EXA2PRO programming environment: Architecture and Applications

Dimitrios Soudris,
Lazaros Papadopoulos
National Technical University of Athens, Greece
dsoudris@microlab.ntua.gr
lpapadop@microlab.ntua.gr

Christoph W. Kessler
PELAB, Department of Computer and Information Science
Linköping University, Sweden
christoph.kessler@liu.se

Dionysios D. Kehagias,
Athanasios Papadopoulos,
Panos Seferlis
CERTH, Greece
diok@iti.gr
spapadopoulos@cperi.certh.gr
seferlis@cperi.certh.gr

Alexander Chatzigeorgiou,
Apostolos Ampatzoglou
CERTH, Greece
achat@uom.gr
apostolos.ampatzoglou@gmail.com

Samuel Thibault,
Raymond Namyst
Inria Bordeaux
Sud-Ouest Bordeaux, France
samuel.thibault@inria.fr
raymond.namyst@inria.fr

Dirk Pleiter
Forschungszentrum Jülich
Germany
d.pleiter@fz-juelich.de

Georgi Gaydadjiev,
Tobias Becker
Maxeler Technologies Ltd.
georgi@maxeler.com
tbecker@maxeler.com

Matthieu Haefele
Maison de la Simulation, CEA, CNRS
University of Paris-Sud, France
matthieu.haefele@
maisondelasimulation.fr

ABSTRACT

The EXA2PRO programming environment will integrate a set of tools and methodologies that will allow to systematically address many exascale computing challenges, including performance, performance portability, programmability, abstraction and reusability, fault tolerance and technical debt. The EXA2PRO tool-chain will enable the efficient deployment of applications in exascale computing systems, by integrating high-level software abstractions that offer performance portability and efficient exploitation of exascale systems' heterogeneity, tools for efficient memory management, optimizations based on trade-offs between various metrics and fault-tolerance support. Hence, by addressing various aspects of productivity challenges, EXA2PRO is expected to have significant impact in the transition to exascale computing, as well as impact from the perspective of applications. The evaluation will be based on 4 applications from 4 different domains that will be deployed in JUELICH supercomputing center. The EXA2PRO will generate exploitable results in the form of a tool-chain that support diverse exascale heterogeneous supercomputing centers and concrete improvements in various exascale computing challenges.

1 INTRODUCTION

Exascale computing is expected to benefit various aspects of human activity, by driving discoveries across a wide spectrum of scientific domains, such as biotechnology, energy and material science [5], [4]. The development of exascale systems will provide greater predictive capability and enable more realistic simulations than today's high-end petascale computer systems. The expected breakthroughs in various scientific areas will transform the global economy and will have profound impact on quality of everyday life.

Some of the most important research challenges towards the goal of exascale computing are the exploitation of the billion-way parallelism by applications, the heterogeneity, resilience, data and memory management, and energy efficiency. These challenges impose high requirements on the programming environments for emerging exascale systems. The suitable programming environments are expected to be resilience-aware and to provide increased data locality and energy efficiency. Therefore, existing software languages, tools and methodologies should be extended and enhanced to address the upcoming exascale challenges.

Scientific progress is expected to be significantly affected by the ability to effectively deploy to exascale systems complex applications that provide novel modeling and simulation solutions. Applications from scientific fields such as astrophysics, materials and energy production required many years to be developed and validated. However, as supercomputing advances, legacy applications should adapt to the new hardware and software technologies. Therefore, emerging exascale computing programming environments should be developed to provide the necessary tools and methodologies to assist this transition.

Heterogeneity is expected to play a central role in exascale systems. High-end hardware accelerators, such as GPUs and reconfigurable architectures will provide increased performance along with high energy efficiency. Programming environments are expected to provide the necessary features to efficiently exploit their capabilities by offloading computationally intensive tasks on them, while providing the necessary support to hide complexity from users.

A productive programming environment should support a broad spectrum of applications and workloads. Additionally, it should enable application developers from various scientific fields to productively write and extend with new features highly parallel applications that target complex heterogeneous exascale architectures. These applications are often developed and maintained by scientists from various domains, rather than by experts HPC application developers. Therefore, programming environments are expected to provide high productivity, both by providing features that assist non-experts to effectively develop complex applications and by optimising exascale systems resources usage.

The EXA2PRO programming environment will integrate a set of tools and methodologies that will be either developed from scratch or they will be extended to address significant challenges of the exascale systems. The main components of the EXA2PRO framework are:

- The high-level software abstractions, based on skeletons and multi-variant components that will be extended for cluster-level and heterogeneity support.
- The Composition framework which is based on an intermediate representation of applicationsâĂŹ source code. EXA2PRO "Plug-ins" to the composition framework are:
  - Static optimizations, performance analysis, and architecture and performance modeling.
  - Memory management tools that will be extended to support heterogeneity.
  - Fault tolerance support by user-exposed fault-tolerance mechanisms and by mechanisms integrated within the runtime system.
  - Technical Debt Management set of tools, that will be extended to automate trade-offs between design and runtime qualities.
- A Runtime System, extended to address various exascale challenges, including load balancing, fault tolerance and support for heterogeneity.

