Viewing Application/Machine Interactions through Computational Idioms

Michael A. Laurenzano[†] Laura Carrington[†] Adam Jundt[†] Ananta Tiwari[†]
Joshua Peraza[†] William A. Ward, Jr.[*] Roy Campbell[*]

[†]EP Analytics
[*]DoD High Performance Computing Modernization Program

Abstract

Models of application behavior are one of the keys to bridging the gap between current large-scale system design practices and upcoming exascale system designs. Processor/accelerator specialization and heterogeneity have been proposed as possible paths forward for attaining the significant energy efficiency improvements necessary to achieve exascale-level computing capabilities within an acceptable power envelope. To have an impact on the exascale system design process, the models must be (1) abstract, containing information that is relevant and actionable across a wide range of programming and execution models and (2) complementary to a well-defined and standardized machine characterization methodology.

We argue that a key component of this modeling paradigm is what we term an idiom*, a small computational or memory access pattern. We hypothesize that much of the computational work within HPC can be expressed as the combination of a reasonably small number of basis idioms. Understanding application composition and machine characteristics in terms of how they behave in the presence of (combinations of) this small number of idioms allows us to bridge the gap between large workloads and an increasingly diverse and complex landscape of hardware options.*

1. Background

The difficulty in realizing exascale computing stems primarily from the requirements that it be simultaneously more energy efficient and resilient to errors. Realizing energy efficient computing is becoming increasingly more difficult as the technology trends (Dennard scaling and Moore's law) that allowed computer users (particularly the HPC community) to effortlessly experience improved performance and energy efficiency appear to be coming to an end. Without such improvements, it also becomes apparent that delivering higher computational capability to large scale computations will require an increasing number of hardware components, introducing difficulties to the previously-enjoyed notion of resilient and stable hardware at the application level.

One of the most promising areas for realizing the vast improvements in energy efficiency required for exascale computing is *hardware specialization*. Specialization is motivated by the argument that energy efficiency is best served by delivering to a computation the specific design and architectural features that give maximum performance, while avoiding wasting energy on features that have little or no benefit. The problem, however, is that different computational problems show proclivities for different designs and architectural features. Traditionally, this problem has been addressed by large scale system design and procurement strategies that seek the best average performance, perhaps with small amounts of variation from the average to account for important special cases (e.g., large memory nodes or nodes with accelerators). This limitation in the amount of tolerable variety within a single HPC system is likely to be (reluctantly) discarded as system designers strive to achieve energy-efficiency improvements across diverse workloads.

As system designers consider a set of heterogeneous and increasingly exotic hardware options, it becomes difficult to see how current modeling approaches will provide them with a set of tools that allows them to meaningfully answer important questions about how their workload will behave on each hardware platform, what tradeoffs exist between their design options, and what combination of platforms match their design goals. Current approaches either do not scale well because they involve the guidance of experts or significant levels of effort to incorporate new and non-traditional architectures [5, 10]. We argue in this paper that *idioms*, key computational and memory patterns that are common in HPC codes, can be used as a rubric through which we can view diverse sets of HPC applications and hardware platforms and may be instrumental in the successful design and realization of exascale systems.

2. The Role of Idioms

Idioms play an important role in understanding the interactions between an application and hardware. An idiom is a computational or memory access pattern that appears across a range of HPC codes. An idiom should not be confused with a dwarf [4, 2]; idioms are limited, simple statements expressed in source code while dwarfs are high level algorithms representing classes of problems/algorithms. Some examples of idioms include matrix-matrix multiply, stream, transpose, reduction, stencil, gather and scatter. Idioms are small and concise expressions, often capable of being expressed in only a few lines of source code. The limited size and scope of an

idiom suits it well for being the unit that is analyzed, ported, and run on specialized hardware. Indeed, prior work shows that idioms can be used to effectively map certain parts of HPC applications into existing co-processor technologies like GPGPUs [3], FPGAs [7], and the Intel Xeon Phi [9]. The hypotheses motivating this paper are:

- An application can largely be described in terms of its constituent idioms.
- A covering set of idioms which can be used to describe much of the computational work done in HPC is reasonably small.

We envision that the adoption of the idiom-centric approach to analysis and design favors an environment in which heavily optimized library implementations akin to LAPACK [1] are provided by hardware vendors or researchers. We note that looking just at the idiom composition of a given application does not provide the entire story for a particular application/machine interaction. Other properties, such as the idiom's working set size, how an idiom interacts with other nearby idioms, and how idioms synchronize/communicate across computing elements, play important roles as well. To systematically consider these relevant properties, our idiom-centric approach adopts a three-step process known as the convolution method [10] in which information about an application (called an *application signature*) is convolved with information about a machine (called a *machine profile*) in order to produce a model of the behavior of that application on that machine.

