The ANTAREX Approach to Autotuning and Adaptivity for Energy Efficient HPC Systems

Cristina Silvano, Giovanni Agosta, Stefano Cherubin, Davide Gadioli, Gianluca Palermo
DEIB – Politecnico di Milano
name.surname@polimi.it

Andrea Bartolini, Luca Benini
IIS – Eidgenössische Technische Hochschule Zürich
{barandre, lbenini}@iis.ee.ethz.ch

Jan Martinovič, Martin Palkovič, Kateřina Slaninová
IT4Innovations, VSB – Technical University of Ostrava
name.surname@vsb.cz

João Bispo, João M. P. Cardoso, Pedro Pinto
FEUP – Universidade do Porto
{jbispo, jmpc, pmsp}@fe.up.pt

Carlo Cavazzoni, Nico Sanna
CINECA
n.surname@cineca.it

Andrea R. Beccari
Dompé Farmaceutici SpA
andrea.beccari@dompe.it

Radim Cmar
Sygic
rcmar@sygic.com

Erven Rohou
INRIA Rennes
erven.rohou@inria.fr

ABSTRACT

The main goal of the ANTAREX [1] project is to express by a Domain Specific Language (DSL) the application self-adaptivity and to runtime manage and autotune applications for green and heterogeneous High Performance Computing (HPC) systems up to the Exascale level. Key innovations of the project include the introduction of a separation of concerns between self-adaptivity strategies and application functionalities. The DSL approach will allow the definition of energy-efficiency, performance, and adaptivity strategies as well as their enforcement at runtime through application autotuning and resource and power management.

Keywords

High Performance Computing, Autotuning, Adaptivity, DSL, Compilers, Energy Efficiency

1. INTRODUCTION

High Performance Computing (HPC) has been traditionally the domain of grand scientific challenges and a few industrial domains such as oil & gas or finance, where investments are large enough to support massive computing infrastructures. Nowadays HPC is recognized as a powerful technology to increase the competitiveness of nations and their industrial sectors, including small scale but high-tech businesses – *to compete, you must compute* has become an ubiquitous slogan [1]. The current road-map for HPC systems aims at reaching the Exascale level (10^{18} FLOPS) within the $2023-24$ timeframe – with a $\times 1000$ improvement over Petascale, reached in 2009, and a $\times 100$ improvement over current systems. Reaching Exascale poses the additional challenge of significantly limiting the energy envelope, while providing massive increases in computational capabilities – the target power envelope for future Exascale system ranges between 20 and 30 MW. Thus, "Green" HPC systems are being designed aiming at maximizing a FLOPS/W metric, rather than just FLOPS, and employing increasingly heterogeneous architectures with GPGPU or MIC accelerators. On average, the efficiency of heterogeneous systems is almost three times that of homogeneous systems (i.e., $7,032$ MFLOPS/W vs $2,304$ MFLOPS/W[2]). This level of efficiency is still two orders of magnitude lower than that needed for supporting Exascale systems at the target power envelope of 20 MW. To this end, European efforts have recently been focused on building supercomputers out of the less power-hungry ARM cores and GPGPUs [2]. On the other hand, the wide margin provided by modern chip manufacturing techniques, combined with the inability to exploit this space to produce faster, more complex cores due to the breakdown of Dennard scaling [3], has given rise to a pervasive diffusion of a number of parallel computing architectures, up to the point where even embedded systems feature multicore processors. The huge design effort has led to a variety of approaches, in terms of core interconnection and data management. Thus, the ability to port applications designed for current platforms, based on GPGPUs like the NVIDIA Kepler or Tesla families, to heterogeneous systems such as those currently designed for embedded systems is critical to provide software support for future HPC.

Designing and implementing HPC applications is a difficult and complex task, which requires mastering several specialized languages and tools for performance tuning. This is incompatible with the current trend to open HPC infrastructures to a wider range of

[1]ANTAREX is supported by the EU H2020 FET-HPC program under grant 671623

[2]www.green500.org, June 2015

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

users. The current model where the HPC center staff directly supports the development of applications will become unsustainable in the long term. Thus, the availability of effective standard programming languages and APIs is crucial to provide migration paths towards novel heterogeneous HPC platforms as well as to guarantee the ability of developers to work effectively on these platforms. To fulfil the 20MW target, energy-efficient heterogeneous supercomputers need to be coupled with a radically new software stack capable of exploiting the benefits offered by heterogeneity at different levels (supercomputer, job, node).

