Raexplore: Enabling Rapid, Automated Architecture Exploration for Full Applications

Yao Zhang, Prasanna Balaprakash, Jiayuan Meng*, Vitali Morozov, Scott Parker, and Kalyan Kumaran
Argonne National Laboratory, Argonne, IL
{yaozhang, pbalapra, jmeng, morozov, sparker, kumaran}@anl.gov

I. Introduction

Over 20 years ago, supercomputer pioneer Seymour Cray famously made an analogy on computer design "If you were plowing a field, which would you rather use: two strong oxen or 1024 chickens?" Today, we see both types of computers in the marketplace, and computer architects face an even wider spectrum of design choices on core complexity, memory hierarchies, parallelism, and special-purpose accelerators. Furthermore, the processor design landscape is becoming increasingly more dynamic, as we see mainstream processors both trickling up (e.g., ARM, DSP, GPU) and trickling down (e.g., Atom, Xeon Phi) in their design space to meet the demands for a range of emerging applications in scientific computing, data analytics, gaming, wearable devices, human computer interactions, etc.

For a big-picture view of this background, Figure 1 sketches today's processor landscape and macro trends in terms of single-thread and throughput performance. In Figure 2, we select eight representative processors and position them in a multi-dimensional design space in terms of their architectural features (dimensions). The main observation is that the design space is vast in terms of both high dimensionality and large dynamic range for each dimension. We list eight major architectural features ranging from core complexity to memory hierarchies, not to mention other relatively minor features such as branch prediction, prefetching, and memory management. We use the ratio between the highest and lowest value to measure the span of the dynamic range of each feature. The observed span ranges from $4\times$ (issue width) up to $78\times$ (last level cache per core).

Fundamentally, architecture design is driven by applications. Architectural evaluation and comparison for a diverse set of current processors are challenging because they often require significant human efforts to port applications to different architectures [1]. Studying the performance of future processors is even more challenging, as the hardware is not yet available. While commonly used simulation-based techniques could provide highly accurate results, they are prohibitively slow to handle the combinatorial explosion of design choices in a multi-dimensional space and thus often limited to studying kernels and benchmarks rather than larger programs, mini-apps, and full applications [2]–[4].

In this work, we aim to address this architecture design challenge by developing Raexplore (**R**apid **a**rchitecture **explore**, pronounced as *ray-xplore*), a performance modeling framework to reduce the needs for application porting in architectural comparison, and to serve as a fast, first-order

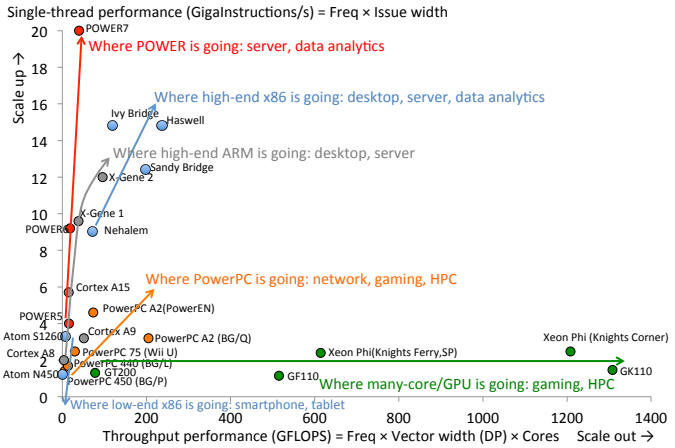

Fig. 1: Processor landscape and trends. Processors in the same family/catergory are color coded.

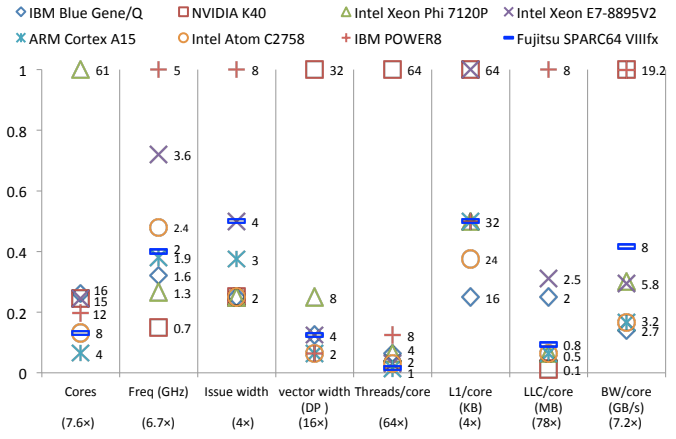

Fig. 2: Mutli-dimensional processor design space. For each architectural feature, the numbers are normalized between 0 and 1 based on the highest value. The numbers to the right of each marker are absolute values. The numbers in parentheses below the horizontal axis are the ratios between the highest and lowest value, which measure the span of the dynamic range of each feature (dimension) for the eight processors.

