Adaptive Optimization and Enforcement of Extra-Functional Properties in High Performance Computing: The ANTAREX Project

Cristina Silvano, Giovanni Agosta, Andrea Bartolini, Andrea R. Beccari, Luca Benini, Loïc Besnard, João Bispo, Radim Cmar, João M. P. Cardoso, Carlo Cavazzoni, et al.

Adaptive Optimization and Enforcement of Extra-Functional Properties in High Performance Computing: The ANTAREX Project

Cristina Silvano*, Giovanni Agosta*, Andrea Bartolini†, Andrea R. Beccari‡, Luca Benini†, Loïc Besnard‡‡, João Bispo§, Radim Cmar††, João M. P. Cardoso§, Carlo Cavazzoni¶, Daniele Cesarini†, Stefano Cherubin*, Federico Ficarelli¶, Davide Gadioli*, Martin Golasowski‖, Imane Lasri**, Antonio Libri†, Jan Martinovič‖, Gianluca Palermo*, Pedro Pinto§, Erven Rohou**, Nico Sanna¶, Kateřina Slaninová‖, Emanuele Vitali*
*DEIB – Politecnico di Milano, †IIS – Eidgenössische Technische Hochschule Zürich,
‡Dompé Farmaceutici SpA, §FEUP – Universidade do Porto, ¶CINECA,
‖IT4Innovations, VSB – Technical University of Ostrava, **INRIA Rennes, ††Sygic, ‡‡IRISA/CNRS
Email: *name.surname@polimi.it †{barandre,lbenini}@iis.ee.ethz.ch, ‡andrea.beccari@dompe.it, §{jbispo,jmpc,pmsp}@fe.up.pt,
¶n.surname@cineca.it, ‖name.surname@vsb.cz, **name.surname@inria.fr, ††rcmar@sygic.com, ‡‡name.surname@irisa.fr

***Abstract*—The ANTAREX project developed an approach to the performance tuning of High Performance applications based on a Aspect-oriented Domain Specific Language (DSL), with the goal to simplify the enforcement of extra functional properties in large scale applications. The project aims at demonstrating its tools and techniques on two relevant use cases, one in the domain of computational drug discovery, the other in the domain of online vehicle navigation. In this paper, we present an overview of the project and its main achievements, as well as of the large scale experiments that have been planned to validate the approach.**

***Keywords*—High Performance Computing, Autotuning, Adaptivity, DSL, Compilers, Energy Efficiency**

I. Introduction

The design and optimisation of applications for High Performance Computing systems is an extremely challenging problem, even more so when taking into account the need for energy efficiency and extreme scalability. Indeed, to achieve an Exascale supercomputer, a radical improvement in energy efficiency is necessary, as the power demand under current technology constraints would reach hundreds of MW, against a state target of 20-30 MW. According to the Green500 list [1], which ranks supercomputers according to the GigaFlops per Watt figure of metric, the "most green" supercomputer SHOUBU SystemB installed in Japan exceeds 18 GigaFlops/W during its 857-TeraFlop/s Linpack performance run. The top positions in Green500 are all occupied by heterogeneous systems based on high-performance processors and co-processors such as the latest NVIDIA Volta GV100 GPU and PEZY SC2 accelerator to further accelerate the computation. The dominance of heterogeneous systems in the Green500 list is expected to continue for the next coming years to reach the target of 20MW Exascale supercomputers. Furthermore, even the Top500 list [2], which measures pure performance, is dominated by the Summit supercomputer installed in the USA, which reaches a peak performance of over 187 PetaFlops with a power envelop of less than 9 MW – still above the target for Exascale.

To fulfil the Exascale target, a goal that the European Union aims at reaching by 2023 [3], energy-efficient supercomputers need to be coupled with a radically new software stack capable of exploiting the benefits offered by architecture heterogeneity at different abstraction levels to meet the scalability and energy efficiency required by the Exascale era. Furthermore, the current development model where the HPC center staff directly supports the development of applications will become unsustainable in the long term, due to the inherent challenges of developing applications for heterogeneous systems. 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 conclude, heterogeneous systems currently dominate the top of the Top500 and Green500 lists and this dominance is expected to be a trend for the next coming years to reach the target of 20MW Exascale supercomputers. To fulfil the 20MW target, energy-efficient heterogeneous supercomputers need to be coupled with radically new software stacks to exploit the benefits offered by heterogeneity at all levels (supercomputer, job, node) to meet the scalability and energy efficiency required by the Exascale era.

Goals of the project. The ANTAREX [1, 2, 3, 4] project aims at providing 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 ANTAREX is to express by means of a DSL the application self-adaptivity and to runtime manage and autotune applications for green heterogeneous HPC systems up to the Exascale level. The use of a DSL allows the introduction of a separation of concerns, where self-adaptivity and energy efficient strategies are specified separately from the application functionalities. This is promoted by the definition of a DSL inspired by aspect-oriented programming concepts for heterogeneous systems. The DSL is based on previous efforts regarding the LARA language [5, 6] and makes possible to express 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 in 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.

[1] www.green500.org, June 2018
[2] www.top500.org, June 2018
[3] https://www.consilium.europa.eu/en/press/press-releases/2018/06/25/supercomputers-council-agrees-to-develop-high-tech-infrastructure/

Key innovations. The ANTAREX approach provides: (1) 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 project is driven by two use cases taken from highly relevant HPC application scenarios: (1) a biopharmaceutical application for drug discovery deployed on the Marconi system at CINECA and (2) a self-adaptive navigation system for smart cities deployed on the server-side on the Anselm and Salomon systems provided by IT4Innovations National Supercomputing Center.