ANTAREX addresses these challenging problems through a holistic approach spanning all the decision layers composing the supercomputer software stack and exploiting effectively the full system capabilities (including heterogeneity and energy management). The main goal of the ANTAREX project is to express by a DSL the application self-adaptivity and to runtime manage and autotune applications for green heterogeneous HPC systems up to Exascale.

One key innovation of the proposed approach consists of introducing a separation of concerns, where self-adaptivity and energy efficient strategies are specified aside to the application functionalities. This is promoted by the definition of a DSL inspired by aspect-oriented programming concepts for heterogeneous systems. The new DSL will be introduced for expressing at compile time the adaptivity/energy/performance strategies and to enforce at runtime application autotuning and resource and power management. The goal is to support the parallelism, scalability and adaptivity of a dynamic workload by exploiting the full system capabilities (including energy management) for emerging large-scale and extreme-scale systems, while reducing the Total Cost of Ownership (TCO) for companies and public organizations.

ANTAREX approach will be based on: (1) Introducing a new DSL for expressing adaptivity and autotuning strategies; (2) Enabling the performance/energy control capabilities by tuning software knobs (including application parameters, code transformations and code variants); (3) Designing scalable and hierarchical optimal control-loops capable of dynamically leveraging them together with performance/energy control knobs at different time scale (compile-, deploy- and run-time) to always operate the supercomputer and each application at the maximum energy-efficient and thermally-safe point. This can be done by monitoring the evolution of the supercomputer as well as the application status and requirements and bringing this information to the ANTAREX energy/performance-aware software stack.

The ANTAREX project is driven by two use cases taken from highly relevant HPC application scenarios: (1) A biopharmaceutical HPC application for drug discovery deployed on the 1.21 PetaFlops heterogeneous NeXtScale Intel-based IBM system based at CINECA and (2) A self-adaptive navigation system for smart cities deployed on the server-side on the 1.46 PetaFlops heterogeneous Intel® Xeon Phi™ based system provided by IT4Innovations National Supercomputing Center. All the key ANTAREX innovations will be designed and engineered since the beginning to be scaled-up to the Exascale level. Performance metrics extracted from the two use cases will be modelled to extrapolate these results towards Exascale systems expected by the end of 2023.

The ANTAREX Consortium comprises a wealth of expertise in all pertinent domains. Four top-ranked academic and research partners (Politecnico di Milano, ETHZ Zurich, University of Porto and INRIA) are complemented by the Italian Tier-0 Supercomputing Center (CINECA), the the Tier-1 Czech National Supercomputing Center (IT4Innovations) and two industrial application providers, one of the leading biopharmaceutical companies in Europe (Dompé) and the top European navigation software company (Sygic).

