



<http://www.diva-portal.org>

Preprint

This is the submitted version of a paper presented at *NBiS 2015*.

Citation for the original published paper:

Abraham, E., Bekas, C., Brandic, I., Genaim, S., Johnsen, E. et al. (2015)

Preparing HPC Applications for Exascale: Challenges and Recommendations.

In: *Proceedings: 2015 18th International Conference on Network-Based Information Systems, NBiS 2015* (pp. 401-406).

<http://dx.doi.org/10.1109/NBiS.2015.61>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-48318>

# Preparing HPC Applications for Exascale: Challenges and Recommendations

(ADPNA at NBIS-2015, ©IEEE)

Erika Ábrahám\*, Costas Bekas†, Ivona Brandić‡, Samir Genaim§, Einar Broch Johnsen¶,

Ivan Kondov||, Sabri Pllana\*\* and Achim Streit||

\* RWTH Aachen University - Aachen, Germany

† IBM Research - Zurich, Switzerland

‡ Vienna University of Technology, Vienna, Austria

§ Complutense University of Madrid, Madrid, Spain

¶ University of Oslo, Department of Informatics, Norway

|| Karlsruhe Institute of Technology, Steinbuch Centre for Computing, Germany

\*\* Linnaeus University, Department of Computer Science, Växjö, Sweden

**Abstract**—While the HPC community is working towards the development of the first Exaflop computer (expected around 2020), after reaching the Petaflop milestone in 2008 still only few HPC applications are able to fully exploit the capabilities of Petaflop systems. In this paper we argue that efforts for preparing HPC applications for Exascale should start before such systems become available. We identify challenges that need to be addressed and recommend solutions in key areas of interest, including formal modeling, static analysis and optimization, runtime analysis and optimization, and autonomic computing. Furthermore, we outline a conceptual framework for porting HPC applications to future Exascale computing systems and propose steps for its implementation.

## I. INTRODUCTION

Exascale computing [1] is expected to revolutionize computational science and engineering by providing 1000x the capabilities of currently available computing systems, while having a similar power footprint. The total performance of the 500 systems in the 44th TOP500 list (18 Nov 2014, <http://top500.org/>) is about 0.3 exaFLOPS. The HPC community [2] is now working towards the development of the first Exaflop computer, expected around 2020, after reaching the Petaflop milestone in 2008. However, only a few HPC applications are so far able to fully exploit the capabilities of Petaflop systems [3]. Examples of typical scalability for commonly used HPC applications in our organizations are provided in Table I. As the existing HPC applications are the major HPC asset, it is important and challenging to increase their scalability and lifetime by making them Exascale-ready before 2020.

The major challenge for preparing HPC applications for Exascale is that there is no Exascale system available yet. Currently all we have are assumptions about Exascale systems. Therefore the commonly used measurement-based approaches for reasoning about performance issues are not applicable. Pre-exascale systems (known as *Summit* and *Sierra*) that IBM [4] is developing for the U.S. Department of Energy will exceed 100 petaflops and may provide hints about the extreme-scale architectures of the future.

This paper argues that efforts for preparing HPC applications for Exascale should start before such systems become available. We identify challenges that need to be addressed and recommend solutions in areas that are relevant for porting HPC application to future Exascale computing systems, including formal modeling, static analysis and optimization, runtime analysis and optimization, and autonomic computing.

We suggest that porting of HPC applications should be made by successive, stepwise improvements based on the currently available assumptions and data about Exascale systems. This approach should support application improvement each time new information about future Exascale systems becomes available, including the time when the application is actually deployed and runs on a concrete Exascale system. A high-level application representation that captures key functional and non-functional properties in conjunction with the abstract machine model will enable programmers and tools to reason about and perform application improvements, and will serve as input to runtime systems to handle performance and energy optimizations and self-aware fault management. A tunable abstract machine model encapsulates current assumptions for future Exascale systems and enables a priori application improvement before the concrete execution platform is known. At runtime, the model is a posteriori tuned to support activities such as feedback-oriented code improvement or dynamic optimization.

