCOACH Extension for a Full-Interprocedural Collectives Verification*, in "Second International Workshop on Software Correctness for HPC Applications", Dallas, United States, November 2018, https://hal.inria.fr/hal-01937316

[12] E. SAILLARD, K. SEN, W. LAVRIJSEN, C. IANCU. *Maximizing Communication Overlap with Dynamic Program Analysis*, in "International Conference on High Performance Computing in Asia-Pacific Region", Tokyo, Japan, January 2018, https://hal.inria.fr/hal-01937407

National Conferences with Proceedings

[13] V. G. PINTO, L. MELLO SCHNORR, A. LEGRAND, S. THIBAULT, L. STANISIC, V. DANJEAN. *Detecção de Anomalias de Desempenho em Aplicações de Alto Desempenho baseadas em Tarefas em Clusters Híbridos*, in "WPerformance 2018 - 17º Workshop em Desempenho de Sistemas Computacionais e de Comunicação", Natal, Brazil, July 2018, pp. 1-14, https://hal.inria.fr/hal-01842038

Conferences without Proceedings

[14] R. PRAT, L. COLOMBET, R. NAMYST. *Combining Task-based Parallelism and Adaptive Mesh Refinement Techniques in Molecular Dynamics Simulations*, in "ICPP18, International Conference on Parallel Processing", Eugene, United States, August 2018 [*DOI :* 10.1145/3225058.3225085], https://hal.archives-ouvertes.fr/hal-01833266