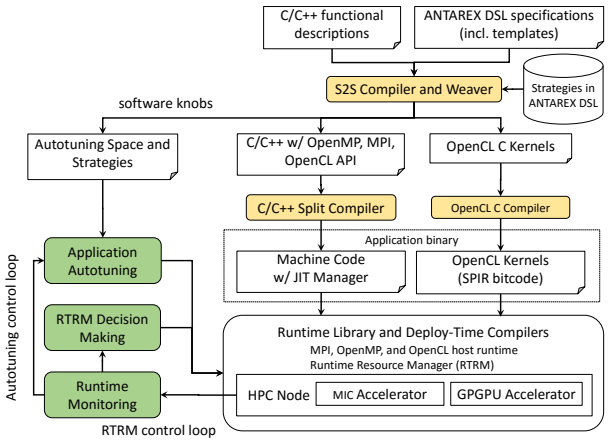

Figure 1: The ANTAREX Tool Flow

2. THE ANTAREX APPROACH

The ANTAREX approach and related tool flow, shown in Figure 1, operates both at design-time and runtime. The application functionality is expressed through C/C++ code (possibly including legacy code), whereas the non-functional aspects of the application, including parallelisation, mapping, and adaptivity strategies are expressed through the DSL developed in the project. One of the benefits consists of facilitating the reuse of legacy code. In the definition of these strategies, the application developer or system integrator can leverage DSL templates that encapsulate specific mechanisms, including how to generate code for OpenCL or OpenMP parallelisation, and how to interact with the runtime resource manager. The DSL weaver and refactoring tool will then enhance the C/C++ functional specification with the desired adaptivity strategies, generating a version of the code that includes the necessary libraries as well as the partitioning between the code for the general-purpose processors and the code for the accelerators (such as GPGPUs and MIC accelerators [4]). A mix of off-the-shelf and custom compilers will be used to generate code, balancing development effort and optimization level.

The ANTAREX compilation flow leverages a runtime phase with compilation steps, through the use of split-compilation techniques. The application autotuning is delayed to the runtime phase, where the software knobs (application parameters, code transformations and code variants) are configured according to the runtime information coming from the execution environment. Finally the runtime resource and power manager are used to control the resource usage for the underlying computing infrastructure given the changing conditions. At runtime, the application control code, thanks to the design-time phase, now contains also runtime monitoring and adaptivity strategies code derived from the DSL extra-functional specification. Thus, the application is continuously monitored to guarantee the required Service Level Agreement (SLA), while communication with the runtime resource-manager takes place to control the amount of processing resources needed by the application. The application monitoring and autotuning will be supported by a runtime layer implementing an application level collect-analyse-decide-act loop.

3. THE ANTAREX DSL

HPC applications might profit by adapting to operational and situational conditions, such as changes in contextual information (e.g., workloads), in requirements (e.g., deadlines, energy), and

in resources availability (e.g., connectivity, number of processor nodes available). A simplistic approach to both adaptation specification and implementation (see, e.g., [5]) employs hard coding of, e.g., conditional expressions and parameterizations. In our approach, the specification of runtime adaptability strategies will rely on a DSL implementing key concepts from Aspect-Oriented Programming (AOP) [6]. AOP is based on the idea that the certain requirements of a system (e.g., target-dependent optimizations) should be specified separately from the source code that defines the functionality of the program. An extra compilation step, performed by a *weaver*, basically merges the original source code and the aspects into the intended program [7]. Using aspects to separate secondary concerns from the core objective of the program can result in cleaner programs and increased productivity (e.g., higher reusability of target-dependent strategies). Moreover, the development process of HPC applications typically involves two types of experts (application-domain experts and HPC system architects) that can split their responsibilities along the boundary of functional description and extra-functional aspects.

3.1 LARA for Runtime Adaptivity and Compiler Optimization

LARA [8, 9] is an AOP approach to developers to capture non-functional requirements and concerns in the form of strategies, which are decoupled from the functional description of the application. Compared to other approaches that usually focus on code injection (e.g., [10] and [11]), LARA provides access to other types of actions, e.g., code refactoring and inclusion of additional information. Additional types of actions may be defined in the language specification and associated weaver, such as software/hardware partitioning [12] or compiler optimization sequences [13].

Figure 2 shows an example of a LARA aspect that profiles function calls in order to gather information about argument values and their frequency. It injects code for an external C library that monitors and stores the name of the function being called, its location and the value of the arguments. A concern intended to be applied over the target application is expressed as one or more aspect definitions, or *aspectdef*, the basic modular unit of LARA. Aspects, similarly to functions, can receive inputs, and return outputs.

```
aspectdef ProfileArguments
  input funcName end
  select fCall end
  apply
    insert before %{profile_args('[[funcName]]',
                   [[$fCall.location]],
                   [[$fCall.argList]]);
    }%;
  end
  condition $fCall.name == funcName end
end
```

Figure 2: Example of LARA aspect for profiling.

An aspect is comprised of three main steps. Firstly, one captures the points of interest in the code using a *select* statement, which in the example selects function calls. Then, using the *apply* statement, one acts over the selected program points. In this case, it will insert code before the function call. We can then define a *condition* statement to constrain the execution of the *apply* (i.e., only the function with the provided name is selected). LARA promotes modularity and aspect reuse, and to allow the definition of more sophisticated concerns, it is possible to embed JavaScript code inside aspects.