The ANTAREX Consortium. To achieve the ANTAREX goals, a broad range of technical expertise is needed. As a consequence, the ANTAREX Consortium comprises some of the foremost institution in European research. The consortium is led by Politecnico di Milano, one of the largest technical universities in Europe, and comprises top-ranked academic partners such as ETH Zürich, Universidade do Porto, and INRIA (with additional support from IRISA/CNRS). Real-world applications are provided by one of the leading biopharmaceutical companies, Dompè, and by the top navigation software company in Europe, Sygic. To effectively validate the tool flow, as well as to support the deployment and scale-up of the target applications, the consortium also includes two supercomputing centers, CINECA in Italy and IT4Innovations in Czech Republic. CINECA's Marconi supercomputer is ranked 18 in the Top500 as of June 2018, making it the 3rd most powerful supercomputer in the European Union, while IT4Innovations' Salomon is ranked 139.

Organization of the paper The rest of this paper is organized as follows. In section II we outline the ANTAREX approach, while in section II-A we introduce the ANTAREX DSL. In sections III and IV we introduce the target platforms and use case scenarios employed for the validation of the ANTAREX tools and methodologies. Finally, in section V we draw some conclusions.

II. The ANTAREX Approach

Figure 1 shows the ANTAREX approach through its tool flow, covering both the design-time and runtime operation. The functionality of the target application is expressed through a standard programming language – the current tools support both C and C++ – and can therefore include legacy code. On the other hand, the extra-functional aspects of the application are expressed through a separate specification, written in the ANTAREX DSL. The ANTAREX DSL is based on LARA [7], extended to provide support for parallelisation, mapping, precision tuning and adaptivity strategies. In this way, the expression of extra-functional concerns is fully decoupled from the functional code, allowing the application domain expert and the performance tuning expert to operate as independently as possible.

The DSL specification is *weaved* into the functional code at compile time, through a source-to-source transformation tool, *Clava*, which performs a refactoring of the application code based on the

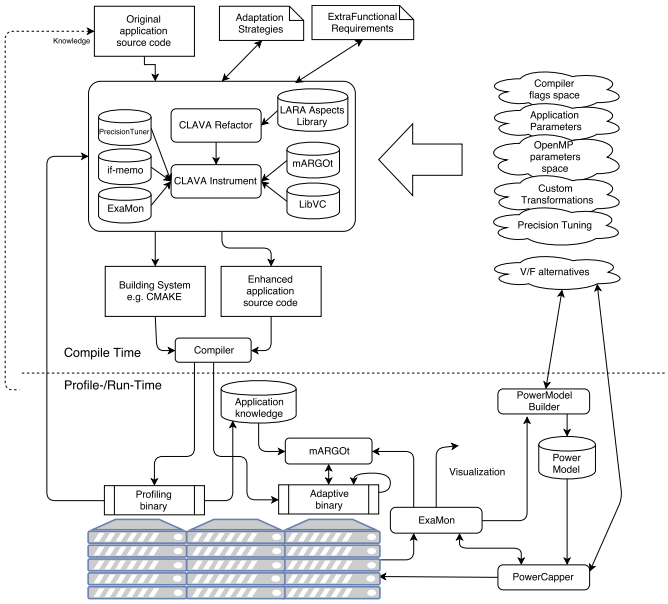

Fig. 1. The ANTAREX Tool Flow

aspects provided, and instruments it with the necessary calls to other components of the tool flow.

a) Dynamic Compilation: Partial dynamic (re)compilation is a technique used as part of continuous program optimisation [8]. It allows the compiler to further optimise the code, during the execution of long runs of an application, which are typical of HPC scenarios. While most high level languages include mechanisms for selective compilation which can be exploited for fine tuning the dynamic compilation options, e.g. for hiding compilation latencies [9], the C/C++ applications commonly used in HPC scenarios generally lack this option. Some support is provided by domain-specific tools, such as RuntimeCompiledC++ [10], which focuses on interactive modification and recompilation of program fragments by the programmer. To enable partial dynamic compilation, ANTAREX DSL aspects can introduce calls to a support library, *libVC* [11], allowing to weaken the boundary between compile-time and runtime, and enabling continuous optimisation.

b) Precision Tuning: Among the optimisation techniques, ANTAREX developed in particular tools for precision tuning, which has emerged as a promising approach to improve power/performance trade-offs. Precision tuning originates from the fact that many applications can tolerate some loss of quality during computation, as in the case of media processing (audio, video and image), data mining, machine learning, etc. Error-tolerating applications are increasingly common in the emerging field of real-time HPC. In ANTAREX, we explored both precision tuning of floating point computation on GPGPU accelerators [12] and floating to fixed point conversion, followed by tuning of the fixed point representation in terms of bit width and point position [13, 14].

c) Memoization: Memoization has been employed for a long time as a performance optimization technique, albeit primarilyin functional languages. A survey of techniques to handle memoization at programming language level, as well as references to applications can be found in [15]. More recently, memoization has been shown as a promising path for energy saving in computation-intensive workloads [16, 17]. In ANTAREX, a library to support memoiza-

tion [18, 19] has been integrated in the DSL, allowing automated exploration of memoization opportunities.

d) System Monitoring: System monitoring is a key aspect of HPC infrastructures. In ANTAREX, we leverage the ability of processing elements to provide performance metrics at hardware level through the *ExaMon* tool [20], which provides a virtualisation of the performance and power monitoring, decoupling sensor readings from sensor value use. This approach increases the scalability of the monitoring, and at the same time provides an easy access to monitoring APIs through the ANTAREX DSL.

e) Performance Tuning and Power Management: Techniques for controlling the performance of one or more applications according to system parameters such as power consumption have been developed through the last decade in both high performance and embedded computing [21]. Typically, a design-time phase leverage optimisation tools [22] to allow the identification of specific operating points, among which the runtime controller selects the actual configuration depending on the workload, as well as the system parameters.