2 CONCEPT AND METHODOLOGY

In this section we provide detailed description of each component of the EXA2PRO tool-chain and present the EXA2PRO compilation flow, which are depicted in Figure 1 and in Figure 2, respectively.

2.1 High-level software abstractions

EXA2PRO adopts the multi-variant component and skeleton approach to properly address code and performance portability in heterogeneous systems. **Skeletons** [21] are predefined generic program building blocks for frequently occurring computation patterns such as data-parallel map, reduce, scan, stencil etc., for which efficient platform-specific implementations may exist. Where computations can be expressed with the given set of skeletons, portability

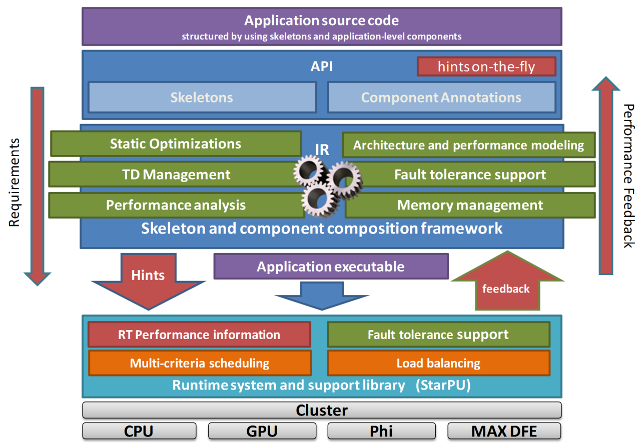

Figure 1: EXA2PRO framework architecture.

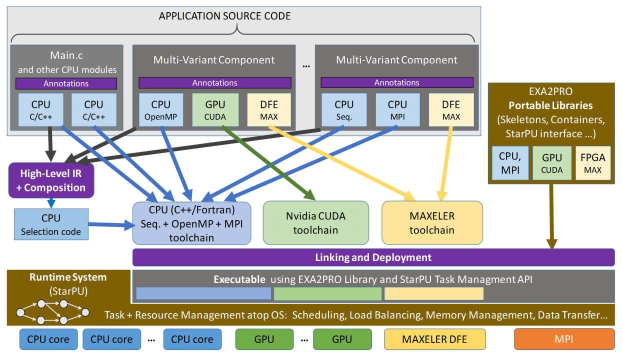

Figure 2: EXA2PRO compilation flow.

and the management of parallelism, heterogeneity, communication and synchronization come for free.

Where the existing skeletons do not fit well for some computation, a portable and flexible extension mechanism is required that allows programmers to write one or several (possibly, platform-specific) implementations in suitable programming models: **Multi-variant components** [24] are abstractions of functions with multiple implementation variants exposed to the composition framework, which will be free to select among them at each call in order to optimize execution flow for time, energy, resource usage or other objective, based on the current execution context (e.g., operand data sizes and locations, resource availability etc.). By user annotations of components and their implementations, metadata (e.g., resource requirements) and computation-specific tuning parameters (e.g., tile or buffer sizes, data structure options etc.) can be exposed. Dependences on platform properties (such as availability of platform-specific libraries, requirement of a number of nodes or cores that is, e.g., a power of 2, or temporary storage requirements) can be expressed as constraints on implementation selectability in terms of entities (names) defined in the platform model. For the latter, a flexible and extensible platform description language like XPDL [25] will be adopted and extended. For the operands to skeletons and components, we adopt and extend "smart" data-container

[19] abstractions for vectors, matrices, graphs and other exascale-relevant data structures. These are wrappers for the corresponding application data structures that abstract away the distribution of memory and possibly other storage properties such as data layout, and internally optimize data transfers and accesses.

Applications deployed in exascale systems can leverage such high level software abstractions very effectively. In the [1] and [2] EC projects, the use of skeletons and multi-variant components has been evaluated in standard benchmarks (e.g. from Rodinia) and test applications such as N-body and fluid dynamics simulations which are often deployed in supercomputing centres. Existing legacy implementations e.g., of basic linear algebra functionality, used for example in iterative solvers or in deep learning, can be easily wrapped or refactored. For example, the "Chemical processes for cleaner environment" application in EXA2PRO makes heavy use of the LU decomposition linear algebra procedure, which is already supported by the high-level software abstractions.

Skeletons and multi-variant components can significantly increase developers' productivity in exascale systems. They provide a high degree of abstraction and their implementations encapsulate all low-level platform-specific details such as parallelization, synchronization, communication, memory management and accelerator usage. Also, they are designed with emphasis on flexibility and type-safety. Moreover, due to their known (for skeletons) or declared (for user-defined components) data access and dependence structure at a coarse level of granularity, they are better suited as the basic operations in global static and dynamic program analysis, compared to non-annotated, low-level and possibly platform-specific scalar operations. For the same reason, they are well amenable to internal optimizations, such as platform-specific parameter tuning, data access optimization, or granularity adaptation. Hence, such high-level software abstractions can both improve performance and significantly reduce developersâĂŹ effort of writing and extending complex applications in exascale heterogeneous computing systems.