Major contributions of this paper include,

- identification of challenges and recommendation of solutions in formal modeling (Section II-A), static analysis and optimization (Section II-B), runtime analysis and optimization (Section II-C), autonomic computing (Section II-D);
- a conceptual framework for preparing HPC applications for Exascale that supports *a priori* application improvements before the concrete execution platform is known as well as *a posteriori* optimization at runtime (Section III-A);

Table I  
TYPICAL CURRENT SCALABILITY (IN PROCESSOR CORES) OF COMMONLY USED HPC APPLICATIONS IN OUR ORGANIZATIONS.

Code	Application Domain	Language	Scalability
WIEN2k	Materials Science	F90	1024
SIMONA	Nano Science	C++	16384
ECHAM/MESSy	Environmental Science	F77/F90	1000
CORSIKA	Astroparticle Physics	F77/F90	2500
OpenFOAM	Computational fluid dynamics	C++	16384
IBM Watson	Graph Analytics	C++	32768
Bifrost	Stellar atmosphere simulation	F90	6500

- a discussion of the related work (Section IV).

## II. CHALLENGES AND RECOMMENDATIONS

In this section we identify challenges and recommend solutions in formal modeling, static analysis and optimization, runtime analysis and optimization, and autonomic computing.

### A. Formal modeling

Our goal is to adapt HPC application code to Exascale execution platforms to achieve good utilization of resources. For this, we need to address questions such as:

- 1) What would happen if we change application or hardware layout?
- 2) What would happen if we change some parameters of the execution platform?
- 3) What would happen if we use a different execution platform?

Unfortunately, answering these questions cannot be done experimentally at the concrete level because such platforms do not yet exist. An alternative is to address these questions at an abstract level, focusing only on *relevant information* without actually executing the program.

We believe that relevant information in this context is not what the code aims to achieve (the result of the computation) but its corresponding *resource footprints*, that is, how computational tasks communicate and synchronize, the amount of resources (such as memory and computing time) these tasks require, and how they access and move data.

In order to adapt the HPC code to a particular architecture we need to capture such resource footprints of software modules at different levels of granularity (e.g., program statements, blocks in procedure bodies and whole procedures), and be able to compare different task compositions. Consequently, the modeling language must feature massively parallel operators over such task-level resource footprints [5]. A similar notion of resource footprints and composition can be used to express the properties of the architecture in a machine model to capture the resources that the architecture can make available to the code.

Working with resource footprints can be supported by an abstract behavioral specification language [6], in which

models describe both tasks and deployments. These models can be used to predict the non-functional behavior of code before it is deployed, and to compare deployments using formal methods. This requires a formal semantics for the specification language that can be used to devise static analysis techniques.

When developing code from scratch using a model-based approach, the resource footprints can be specified in tandem with the standard model in a model-driven development [7], [8], [9], [10]. However, when building such models from existing HPC code, monitoring profiles of low- and medium-scale systems can be used to extract resource footprints that approximate the resource consumption in terms of probabilistic distributions.

### B. Static analysis and optimization

The application of formal methods to parallel programs for analyzing functional properties, such as safety and liveness, has a long tradition. For *non-functional properties*, such as execution time and energy consumption, most performance analysis approaches use *monitoring* and present statistical information to the user. These approaches are helpful to improve HPC application code, but they also have some shortcomings:

- 1) Due to non-determinism, different program executions might lead to different observations. As a consequence, these methods are not able to provide reliable probabilistic information about average or worst-case execution times.
- 2) They are based on execution on a real platform, thus they cannot be used to predict performance on Exascale computers, which are not available yet.
- 3) These methods can be used to identify execution bottlenecks, but they cannot explain the reasons for these bottlenecks, and thus they do not offer any concrete support for code improvement.