In ANTAREX, at runtime, the *mARGOt* tool [23] configures the available software knobs (application parameters, code transformations and code variants) according to the runtime information coming from application self-monitoring and system monitoring, thus creating an autotuning control loop. Finally, the runtime power manager, *PowerCapper*, is used to control the resource usage for the underlying computing infrastructure given the changing conditions [24, 25].

At runtime, the application control code, thanks to the design-time phase, now contains also runtime monitoring and adaptivity strategy 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 is supported by a runtime layer implementing an application level collect-analyse-decide-act loop.

A. The ANTAREX DSL

HPC applications might profit from adapting to operational and situational conditions, such as changes in contextual information (e.g., workloads), in requirements (e.g., deadlines or energy budgets), and in availability of resources (e.g., connectivity or number of processor nodes available). A simplistic approach to both adaptation specification and implementation (see, e.g., [26]) employs hard coding of conditional expressions and parameterizations, among others. In our approach, the specification of runtime adaptability strategies relies on a DSL implementing key concepts from the Aspect-Oriented Programming (AOP) paradigm [27]. We use this language to target specific execution points and specify adaptation concerns.

Our approach is based on the idea that certain application/system requirements (e.g., target-dependent optimizations, adaptivity behavior and concerns) should be specified separately from the source code that defines the main functionality, and without changes to the original application source code. Those requirements are expressed as DSL aspects that embody strategies. An extra compilation step, performed by a *weaver*, merges the original source code and aspects into the intended program [28]. Using aspects to separate concerns from the core objective of the program results in cleaner programs. Moreover, the reusability of strategies can be a viable path to increase productivity. The development process of HPC applications typically involves two types of experts (application-domain experts and HPC system architects) that split their responsibilities along the boundary of functional description and extra-functional aspects. Our DSL-aided toolflow provides a suitable approach for helping to express their concerns.

The ANTAREX DSL relies on the already existing LARA language and framework [5, 6], which was adopted and extended. We developed the *Clava*[4] weaver to leverage the rest of the ANTAREX tool flow through aspects and APIs. Compared to other approaches that usually focus on code injection (e.g., [29]), LARA provides access to other types of actions, e.g., code refactoring, compiler optimizations, and inclusion of additional information, all of which can guide compilers to generate more efficient implementations. Additional types of actions may be defined in the language specification and associated weaver, such as software/hardware partitioning [30] or compiler optimization sequences [31].

The LARA language is compatible with the ECMAScript 5 specification[5], which means that JavaScript code that conforms to that specification is usually considered valid LARA code. Besides supporting plain JavaScript in `.js` files, LARA extends JavaScript with several new keywords and syntax constructs, which can only be used in `.lara` files.

On a more detailed level, the keyword `aspectdef` marks the beginning of an aspect. An aspect is similar to a JavaScript function but where LARA keywords and constructs can be used. The weaver, before execution of a `.lara` file, implicitly parses the target source-code (e.g., a C++ program) and creates an abstract representation that will be accessible during the execution of the LARA strategy. The keyword `select` allows specifying the points in the code (e.g., function calls) that one wants to analyze or transform. The selection is hierarchical and similar to a query. For instance, `select function.loop end` selects all the loops inside all the functions in the target source code. The `apply` block is similar to a for loop that iterates over all the points of the previous selection, herein simply referred as join point. Each join point has a set of attributes, which can be accessed, and a set of actions, which can be used to transform the code. Finally, the `condition` keyword can be used to filter a join point selection.

These are the basic elements of the LARA language that can be used to define strategies for code transformation and runtime adaptability. For more specific operations, the Clava weaver provides APIs that implement commonly used strategies (e.g., time measurement around program segments), as well as APIs for the integration of other ANTAREX components.

As an example of an integration API, consider the LARA code presented in Figure 2, which is an excerpt of a LARA strategy that fully integrates the mARGOt autotuner into an application. The shown strategy will generate the needed configuration files for mARGOt and also enhance the original application with calls to the mARGOt API in order to provide runtime adaptation capabilities. The aspect is parameterized with the name of function whose calls we will be targeting, as well as the paths to configuration files. There is a top level aspect, `MargotIntegration`, that calls the other aspects, which one performing a specific task. in `MargotConfig`, using the API provided with Clava, the user configures all information for mARGOT. This is done in lines 13–26 and configures the knob the autotuner will set, the metric to be collected and the objective function

[4] https://github.com/specs-feup/clava

[5] Support for some features of more recent specifications has been added, such as the `for...of` statement, when used in `.lara` files.

```
1  aspectdef MargotIntegration
2    input targetName, configPath, opListPath end
3
4    cfg = call MargotConfig(targetName, configPath);
5    call MargotDse(targetName, opListPath, cfg.dseInfo);
6    call MargotCodeGen(targetName, cfg.codegen);
7  end
8
9  aspectdef MargotConfig
10   input targetName, configPath end
11   output dseInfo, codegen end
12
13   config = new MargotConfig();
14   targetBlock = config.newBlock(targetName);
15
16   targetBlock.addKnob('Knob1', 'knob1', 'int');
17
18   targetBlock.newTimeMonitor('timer');
19
20   targetBlock.addMetric('exec_time', 'float');
21   targetBlock.addRuntimeProvider('exec_time', 'timer', 1);
22
23   problem = targetBlock.newState('default');
24   problem.minimizeMetric('exec_time');
25
26   config.build(configPath);
27
28   dseInfo = MargotDseInfo.fromConfig(config, targetName);
29   codegen = MargotCodeGen.fromConfig(config, targetName);
30 end
31
32 aspectdef MargotDse
33   input targetName, opListPath, dseInfo end
34
35   select function.body.call{targetName} end
36   apply
37     dseInfo.setScope($body);
38     dseInfo.setMeasure($call);
39   end
40
41   dseInfo.setDseRuns(30);
42   dseInfo.addTimeMetric('exec_time', TimeUnit.micro());
43   dseInfo.setKnobValues('Knob1', 2, 4, 8, 16, 32);
44
45   dseInfo.execute(opListPath);
46 end
47
48 aspectdef MargotCodeGen
49   input targetName, codegen end
50
51   select call{targetName} end
52   apply
53     codegen.init($call);
54     codegen.update($call);
55   end
56 end
```