2.2 Composition framework, static optimizations, modeling and performance analysis

The skeletons and multi-variant components will be implemented by an integrated composition framework which will perform the static parts of application analysis, optimization, code generation and deployment. It will be organized around a generic, whole-program compiler-like core framework that browses and processes the annotations of multi-variant application components, translates skeleton calls and data container uses, builds a whole-program high-level intermediate representation (IR), and coordinates program analysis, code optimizations and composition code generation for the application to be deployed. Part of this core functionality is inspired from technical contributions to EC projects PEPPHER [20] and EXCESS [26].

The composition framework is designed to be extensible and will integrate various software modules as core plug-ins that each can access the exascale application's IR. Such modules will, e.g., provide static analysis of code quality/maintainability, design-space exploration for multi-objective (performance, energy, âĂę) optimization given Quality-of-Service (QoS) constraints, coordinate automated construction of performance and energy models for user-level components, and add extensions that address resilience of the resulting executable code. Plug-ins could be contributed even by third party providers as long as they adhere to the IR access API. The plug-in based software architecture with central IR access API is partly inspired by the CoSy compiler framework [3], but it is simpler than CoSy as the core framework as well as all plug-ins will be written in the same implementation language, namely, C++.

Static optimizations include global selection optimization [18] which, based on off-line performance and energy models, prepares co-optimized variant selections for groups of calls such as iterative loops, and global optimizations for data locality such as affinity based tiling, kernel fusion, and delayed evaluation.

The composition framework can access a high-level platform model, written in a formal language like XPDL [25], which allows to specify platform-specific selectability constraints, to provide introspection of the execution environment for retargetable tools/plug-ins and for the executable code to enable adaptivity, and to coordinate the deployment and execution of microbenchmarking code to bootstrap performance and energy models for a new platform. Finally, it interacts with the runtime system and coordinates the collection of profiling information e.g., for off-line performance modeling, in order to assist the efficient dynamic resource management by the lower layers.

2.3 Memory management optimizations

Memory management optimization techniques can significantly contribute in efficient utilization of system resources and in performance improvements. They focus on the efficient management of the available memory space, by offering customized application-specific solutions. Memory management optimizations are often based on source-to-source design space exploration methodologies and they focus on the identification of trade-offs between performance, energy consumption and memory utilization. Loop transformations for efficient exploitation of caches [22], exploration of the data structures design space [23], [27] and exploration of customized application-specific memory managers [13] are techniques that improve the utilization of memory by increasing data locality, reducing memory fragmentation and memory size requirements.

EXA2PRO will provide a set of tools for both static and dynamic memory management optimizations of exascale applications. Data Transfer and Storage Exploration (DTSE) tool performs loop transformations for efficient utilization of cache and scratchpad memories. Dynamic Data Type Refinement (DDTR) [10] and Dynamic Memory Management (DMM) [30] tools explore a wide range of the design space of application's data structures and of memory managers respectively and propose Pareto-optimal solutions in terms of performance and memory utilization.

In applications deployed in exascale architectures memory management techniques can assist in performance improvements in several ways. Low arithmetic intensity often prohibits the efficient utilization of the computational power offered by (pre-)exascale architectures. (For instance, EXA2PRO LQCD application and similar applications suffer from low arithmetic intensity). Improvements in arithmetic intensity will be examined by using the EXA2PRO memory management tools that enable source-to-source transformations.

2.4 Fault-tolerance mechanisms

Current large-scale HPC systems experience failures on the order of every few days. However, models indicate that an exascale-sized system will likely experience system failures several times an hour [28]. EXA2RPO will provide a set of fault-tolerance mechanisms based mainly on data replication. The StarPU runtime system will be extended accordingly to support these mechanisms. Application profiling at design-time will assist the identification of mechanisms which are the most suitable for each application under deployment.

Apart from cluster-level fault-tolerant support, EXA2PRO will provide mechanisms for fault-tolerance at intra-node level, as well. The runtime system will extended to provide support for virtual data [8] to keeping replicates of critical data in multiple memory storages. In case of faults, dynamic scheduling will be able to rebalance the load according to the remaining resources and the availability of valid replicates.

In addition to the above, EXA2PRO will develop and expose to the Maxeler's Data Flow Engines code designer basic primitives to easily create highly customized fault tolerant dataflow implementations at arbitrary granularity levels. DataFlow Engines (DFEs) are FPGA based PCIe devices specially designed for streaming dataflow acceleration. By integrating fault tolerance features, the area of the resulting kernel will increase and sometimes the operating frequency might also be penalized. We will evaluate the applicability of classic techniques, e.g., triple modular redundancy, data replay, rollback and rollforward, just to name few. All techniques found appropriate will be exposed through the MaxJ programming language (the first implementation of OpenSPL) and the generation of the corresponding structures will be automated.

2.5 Technical Debt Management

Technical debt (TD) is a metaphor that is used to draw an analogy between financial debt as defined in economics and the situation in which an organization decides to produce immature software artefacts (e.g., designs or source code), in order to deliver the product to market within a shorter time period [16] or to reach desired levels on run-time qualities. TD is accumulated during all development phases, i.e. requirements analysis, architectural/detailed design, and implementation, and therefore should be managed during the complete software lifecycle. TD, due to its economic valuation offers the benefit of providing an acceptable communication vehicle between technical and project managers. In particular, it acts as a mean for assessing the importance of established software engineering (SE) concepts, like quality assessment, software analytics, software aging, etc., that until recently were not easily understood by non-technical staff.

