Run-time Exploitation of Application Dynamism for Energy-efficient Exascale Computing (READEX)

Yury Oleynik*, Michael Gerndt*, Joseph Schuchart†, Per Gunnar Kjeldsberg‡, and Wolfgang E. Nagel†
*Chair of Computer Architectures
Technische Universität München
Garching, Germany
†Center for Information Services and High Performance Computing
Technische Universität Dresden
Dresden, Germany
‡Department of Electronics and Telecommunications
Norwegian University of Science and Technology
Trondheim, Norway

***Abstract*—Efficiently utilizing the resources provided on current petascale and future exascale systems will be a challenging task, potentially causing a large amount of underutilized resources and wasted energy. A promising potential to improve efficiency of HPC applications stems from the significant degree of dynamic behavior, e.g., run-time alternation in application resource requirements in HPC workloads. Manually detecting and leveraging this dynamism to improve performance and energy-efficiency is a tedious task that is commonly neglected by developers. However, using an automatic optimization approach, application dynamism can be analyzed at design-time and used to optimize system configurations at run-time.**

The European Union Horizon 2020 READEX project will develop a tools-aided scenario based auto-tuning methodology to exploit the dynamic behavior of HPC applications to achieve improved energy-efficiency and performance. Driven by a consortium of European experts from academia, HPC resource providers, and industry, the READEX project aims at developing the first of its kind generic framework for split design-time run-time automatic tuning for heterogeneous system at the exascale level.

I. Introduction

High Performance Computing (HPC) is a major driving force for research and innovation in many scientific and industrial areas. A constantly growing demand for computing performance leads to the installation of increasingly powerful and ever more complex systems characterized by a rising number of CPU cores as well as increasing heterogeneity. This makes optimization of HPC application a complex task demanding severe programming effort and high level of expertise.

With growing computational performance, there is typically also an increase in a system's energy consumption, which in turn is a major driver for the total cost of ownership of HPC systems. Limitations to chip temperature and cooling capabilities can furthermore make the performance of exascale HPC systems power-bound. However, developers commonly focus on the implementation and improvement of algorithms with regards to accuracy and performance, neglecting possible improvements to energy-efficiency. The fact that programmers in general lack the platform and hardware knowledge required to exploit these measures is an important obstacle for their use.

The European Union READEX Project [1] tackles the challenges above by embracing the significant potential for improvements to performance and energy-efficiency stemming from the fact that HPC applications commonly exhibit dynamic resource requirements, e.g., alternating application regions or load-changes at application run-time. We will refer to this as application *dynamism* throughout the paper. Examples of dynamism can be found in many current HPC applications, including weather forecasting, molecular dynamics, or adaptive mesh-refinement applications.

It is expected that applications running on future extreme-scale systems will exhibit even higher levels of dynamism. This will be mainly due to the increased demand on data movement between processing elements, both on intra- and inter-node levels, and more complex levels of the memory hierarchy. Furthermore, the rise of many-core co-processors and accelerators introduced new degrees of freedom such as offloading and scheduling. These new types of hardware will play an increasingly important role in the race to Exascale computing. At the same time, dynamic changes in the performance of the available resources will require applications to adapt at run-time. Especially in the area of extreme data applications, the mix of I/O and computational phases announces a high degree of dynamic behaviour.

The READEX project will develop and implement a tools-aided methodology that enables HPC application developers to exploit dynamic application behaviour when run on current and future extreme parallel and heterogeneous multi-processor platforms. READEX combines and extends state-of-the-art technologies in performance and energy-efficiency tuning for HPC with dynamic energy optimization techniques for embedded systems.

The general concept of the READEX project is to handle application energy-efficiency and performance tuning by taking a complete application life-cycle approach, in contrast to other approaches that regard performance and energy tuning as a static single activity taking place in the application development phase. With inspiration from systems scenario based design in the embedded systems domain, READEX will develop a *dynamic* auto-tuning methodology spanning over the development (design-time) and production/maintenance (run-time) phases of the application life-cycle.