Fig. 2. Excerpt of a LARA strategy for the integration of the mARGOt autotuner into an application.

to minimize. At the end of this aspect, the configuration object is used to generate and return additional information for the following steps. Then, in `MargotDse`, in lines 32–43, this strategy performs a design-space exploration. This generates a list of operating points that map each tested value of the knob to a value of the collected metric. In this specific case, the application will be tested 30 times with the values defined in line 41 and the average execution time for each of those values is reported. This will be written to a file that represents the initial knowledge base of the autotuner. Finally, in the `MargotCodeGen` aspect (lines 47–54), the application code is changed to include calls to the autotuner interface. These are the calls that will cause mARGOt to check its internal state and update the values of the knob. The strategy selects function calls using their name as a filter and then pass the selected points to the API calls that will surround them with the necessary code. Using such an API, one does not need to know what code is going to be inserted, but rather needs to specify what actions need to be performed.

III. Target Platforms

The target platforms are two PetaFlop class systems, the CINECA's top-level supercomputer, Marconi, and IT4Innovations Salomon supercomputer.

A. CINECA Platform & Roadmap

CINECA, to further its institutional mission to support the competitiveness of the Italian research infrastructure, including its participation to the international supercomputing enviroment in Europe (PRACE), operates an evolving, massive infrastructure aiming at a convergence of computing and data handling tasks. CINECA aims at growing its infrastructure to bridge the gap to Exascale around 2023, as well as to support emerging paradigms such as Big Data analytics, real-time HPC, and cloud HPC.

The current supercomputing machine (Marconi), in production from the second-half of 2016, is set up as a scalable hybrid cluster starting at 10 PFLOPS, and currently benchmarked at a peak of 16 PFPLOS[6]. The CINECA roadmap will lead to a scale up of the Marconi machine to over 20 PFLOPS in 2019, and up over 200 PFLOPS by 2022, with a power envelop growing from the current 3 MW to around 10 MW.

However the path following to high-end computing and data managing has revealed during the last ten years or so, a major constraint in the exponential growth of power consumption in a way that is now commonly accepted the sustainable limit of 20MW for the first exaFLOPS supercomputing system. With this limit in mind, CINECA is modelling its strategy for future HPC/DA systems in order to experiment, design and deploy energy-aware supercomputing facilities. To this end, CINECA has been involved in several projects at national and EU level to take advantage of the most promising techniques for limiting the overall requirements of energy-to-solution computing workflow [33, 34, 35, 36].

In the context of ANTAREX, CINECA is employing the A2 partition of Marconi [7], which includes 3,600 nodes, each equipped with a 68-cores Intel Xeon Phi7250 (Knights Landing), clocked at 1.4 GHz, for a total of 244,800 cores. Each node is equipped with 16 GB/node MCDRAM + 96 GB/node DDR4. The partition has a peak performance of 11 PFLOPS.

B. IT4Innovations Platform & Roadmap

IT4Innovations as the National Supercomputing Center of the Czech Republic operates two HPC clusters as of 2018. The first one is a 2 PFLOP/s system called *Salomon* having 1008 nodes based on Intel Haswell CPUs [8]. Half of its computing power is provided by Intel Xeon Phi accelerators [9] based on the Knights Corner architecture. It uses InfiniBand FDR56 interconnect in 7D Enhanced hypercube topology.

[6] https://www.top500.org/list/2018/06
[7] https://wiki.u-gov.it/confluence/pages/viewpage.action?pageId=131696536
[8] 2 x Intel Xeon E5-2680v3, 2.5 GHz, 12 cores, 128 GB RAM per node
[9] 2 x Intel Xeon Phi 7120P, 61 cores, 16 GB RAM

The second cluster is a smaller 94 TFLOP/s system called *Anselm*. It has 209 nodes based on Intel Sandy Bridge CPUs [10]. It also contains 23 GPU accelerated nodes[11] and 4 MIC accelerated nodes[12].

Each cluster has its own dedicated storage available, including distributed filesystem LUSTRE. Apart from this, each cluster has a set of nodes dedicated for virtualization with its own storage which can be used to run virtual machines providing auxiliary services. The ANTAREX project used both machines including their virtualization infrastructure extensively.

During the project a small portion (18 nodes) from the Anselm cluster has been removed from the PBS scheduler queues and dedicated entirely for development and integration activities related to the UC2. The dedicated partition has been isolated in a private network segment accessible only through a VPN operated by IT4Innovations. Access to the this infrastructure has been restricted only for members of the ANTAREX consortium involved in the UC2 activities.

Main purpose of the dedicated partition was to gain a privileged access to the computing hardware for testing of various network settings and running performance monitoring tools which require privilege escalation. It has been used mainly for verification and validation of the server-side routing service using the traffic simulator. Integration of the mARGOt autotuner has been also tested there. The next step is to use the dedicated partition to test tools which require privilege escalation, mainly mARGOt autotuner with energy constraints measured by PAPI and other related experiment.

The remaining production part of the Anselm cluster has been used for scalability and performance tests of the experimental server-side routing service. It is also used for ongoing verification and validation of the developed service using the traffic simulator and large number Sygic navigation app instances.