Thus, developers should be aware of the TD that they incur during application development, be informed about the particular design or code inefficiencies that they introduce and be offered the possibility to automatically address these inefficiencies by applying appropriate software refactorings. Thus, tools to continuously manage TD are necessary, to guide the partial and efficient TD repayment, the prioritization of refactoring opportunities, and the provision of design decisions that will improve the quality of software. We envision that such tools should be available both for greenfield software development (i.e. during the design and implementation of new HPC applications) as well as for brownfield software development (i.e. when scientists decide to develop/improve upon an existing HPC application). The required methods should consider all involved parameters including the effort, time and risk for performing changes, resulting benefit and implied costs. Moreover, in the context of HPC applications any decision to repay TD should be evaluated against its impact on run-time qualities of paramount importance for HPC software such as execution time and energy consumption. Although the current methods, tools and techniques for monitoring and managing TD are continually refined by researchers and practitioners, there is a lack of HPC-specific methods placing emphasis on the interplay between TD and performance and energy.

Applications deployed in exascale systems require many years to be developed and validated. They are usually developed and managed by the scientific community around the end user applications rather than professional software developers. Thus, source code maintainability and reusability are often overlooked in favor of performance and achieving scalability to larger-scale problems. EXA2PRO TD tools will enable improvements in source code quality accompanied by estimates of developersâĂŹ effort required to perform adaptive and perfective maintenance on the corresponding exascale applications. This is especially important for applications targeting exascale systems, whose evolution and incorporation of new features is often a considerably tedious task, due to the complexity of the application and the fact that the quality of the source code has often been neglected.

2.6 Run-time system

The ground of EXA2PRO will be a runtime system based on StarPU [9]. It will provide support for the combination of various programming models: it currently supports CUDA, OpenMP and OpenCL front-ends, and in the context of EXA2PRO it will use MaxJ for the support of the DFEs architecture. This will allow the composition framework to seamlessly retarget application tasks across diverse architectures.

The runtime system will also carry out efficient resource management. State-of-the-art dynamic task scheduling [7] will be extended to selecting among the task variants proposed by the composition framework, as well as combining multiple scheduling criteria, including task performance, data transfers [8], and energy consumption. This will allow to abstract the question of performance out of the writing of application software. The runtime will however provide performance feedback through an API targeted to the composition framework as well as to the end-user. This will instigate a performance analysis loop, between actual execution results and the composition framework performance models, while showing the results within a dashboard, so the end-user gets a glimpse into the optimization process, and gets hints on which parts of the application algorithm needs rethinking for more parallelism, for instance. Malleability of the execution of the application will

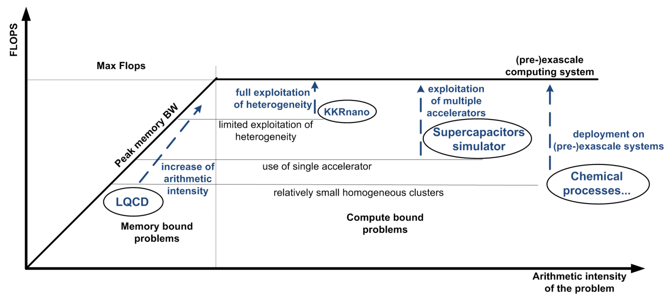
FLOPS
Max Flops
(pre-)exascale computing system
Peak memory BW
full exploitation of heterogeneity
KKRnano
limited exploitation of heterogeneity
exploitation of multiple accelerators
Supercapacitors simulator
deployment on (pre-)exascale systems
increase of arithmetic intensity
use of single accelerator
Chemical processes...
LQCD
relatively small homogeneous clusters
Memory bound problems
Compute bound problems
Arithmetic intensity of the problem

Figure 3: Summarizes imitations of applications and goals in the context of EXA2PRO.

be supported through dynamic load balancing, thus allowing the architecture to be reconfigurable dynamically. Finally, extensions to runtime system for fault-tolerance support have been described in the "Fault-tolerance mechanisms" subsection above.

3 APPLICATIONS

The selection of the applications for the evaluation of EXA2PRO environment is based on the following two criteria:

- *Exploitation of exascale systems.* The applications provide the required features to exploit the exascale computing technology. By deployed in emerging exascale systems effectively, they are expected to provide significant results (breakthrough solutions to important scientific challenges) that are not possible to provide with the computational power offered by existing petascale technologies.
- *Impact to various domains in both the scientific community and the industry.* The results that will be obtained by the deployment of the applications in exascale systems are expected to have significant impact in a wide range of scientific and industrial domains. The applications belong to the following 4 areas: a) high-energy physics, b) material science, c) chemical processes for cleaner environment and d) energy storage.

The goals in the context of the EXA2PRO for each application are summarized in Figure 3 and detailed in the following subsections.