Figure 3 presents an example of a LARA aspect that unrolls innermost `FOR` loops whose iteration count is less than a given threshold (it uses an action supported by the weaver - keyword `do`). In this example this aspect will be called in a dynamic context after we specialize a function for a given argument value.

```
aspectdef UnrollInnermostLoops
  input $func, threshold end
  select $func.loop{type=='for'} end
  apply
    do LoopUnroll('full');
  end
  condition
    $loop.isInnermost && $loop.numIter <= threshold
  end
end
```

Figure 3: Example of LARA aspect for loop unrolling.

Figure 4 shows an example of a possible dynamic aspect. This aspect specifies the monitoring of calls to the function `kernel` in runtime, and specialize it if the runtime value of parameter `size` is between a range defined by `lowT` and `highT`. First, it statically prepares the function call to support several versions of the function, according to the parameter `size` (the keyword `call` calls an aspect). Then, dynamically, it specializes the function call for the current value of the parameter `size`, and unrolls the loops of the newly developed specialized function. Finally, it adds the specialized version as one of the possible function variants that can be called.

```
aspectdef SpecializeKernel
  input lowT, highT end

  call spCall: PrepareSpecialize('kernel','size');

  select fCall{'kernel'}.arg{'size'} end
  apply dynamic
    call spOut : Specialize($fCall, $arg.name,
      $arg.runtimeValue);
    call UnrollInnermostLoops(spOut.$func,
      $arg.runtimeValue);
    call AddVersion(spCall, spOut.$func,
      $arg.runtimeValue);
  end
  condition
    $arg.runtimeValue >= lowT &&
    $arg.runtimeValue <= highT
  end
end
```

Figure 4: Example of LARA aspect with dynamic weaving.

3.2 ANTAREX DSL Concepts

The current LARA infrastructure represents a solid foundation to build a more sophisticated DSL that will enable us to specify runtime adaptability strategies.

The ANTAREX DSL approach aims at reaching a higher abstraction level, to separate and express data communication and computation parallelism, and to augment the capabilities of existing programming models by passing hints and metadata to the compilers for further optimization. The approach aims at improving performance portability with respect to current programming models, such as OpenCL, where fine-tuning of performance (which is very sensitive to even minimal variation in the architectural parameters [14, 15]) is left entirely to the programmer. This is done by

exploiting the capabilities of the DSL to automatically explore the configuration space for the parallel code.

To this end, iterative compilation [16] techniques are attractive to identify the best compiler optimizations for a given program/code fragment by considering possible trade-offs. Given the diversity of heterogeneous multiprocessors and the potential for optimizations provided by runtime information, runtime optimization is also desirable. To combine the two approaches, *split compilation* will be used. The key idea is to split the compilation process in two steps - offline, and online - and to offload as much of the complexity as possible to the offline step, conveying the results to runtime optimizers [17]. We will express code generation strategies to drive a dynamic code generator in response to particular hardware features as well as dynamic information. This combination of iterative- and split-compilation will have a significant impact on the performance of applications, but also on the productivity of programmers by relieving programmers from the burden of repeatedly optimizing, tuning, compiling and testing.

4. SELF-ADAPTIVITY & AUTOTUNING

The management of system adaptivity and autotuning is a key issue in HPC systems, the system needs to react promptly to changing workloads and events, without impacting too much the extra-functional characteristics, such as energy and thermal features [18, 19]. The motivation can be easily explained by the requirement to meet the maximum performance/power ratio across all the possible deployments of the applications. This is especially important when considering the rapid growth of computing infrastructures that continue to evolve on one hand by increasing computing nodes, while on other hand by increasing the performance exploiting heterogeneity in terms of accelerators/co-processors. Thus, there is a requirement on applications to become adaptive with respect to the computing resources. In this direction, another interesting effect is that there is a growing need of guaranteeing SLA both at the server- and at the application-side. This need is related to the performance of the application, but also to the maximum power budget that can be allocated to a specific computation. In this context, efforts are mainly focused on two main paths: i) the development of an autotuning framework to configure and to adapt application-level parameters and ii) to apply the concept of precision autotuning to HPC applications.