The Salomon cluster is used mainly for computing what-if scenarios using the betweenness centrality algorithm. It has been also used for performance and scalability testing of multi-node implementation of the algorithm and also to verify and validate tools developed by INRIA integrated in the algorithm.

IV. Application Scenarios

To demonstrate the impact of the ANTAREX toolchain, the project leverages two applications from emerging domains.

The first application is drawn from the computational chemistry domain, and aims at speeding up dramatically the early phases of the drug discovery process, allowing a fast screening of a large amount of potential drug molecules. The second application is drawn from the navigation system domain, and aims at introducing a server-side aspect to the navigation, with the final goal of providing directions to vehicles in a coordinated manner, to ensure a near-optimal transit time to each vehicle.

The two applications are well positioned, since they represent two critical societal challenges, and map to two different classes of HPC applications – the massively parallel class, and the real-time HPC class.

A. Drug Discovery

The goal of a drug discovery process is to find novel drugs starting from a huge exploration space of possible molecules. Typically, this process involves several *in vivo*, *in vitro* and *in silico* tasks ranging from chemical design to toxicity analysis. Molecular docking is one stage of this process. It aims at estimating the three-dimensional pose of a given molecule, named *ligand*, when it interacts with the target protein. The ligand is much smaller than the target protein, therefore we focus a small region of the target protein (or receptor), named *pocket* (or binding site). Given the three-dimensional pose of the ligand within the pocket, we are able to estimate the strength of the chemical and physical interactions between the ligand and the pocket by computing a geometric fitting score.

The evaluation of the pose of each ligand is independent from the evaluation of all the other candidates. Given that in drug discovery the number of ligands that we are interested in analyzing is above the billion of units, this problem can be considered embarrassingly parallel. However, to find the three-dimensional pose of the ligand when it interacts with the pocket, we have to deal with a large number of degrees of freedom.

As evaluating the chemical and physical interactions between the ligand and the pocket is a computationally intensive problem, the of splitting the pose prediction task from the virtual screening task is a clear consequence. The pose prediction task focuses on providing the best pose for a given ligand within a given binding site, whereas the virtual screening task aims at selecting among a huge database of candidates a small set of promising ligands which best fit the given binding site. A remarkable difference between the pose prediction and the virtual screening task lies in the approach to the estimation of the chemical and physical interactions between the ligand and the pocket, and how to determine the pose to be evaluated.

The estimation of the interactions between the ligand and the pocket can be done with either a geometrical or a pharmacophoric approach. The geometrical approach estimates the ligand-pocket interactions by only using the shape and volume information, whilst the pharmacophoric approach evaluates the actual chemical and physical interactions. Although the best solution according to a pharmacophoric approach has also a very good geometrical score, the best geometrical solution does not guarantee to be either a valid solution or a good solution from a pharmacophoric perspective. On the other side the geometrical evaluation is orders of magnitude faster than the pharmacophoric.

On the other side, to estimate the poses to be evaluated we have also 2 alternatives: a pure geometrical and a molecular dynamic based approach. The former is based on playing on the molecule flexibility to geometrically fit the ligand in the pocket, while the latter is based on a molecular simulation that places the atoms on a minimal energy point. Also in this case, while the geometrical approach provides a faster time to solution, the other one is more accurate in finding better poses.

The use of molecular simulations together with a pharmacophoric estimation represents the most accurate solution but also the most computational-intensive approach. it is regularly exploited on the pose prediction task. The combined geometrical approach on the other side ca be exploited for the virtual screening phase.

In ANTAREX, we focused on the combination of the geometrical approaches with the goal of screening a huge amounts of ligand. Indeed, we developed within LIGEN a completely new module *GeoDock* for molecular docking that is based only on geometric evaluations. This reduces a lot the cost of the docking process by enabling the user to increase the size of the ligand database to be processed during the virtual-screening task. Given that, it is possible to process each ligand of the chemical library in an independent way,

[10] 2 x Intel Sandy Bridge E5, 2.4 GHz, 8 cores, 64/96 GB RAM, per node
[11] NVIDIA Kepler K20m
[12] Intel Xeon Phi 5110P

and given the size of the target ligand library, the problem can be considered embarrassing parallel. *GeoDock* is a typical example of batch application optimized for homogeneous HPC platforms that can scale up to exascale.

Figure 3 and 4 show a preliminary analysis of the week and strong scaling for the *GeoDock* module while running on the MARCONI-A2 partition composed of Intel KNL nodes. In particular, Figure 3 shows the elapsed time of a *GeoDock* run while considering 100 ligands. The considered number of nodes ranges from 8 to 128, while each node runs 68 MPI processes. From the figure it is possible to notice how starting from 128 nodes we see a bad weak-scaling behavior. While the small oscillations up to 64 are due to the different types of ligands we are screening (the computation is data-dependent), at 128 nodes we noticed a bottleneck on the I/O that almost double the execution time. These initial results have been used to optimize the I/O part of the *GeoDock* module. On the opposite, Figure 4 show how the application throughput varies while increasing the number of ligands per MPI process while considering a setup with 32 nodes and 68 MPI procesess per node. Given that there is no synchronization inside the application, this experiment can be considered as a proxy of a strong scaling. In particular we can notice a smooth performance degradation starting from around 30 ligands per MPI process. As for the previous case, the small oscillations we can observes for values larger than 30 are due to the data variability.