3.1 LQCD

High Energy Physics: Lattice Quantum Chromodynamics (LQCD) Typically, the most compute intensive part of LQCD simulations is a linear solver, for which iterative algorithms are used. These algorithms require repeated multiplications of a sparse matrix and a vector. Due to the regular structure of the sparse matrix, within the scheme of the Berkeley Dwarfs this numerical task falls in the class of Structured Grids. LQCD applications feature a high level of intrinsic parallelism within a single performance dominating kernel, which facilitates porting to different massively-parallel architectures.

As for many other applications from this class, a relatively low arithmetic intensity is a major challenge for exploiting the performance of current compute devices. Towards exascale, applications from this field have to become significantly more parallel. Partially, this will happen by increasing the extension of the simulation lattice in all 4 dimensions. Due to its communication patterns, scaling to significantly larger systems compared to those available today can be considered feasible.

3.2 Materials science: KKRnano

This application implements a particular formulation of the Density Functional Theory (DFT) methods. It addresses the challenge of the unfavourable scaling behavior of DFT as a function of the number of atoms by systematic truncation of the long-range interactions. For many interesting classes of materials it is important to include an increasingly large number of atoms. Simulating 10,000s of atoms requires petascale supercomputers.

The most time consuming part of the application consists of an iterative solver based on a variant of the QuasiMinimal Residual (QMR) method. The numerical task thus falls into the class "Dense Linear Algebra". Future exascale systems should allow using KKRnano for performing calculations involving 105-106 and more atoms. This would allow to explore the properties of completely new classes of materials.

3.3 Chemical processes for cleaner environment: Material and process design for CO2 capture

The efficient capture of CO2 is reasonably regarded as one of the grand challenges for the 21st century [17], considering that a single coal-fired plant may emit up to 25 million tons of CO2 per year. The design of CO2 capture materials and processes enabling low cost penalties is a very complex problem with a huge design space, due to the extremely large number of process design decisions and the variety of material structures. Since conventional CO2 capture unit installation cost on a power plant costs hundreds of million of Euros, cost reduction of a modest 2-3% is very significant. In the context of EXA2PRO, the application will be implemented on large clusters and exploit acceleration to provide optimal CO2 capture materials and systems.

3.4 Energy storage: Supercapacitors simulation (MetalWalls)

The urgent need for efficient energy storage has resulted in a widespread and concerted research effort in the past ten years. Current battery technologies are very efficient in terms of energy storage density but reach their limit when large amounts of energy have to be stored or retrieved on short time scales. Supercapacitors can be seen as complementary devices with smaller energy storage densities, but which can be operated on short time scales. These devices are already replacing batteries in high power applications. In the future this trend is expected to accelerate, particularly for the recovery of kinetic energy in electric vehicles [15]. A major challenge remains to be solved in order to determine the relevant quantities for the target objectives: electrical capacitance, amount of adsorbed ions and diffusion coefficient as a function of the electrolyte composition and of the potential difference between the electrodes. This information cannot easily be obtained from experiments and traditional models do not work in this case where interactions at the molecular level play an essential role. "Metalwalls" is a classical molecular dynamics code able to simulate supercapacitors with an accuracy that put this numerical tool in a world leading position [12]. In the context of this project, Metalwalls will leverage the EXA2PRO environment to be efficiently implemented in a heterogeneous computing system.

3.5 EXA2PRO benchmarks

The EXA2PRO benchmarks will be used mainly in the early and middle phases of the project for providing proofs of concepts and for early evaluation of tools and methodologies during their development. They have been carefully selected to provide the required complexity, however, the evaluation of ideas and methodologies in the benchmarks is expected to require significantly less programming effort than in the 4 aforementioned applications.

Neuron Computing: The InfOli simulator is a neuron network simulator, specializing in providing accurate, biophysically-meaningful insight into how a network comprised of human neurons operates [14]. The simulation of massive dense networks of the neuron model is very challenging, due to the high demand in floating-point operations per simulation cycle. Single-node deployments in multiple platforms [29], include x86-based architectures (Intel Xeon and Intel Xeon Phi), GPUs and MAX DFEs. The simulation will be used during the implementation and the early evaluation of the heterogeneity and multi-node extensions of the EXA2PRO environment.

Collection of Small Multi-Variant Benchmarks: For the necessary broad quantitative evaluation of performance/energy prediction and selection optimization techniques for multi-variant computations to be developed in this project, we will extract a set of relatively small C/C++-based benchmark programs, covering different domains and characteristics, from existing benchmark suites. These will complement the larger applications of the project as listed above. In particular, we will select benchmarks that already provide multiple implementation variants (e.g. CPU, GPU, multi-threaded, and/or multi-node) from third parties to allow for a direct comparison with the base-line programming models, also in programmability and code complexity aspects. Such benchmark codes will be taken from well-established benchmark suites in parallel and heterogeneous computing, e.g. from Rodinia and PARSEC.

Dense Linear Algebra: The Chameleon solver is a dense linear algebra library, which notably includes the classical Cholesky, LU, QR factorizations. It is part of the MORSE (Matrices Over Runtime Systems @ ExaScale) project , which aims at leveraging runtime systems for dense and sparse linear algebra. Chameleon is thus a collection of state-of-the-art dense linear algebra algorithms, which are expressed as task graphs [6], allowing for more parallelism than the previous vectorized approaches (such as LINPACK) or panelled approaches (such as LAPACK), thanks to more task pipelining. It also seamlessly includes support for GPU accelerators and distributed MPI execution, thanks to the underlying runtime system. Chameleon currently supports the PARSEC, the Quark, and the StarPU runtime systems. The Cholesky factorization of Chameleon was shown to scale over the StarPU runtime system, but does not cope with faults yet, therefore it will be used as initial work on fault tolerance support in the context of EXA2PRO. It also supports only simple performance reporting, and will benefit from more advanced performance feedback.