We expect that *formal methods* can address these limitations to provide performance analysis tools that considerably go beyond the state-of-the-art. A major step in this direction will be the usage of resource footprints which describe both HPC applications and execution platforms as abstract probabilistic models. Formal analyses can be

applied to these models to predict their probabilistic behavior. While a range of techniques are available for non-probabilistic programs, the analysis of parallel probabilistic programs still need development effort. To achieve a reasonable balance between scalability and precision for challenging HPC applications, it seems fundamental to use *hybrid approaches* [11] that combine techniques such as static analysis, dynamic analysis, simulation, (parametric) model checking [12], counterexample-guided abstraction refinement [13], deductive approaches, etc.

To deliver the envisaged performance analysis tools, we face the following challenges: (1) determining the computation of cost properties that are given by means of probabilistic distributions; (2) the inference of average cost in addition to the traditional worst-case cost; (3) take into account the underlying platform through a set of probabilistic parameters; (4) deal with massive and heterogeneous parallelism [14], [15], [16], [17] which is challenging for program analysis in general; and (5) develop multi-objective resource usage analyses and optimizations.

### C. Runtime analysis and optimization

Formal modeling and static analysis should be enhanced with analysis of measurements at runtime. Plenty of tools (for instance, <http://www.vi-hps.org>) have been developed for performance measurement and analysis of HPC applications at runtime. However, these tools will experience several issues when applied to Exascale. The collection rates and the overall volume of monitoring data in an Exascale computing environment will exceed the scalability of present performance tools. Therefore, throttling the data volume will have to be applied online in order to store as less data as possible and as much as necessary for later *post mortem* analysis. However, simple profiling will not be sufficient due to loss of temporal information, thus a hybrid approach will have to be applied that performs on-the-fly trace analysis in order to discard irrelevant data, while retaining the same amount of information.

The metric classification should be based on the formal model (see Section II-A). Such an approach will provide a generic insight into the performance of an HPC application that can be used for detecting performance bottlenecks. The instrumentation and hardware counter monitoring should follow a similar procedure where source code probing should be applied automatically by using tools such as OPARI [18]. While many tools for collecting metrics of computing performance have been developed, very few analysis tools exist for energy consumption metrics in adequate accuracy and time resolution necessary for the runtime performance analysis [19], [20].

Currently, common approaches (see for example PRACE best practices [21]) for optimizing HPC applications require per-case inspection of runtime performance measurement data, such as profiling and tracing data. After the critical re-

gion has been determined, diverse heuristic approaches, such as “*trial and error*”, “*educated guess*” or “*rule of thumb*”, are applied to make changes in the affected source code sections. The most significant limitation of these heuristics and knowledge-based approaches is that,

- 1) all changes are made directly and manually in the source code, and
- 2) the effect of the changes does not always lead to an improved performance which makes necessary the repeating of all steps several times.

Moreover, Exascale computers pose a multi-objective optimization problem, weighing out the effects of several sometimes incongruent requirements. Therefore, a systematic and automatic approach for the optimization problem is essential to find the optimal solution. Another problem is that the critical section in an application typically changes with the optimization iterations and/or with upscaling, due to the law of diminishing returns, which makes the manual analysis and source code changes even more laborious and inefficient, even if done by an experienced HPC developer. Thus instrumentation, collection/measurement and analysis steps should be automated, for example based on high-level scalable tools [22], [23], [24], and integrated into a feedback loop (see Section II-D).

### D. Autonomic computing

During the execution of an application, failures may occur or the application performance may be below the expectation. These issues are addressed typically by programmers in a “*trial and error*” manner, i.e. by manually changing and adapting their code to handle the failures and improve the performance. Our proposed framework (Section III-A) provides means for model-based failure handling or performance improvement based on *autonomic computing*. Autonomic computing addresses self-managing characteristics of distributed computing resources with the facilities to adapting to unpredictable changes while hiding management complexity to operators and users [25]. Among the explored categories, *advanced-control based methods* and more specifically *distributed controllers* are the first candidates to realize autonomic computing in Exascale systems.

We propose to devise methodologies to efficiently collect runtime information balancing the amount and cost for storage of monitoring data with the quality of monitored data necessary to make deductions about the application behavior (e.g. trace analysis). The goal is thereby to define methodologies to scale current monitoring tools to Exascale, balancing between quality and volume of monitoring data.