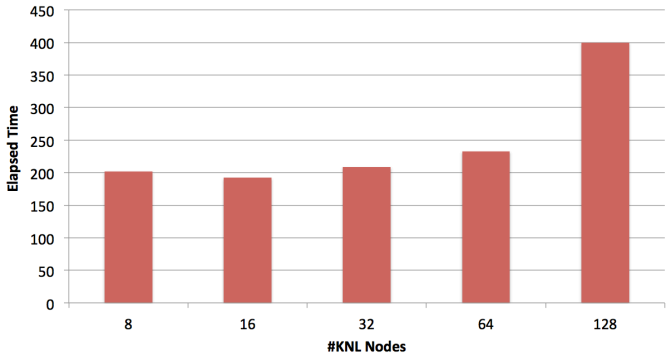

Fig. 3. Weak scaling analysis for the *GeoDock* module.

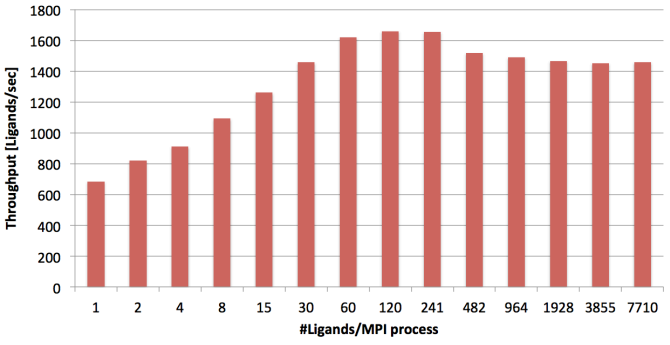

Fig. 4. Strong scaling analysis for the *GeoDock* module.

B. Navigation

Main focus of this use case is to integrate and verify the ANTAREX tools in an experimental server-side navigation system. The navigation system provides an on-line navigation service which maintains a global view on state of the road network at a given geographical area as well as the means for executing a hypothetical traffic scenarios which can occur in the road network.

Many modern cities need to cope with rapid growth of traffic volume in the recent years. Expansion of the road network is a very expensive and long-term task and in some cases (city centers, dense agglomerations) virtually impossible due to lack of available space. The other option is to divert the traffic in a way which would be nearly optimal for each individual car. This can be achieved by various means such as dynamic road signs and large displays, lane switching or dynamic signaling plans for junctions. These means can be inefficient because it is impractical to deploy them on every street in a city. Another approach is to implement an on-line navigation service and use smart phones as endpoints for such system. Modern phones have all capabilities needed for using such system. The GPS receiver is used to get actual position with sufficient precision and mobile data connection can be used to communicate with the service in real time. Both approaches can be combined as the server-side routing service can be enriched by data from various other sources (weather models, stationary traffic sensors) and the traffic infrastructure can from results of this data fusion to provide more precise information for the dynamic elements of the infrastructure (junction signaling, overhead highway displays and others).

The communication with the service can be bi-directional, as the mobile phone obtains the optimal route from the system and can provide feedback which would enhance the service awareness of the traffic network current state. In order to provide consistent service for the user, the smartphone application should be able to switch between offline and online navigation, in case there is a lack of mobile phone coverage and optimize the amount of communication needed for sending the feedback.

In the context of UC2, the server-side routing service is developed by IT4Innovations in collaboration with Sygic who provides its navigation smartphone app as a client-side endpoint for the routing service. Part of the server-side routing service is also a web-based interface for computing a betweenness centrality of a road network which can be used to model behavior of the traffic under certain conditions [37]. The web-based interface allows defining a number of what-if scenarios, including road closure, event which affects speed on roads in a given area and other options. The system is seamlessly integrated with HEAppE service[13] which is used to submit the computation job with the scenarios on the cluster [38]. Results of the computation can be access using the same interface when the job is finished.

UC2 makes use mainly of these tools, LARA DSL, mARGOt autotuner [23] and library for precision reduction and memoization provided by INRIA. Both parts of the system use the tools. On the server-side, the DSL is used to generate a part of the data access layer for the routing index stored in a HDF5 format [39]. It is also used to inject the integration code for the autotuner. mARGOt itself is integrated in the Probabilistic Time-Dependent Routing (PTDR) algorithm in the routing pipeline [40]. It is used to estimate a number of Monte Carlo samples needed to obtain estimation of the travel time with sufficient precision. The autotuner is also used in a management process which facilitates communication with the actual routing workers to estimate number of workers of each type needed to

[13] http://heappe.eu

satisfy current intensity of requests while conforming to a given SLA. Also the client side makes use of the the mARGOt autotuner with the goal to optimize the requesting frequencies towards server-side services subject to constraints on navigation quality and data transfer limits. This is to alleviate server-side processing load to make system scalable and more robust across dynamic conditions.

In order to validate function of the server-side routing a traffic simulator has been developed. This software simulates cars driving around a given region along routes provided by the routing service [41]. A snapshot of such simulation in progress is shown in Figure 5. Global view on the traffic situation in this simulated traffic network is derived from the output of the simulator and fed back to the service by the generated probabilistic-speed profiles. This simulator will be used to prepare a set of deterministic scenarios with given traffic obstacles which will be used to validate the integrated tools. This part is the main experiment of the UC2, where the routing service will be validated along with the individual tools.

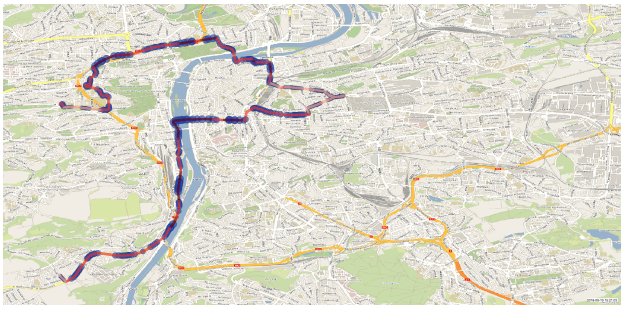

Fig. 5. Snapshot of the traffic simulation in progress, with both simulated floating car data (points) and state of the virtual traffic network (lines) visible.