Application Autotuning. Two types of approaches have been investigated so far to support application autotuning depending on the level of knowledge about the target domain: white-boxes and black-boxes. White-box techniques are those approaches based on autotuning libraries that deeply use the domain specific knowledge to fast surf the parameter space. On the other side, black-box techniques do not require any knowledge on the underlying application, but suffer of long convergence time and less custom possibilities. The proposed framework falls in the area of grey-box approaches. Starting from the idea of non-domain knowledge, it can rely on code annotations to shrink the search space by focusing the autotuner on a certain subspace. Moreover, the framework includes an application monitoring loop to trigger the application adaptation. The monitoring, together with application properties/features, represents the main support to the decision-making during the application autotuning phase since it is used to perform statistical analysis related to system performance and other SLA aspects. Continuous on-line learning techniques are adopted to update the knowledge from the data collected by the monitors, giving the possibility to autotune the system according to the most recent operating conditions. Machine learning techniques are also adopted by the decision-making engine to support autotuning by predicting the most promising set of parameter settings.

Precision Autotuning. In recent years, customized precision has emerged as a promising approach to achieve power/performance trade-offs when an application can tolerate some loss of quality. In ANTAREX, the benefits of customized precision HPC applications will be investigated in tight collaboration with the domain experts of the two use cases. We also plan to apply fully automatic dynamic optimizations, based on profiling information, and data acquired at runtime, e.g. dynamic range of function parameters.

In both cases, the use of the ANTAREX DSL will be crucial to decouple the functional specification of the application from the definition of software knobs (such as code variants or application parameters) and from the precision tuning phase.

5. RUNTIME RESOURCE & POWER MANAGEMENT

ANTAREX focuses on a holistic approach towards next-generation energy-efficient Exascale supercomputers. While traditional design of green supercomputers relies on the integration of best-in-class energy-efficient components [20], recent works [21, 22] show that as an effect of this design practice supercomputers are nowadays heterogeneous system. Indeed, supercomputers are not only composed by heterogeneous computing architectures (GPGPUs and CPUs), but different instances of the same nominal component execute the same application with 15% of variation in the energy-consumption. Different applications on the same resources show different performance-energy trade-offs, for which an optimal selection of operating points can save from 18% to 50% of node energy with respect to the default frequency selection of the Linux OS power governor. Moreover it has been recently shown that environmental conditions, such as ambient temperature, can significantly change the overall cooling efficiency of a supercomputer, causing more than 10% *Power usage effectiveness* (PUE) loss when transitioning from winter to summer [23]. These sources of heterogeneity, coupled with current worst case design practices, lead to significant loss in energy-efficiency, and to some missed opportunities for runtime resource and power management (RTRM, RTPM). ANTAREX leverages RTRM and RTPM by combining: (1) novel introspection points, application progress and dynamic requirements; (2) autotuning capabilities enabled by the DSL in the applications; and (3) information coming from the processing elements of the Exascale machine and IT infrastructure and their respective performance knobs (such as resource allocation, Dynamic Voltage and Frequency Scaling, cooling effort, room temperature). Information flows converge in a scalable multilayer resource management infrastructure. The information will be used to allocate to each application the set of resources and their operating points to maximize the overall supercomputer energy-efficiency, while respecting SLA and safe working conditions. The latter will be ensured by the resource management solution by optimal selection of the cooling effort and by a distributed optimal thermal management controller. The ANTAREX power management and resource allocation approach is based on: (1) Expanding the energy/performance control capabilities by introducing novel software control knobs (such as software reconfigurability and adaptability); (2) Designing scalable and hierarchical optimal control-loops capable of dynamically leveraging the control knobs together with classical performance/energy control knobs (job dispatching, resource management and DVFS) at different time scale (compile time, deployment time and runtime); (3) Monitoring the supercomputing evolution, the application status and requirements, bringing this information to the energy/performance-aware software stack. This approach will always enable the supercomputer and each application to operate at the most energy-

efficient and thermally-safe point.

6. APPLICATION SCENARIOS

The ANTAREX project is driven by two industrial HPC applications chosen to address the self-adaptivity and scalability characteristics of two highly relevant scenarios towards the Exascale era.