Combining the information from both static code analysis and runtime analysis, as outlined above, we will iteratively apply objective-oriented transformations to legacy application code at a formal level based on the Exascale DSL model (see Section II-A). To this end, we will automate the analysis, optimization and transformation processes by

implementing a generic feedback loop independent of the concrete programming language, algorithms used and target hardware architecture. A feedback loop driver enables to link the static analysis tool, the runtime analysis tool, the knowledge database and multi-parameter multi-objective optimization. As output, a set of rules (policies) is generated which is then applied to transform the formal application model and to adapt the runtime environment parameters (cf. Figure 1). After the transformations a new application executable is built and started in the adapted runtime environment. This described loop is iterated until convergence of the optimization.

### III. CONCEPTUAL FRAMEWORK AND BENEFITS

In this section we propose a conceptual framework for porting HPC applications to Exascale computing systems. Furthermore, we highlight benefits of our conceptual framework in the context of Exascale computing.

#### A. Conceptual Framework

Our proposed approach for preparing HPC applications for Exascale is depicted in Figure 1. The usage of a *Domain-Specific Exascale Language (DSEL)* facilitates the programmer to express non-functional aspects (like required time to solution, resilience or energy-efficiency) of the execution of scalable parallel HPC codes. DSEL has a formal operational semantics that enables the formal analysis of the code. The aim of the *Scalable Model-based Analyzer (SMA)* is to address non-functional properties of HPC codes, with a particular focus on scalability while complying with the crucial dimensions of resource consumption for Exascale computing: time, energy, and resilience. The SMA is responsible for analyzing resource consumption in terms of time, energy, and resilience, based on developed DSELS. The *Exascale Runtime Data Collector (ERDC)* is responsible for scalable monitoring to *extract important monitoring data* through the utilization of various techniques like *filtering*, *streaming*, or *data mining*. The runtime information is used to verify or to tune the model of the code via the *Autonomous Feedback Loop (AFL)*. To endow the system with self-adaption, control-theoretical concepts are incorporated in autonomic computing paradigm. Based on the autonomic technology for application optimization, programmers will be less dependent on the currently used “*trial-and-error*” approach.

Our approach considers optimization opportunities during the application life cycle comprising improvements based on static code analysis, deployment-time optimization, and run-time optimization. The developed models are used to identify the potential for improvement of the scalability for HPC applications under study and suggest application modifications that may lead to better scalability.

#### B. Benefits

Exascale computing is not simply the continuation of a computational capability trend that has been proven true for the last five decades. First, while clock rate scaling is limited, complex multicore architectures and parallel computing still follow Moore’s law. Second, Exascale computing capability will finally allow complex real-life simulations and data analytics. The latter will greatly expand the horizons of scientific discovery and enable the new data-driven economy to become a reality.

However, the Exascale promise faces a series of obstacles, with the most difficult being energy, scalability, reliability and programmability. Our proposal is to develop a holistic, unifying and mathematically founded framework to systematically attack the roots of these problems. That is, instead of attacking these problems separately, we propose a holistic approach to study them as a multi-parametric problem which will allow us to deeply understand their interplay and thus make the right decisions to navigate in this complex landscape.

The benefits are targeting the full spectrum of actors and beneficiaries. System developers will have a much better path to design, while end users and application developers will benefit from increased scalability, performance, reliability and programmability. HPC centers will see a great increase in overall system usability and an energy budget that is affordable. This in turn has the potential to greatly limit and contain the overall impact of high end HPC to the environment.

### IV. RELATED WORK

In a prospective analysis of issues with extreme scale systems [26], the importance of concurrency, energy efficiency and resilience of software, as well as software–hardware co-design has been elucidated.

Focusing on energy-aware HPC numerical applications, the EXA2GREEN project (<http://exa2green-project.eu>) has developed energy-aware performance metrics [27], as well as energy-aware basic algorithm motifs such as linear solvers [28]. Further work will strongly benefit from these results. Different power measurement interfaces available on current architecture generations have been evaluated and the role of the sampling rate has been discussed [29].