The requirements for the READEX tool-suite listed below outline the technical ambitions of the project:

- (Semi)-automatic dynamic tuning
- Support for heterogeneous systems
- Capability for exascale deployment
- Multi-objective tuning
- Programming paradigm for application domain dynamism

II. Related Work

While a small number of dynamic auto-tuning methodologies and tools exist for run-time optimizations [2], [3], no single standalone dynamic auto-tuning framework currently exists with the capability to target the full breadth of large-scale HPC applications being used in academia and industry both now and on the road to Exascale.

Several leading EU research projects are approaching the challenge of tuning for performance and energy-efficiency by either introducing entirely new programming models or leveraging existing prototype languages. An example of the latter approach is the ENabling technologies for a programmable many-CORE (ENCORE) project [4], which aims to achieve massive parallelism relying on tasks and efficient task scheduling using the OmpSs programming model [5]. The READEX project takes a different approach by developing a new generic programming paradigm allowing to express and to utilize dynamism of applications in the automatic tuning process.

Further European projects are focusing their optimisation efforts on heterogeneous systems. For instance, the Performance Portability and Programmability for Heterogeneous Many-core Architectures PEPPHER project [6] has developed a methodology and framework for programming and optimizing applications for single-node heterogeneous many-core processors to ensure performance portability. With Intel as a key partner of the READEX project, we will go one step further and provide a framework that supports the heterogeneity of the system in the form of tuning parameters that allows for large-scale heterogeneous applications to dynamically (and automatically) adapt to heterogeneous resources according to run-time requirements.

To the best of our knowledge, no dynamic auto-tuning framework exists yet that shows the potential to scale to future Exaflop machines. Auto-tuning frameworks typically follow a centralized approach, where the central agent will become a bottleneck when deployed on these systems. In contrast, READEX aims to develop the concept of (semi)-distributed dynamic tuning that minimises centralisation of control.

III. Background

READEX synergistically combines and extends two technologies from the opposite ends of the computing continuum, namely the systems scenario methodology for dynamic tuning from the field of embedded systems with the automatic static tuning from the area of HPC.

A. System Scenarios Methodology

In the embedded systems domain a scenario-based methodology [7]–[9] has been developed to enable exploitation of application dynamism through fine-grained run-time system tuning.

The methodology consists of exploiting detailed knowledge about the application(s) to be run on the system, extracted through profiling and (semi-)automatic code inspection. Using these techniques, different Run-Time Situations (RTSs) of the application are identified that have different costs related to them, e.g., execution time, energy consumption, and memory foot-print.

At design-time, RTSs are grouped into scenarios with similar multidimensional system costs. Optimized platform configurations are generated for each scenario. Furthermore, efficient and possibly application-specific scenario prediction and scenario switching mechanisms are developed.

At run-time, the upcoming scenario is predicted and a platform configuration switch is performed based on the mechanisms developed at design-time. Should the application experience an RTS which was not seen at design-time a back-up scenario guaranteed to satisfy any RTS is used.

Examples of more than 30% energy reductions have been reported for the system scenarios methodology in the embedded systems field [8]. This type of technique has not been applied in the HPC domain and extensive research is needed for it to happen.

B. Static Auto-tuning of HPC Applications

Most of the current tools for performance engineering focus on collecting and presenting information for the users, while only few focus on the automation of the performance optimization process (auto-tuning), e.g., the Periscope Tuning Framework (PTF) developed in the EU FP7 ICT AutoTune project [10], [11]. PTF automatically finds optimized system configurations for whole application runs, effectively averaging the benefits of system adaptation over the whole run-time of the application (static tuning). With these static auto-tuning techniques, improvements in energy-efficiency of up to 10% for application runs have already been achieved while keeping the performance degradation to a few percent [12].