Use Case 1: Computer Accelerated Drug Discovery Computational discovery of new drugs is a compute-intensive task that is critical to explore the huge space of chemicals with potential applicability as pharmaceutical drugs. Typical problems include the prediction of properties of protein-ligand complexes (such as docking and affinity) and the verification of synthetic feasibility. These problems are massively parallel, but demonstrate unpredictable imbalances in the computational time, since the verification of each point in the solution space requires a widely varying time. Moreover, different tasks might be more efficient on different type of processors, especially in a heterogeneous system. Dynamic load balancing and task placement are critical for the efficient solution of such problems [24, 25].

Use Case 2: Self-Adaptive Navigation System To solve the growing automotive traffic load, it is necessary to find the best utilization of an existing road network, under a variable workload. The basic idea is to provide contextual information from server-side to traditional mobile navigation users and vice versa. The approach will help to overcome the major shortcomings of the currently available navigation systems exploiting synergies between server-side and client-side computation capabilities. The efficient operation of such a system depends strongly on balancing data collection, big data analysis and extreme computational power [26, 27].

Prototypes of these two use cases will be developed, integrated and validated in relevant environments to practically assess the benefits of the ANTAREX self-adaptive holistic approach, as well as the scalability of the proposed approach towards Exascale systems.

Target Platforms The target platforms are the CINECA's Tier-1 IBM NeXtScale hybrid Linux cluster, based on Intel TrueScale interconnect as well as Xeon Haswell processors and MIC accelerators [4], and IT4Innovations Salomon supercomputer, which is a PetaFlop class system consisting 1008 computational nodes. Each node is equipped with 24 cores (two twelve-core Intel Haswell processors). These computing nodes are interconnected by InfiniBand FDR and Ethernet networks. Salomon also includes of 432 compute nodes with MIC accelerators.

7. EARLY EVALUATION

To provide an early analysis of the combined impact of parallelisation, precision tuning, compiler optimizations, we used a miniapp extracted from the Drug Discovery application. The miniapp implements one of the most time consuming computational kernel of the LiGen de-novo drug design software [25]. In particular the miniapp implements the computation of the internal inter-atomic distances of a drug molecule (*SumOfIntervalDistance*) and the computation of the overlap between the drug molecule and a set of protein active site probes (*MeasureOverlap*). Both computations are heavily used to compute a geometrical component of drug docking score. The miniapp does not introduce any change with respect to the LiGen code, preserving the same source and class hierarchy, whereas all the proprietary components and those not relevant for performances have been stripped out.

We employ test configurations with 4000 atoms per molecule and 400 ligands. We applied parallelisation via an OpenMP parallel for with reduction template applied to the outermost loop of each kernel. We applied precision tuning, allowing the computation to be run in any of a range of floating point types, as well as using integers. Finally, we applied all high-level optimization flags (*-Ox*) exposed by GCC. The overall Design Space for the combined set of optimizations is reported in Table 1. We then performed a full search of the Design Space on three different target machines, a quad-core Intel i5, a 4x quad-core AMD NUMA, and a 2x octa-core Intel Xeon with hyperthreading. For each hardware platform and each kernel, we consider as the baseline the most effective optimization level using maximum precision and 16 OpenMP threads.

Table 1: Design Space characterization

Parameter	Values
OpenMP Threads	none, 2, 4, 8, 16
Precision	__float128, double, float, int
Optimization	-O0, -O1, -O2, -O3, -O

In Table 2, we report for each kernel and hardware platform the combination of parameters that provides the best speedup over the baseline, while limiting error to less than 5%. Timings are averaged over 30 computations with the same input data and configuration. It is worth noting that, if the input size is small enough, for the shorter kernel parallelisation at this level is not useful on the Xeon machine. While integer computation does not provide sufficient precision, single and double precision floating point arithmetics are generally a good match to the data (originally stored as double precision floating point) both for speed and precision. It is worth noting that the parallel reduction is inherently more precise than the sequential version, so __float128, which provides more precision for the sequential implementation of *SumOfIntervalDistance* is never actually needed in the parallel codes. Finally, as expected the more aggressive optimization levels often fail at matching the features of the target architecture, which points to the need for finer control of the compiler transformations.