The AutoTune approach [30] employs the Periscope tuning framework [31] to automate performance analysis and tuning of HPC applications with the goal to improve performance and energy efficiency. Therein, both performance analysis and tuning are performed automatically during a single run of the application.

The DEEP project [32] has developed a novel Exascale-enabling supercomputing architecture with a matching software stack and a set of optimized grand-challenge simulation applications. The goal of the DEEP architecture is to enable

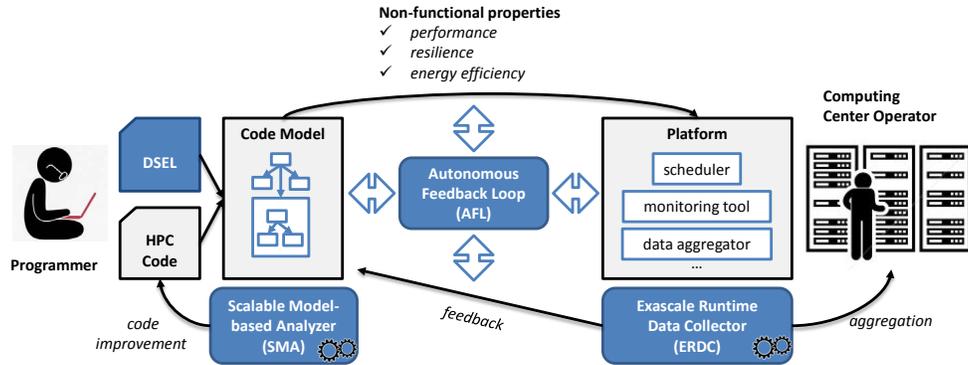


Figure 1. Our conceptual framework for porting HPC applications to Exascale computing systems

unprecedented scalability and with an extrapolation to millions of cores to take the DEEP concept to an Exascale level. The follow-up DEEPer project (<http://www.deep-er.eu>) is mainly focusing on I/O and resiliency aspects.

The CRESTA project (<http://www.cresta-project.eu>) has adopted a co-design strategy for Exascale, including all aspects of hardware architectures, system and application software. A major asset from the CRESTA project is the Score-P measurement system [33] on which an integration and automation of performance analysis tools (cf. Section II-C) can be based. In addition, efforts have been made on developing a domain-specific language for expressing parallel auto-tuning specifications and an adaptive runtime support framework.

## V. SUMMARY

Exascale computing will revolutionize high-performance computing, but the first Exascale systems are not expected to appear before 2020. In this paper we have argued that the effort for preparing HPC applications for Exascale should start now. We have proposed that porting of HPC applications should be made by successive, stepwise improvements based on the currently available assumptions and data about Exascale systems. This approach should support application improvement each time new information about future Exascale systems becomes available, including the time when the application is actually deployed and runs on a concrete Exascale system. We have identified challenges that need to be addressed and recommended solutions in key areas of interest for our approach including: formal modeling, static analysis and optimization, runtime analysis and optimization, and autonomic computing. Our future research will address the development of a framework that supports the conceptual framework presented in this paper.