PTF's main principles are the use of formalized expert knowledge and strategies, an extensible and modular architecture based on tuning plugins, automatic execution of experiments, and distributed, scalable processing. PTF provides a number of predefined tuning plugins, including:

- Dynamic Voltage Frequency Scaling,
- Compiler flags selection,
- MPI run-time settings,
- OpenMP parallelism capping,
- MPI master-worker pattern settings

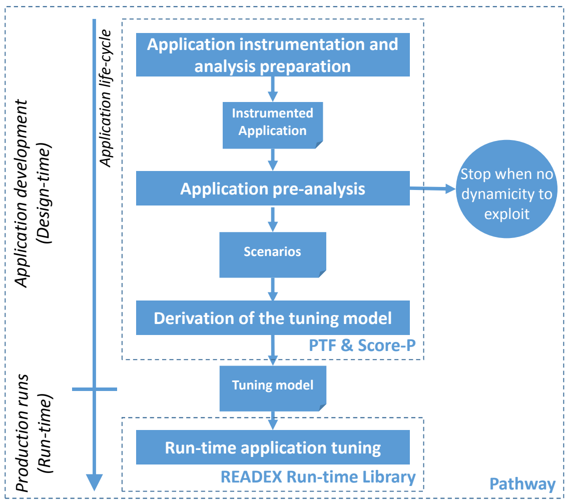
Application development (Design-time)
Production runs (Run-time)
Application life-cycle
Application instrumentation and analysis preparation
Instrumented Application
Application pre-analysis
Stop when no dynamicity to exploit
Scenarios
Derivation of the tuning model
PTF & Score-P
Tuning model
Run-time application tuning
READEX Run-time Library
Pathway

Fig. 1. Overview of the READEX methodology.

PTF also provides the Tuning Plugin Interface for the development of new plugins. It builds on the common measurement tools infrastructure Score-P [13].

IV. Approach

The READEX approach combines the concept of system scenarios with automatic energy and performance tuning into a holistic tools-aided methodology spanning over the major parts of the HPC application life-cycle, namely, application development and performance tuning at design-time and production runs at run-time. Figure 1 provides a high-level overview of the methodology.

In order to support the user as well as to allow for a smooth user interaction during the whole process, the Pathway [14] performance engineering workflow tool will be extended with the READEX workflow allowing to automate routine tasks as well as to capture and to protocol the progress. The READEX workflow can also be used with other workflow tools or carried out manually.

A. Application Instrumentation and Analysis Preparation

During the first step of the methodology, the application is instrumented and application-domain knowledge is provided by the user if desired using a new programming paradigm to be developed in the project. It will provide a possibility for exposing parameters that define dynamic behavior of the application to the READEX tool-suite. These application domain parameters, referred to as identifiers, will enhance distinguishing of different system scenarios and correspondingly adapting system configurations in order to improve overall application performance and energy characteristics.

The paradigm will also allow developers to expose additional application-level tuning parameters to the tuning process, e.g., alternative code-paths that will be chosen based on the provided identifiers.

In addition to the optional information provided by the user, the application is instrumented by automatically inserting probe-functions around relevant code regions using existing technologies from the Score-P infrastructure which allows for a fine-grained analysis and tuning.

B. Application Pre-analysis

Based on the performance dynamics analysis capabilities of PTF, the READEX analysis strategy will be developed. It will automatically characterize present dynamism and indicate the optimization potential. The latter gives the user an estimate for the performance and energy-efficiency gains which could be realized using the READEX methodology.

Following this strategy, the application is run with a representative data set and relevant performance and energy metrics are collected over the repetitive regions of the application. This results in time-series of measurements representing temporal evolution of each regions performance in a multi-dimensional space. According to the system scenario methodology, each region's execution, represented by a point in this space, corresponds to a run-time situation (RTS). By grouping the points according to a multi-dimensional similarity function, clusters of points with similar performance and energy properties, so called system scenarios, are obtained.