8. CONCLUSIONS

Exascale HPC systems will need the definition of new software stacks to fully exploit heterogeneity, while meeting power efficiency requirements. The goal of ANTAREX is to provide a holistic system-wide adaptive approach for next generation HPC systems. Our long-term vision is to explore an innovative application programming paradigm and description methodology to decouple functional and extra-functional aspects of the application. The impact and benefits of such technology are far reaching, beyond traditional HPC domains.

References

[1] J. Curley, "HPC and Big Data," *Innovation*, vol. 12, no. 3, Jul. 2014.

[2] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with Commodity CPUs: Are Mobile SoCs Ready for HPC?" in *Proc. Int'l Conf. on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, pp. 40:1–40:12.

[3] M. Bohr, "A 30 Year Retrospective on Dennard's MOSFET Scaling Paper," *IEEE SSCS Newsletter*, vol. 12, no. 1, pp. 11–13, Winter 2007.

[4] G. Chrysos, "Intel® Xeon Phi™ Coprocessor-the Architecture," Intel Whitepaper, 2014.

Table 2: Exploration of the compiler parameters for the UC1 MiniApp. Speedups are computed over the fastest version with maximum parallelism (16) and precision (__float128). A constraint limiting the maximum error to 5% was imposed.

Kernel	Hardware	OpenMP Threads	Precision	Optimization	Speedup	Error
MeasureOverlap	4-core Intel i5-2500 3.30GHz	16	double	-Os	6.1	0.0%
SumOfIntervalDistance	4-core Intel i5-2500 3.30GHz	4	double	-Os	4.0	0.0%
MeasureOverlap	4x4-core AMD Opteron 8378 2.40GHz	16	float	-O3	2.7	0.0%
SumOfIntervalDistance	4x4-core AMD Opteron 8378 2.40GHz	16	double	-Os	2.7	0.0%
MeasureOverlap	2x8-core Intel Xeon E5-2630v3 2.40GHz	none	double	-O2	2.4	0.0%
SumOfIntervalDistance	2x8-core Intel Xeon E5-2630v3 2.40GHz	16	float	-Ofast	3.6	0.00065%

[5] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven, "Using architecture models for runtime adaptability," *IEEE Softw.*, vol. 23, no. 2, pp. 62–70, Mar. 2006.

[6] J. Irwin, G. Kickzales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, and J.-M. Loingtier, "Aspect-oriented Programming," in *ECOOP'97 – Object-Oriented Programming*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, vol. 1241, pp. 220–242.

[7] T. Elrad, R. E. Filman, and A. Bader, "Aspect-oriented Programming: Introduction," *Communications of the ACM*, vol. 44, no. 10, pp. 29–32, 2001.

[8] J. M. P. Cardoso, T. Carvalho, J. G. F. Coutinho, W. Luk, R. Nobre, P. Diniz, and Z. Petrov, "LARA: An Aspect-oriented Programming Language for Embedded Systems," in *Proc. 11th Annual Int'l Conf. on Aspect-oriented Software Development*. ACM, 2012, pp. 179–190.

[9] J. M. P. Cardoso, J. G. F. Coutinho, T. Carvalho, P. C. Diniz, Z. Petrov, W. Luk, and F. Gonćalves, "Performance-driven instrumentation and mapping strategies using the LARA aspect-oriented programming approach," *Software: Practice and Experience*, Dec. 2014.

[10] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "An Overview of AspectJ," in *ECOOP 2001 – Object-Oriented Programming*, 2001, pp. 327–354.

[11] O. Spinczyk, A. Gal, and W. Schröder-Preikschat, "AspectC++: An Aspect-oriented Extension to the C++ Programming Language," in *Proc. 40th Int'l Conf on Tools Pacific: Objects for Internet, Mobile and Embedded Applications*, 2002, pp. 53–60.

[12] J. M. Cardoso, T. Carvalho, J. G. Coutinho, R. Nobre, R. Nane, P. C. Diniz, Z. Petrov, W. Luk, and K. Bertels, "Controlling a complete hardware synthesis toolchain with LARA aspects," *Microprocessors and Microsystems*, vol. 37, no. 8, pp. 1073–1089, 2013.