## REFERENCES

- [1] E. D'Hollander, J. Dongarra, I. Foster, L. Grandinetti, and G. Joubert, Eds., *Transition of HPC Towards Exascale Computing*, ser. Advances in Parallel Computing. IOS Press, 2013, vol. 24.
- [2] J. Dongarra and et al., "The international exascale software project roadmap," *Int. J. High Perform. Comput. Appl.*, vol. 25, no. 1, pp. 3–60, 2011.
- [3] D. A. Bader, Ed., *Petascale Computing: Algorithms and Applications*, ser. Chapman & Hall/CRC Computational Science series. Chapman & Hall/CRC, 2007.
- [4] "U.S. Department of Energy Selects IBM Data Centric Systems to Advance Research and Tackle Big Data Challenges," 2014. [Online]. Available: <http://www-03.ibm.com/press/us/en/pressrelease/45387.wss>
- [5] S. Pllana, I. Brandic, and S. Benkner, "A Survey of the State of the Art in Performance Modeling and Prediction of Parallel and Distributed Computing Systems," *International Journal of Computational Intelligence Research (IJ CIR)*, vol. 4, no. 1, pp. 17–26, January 2008.
- [6] E. B. Johnsen, R. Hähnle, J. Schäfer, R. Schlatter, and M. Steffen, "ABS: A core language for abstract behavioral specification," in *9th International Symposium on Formal Methods for Components and Objects (FMCO'10)*, ser. Lecture Notes in Computer Science, vol. 6957. Springer, 2011, pp. 142–164.
- [7] E. Arkin, B. Tekinerdogan, and K. M. Imre, "Model-driven approach for supporting the mapping of parallel algorithms to parallel computing platforms," in *Model-Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 8107, pp. 757–773.
- [8] S. Pllana and T. Fahringer, "On customizing the UML for modeling performance-oriented applications," in *UML*, ser. Lecture Notes in Computer Science, vol. 2460. Springer, 2002, pp. 259–274.
- [9] S. Pllana, S. Benkner, E. Mehofer, L. Natvig, and F. Xhafa, "Towards an intelligent environment for programming multi-core computing systems," in *Euro-Par 2008 Workshops - Parallel Processing*, ser. Lecture Notes in Computer Science, vol. 5415. Springer, 2008, pp. 141–151.
- [10] T. Fahringer, S. Pllana, and J. Testori, "Teuta: Tool support for performance modeling of distributed and parallel applications," in *Computational Science - ICCS 2004*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3038, pp. 456–463.

