Exascale Workload Characterization and Architecture Implications

Prasanna Balaprakash*, Darius Buntinas*, Anthony Chan*, Apala Guha†,
Rinku Gupta*, Sri Hari Krishna Narayanan*, Andrew A. Chien*, Paul Hovland*, and Boyana Norris*
*Argonne National Laboratory, Mathematics and Computer Science Division
†Department of Computer Science, University of Chicago

I. Introduction

Emerging exascale architectures bring forth new challenges related to heterogeneous systems power, energy, cost, and resilience. These new challenges require a shift from conventional paradigms in understanding how to best exploit and optimize these features and limitations.

Our objective is to identify the top few dominant characteristics in a set of applications. Understanding these characteristics will allow the community to build and exploit customized architectures and tools best suited to optimize each dominant characteristic in the application domain. Every application will typically be composed of multiple characteristics and thus will use several of the customized accelerators and tools during its execution phases, with the eventual goal of using the entire system efficiently.

In this poster, we describe a hybrid methodology, based on binary instrumentation, for characterizing scientific applications such as instruction mix and memory access patterns. We apply our methodology to proxy applications that are representative of a broad range of DOE scientific applications. With this empirical basis, we develop and validate statistical models that extrapolate application properties as a function of problem size. These models are then used to project the first quantitative characterization of an exascale computing workload, including computing and memory requirements. We evaluate the potential benefit of processor under memory, a radical new exascale architecture customization and understand how these new customization can impact applications.

II. Research Focus

Our approach [1], [2] addresses several questions and challenges. Prominent among them are: (a) understanding what characteristics may be termed as dominant for various applications, and understanding their trend based on input size, scalability, and architecture heterogeneity at the current scale; (b) understanding and building tools for characterizing applications at the exascale level; (c) understanding emerging accelerators, tools, and technologies that will eventually be dominant at exascale, and building tools for mapping the dominant characteristics to these technologies; and (d) researching the right balance of general and customized architecture, keeping in mind energy efficiency and performance optimization.

III. Research Results

A major challenge in understanding application characterization is that no single analysis tool provides a complete characterization of applications. We adopt a methodology that employs multiple tools such as PIN, PBound, and HPCToolkit, to build a detailed picture of application behavior. These tools focus on source code analysis, binary instrumentation and hardware performance counter analysis to capture application characteristics. We then apply our methodology to a collection of proxy applications representative of scientific workloads. Specifically, we focus on understanding the Mantevo [3] suite (miniFE, miniMD, and HPCCG), Nek5000-based applications [4] (eddy, vortex and turbChannel), and the PETSc [5] suite of applications.

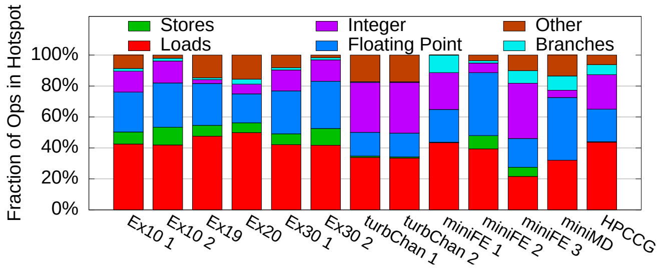

Fig. 1. Hotspots for various applications

We focus on three attributes: basic application properties (e.g., compute operations, memory operations); number, identity, and importance of performance-critical regions (hotspots); and variation of basic application properties across the workload (e.g., memory bandwidth requirements). All these attributes are characterized as a function of application input (dataset) size. We use our tools for this characterization and compare the results across the tools in order to demonstrate close alignment of results.

As an example of our characterization research, Figure 1 shows the top hotspots in our application codes and their instruction mix. Hotspots allow us to focus on particular section of codes for reduced complexity.

Using the characterization data, we create statistical models for extrapolating key characteristics of proxy applications. The models can be used to explore the impact of machine constraints on feasible application configurations and, conversely, the impact of application configurations on appropriate architecture choices. While details of the models are beyond the scope of this abstract, we note that our model follows a classical methodology in which we first use linear models with first- and second-order interactions and iteratively include quadratic and cubic terms in the models if required. To evaluate

TABLE I. MEMORY SIZE PROJECTION MODEL FOR EXASCALE WORKLOADS

Appl.	Exascale Projection Models, where $N = n_1 \cdot n_2 \cdot n_3$, and c_1, c_2, c_3, and c_4 are constants
Ex19,Ex30	$f(n_1, n_2, n_3) = c_1 + c_2 \cdot N + c_3 \cdot (n_1 \cdot n_2) + c_4 \cdot n_1$
Ex20	$f(n_1, n_2, n_3) = c_1 + c_2 \cdot (n_1 \cdot n_2) + c_3 \cdot (n_1 \cdot n_2)^2$
miniFE,miniMD HPCCG	$f(n_1, n_2, n_3) = c_1 + c_2 \cdot N$