We expect that the routing service will optimize traffic for the specified scenarios such that all simulated cars will arrive at their destinations in shorter time in case they are using the server-side navigation that in case when only offline navigation is used. This will be tested by generating probabilistic speed profiles for each scenario with given traffic event and comparing the results with simulation executed only with simple routing algorithm which is not aware of the traffic network state.

The next step is to validate function of the autotuning tools. Operating points for the mARGOt autotuner used in the PTDR algorithm will be derived from the profiles, which in turn are derived from the traffic scenarios described above. The same scenario will be executed again with the autotuner, while the expectation is that it will not affect the SLA and the sum of Monte Carlo samples required for the entire simulation will be lower than in the case without any autotuning. Simulation logs will provide timing of the individual requests passing through the stages of the routing pipeline. These data will be used to generate operating points for the autotuner used in the management process for workers autotuning. The same scenario will be executed again, this time with workers autotuning enabled. The expectation is that the SLA is again not violated and the number of running workers at any time during the simulation will be less or same as without the workers autotuning enabled. Similarly the autotuner on the client-side, derived from data transfer statistics of simulations and augmented with the definition of quality functions, will prove to achieve comparable navigation quality using less number of requests towards server-side services.

The Betweenness algorithm implementation utilizes tools developed by INRIA for reduction of floating point precision and memoization. Small scale tests (single city) have been executed with promising results, the results are shown in Figure 6. Large scale tests (entire region or country) are planned for near future to be executed on the Salomon cluster. We expect that tools provided by INRIA will improve the scalability of the algorithm while maintaining its sufficient precision.

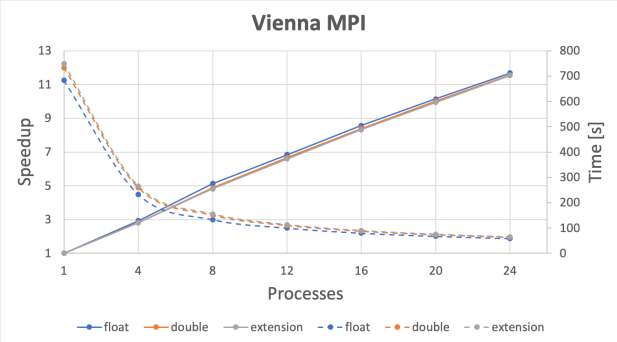

Fig. 6. Small scale test of the multi-node implementation of the betweenness algorithm executed on road network of Vienna with various floating point types.

V. Conclusions

To fully exploit the heterogeneous resources of future Exascale HPC systems, new software stacks are needed to provide power management, optimization, and autotuning to the parallel applications deployed on such systems. The ANTAREX project provides a holistic system-wide adaptive approach for next generation HPC systems, centered around a domain specific language that allows a full decoupling of functional and extra-functional specifications for each application, providing integration with a wide range of support tools.

We have shown how the ANTAREX tool flow allows developers to control the precision of a computation, to manage dynamic code specialization, monitoring, power capping, and dynamic autotuning. The impact and benefits of such technology are far reaching, beyond traditional HPC domains.

The ANTAREX tools are being used to support large scale experiments on supercomputing systems with peak performances in the order of one to ten PFLOPS, allowing the project to project its results towards the incoming pre-Exascale machines that will be deployed in Europe in the next couple of years.

Acknowledgements

The ANTAREX project is supported by the EU H2020 FET-HPC program under grant 671623. The IT4Innovations infrastructure is supported by the Large Infrastructures for Research, Experimental Development and Innovations project “IT4Innovations National Supercomputing Center – LM2015070”. The CINECA infrastructure is supported by the EU and the Italian Ministry for Education, University and Research (MIUR) under the PRACE project.

References

[1] C. Silvano, G. Agosta *et al.*, “AutoTuning and Adaptivity appRoach for Energy efficient eXascale HPC systems: the ANTAREX Approach,” in *Design, Automation, and Test in Europe*, Dresden, Germany, Mar. 2016.

[2] C. Silvano, G. Agosta, S. Cherubin *et al.*, "The ANTAREX approach to autotuning and adaptivity for energy efficient HPC systems," in *2016 ACM Int'l Conf on Computing Frontiers*, 2016, pp. 288–293.

[3] C. Silvano, A. Bartolini, A. Beccari, C. Manelfi, C. Cavazzoni, D. Gadioli, E. Rohou, G. Palermo, G. Agosta, J. Martinovič *et al.*, "The ANTAREX Tool Flow for Monitoring and Autotuning Energy Efficient HPC Systems (Invited paper)," in *SAMOS 2017-Int'l Conf on Embedded Computer Systems: Architecture, Modeling and Simulation*, 2017.

[4] C. Silvano, G. Agosta *et al.*, "Antarex: A dsl-based approach to adaptively optimizing and enforcing extra-functional properties in high performance computing," in *Proceedings of the 2018 Euromicro Conference on Digital Systems Design*, 2018.

[5] 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.

[6] 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.

[7] J. M. Cardoso *et al.*, "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.

[8] T. Kistler and M. Franz, "Continuous program optimization: A case study," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 25, no. 4, pp. 500–548, 2003.

[9] S. Campanoni, M. Sykora, G. Agosta, and S. C. Reghizzi, "Dynamic look ahead compilation: a technique to hide jit compilation latencies in multicore environment," in *International conference on compiler construction*. Springer, 2009, pp. 220–235.

[10] D. Binks, M. Jack, and W. Wilson, "Runtime compiled c++ for rapid ai development," *Game AI Pro: Collected Wisdom of Game AI Professionals*, p. 201, 2013.