[Machine Profiles] The first step is to fully characterize the set of potential hardware platforms, not in terms theoretical specifications offered by the vendor, but by the actual performance and power behavior for a series of computations that are relevant to the idioms that make up HPC codes. We refer to this set of computations as *idiom-aware benchmarks*. This set of idiom-aware benchmarks should be used to probe the hardware to develop a detailed picture of how it performs in the presence of each particular computation. We view parameterized benchmarking [6], in which the benchmark exposes a series of tunable knobs, to be a plausible methodology for approaching this problem.

A set of idiom benchmark results aggregated across hardware platforms can be used to highlight the set of tradeoffs that might be made for that computation across those platforms, while the aggregation of idiom benchmark results taken for a single hardware platform can be used in tandem with a machine profile to develop a detailed model of that platform.

[Application Signatures] The second step is to isolate the set of application characterizing idioms, as is shown in [8]. Identifying the idiom coverage of an application can be used not only in understanding the advantages of various proposed hardware but also in understanding the computational similarities among applications. While from an algorithmic view two applications might seem quite different, from a hardware view the applications may appear quite similar for some computational phases. It is these similarities that aid in the co-design of an exascale system for a diverse workload.

[Convolution] The final stage is in mapping the application's idiom characterization to the hardware to determine which specialized hardware would benefit the workload in a co-design process. The mapping is enabled through a combination of static and dynamic characterization processes. The static component involves combing the source code of large HPC applications for computational idioms automatically [8], while the dynamic component exposes other important features of behavior (e.g. cache hit rate, data locality, memory operations, floating-point operations, dependencies) that occur during the computation. The combination of these static and dynamic components yields a set of features that describes the application, a feature set mirrored by tunable knobs within the idiom-aware benchmarks. By implementing tunable knobs in the idiom-aware benchmarks that vary the important features above, and then tuning the knobs to characterize the application in terms of the benchmarks, it is possible to provide a mapping of the application to the hardware's features.

Many challenges lay ahead on the path to exascale computing. A major impediment to realizing exascale is the lack of a practical framework for models that accurately describes the array of application/machine interactions that must be well understood before making system design decisions. In this paper we have proposed using computational idioms as an approach to realizing this framework.

References

[1] E. Anderson, *LAPACK Users' guide*. Siam, 1999, vol. 9.

[2] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams *et al.*, "The landscape of parallel computing research: A view from berkeley," Tech. Rep., 2006.

[3] L. Carrington, M. M. Tikir, C. Olschanowsky, M. Laurenzano, J. Peraza, A. Snavely, and S. Poole, "An idiom-finding tool for increasing productivity of accelerators," in *Proceedings of the international conference on Supercomputing*. ACM, 2011, pp. 202–212.

[4] P. Colella, "Defining software requirements for scientific computing," 2004.

[5] D. J. Kerbyson, H. J. Alme, A. Hoisie, F. Petrini, H. J. Wasserman, and M. Gittings, "Predictive performance and scalability modeling of a large-scale application," in *Proceedings of the 2001 ACM/IEEE conference on Supercomputing (CDROM)*. ACM, 2001, pp. 37–37.

[6] M. A. Laurenzano, M. Meswani, L. Carrington, A. Snavely, M. M. Tikir, and S. Poole, "Reducing energy usage with memory and computation-aware dynamic frequency scaling," in *Euro-Par 2011 Parallel Processing*. Springer, 2011, pp. 79–90.

[7] M. R. Meswani, L. Carrington, D. Unat, A. Snavely, S. Baden, and S. Poole, "Modeling and predicting application performance on hardware accelerators," in *Workload Characterization (IISWC), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 73–73.

[8] C. Olschanowsky, A. Snavely, M. R. Meswani, and L. Carrington, "Pir: Pmac's idiom recognizer," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*. IEEE, 2010, pp. 189–196.

[9] J. Peraza, A. Tiwari, M. A. Laurenzano, L. Carrington, W. A. Ward, and R. Campbell, "A methodology for understanding performance of stencil computations on intel's xeon phi," in *(in submission)*.

[10] A. Snavely, L. Carrington, N. Wolter, J. Labarta, R. Badia, and A. Purkayastha, "A framework for performance modeling and prediction," in *Supercomputing, ACM/IEEE 2002 Conference*. IEEE, 2002, pp. 21–21.