TABLE II. SCALING LIMITS - RUNTIME (EXAOP) AND MEMORY (100 PB)

Appl	Exascale Limit		Exascale Limit		Crit. Limit	Size
	24 hrs	100 PB	Time	MemCap		
miniMD	41G	600G	**27 hrs**	108 PB	Compute	41G
Ex20	92G	500G	**25 hrs**	98 PB	Compute	92G
Ex30	130G	1500G	**27 hrs**	110 PB	Compute	130G
Ex19	5000G	1000G	23 hrs	**125 PB**	MemCap	1000G
miniFE	5000G	250G	23 hrs	**110 PB**	MemCap	250G
HPCCG	5000G	250G	23 hrs	**110 PB**	MemCap	250G

model accuracy, we adopt the leave-one-out cross-validation technique: Given a set s of input sizes $\{$A, B, C, D, E, F, G$\}$, a single input size from s is used as the validation point, and the remaining as the training points. This is repeated such that each input size in s is used once as the validation data.

As an example, Table I summarizes our memory intensity projection model (n_1, n_2, and n_3 refer to input parameters for defining application size). The memory projection model is based on the memory usage of each application (varied by its problem size) and on the impact of the number of instructions, loads, stores, branches, and floating-point operations that make up the application instruction mix.

Using our models, we project the feasible application sizes on an exascale system based on realistic runtimes (24 hours at exaflop sustained rate) and memory capacity (100 PB). Because of different scaling properties, the applications can be subdivided as shown in Table II into two groups: three (miniMD, Ex20, and Ex30) compute-limited and three (Ex19, minFE, HPCCG) memory-capacity-limited applications. The memory-capacity constraints are extreme for miniFE and HPCCG, since the 100 PB constraint requires approximately 30 exaops, or around thirty seconds if high speedups can be achieved. Ex19 is also memory-capacity limited but at a run size approximately 100 times larger than miniFE and HPCCG, or less than one hour at high speedups.

Given the feasible dataset sizes for each scientific application, we estimate the memory bandwidth requirements and then assess the potential benefit of radical proposed changes such as processor under memory (PUM). The PUM organization consists of close physical connection from the processing unit to a stacked set of memory dies, which can achieve a tenfold greater memory bandwidth (10 TB/s, compared with the typical 1 TB/s found in current architectures).

For each application, we use two models to estimate memory-bandwidth requirements for the exascale workload. One is an application-level model that projects the overall memory bandwidth for the entire application; another is a node-level model that projects the node-level memory-bandwidth needs by dividing the exascale application data size by the number of nodes in the exascale machine. We hypothesize that the node-level estimates are conservative and provide a tighter bound on the memory bandwidth. Based on these two projection models, we estimate the ability of the PUM enhancements to improve performance. The results in Table III show that for compute-limited applications (Ex20, Ex30, and miniMD), adding PUM gives no benefit: there is no runtime reduction. For the memory-capacity limited applications, however, there is a significant runtime reduction, showing a promise of 2.97x to 10x improvement.

App	Scaling Limit Exascale	PUM Improvement		Key Limit PUM	Program -ming Change
		App-level	Node-level		
Ex19	MemCap	2.35	2.97	MemCap	Local
Ex20	Compute	4.02	1.00	Compute	Local
Ex30	Compute	4.08	1.00	Compute	High
miniMD	Compute	6.75	1.00	Compute	Local
miniFE	MemCap	6.57	6.56	MemCap	Moderate
HPCCG	MemCap	10.00	10.00	MemCap	Local

TABLE III. PERFORMANCE LIMITS AND POTENTIAL IMPROVEMENT BASED ON APPLICATION- AND NODE-LEVEL SCALING

IV. FUTURE WORK

We will focus on memory-access patterns of various scientific applications and will study the impact of memory optimization. We will develop rigorous statistical and machine learning models of the exascale workloads for exploring the architecture design space. We will continue researching emerging architectures and their impact on dominant characteristics of scientific applications.

Acknowledgments

This work was supported by the U.S. Department of Energy Office of Science DE-AC02-06CH11357 and NSF OCI-1-57921.

REFERENCES

[1] M. Gahagan, A. Snavely, and A.Chien, "10x10 a General-purpose Architectural Approach to Heterogeneity and Energy-efficiency," in *Proceedings of the International Conference on Computational Science (ICCS 201)*, 2011.

[2] A. Guha and A. Chien, "The 10x10 foundation for heterogeneity," UChicago, CS TR 2012-01, 1 2012.

[3] M. A. Heroux, D. W. Doerer, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, and R. W. Numrich, "Improving performance via mini-applications," Sandia National Laboratories, Tech. Rep. SAND2009-5574, Sept. 2009.

[4] H. M. Tufo and P. F. Fischer, "Terascale spectral element algorithms and implementations," in *ACM/IEEE conference on Supercomputing*, Portland, OR, 1999.

[5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, Eds. Birkhäuser Press, 1997, pp. 163–202.