* Now affiliated with Google Inc.

architecture explorer to complement slower but more accurate simulation-based techniques. In particular, we develop a methodology that combines experimental performance characterization and analytical performance modeling to enable rapid and systematic architecture exploration. We also develop analytical models for two recent manycore processors IBM Blue Gene/Q compute chip and Intel Xeon Phi. We show that our models could capture complex interactions between architectural components including instruction pipeline, cache, and memory, and achieve a 3–22% error for same-architecture and cross-architecture performance predictions. Lastly, using our framework, we analyze processor performance for a set of scientific applications, and suggest and evaluate a list of architectural design choices.

II. Methodology

We have three goals in mind in developing a performance modeling methodology: (1) handle real-world large programs/applications, instead of kernels or benchmarks, as it is not common in practice that a single kernel dominates the application runtime; (2) explore architecture design space rapidly; and (3) capture complex effects and interactions of major architectural features and predict performance accurately. To this end, we develop a methodology that combines experimental hardware counter-based performance characterization and analytical performance modeling. The hardware counter-based approach provides a fast way to characterize performance for large applications on an existing baseline architecture. To project performance for future or different architectures, we develop analytical models that take in baseline performance characteristics and produce performance predictions. Analytical modeling reduces the process of architecture exploration to a matter of merely evaluating a set of mathematical formulas, enabling fast search of the vast design space. This combined experimental and analytical approach is similar to that of the work by Kerbyson [5] but with two major differences: (1) this work focuses on microarchitectural features while theirs is on system-level features such as network topology, and (2) this work uses hardware counters to automatically gather application characteristics, while theirs relies on manually built algorithmic-level application models.

Figure 3 shows the schematic of our performance modeling framework. It first takes in a set of targeted applications and characterizes their performance on a baseline architecture. The performance characteristics are a set of performance events measured by hardware counter-based tools such as PAPI, Intel VTune, or IBM HPM. The performance characteristics are then calibrated for a target architecture configuration using analytical performance models. The target could be either a future design of the baseline architecture, or a different current architecture (e.g., Blue Gene/Q as baseline and a hypothetical Blue/Gene processor as target, or Blue Gene/Q as baseline and Xeon Phi as target). Finally, the analytical models produce performance analysis for the baseline architecture and performance predictions for the future/target architecture. Note that the analytical models include both the models for performance analysis (e.g., to derive the time of instruction execution, memory access, and their overlap) and the models for performance characteristics calibration to account for differences in architecture features such as cache size and instruction latency.

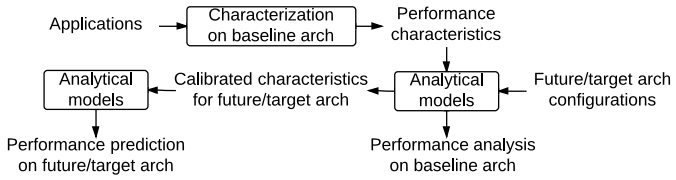

Fig. 3: Performance modeling framework.

III. Results and future work

We have validated our models for same-architecture and cross-architecture performance prediction for various code blocks in a number of scientific applications. Our preliminary results show our models are able to capture complex effects and interactions of architectural components within a 3-22% error. We also demonstrate our performance modeling framework for evaluating the design balance and exploring future scaling options of BGQ for full applications. The studied architectural features include core count, L1 and LLC size, and memory bandwidth. We find that for a future processor based on BGQ, more cores, together with correspondingly larger L1 cache and higher bandwidth, will continue to scale the performance, while the LLC size could be kept same or even shrink. In our future work, we plan to (1) study architectures different from BGQ and Xeon Phi such as GPUs, multicores, and ARM, (2) incorporate chip area and power models to enable constraint-based architecture exploration, (3) incorporate system-level models for network, I/O, and storage, and (4) introduce algorithmic and program transformation parameters to the optimization space for hardware and software co-design.

Acknowledgment

This work is supported by the U.S. Department of Energy under contract DE-AC02-06CH11357 and the LDRD X-PECT project 2013-213-NO. The research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

References

[1] J. Krueger, D. Donofrio, J. Shalf, M. Mohiyuddin, S. Williams, L. Oliker, and F.-J. Pfreund, "Hardware/Software co-design for energy-efficient seismic modeling," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 73:1–73:12.

[2] D. Sanchez and C. Kozyrakis, "ZSim: Fast and accurate microarchitectural simulation of thousand-core systems," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, 2013, pp. 475–486.

[3] A. Rico, F. Cabarcas, C. Villavieja, M. Pavlovic, A. Vega, Y. Etsion, A. Ramirez, and M. Valero, "On the simulation of large-scale architectures using multiple application abstraction levels," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 4, pp. 36:1–36:20, Jan. 2012. [Online]. Available: http://doi.acm.org/10.1145/2086696.2086715

[4] J. Shalf, D. Quinlan, and C. Janssen, "Rethinking hardware-software codesign for exascale systems," *Computer*, vol. 44, no. 11, pp. 22–30, 2011.

[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*, ser. SC '01, 2001, pp. 37–37.