

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

**Performance modeling for exascale autotuning:
An integrated approach***

Prasanna Balaprakash, Stefan M. Wild, and Paul D. Hovland

Mathematics and Computer Science Division

Preprint ANL/MCS-P5000-0813

July 2013

*Support for this work was provided through the SciDAC program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract No. DE-AC02-06CH11357.

Performance modeling for exascale autotuning: An integrated approach

Prasanna Balaprakash*, Stefan M. Wild, and Paul D. Hovland

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

The usual suspects—shrinking integrated circuit feature sizes, heterogeneous nodes with many-core processors, deep memory hierarchies, an ever-present power wall, energy efficiency demands, and resiliency concerns—make exascale application and system co-design a daunting, complex task. Providing effective model-driven prediction and optimization capabilities at runtime and a software stack that includes model-informed autotuning are key to mitigating this complexity. We define autotuning for application-system co-design as a systematic process of navigating the space defined by *both software and hardware parameters* that affect the performance metrics of the application and the system.

Autotuning should orchestrate hardware- and software-provided knobs to reduce execution time, power draw, energy consumption, and other constituent features such as memory footprints. Current autotuning approaches, however, are unlikely to be successful for application-system co-design at exascale: the number of parameters exposed at the hardware and software levels will be large, drastically increasing the decision space; rigorous approaches to optimizing multiple conflicting objectives simultaneously are absent; and there is a lack of multiple-metric performance models. Significant research is required to develop an *integrated modeling, machine learning, and search approach* in order to provide model-driven prediction and optimization capabilities at runtime.

Automatic performance modeling

Given the challenges projected at the exascale, modeling only the execution time of the application is insufficient. Execution time will be one among several, possibly conflicting, system-related metrics (such as energy consumption and system resiliency) that need to be modeled. Therefore, the co-design problem must be tackled as a multiobjective optimization problem, where a search algorithm would optimize (and/or be constrained by) multiple metrics simultaneously. Ideally, analytical performance models must be developed to quantify meaningful differences across the decision space and provide error bounds/distributions associated with their predictions. Models for a variety of metrics should offer a convenient mechanism for exposing sweet spots in the decision space, adaptive pruning of the decision space in online and offline algorithms as empirical information becomes available, and a variety of other search-related tasks.

Automating performance modeling is crucial for application-system co-design. Analytical modeling should be supported by effective static analysis tools that can automatically extract the set of all code-specific characteristics (for example, total instructions, arithmetic intensity, branching instructions, and memory requirements) with respect to the application configuring parameters. The performance models should use algebraic expressions that take code-specific characteristics and the system level parameter settings as input to predict the performance metric(s). We should develop a rich mathematical modeling framework for flexible performance modeling because different metrics might require different mathematical functions. For example, one might expect to encounter polynomial models for memory requirements and execution time, but piecewise models for resilience metrics due to varying demands across code regions. The framework should allow the modelers to test various algebraic combinations such as generalized linear, nonlinear, piecewise, and spline models. It should also allow systematic combination of two or more performance models to obtain other metrics of interest that cannot be modeled directly, such as an energy model from runtime and power models or a bandwidth model from a cache miss model. We should provide a seamless

*Corresponding author, pbalapra@mcs.anl.gov

integration of the modeling framework with the simulator and/or system for model validation and calibration during the application-system co-design phase.

Machine learning for dynamic and predictive empirical modeling

Given the complexity of exascale application-system co-design, analytical performance models based on closed-form mathematical expressions may not be adequately informative for all metrics of interest. When analytical performance models become too restrictive—in particular, when dynamic changes must be taken into account—empirical performance modeling can bridge the gap. In this approach, a small set of co-design configurations is evaluated to measure the required performance metrics, and a predictive model is built by using statistical/machine learning approaches.

Machine learning typically addresses settings where one has a relatively large number of design point evaluations and the goal is to obtain a best model. This approach is well suited to so-called *big data* applications, where the goal is to extract high-quality information from an abundance of data. However, it is a poor fit for empirical modeling at exascale because the high computational expense of design point evaluations results in a relative paucity of data. In this setting, we should focus on developing online learning algorithms for *small data*, which must be informed by available models. A promising approach to tackle this problem is through a Bayesian statistical modeling framework. A strength of this framework lies in the uncertainty quantification of the model predictions. We can exploit this property to develop an effective design-of-experiments strategy to evaluate the configurations that improve the model prediction accuracy in decision-space regions of interest. Moreover, Bayesian learning allows one to develop models for multiple performance metrics with a single methodology. We also emphasize the need for rigorous subsidiary procedures that infuse application- and architecture-specific knowledge to the statistical models.

Modeling the autotuning search space

Developing search algorithms for multiobjective autotuning is a challenging task. Autotuning can be formulated as a noisy, mixed-integer, nonlinear mathematical optimization problem over a decision space comprising continuous, discrete, and categorical parameters. We must move beyond “black-box” optimization in order to make these search problems tractable. Application- and architecture-specific knowledge captured in the form of models should be exploited early, often, and automatically (for example, parameters with powers of two or a multiple of cache line size). In addition, by treating metrics, such as those based on power and energy, as objectives rather than solely as constraints, one can obtain a hierarchy of solutions and quantify the sensitivities associated with changing constraint bounds. Furthermore, optimal solution sets obtained offline can significantly simplify (or even trivialize) online optimization at runtime, when quantities such as the price of electricity, resource availability, and system state/health are known.

An integrated approach to autotuning

We believe that a systematic integration of analytical and statistical performance models with search algorithms is key to automating application-system co-design. First, we have to develop analytical models that capture the relationship among the design parameters, application inputs, and performance metrics. These models should inform statistical performance models in cases where analytical models fall short or are infeasible. At runtime, the performance models can be further refined and updated to take into account dynamic changes in the system and errors in the models. A search algorithm should use the models and the resulting reduced/transformed decision space to find the best parameter setting for the running state of the system with significantly reduced empirical expense. Analytical and statistical performance models can also be employed to inform and capture the correlations among the state of the system, the best co-design parameters, and the multiple metrics of interest to further improve multiobjective search tasks.

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.