[13] R. Nobre, L. G. Martins, and J. M. Cardoso, "Use of Previously Acquired Positioning of Optimizations for Phase Ordering Exploration," in *Proc. of Int'l Workshop on Software and Compilers for Embedded Systems*. ACM, 2015, pp. 58–67.

[14] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "Towards Transparently Tackling Functionality and Performance Issues across Different OpenCL Platforms," in *Int'l Symp. on Comp. and Networking (CANDAR)*, Dec 2014, pp. 130–136.

[15] G. Agosta, A. Barenghi, A. Di Federico, and G. Pelosi, "OpenCL Performance Portability for General-purpose Computation on Graphics Processor Units: an Exploration on Cryptographic Primitives," *Concurrency and Computation: Practice and Experience*, 2014.

[16] F. Bodin, T. Kisuki, P. Knijnenburg, M. O'Boyle, and E. Rohou, "Iterative compilation in a non-linear optimisation space," 1998.

[17] A. Cohen and E. Rohou, "Processor virtualization and split compilation for heterogeneous multicore embedded systems," in *Proc. 47th Design Automation Conference*. ACM, 2010, pp. 102–107.

[18] E. Paone, D. Gadioli, G. Palermo, V. Zaccaria, and C. Silvano, "Evaluating orthogonality between application auto-tuning and run-time resource management for adaptive opencl applications," in *IEEE 25th Int'l Conf. on Application-Specific Systems, Architectures and Processors, ASAP 2014, Zürich (CH), June 18-20, 2014*, 2014, pp. 161–168.

[19] E. Paone, F. Robino, G. Palermo, V. Zaccaria, I. Sander, and C. Silvano, "Customization of opencl applications for efficient task mapping under heterogeneous platform constraints," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, 2015, pp. 736–741.

[20] B. Subramaniam, W. Saunders, T. Scogland, and W.-c. Feng, "Trends in Energy-Efficient Computing: A Perspective from the Green500," in *Int'l Green Computing Conf.*, June 2013.

[21] F. Fraternali, A. Bartolini, C. Cavazzoni, G. Tecchiolli, and L. Benini, "Quantifying the Impact of Variability on the Energy Efficiency for a Next-generation Ultra-green Supercomputer," in *Proc. 2014 Int'l Symp. on Low Power Electronics and Design*. ACM, 2014, pp. 295–298.

[22] A. Auweter, A. Bode, M. Brehm, L. Brochard, N. Hammer, H. Huber, R. Panda, F. Thomas, and T. Wilde, "A Case Study of Energy Aware Scheduling on SuperMUC," in *Supercomputing*. Springer, 2014, vol. 8488, pp. 394–409.

[23] A. Borghesi, C. Conficoni, M. Lombardi, and A. Bartolini, "MS3: a Mediterranean-Stile Job Scheduler for Supercomputers - do less when it's too hot!" in *2015 Int'l Conf. on High Perf. Comp. & Simulation*. IEEE, 2015.

[24] C. Beato, A. R. Beccari, C. Cavazzoni, S. Lorenzi, and G. Costantino, "Use of experimental design to optimize docking performance: The case of ligendock, the docking module of ligen, a new de novo design program," *J. Chem. Inf. Model.*, vol. 53, no. 6, pp. 1503–1517, 2013.

[25] A. R. Beccari, C. Cavazzoni, C. Beato, and G. Costantino, "LiGen: A High Performance Workflow for Chemistry Driven de Novo Design," *J. Chem. Inf. Model.*, vol. 53, no. 6, pp. 1518–1527, 2013.

[26] R. Tomis, J. Martinovič, K. Slaninová, L. Rapant, and I. Vondrák, "Time-dependent route planning for the highways in the czech republic," in *Proc. Int'l Conf. on Computer Information Systems and Industrial Management, CISIM 2015*, 2015.

[27] D. Fedorčák, T. Kocyan, M. Hájek, D. Szturcová, and J. Martinovič, "viaRODOS: Monitoring and Visualisation of Current Traffic Situation on Highways," in *Computer Information Systems and Industrial Management*. Springer, 2014, vol. 8838, pp. 290–300.