4 EXA2PRO EXPECTED IMPACT

In this section we highlight the impact of EXA2PRO in the programming environments for exascale computing and the economic impact on the CO2 capture and energy storage scientific and industrial domains.

Programming environments for exascale systems Programming of future supercomputing architectures will become significantly more challenging for various reasons. As the clocks of compute devices can hardly be increased (and might even drop), future architectures will become more parallel and applications will have to be able to exploit the parallelism at different levels. Additionally, memory architectures are likely to become more complex and involve different memory technologies with different performance and capacity characteristics. Finally, architectures might become more heterogeneous involving different type of processor cores and (possibly reconfigurable) accelerators. Users do not only have to be able to exploit the capabilities of these more complex hardware architectures, given the significant investments into the applications, these codes need to remain (performance) portable and able to use different architectures. Addressing this challenge will require to pursue a further split of concerns, to elevate science domain specialists and engineers from the burden of porting/optimising applications to/for different architectures. Programming models will play an important role.

The vast majority of applications on todayâĂŹs supercomputers, as they are operated at JUELICH, are parallelised using MPI. At the node level, multi-threading is used as an additional level of parallelisation, for which most often OpenMP is used. A smaller set of applications also explicitly addresses the data-parallelism provided by increasingly wide SIMD ISAs, e.g. IntelâĂŹs AVX-512 instructions introduced with the most recent generation of Xeon Phi processors. Finally, a subset of applications uses GPUs as accelerators, for which programming languages like CUDA or OpenCL (to a lesser extend) as well as directive-based programming models like OpenACC are used.

In future it can be expected that application developers will also have to address the topic of resilience as in future jobs will depend on the absence of major failures for a larger number of components. As of today the main strategy for coping with such errors is checkpoint-restart. The integration of such techniques as of today largely remains with application developers. Only for optimization of the required I/O operations support is provided by supercomputing centers.

EXA2PRO addresses the above challenges by proposing a programming environment that enables the efficient exploitation of accelerators in heterogeneous exascale computing systems. It will support various programming models (including OpenCL, OpenMP, MPI, MaxJ/OpenSPL, StarPU) and enable the interoperability between them, to provide performance portability. Additionally, it will integrate several fault tolerance mechanisms, at different levels: User-exposed fault-tolerance mechanisms for accelerators will be accessible to application developers. The runtime system will provide both redundancy and rollback-recovery mechanisms. These mechanisms will be evaluated in the 4 EXA2PRO applications and trade-offs between fault-tolerance/performance/energy and source code quality will be investigated.

EXA2PRO will contribute to the exploitation of emerging exascale systems by applications from various scientific areas. We expect large number applications to be deployed in (pre-)exascale systems and to utilize computational resources and heterogeneity more efficiently. Therefore, EXA2PRO will significantly contribute in improving today's predictive capabilities of exascale applications and in the realization of more realistic simulations in various scientific fields. EXA2PRO is expected to have significant impact to the scientific community and to the industry that focuses on applications targeting exascale systems. Emerging exascale systems should be accessible to scientists from a wide range of domains, with limited expertise in application development in large-scale HPC systems. Considering the complexity of the exascale systems and that of the applications targeting them, the EXA2PRO toolchain can significantly contribute towards broadening their usage by scientists from various domains. The fact that the EXA2PRO environment addresses critical exascale systemsâĂŹ challenges, such as performance portability and fault-tolerance, allows application developers to focus on algorithmic development only and therefore it boosts productivity.

EXA2PRO aims at providing a set of tools to assist the deployment of applications from various scientific fields in exascale computing systems. For instance, the high-level software abstractions offered by EXA2PRO (skeletons and multi-variant components) that support multiple programming languages (C/C++ and FORTRAN), along with the tools for improving the maintainability of the source code, will enable applications that have been deployed in conventional HPC systems to exploit the computational power of the emerging exascale architectures. Additionally, the computational power offered by exascale systems may drive the evolution of applications and their enhancement with new features. Thus, EXA2PRO will significantly contribute to the productive development of new features in existing applications with reduced programmersâĂŹ effort, by minimizing the amount of time that developers allocate in issues that are unrelated to development of application algorithms.

4.1 Economic impact

2016 reports show that the HPC market continues to grow and that the annual growth rate is forecasted to reach 8% by the end of the decade, while HPC applications growth rate was 7% in 2016 . Performance-critical application domains including scientific research, security and personalized medicine continue to fuel this growth. Server and storage sales have the highest contribution in the HPC market revenue. EXA2PRO programming environment can contribute to the even higher growth of the HPC market by making HPC and supercomputing systems more accessible to the industry and to the scientific community. Indeed, EXA2PRO will enable scientific applications with significant economic impact to benefit from the computational power offered by exascale systems.

