Performance Portability in the Age of Extreme Heterogeneity

John Schalf *

November 12, 2020

Abstract

Moore's Law is a techno-economic model that has enabled the IT industry to double the performance and functionality of digital electronics roughly every 2 years within a fixed cost, power and area. This expectation has led to a relatively stable ecosystem (e.g. electronic design automation tools, compilers, simulators and emulators) built around general-purpose processor technologies, such as the x86, ARM and Power instruction set architectures. However, the historical improvements in performance offered by successive generations of lithography are waning while costs for new chip generations are growing rapidly.

In the near term, the most practical path to continued performance growth will be architectural specialization in the form of many different kinds of accelerators. New software implementations, and in many cases new mathematical models and algorithmic approaches, are necessary to advance the science that can be done with these specialized architecture. This trend will not only continue but also intensify as the transition from multi-core systems to hybrid systems has already caused many teams to re-factor and redesign their implementations. But the next step to systems that exploit not just one type of accelerator but a full range of heterogeneous architectures will require more fundamental and disruptive changes in algorithm and software approaches. This applies to the broad range of algorithms used in simulation, data analysis and learning. New programming models or low-level software constructs that hide the details of the architecture from the implementation can make future programming less time-consuming, but they will not eliminate nor in many cases even mitigate the need to redesign algorithms. Future software development will not be tractable if a completely different code base is required for each different variant of a specialized system.

The aspirational desire for "minimizing the number of lines of code that must be changed to migrate to different systems with different arrangements of specialization" is encapsulated in the loaded phrase "Performance Portability." However, performance portability is likely not an achievable goal if we attempt to do it using imperative languages like

*Lawrence Berkeley National Laboratory

Fortran and C/C++. There is simply not enough flexibility built in to the specification of the algorithm for a compiler to do anything other than what the algorithm designer explicitly stated in their code. To make this future of diverse accelerators usable and accessible in the former case will require the co-design of new compiler technology and domain- specific languages (DSLs) designed around the requirements of the target computational motifs (the 13 motifs that extended Phil Colella's original Dwarfs of algorithmic methods). The higher levels of abstraction and declarative semantics offered by DSLs enable more degrees of freedom to optimally map the algorithms onto diverse hardware than traditional imperative languages that over-prescribe the solution. Because this will drastically increase the complexity of the mapping problem, new mathematics for optimization will be developed, along with better performance introspection (both hardware and software mechanisms for online performance introspection) through extensions to the roofline model. Use of ML/AI technologies will be essential to enable analysis and automation of dynamic optimizations.