[11] S. Cherubin and G. Agosta, "libVersioningCompiler: An easy-to-use library for dynamic generation and invocation of multiple code versions," *SoftwareX*, vol. 7, pp. 95 – 100, 2018.

[12] R. Nobre, L. Reis, J. Bispo, T. Carvalho, J. M. P. Cardoso, S. Cherubin, and G. Agosta, "Aspect-driven mixed-precision tuning targeting gpus," in *PARMA-DITAM '18*, Jan 2018.

[13] S. Cherubin, G. Agosta, I. Lasri, E. Rohou, and O. Sentieys, "Implications of Reduced-Precision Computations in HPC: Performance, Energy and Error," in *Int'l Conf on Parallel Computing (ParCo)*, Sep 2017.

[14] D. Cattaneo, A. Di Bello, S. Cherubin, F. Terraneo, and G. Agosta, "Embedded Operating System Optimization through Floating to Fixed Point Compiler Transformation," in *Proc of 2018 Euromicro Conference on Digital Systems Design*, 2018.

[15] U. A. Acar, G. E. Blelloch, and R. Harper, "Selective memoization," in *Proceedings of the 30th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ser. POPL '03. New York, NY, USA: ACM, 2003, pp. 14–25.

[16] G. Agosta, M. Bessi, E. Capra, and C. Francalanci, "Dynamic memoization for energy efficiency in financial applications," in *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 2011, pp. 1–8.

[17] ——, "Automatic memoization for energy efficiency in financial applications," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 2, pp. 105 – 115, 2012, iEEE International Green Computing Conference (IGCC 2011). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210537912000066

[18] A. Suresh, E. Rohou, and A. Seznec, "Compile-Time Function Memoization," in *26th International Conference on Compiler Construction*, Austin, United States, Feb. 2017.

[19] A. Suresh *et al.*, "Intercepting Functions for Memoization: A Case Study Using Transcendental Functions," *ACM Trans. Archit. Code Optim.*, vol. 12, no. 2, p. 23, Jul. 2015.

[20] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 1038–1043.

[21] C. Silvano *et al.*, "2parma: Parallel paradigms and run-time management techniques for many-core architectures," in *2010 IEEE Computer Society Annual Symposium on VLSI*, July 2010, pp. 494–499.

[22] ——, "Multicube: Multi-objective design space exploration of multi-core architectures," in *2010 IEEE Computer Society Annual Symposium on VLSI*, July 2010, pp. 488–493.

[23] D. Gadioli, G. Palermo, and C. Silvano, "Application autotuning to support runtime adaptivity in multicore architectures," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), 2015 Int'l Conf on*. IEEE, 2015, pp. 173–180.

[24] A. Bartolini, R. Diversi, D. Cesarini, and F. Beneventi, "Self-aware thermal management for high performance computing processors," *IEEE Design & Test*, 2017.

[25] D. Cesarini, A. Bartolini, and L. Benini, "Prediction horizon vs. efficiency of optimal dynamic thermal control policies in hpc nodes," in *2017 IFIP/IEEE Int'l Conf on Very Large Scale Integration (VLSI-SoC)*, Oct 2017, pp. 1–6.

[26] J. Floch *et al.*, "Using architecture models for runtime adaptability," *IEEE Softw.*, vol. 23, no. 2, pp. 62–70, Mar. 2006.

[27] 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. LNCS. Springer, 1997, vol. 1241, pp. 220–242.

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

[29] 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.

[30] 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.

[31] 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.

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

[33] C. Cavazzoni, "EURORA: A European Architecture Toward Exascale," in *Proc of the Future HPC Systems: The Challenges of Power-Constrained Performance*. New York, NY, USA: ACM, 2012, pp. 1:1–1:4.

[34] A. Bartolini, M. Cacciari, C. Cavazzoni, G. Tecchiolli, and L. Benini, "Unveiling Eurora - Thermal and Power Characterization of the Most Energy-efficient Supercomputer in the World," in *Proc Conf. on Design, Automation & Test in Europe*, 2014, pp. 277:1–277:6.

[35] "Mont-Blanc (FP7-ICT-2011-7 European project): European scalable and power efficient HPC platform based on lowpower embedded technology," 2011.

[36] "Mont-Blanc 2 (FP7-ICT-2013-10 European project), European scalable and power efficient HPC platform based on low-power embedded technology," 2013.

[37] J. Hanzelka, M. Běloch, J. Křenek, J. Martinovič, and K. Slaninová, "Betweenness propagation," in *IFIP International Conference on Computer Information Systems and Industrial Management*. Springer, 2018, pp. 279–287.

[38] V. Svatoň, M. Podhoranyi, R. Vavřík, P. Veteška, D. Szturcová, D. Vojtek, J. Martinovič, and V. Vondrák, "Floreon+: A web-based platform for flood prediction, hydrologic modelling and dynamic data analysis," in *Dynamics in GIscience*, 2018, pp. 409–422.

[39] M. Golasowski, J. Bispo, J. Martinovič, K. Slaninová, and J. M. Cardoso, "Expressing and applying c++ code transformations for the hdf5 api through a dsl," in *IFIP Int'l Conf on Computer Information Systems and Industrial Management*. Springer, 2017, pp. 303–314.

[40] M. Golasowski, R. Tomis, J. Martinovič, K. Slaninová, and L. Rapant, "Performance evaluation of probabilistic time-dependent travel time computation," in *IFIP Int'l Conf on Computer Information Systems and Industrial Management*. Springer, 2016, pp. 377–388.

[41] V. Ptošek, J. Ševčík, J. Martinovič, K. Slaninová, L. Rapant, and R. Cmar, "Real time traffic simulator for self-adaptive navigation system validation," in *Proceedings of EMSS-HMS: Modeling & Simulation in Logistics, Traffic & Transportation*, 2018.