We expect indirect economic impact by improving usability of supercomputers and enabling their use for computational science and engineering applications with high commercial impact. KKRnano, "Chemical processes for cleaner environment" and Supercapacitors simulation EXA2PRO applications cover different industrial areas, for which research on new materials is relevant, as well as research related to the grand societal challenge of the energy transition.

4.2 Economic impact of the Chemical processes for cleaner environment application

For the case of chemical absorption CO2 capture systems, a detailed techno-economic analysis performed in the recent FP7 CAPSOL project has shown that the costs of installing a CO2 capture system are in the order of 400M euros, whereas the augmented annualized costs are in the order of 180M euros/year. The difference in investment costs between a feasible and a locally optimum CO2 capture design is in the order of 30M euros, i.e. the savings may be 10% of the investment costs. The selection of an appropriate material for a process system which is optimized for the chemical characteristics of this material is of great importance. For example, Monoethanolamine (NH2-CH2-CH2-OH) and Monopropanolamine (NH2-CH2-CH2-CH2-OH) are two very similar materials (solvents) used for CO2 capture purposes. As a result of their chemical similarity their molecular properties are also very similar. However, a detailed techno-economic analysis performed in the recent FP7 CAPSOL project has shown that the process investment costs that correspond to each material may differ by over 15M euros.

With the technologies envisaged in EXA2PRO we expect that an overall augmented annualized cost reduction of 30-40% is possible. The capital costs can be reduced by up to 20% and the operating costs can be reduced by up to 50% using advanced solvents and global optimization. This is because we expect to obtain solutions of very low regeneration energy requirements (e.g. 50% lower than conventional MEA due to the use of advanced low energy solvents), which will also result in lower flowrates around the columns and equipment of considerably lower sizes.

For the case of physical absorption CO2 capture systems, [11] has shown that even with local optimization they were able to increase the net present value (NPV) by approximately 90% (i.e. meaning that the investment would be profitable for the 15 years lifetime of the project). This was achieved by simply increasing their solvent design space by 1 chemical group; they introduced the ether group in their initial alkane-based solvent structural options.

In EXA2PRO we will consider over 40 different functional groups and also the size of the equipment as additional decision options. As a result, we will investigate a very wide design space compared to [11]. With such a wide design space we expect that a 180% increase in the NPV compared to using pure alkanes is feasible.

A major impact of the EXA2PRO in the CO2 capture and in the broader chemical industry is the acceleration of time-to-market of new products and processes. The extreme computational power offered by exascale computing will allow the virtual prototyping of chemical products, equipment and processes as it will allow simulations of very high accuracy. This will greatly reduce (and possibly eliminate) the need for costly lab and pilot-plant experiments.

4.3 Economic impact of the Metalwalls application

In contrast to conventional Li-ion batteries, the charging of supercapacitor is not limited by diffusion of the ions into solid materials, and hence high power densities can be achieved. Supercapacitors are therefore used in all applications in which high power deliveries or uptake is needed (electronics, hybrid transport, grid balancing...). The market growth over the past years has been of more than 30% / year. It is now valued at 500M euros in 2015, but it is expected to reach more than 2B euros by 2022. The key factors driving the supercapacitor market are the increase of the storage capabilities, the low equivalent series resistance and additional aspects such as resistance to moisture or light weight.

Over the past ten years, the transfer of knowledge from fundamental research to the industry has led to an increase by a factor of 2 of the storage capability, with devices now reaching specific energies beyond 6-7 Wh / kg. This enhancement of performance is mostly due to the spreading of the use of nanoporous carbon electrodes, which allow to increase the specific capacitance of the electrode/electrolyte interface compared to the conventional mesoporous carbons. The next challenge will be to push this limit further (>10 Wh /kg) in order to establish supercapacitors as ideal candidates in other key applications. In this perspective, molecular simulations are likely to play an important role. Over the past five years, they were shown to be very accurate for a series of well-characterized system. The use of exascale computing will allow to study a wide range of new electrode / electrolyte combinations and to provide to the industry the necessary structure / property relationship that are necessary for finding the capacitance optimum. Simulations will also allow to test new concepts of energy storage, such as pseudo-capacitors or redox supercapacitors. Although very prospective, any breakthrough in the field could lead to very large improvements on the specific energy. Any improvement of the performances predicted from our simulations will immediately be tested in his experimental laboratory, thus ensuring a rapid transfer towards the industry.

5 ACKNOWLEDGEMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programme EXA2PRO under grant agreement No 801015 (www.exa2pro.eu).

REFERENCES

[1] 2012. PEPPHER FP7 project. http://www.peppher.eu/

[2] 2016. EXCESS FP7 project. http://www.excess-project.eu/

[3] 2017. Ace. http://www.ace.nl/compiler/cosy.html

[4] 2017. DOE USA report. https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf

[5] 2017. ETP4HPC report. http://www.etp4hpc.eu/en/sra.html

[6] E. Agullo and et al.. 2011. A Hybridization Methodology for High-Performance Linear Algebra Software for GPU. *GPU Computing Gems, Jade Edition. W Hwu eds. Elsevier* 2 (2011), pp. 473–484.