In order to recognize present system scenarios, a classifier based on the provided identifiers is built. It is used in the later methodology steps for detecting a scenario before it is executed so that the optimal system configuration can be applied. For example, a combination of the function call-path associated with the value of the function arguments can help distinguish the expected function performance and, thus, to univocally identify the upcoming scenario entered by the application.

C. Derivation of the Tuning Model

In order to build a tuning model allowing adaptation of the system (both application and platform) to the dynamically changing requirements, PTF and Score-P will be extended to perform automatic search for optimal system configurations for the scenarios identified in the previous step.

Exploration of the space of possible tuning configurations is controlled by PTF tuning plugins, where each tuning plugin is responsible for a specific tuning parameter. READEX will support and develop a number of plugins for hardware, system software, and application parameters.

To facilitate the determination of optimal platform configurations, PTF will run the instrumented application. For the identified scenarios, various system configurations are evaluated in terms of the requested objective functions and results are stored in the tuning database. Since the search space for optimal configurations is potentially large, new search strategies will have to be developed.

After all relevant system configurations are evaluated for all system scenarios, the plugins create a tuning model based on the information stored in the tuning database. The model contains a look-up table providing the Pareto-optimal system configurations for the known scenarios. The database itself is exported as part of the tuning model, so that it can be populated with new information during production runs.

D. Run-time Application Tuning

When it comes to production runs, the READEX methodology uses the previously obtained knowledge about application dynamism to adapt both the application and the platform to the changing requirements. This task is carried out by the low-overhead READEX Run-time Library (RRL). For the already seen scenarios the optimal configuration is directly extracted from the tuning model. For the un-seen ones an RRL calibration mechanism is used to guess the optimal system configuration based on machine-learning algorithms and the data stored in the tuning model.

In order to qualify for Exascale deployments, the architecture of the RRL will rely on maximal decentralization of tuning decision making. Where un-avoidable, the RRL will rely on scalable tree-like reduction networks for the determination of global decisions. Furthermore, special attention will be given to achieve low overhead.

E. Validation Approach

The READEX project considers two different metrics for evaluation of the project success: the achieved improvement in energy-efficiency, measured in energy-to-solution, and the time and effort required to achieve this improvement compared to the manual tuning.

For evaluation and validation of the project results, READEX employs a co-design process in which the auto-tuning methodology and the tool-suite are developed in parallel with the manual tuning of selected applications and computational libraries.

The evaluation of the improvements in energy-efficiency will be done on a system that has been installed at Technische Universität Dresden in the first quarter of 2015. The system will be equipped with more than 1400 power-instrumented nodes that allow for scalable and accurate energy measurements with a fine spatial and temporal granularity (CPU, memory, and whole node with up to 1000 Samples/s) [15].

We will employ a wide variety of target applications to validate our approach, e.g., the OpenFOAM CFD solver [16] as well as benchmarks from the CORAL benchmark suite [17]. By choosing both industry-grade HPC applications and scalable benchmarks, we aim at maximising the potential impact of the READEX project.

V. Conclusion

Energy and extreme parallelism are the major challenges on the road to exascale computing. The European Union Horizon 2020 READEX project will address these by providing application developers with a tools-aided methodology for automatic tuning of applications at run-time with respect to dynamically changing resource requirements. The aim is to significantly improve energy-efficiency and performance by better exploiting the resources available to the application while reducing the programming effort through the automation.

In achieving its ambitious goals, the project builds on two proven technologies, the static auto-tuning and the system scenarios methodology, to develop the first of its kind generic dynamic auto-tuner. The envisioned methodology encompasses the design-time and run-time parts of the application life-cycle. In the first part the extensive knowledge about application dynamism and optimal system configurations is derived, which is then used to perform low-overhead and scalable dynamic tuning at run-time.

Acknowledgment

The research leading to these results has received funding from the European Union's Horizon 2020 Programme under grant agreement number 671657.