- [11] S. Pllana, S. Benkner, F. Khafa, and L. Barolli, "Hybrid performance modeling and prediction of large-scale parallel systems," in *Parallel Programming, Models and Applications in Grid and P2P Systems*, ser. Advances in Parallel Computing. IOS Press, 2009, vol. 17, pp. 54–82.
- [12] N. Jansen, F. Corzilius, M. Volk, R. Wimmer, E. Ábrahám, J.-P. Katoen, and B. Becker, "Accelerating parametric probabilistic verification," in *Quantitative Evaluation of Systems (QEST'14)*, ser. Lecture Notes in Computer Science, vol. 8657. Springer, 2014, pp. 404–420.
- [13] C. Dehnert, N. Jansen, R. Wimmer, E. Ábrahám, and J.-P. Katoen, "Fast debugging of PRISM models," in *International Symposium on Automated Technology for Verification and Analysis (ATVA'14)*, ser. Lecture Notes in Computer Science, vol. 8837. Springer, 2014, pp. 146–162.
- [14] S. Benkner, S. Pllana, J. Traff, P. Tsigas, U. Dolinsky, C. Augonnet, B. Bachmayer, C. Kessler, D. Moloney, and V. Osipov, "PEPPER: Efficient and Productive Usage of Hybrid Computing Systems," *Micro, IEEE*, vol. 31, no. 5, pp. 28–41, Sept 2011.
- [15] C. Kessler, U. Dastgeer, S. Thibault, R. Namyst, A. Richards, U. Dolinsky, S. Benkner, J. Traff, and S. Pllana, "Programmability and performance portability aspects of heterogeneous multi-/manycore systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, March 2012, pp. 1403–1408.
- [16] M. Sandrieser, S. Benkner, and S. Pllana, "Using Explicit Platform Descriptions to Support Programming of Heterogeneous Many-Core Systems," *Parallel Computing*, vol. 38, no. 1-2, pp. 52–56, January 2012.
- [17] J. Dokulil, E. Bajrovic, S. Benkner, S. Pllana, M. Sandrieser, and B. Bachmayer, "Efficient hybrid execution of C++ applications using intel(r) xeon phi(tm) coprocessor," *CoRR*, vol. abs/1211.5530, 2012. [Online]. Available: <http://arxiv.org/abs/1211.5530>
- [18] B. Mohr, A. D. Malony, S. Shende, and F. Wolf, "Towards a performance tool interface for OpenMP: An approach based on directive rewriting," in *Third European Workshop on OpenMP (EWOMP'01)*, 2001.
- [19] P. Alonso, R. Badia, J. Labarta, M. Barreda, M. Dolz, R. Mayo, E. Quintana-Orti, and R. Reyes, "Tools for power-energy modelling and analysis of parallel scientific applications," in *2012 41st International Conference on Parallel Processing (ICPP'12)*. IEEE, 2012, pp. 420–429.
- [20] A. Bohra and V. Chaudhary, "VMeter: Power modelling for virtualized clouds," in *2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW'10)*. IEEE, 2010, pp. 1–8.
- [21] PRACE, "Best Practice Guides," Accessed 2015-02-03. [Online]. Available: <http://www.prace-ri.eu/best-practice-guides/>
- [22] M. Geimer, F. Wolf, B. Wylie, and B. Mohr, "Scalable parallel trace-based performance analysis," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, ser. Lecture Notes in Computer Science. Springer, 2006, vol. 4192, pp. 303–312.
- [23] B. Mohr and F. Wolf, "KOJAK: A tool set for automatic performance analysis of parallel programs," in *Euro-Par 2003 Parallel Processing*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2790, pp. 1301–1304.
- [24] F. Wolf and B. Mohr, "Automatic performance analysis of hybrid MPI/OpenMP applications," in *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing*. IEEE, 2003, pp. 13–22.
- [25] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003. [Online]. Available: <http://dx.doi.org/10.1109/MC.2003.1160055>
- [26] V. Sarkar, W. Harrod, and A. E. Snavely, "Software challenges in extreme scale systems," *Journal of Physics: Conference Series*, vol. 180, no. 1, p. 012045, 2009.
- [27] C. Bekas and A. Curioni, "A new energy aware performance metric," *Computer Science - Research and Development*, vol. 25, no. 3-4, pp. 187–195, 2010. [Online]. Available: <http://dx.doi.org/10.1007/s00450-010-0119-z>
- [28] P. Klavík, A. C. I. Malossi, C. Bekas, and A. Curioni, "Changing computing paradigms towards power efficiency," *Philosophical Transactions of the Royal Society A*, vol. 372, no. 2018, 2014. [Online]. Available: <http://dx.doi.org/10.1098/rsta.2013.0278>
- [29] M. E. M. Diouri, M. F. Dolz, O. Glück, L. Lefèvre, P. Alonso, S. Cataln, R. Mayo, and E. S. Quintana-Ort, "Assessing power monitoring approaches for energy and power analysis of computers," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 2, pp. 68 – 82, 2014.
- [30] R. Miceli, G. Civario, A. Sikora, E. César, M. Gerndt, H. Haitof, C. Navarrete, S. Benkner, M. Sandrieser, L. Morin, and F. Bodin, "Autotune: A plugin-driven approach to the automatic tuning of parallel applications," in *Applied Parallel and Scientific Computing*, ser. Lecture Notes in Computer Science. Springer, 2013, vol. 7782, pp. 328–342.
- [31] S. Benedict, V. Petkov, and M. Gerndt, "PERISCOPE: An Online-Based Distributed Performance Analysis Tool," in *Tools for High Performance Computing 2009*. Springer, 2010, pp. 1–16.
- [32] N. Eicker, T. Lippert, T. Moschny, and E. Suarez, "The DEEP project - Pursuing cluster-computing in the many-core era," in *2013 42nd International Conference on Parallel Processing (ICPP'13)*, 2013, pp. 885–892. [Online]. Available: <http://dx.doi.org/10.1109/ICPP.2013.105>
- [33] X. Aguilar, J. Doleschal, A. Gray, A. Hart, D. Henty, T. Hilbrich, D. Lecomber, S. Markidis, H. Richardson, and M. Schliephake, "The Exascale Development Environment: State of the art and gap analysis (CRESTA White Paper)," 2013, accessed 2015-02-04. [Online]. Available: <http://cresta-project.eu/>