[7] E. Agullo and et al.. 2015. Bridging the Gap between Performance and Bounds of Cholesky Factorization on Heterogeneous Platforms. In *IEEE International Parallel and Distributed Processing Symposium Workshop*. pp. 34–45.

[8] C. Augonnet, J. Clet-Ortega, S. Thibault, and R. Namyst. 2010. Data-Aware Task Scheduling on Multi-accelerator Based Platforms. In *IEEE 16th International Conference on Parallel and Distributed Systems*. pp. 291–298.

[9] C. Augonnet, S.l Thibault, R. Namyst, and P. Wacrenier. 2011. StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures. *Concurr. Comput. : Pract. Exper.* 23, 2 (2011), pp. 187–198.

[10] A. Bartzas and et al.. 2006. Dynamic data type refinement methodology for systematic performance-energy design exploration of network applications. In *Proceedings of the Design Automation Test in Europe Conference*, Vol. 1.

[11] J. Burger and et al.. 2015. A hierarchical method to integrated solvent and process design of physical CO2 absorption using the SAFT Mie approach. *AIChE Journal* 61, 10 (2015).

[12] Merlet C. and et. al.. 2012. On the molecular origin of supercapacitance in nanoporous carbon electrodes. *Nature materials* 11 (2012).

[13] F. Catthoor, E. de Greef, and S. Suytack. 1998. *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design.* Kluwer Academic Publishers, Norwell, MA, USA.

[14] G. Chatzikonstantis, D. Rodopoulos, C. Strydis, C. I. De Zeeuw, and D. Soudris. 2017. Optimizing Extended Hodgkin-Huxley Neuron Model Simulations for a Xeon/Xeon Phi Node. *IEEE Transactions on Parallel and Distributed Systems* 28, 9 (2017), pp. 2581–2594.

[15] E.B. Conway. 2013. *Electrochemical supercapacitors: scientific fundamentals and technological applications.* Springer Science and Business Media.

[16] W. Cunningham. 1992. The WyCash Portfolio Management System. In *Addendum to the Proceedings on Object-oriented Programming Systems, Languages, and Applications (Addendum) (OOPSLA '92)*. pp. 29–30.

[17] D.M. D'Alessandro, M. Deanna, B. Smit, and J.R. Long. 2010. Carbon dioxide capture: prospects for new materials. In *Angewandte Chemie International Edition 49.35*. pp. 6058–6082.

[18] U. Dastgeer and C. Kessler. 2013. A Framework for Performance-Aware Composition of Applications for GPU-Based Systems. In *42nd International Conference on Parallel Processing*. pp. 698–707.

[19] U. Dastgeer and C. Kessler. 2016. Smart Containers and Skeleton Programming for GPU-Based Systems. *Int. J. Parallel Program.* 44, 3 (2016), pp. 506–530.

[20] U. Dastgeer, L. Li, and C. Kessler. 2012. The PEPPHER Composition Tool: Performance-Aware Dynamic Composition of Applications for GPU-Based Systems. In *SC Companion: High Performance Computing, Networking Storage and Analysis*. pp. 711–720.

[21] J. Enmyren and C. Kessler. 2010. SkePU: A Multi-backend Skeleton Programming Library for multi-GPU Systems. In *Proceedings of the Fourth International Workshop on High-level Parallel Programming and Applications (HLPP '10)*. pp. 5–14.

[22] M. Jayapala and et al.. 2005. Clustered loop buffer organization for low energy VLIW embedded processors. *IEEE Trans. Comput.* 54, 6 (2005), 672–683.

[23] C. Jung and et al.. 2011. Brainy: Effective Selection of Data Structures. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '11)*. pp. 86–97.

[24] C. Kessler and et al.. 2012. Programmability and performance portability aspects of heterogeneous multi-/manycore systems. In *Design, Automation Test in Europe Conference Exhibition (DATE)*. pp. 1403–1408.

[25] C. Kessler, L. Li, A. Atalar, and A. Dobre. 2015. XPDL: Extensible Platform Description Language to Support Energy Modeling and Optimization. In *44th International Conference on Parallel Processing Workshops*. pp. 51–60.

[26] C. Kessler, L. Li, A. Atalar, and A. Dobre. 2016. An Extensible Platform Description Language Supporting Retargetable Toolchains and Adaptive Execution. In *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES '16)*. pp. 194–196.

[27] L. Liu and S. Rus. 2009. Perflint: A Context Sensitive Performance Advisor for C++ Programs. In *International Symposium on Code Generation and Optimization*. pp. 265–274.

[28] E. Meneses, X. Ni, G. Zheng, C. L. Mendes, and L. V. Kale. 2015. Using Migratable Objects to Enhance Fault Tolerance Schemes in Supercomputers. *IEEE Transactions on Parallel and Distributed Systems* 26, 7 (2015), pp. 2061–2074.

[29] G. Smaragdos and et al.. 2017. A node-level heterogeneous accelerator platform for neuron simulations. *Journal of Neural Engineering* (2017).

[30] S. Xydis, A. Bartzas, I. Anagnostopoulos, D. Soudris, and K. Pekmestzi. 2010. Custom multi-threaded Dynamic Memory Management for Multiprocessor System-on-Chip platforms. In *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*. pp. 102–109.