References

[1] Run-time Exploitation of Application Dynamism for Energy-efficient Exascale computing (READEX), last accessed August 10, 2015. [Online]. Available: http://www.readex.eu

[2] E. César, A. Moreno, J. Sorribes, and E. Luque, "Modeling master/worker applications for automatic performance tuning," *Parallel Computing*, vol. 32, no. 7, pp. 568–589, 2006.

[3] A. Tiwari, C. Chen, J. Chame, M. Hall, and J. K. Hollingsworth, "A scalable auto-tuning framework for compiler optimization," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on.* IEEE, 2009, pp. 1–12.

[4] ENabling technologies for a programmable many-CORE (ENCORE), last accessed August 10, 2015. [Online]. Available: http://www.encore-project.eu/

[5] The OmpSs Programming Model, last accessed August 10, 2015. [Online]. Available: https://pm.bsc.es/ompss

[6] S. Benkner, S. Pllana, J. L. Träf, P. Tsigas, U. Dolinsky, C. Augonnet, B. Bachmayer, C. Kessler, D. Moloney, and V. Osipov, "Peppher: Efficient and productive usage of hybrid computing systems," *IEEE Micro*, vol. 31, no. 5, pp. 28–41, 2011.

[7] I. Filippopoulos, F. Catthoor, and P. G. Kjeldsberg, "Exploration of energy efficient memory organisations for dynamic multimedia applications using system scenarios," *Design Automation for Embedded Systems*, vol. 17, no. 3-4, pp. 669–692, 2013.

[8] S. V. Gheorghita, M. Palkovic, J. Hamers, A. Vandecappelle, S. Mamagkakis, T. Basten, L. Eeckhout, H. Corporaal, F. Catthoor, and F. Vandeputte, "System-scenario-based design of dynamic embedded systems," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 1, pp. 3:1–3:45, 2009.

[9] Z. Ma, P. Marchal, D. P. Scarpazza, P. Yang, C. Wong, J. I. Gómez, S. Himpe, C. Ykman-Couvreur, and F. Catthoor, *Systematic methodology for real-time cost-effective mapping of dynamic concurrent task-based systems on heterogenous platforms*. Springer Science & Business Media, 2007.

[10] S. Benkner, F. Franchetti, H. M. Gerndt, and J. K. Hollingsworth, "Automatic Application Tuning for HPC Architectures (Dagstuhl Seminar 13401)," *Dagstuhl Reports*, vol. 3, no. 9, pp. 214–244, 2014. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2014/4423

[11] R. Miceli, G. Civario, A. Sikora, E. César, M. Gerndt, H. Haitof, C. Navarrete, S. Benkner, M. Sandrieser, L. Morin *et al.*, "Autotune: A plugin-driven approach to the automatic tuning of parallel applications," in *Applied Parallel and Scientific Computing*. Springer, 2013, pp. 328–342.

[12] "Automatic Online Tuning (AutoTune)," last accessed August 10, 2015. [Online]. Available: http://www.autotune-project.eu/

[13] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony *et al.*, "Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir," in *Tools for High Performance Computing 2011*. Springer, 2012, pp. 79–91.

[14] V. Petkov, M. Gerndt, and M. Firbach, "Pathway: Performance analysis and tuning using workflows," in *IEEE 10th International Conference on High Performance Computing and Communications (HPCC)*. IEEE, 2013, pp. 792–799.

[15] HDEEM: HIGH DEFINITION ENERGY EFFICIENCY MONITORING, last accessed August 10, 2015. [Online]. Available: https://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/hdeem

[16] H. Jasak, A. Jemcov, and Z. Tukovic, "OpenFOAM: A C++ library for complex physics simulations," in *International workshop on coupled methods in numerical dynamics*, vol. 1000, 2007, pp. 1–20.

[17] "Coral benchmarks," last accessed August 10, 2015. [Online]. Available: https://asc.llnl.gov/CORAL-benchmarks/