Black-Box Kernel-Level Performance Modeling for GPUs

Extended Abstract and Summary

James D. Stevens
Andreas Klöckner

Department of Computer Science
University of Illinois at Urbana-Champaign

***Abstract*—We present a mechanism to symbolically gather performance-relevant operation counts from numerically-oriented subprograms ('kernels') expressed in the Loopy programming system, and apply these counts in a simple, linear model of kernel run time. We use a series of 'performance-instructive' kernels to fit the parameters of a unified model to the performance characteristics of GPU hardware from multiple hardware generations and vendors. We evaluate predictive accuracy on a broad array of computational kernels relevant to scientific computing. In terms of geometric mean, our simple, vendor- and GPU-type-independent model achieves relative accuracy comparable to that of previously published work using hardware specific models.**

1. Introduction

The ability to predict execution time of computational kernels is a key step toward the automation of performance tuning for complicated, modern, vector-based, massively parallel processor architectures. We present a simple, effective model to achieve such a prediction. The model is realized on top of a program transformation system, providing a self-contained foundational building block to aid the developer of automated tuning solutions in exploring the vast search space of possible program variants. We mainly view our model as a more economical alternative to evaluating execution time than, for example, using on-device timing runs. Our system primarily targets the execution paradigm of modern GPU Hardware, but makes no assumptions about the internal hardware organization, and device-specific parameters are obtained through a black-box adaptation process that runs only once on each new piece of hardware that uses the system.

Much of the previous work in GPU performance modeling has focused on constructing analytical models of instruction-level execution based on detailed hardware knowledge and instruction analysis for a single architecture. Many of these models predict well for their specific target architecture. For example, Hong and Kim [2009] present an analytical performance model for Nvidia GPU architectures that estimates memory-level and thread-level parallelism. This model achieves a geometric mean prediction error of 13.3% on the MERGE [Linderman et al., 2008] benchmarks on four Tesla generation Nvidia GPUs. It makes extensive use of hardware performance characteristics, such as timing delays between memory transactions, DRAM access latency, and instruction execution cycles, and requires an analysis of PTX assembly instructions. Baghsorkhi et al. [2010] also use deep analytical knowledge of a (single) GPU, and, unlike Hong and Kim, model branch divergence, bank conflicts, and SIMD pipeline delays. From the perspective of optimization selection, Cavazos et al. [2006] present a probabilistic predictor of transformation selection using a non-analytical, black-box model based on an artificial neural network. Joseph et al. [2006] use techniques from machine learning to identify piecewise nonlinearities in cost metrics. Other approaches emphasize the performance of single subsystems, such as branch prediction [Emer et al., 2002].

Our work differs from previous work in five ways:

- We completely automate the gathering of all performance-relevant kernel properties.
- We model execution time without explicit representation of any hardware characteristics or behavior.
- Our model is hardware vendor- and generation-independent, and we demonstrate performance on an AMD GPU and three generations of Nvidia GPUs.
- Our model is simple and amenable to analysis: the exposed weights have known meanings, allowing reasoning about how specific operations contribute to execution time.
- Prediction evaluation is rapid and simple: Obtaining a cost estimate requires computing a small inner product involving precomputed symbolic expressions dependent on problem size parameters.

1.1. Impact on Supercomputing

In the shifting landscape of large- and extreme-scale computing, where the machine scale is often determined

by power and cooling constraints, individual nodes need to carry increasingly heavy burdens and, as a result, contain increasingly complex parallel computing architectures. Key to leveraging this within-node parallelism is the ability to predict its performance on a given workload, for needs such as load balancing, job scheduling, performance optimization, machine design and qualification, and benchmarking. For array-based workloads, as encountered frequently in scientific computing, these needs are met directly by the modeling machinery presented in this contribution.

2. Model Overview

We model execution time as a linear combination of kernel properties chosen a priori to ensure that they contribute linearly to overall execution time, i.e.,

$$T_{\text{wall}}(\mathbf{n}) \approx \sum_{i=1}^{N_{\text{properties}}} \alpha_i p_i(\mathbf{n}),$$

where α_i is a machine-dependent weighting coefficient for the ith contributing cost component and property $p_i(\mathbf{n})$ is a kernel-dependent symbolic expression that, based on kernel parameters $\mathbf{n}$, accounts for the number of units of cost α_i incurred.

3. Conclusions

We present an alternative to previous GPU performance models that can be easily fitted to new hardware and allows rapid, runtime performance prediction. This speed and versatility requires minor if any sacrifices in prediction accuracy compared to models in the literature. To our knowledge, this is the first GPU performance model that collects all performance-relevant information automatically and utilizes no explicit knowledge of hardware characteristics.

References

S. S. Baghsorkhi, M. Delahaye, S. J. Patel, W. D. Gropp, and W.-m. W. Hwu. An Adaptive Performance Modeling Tool for GPU Architectures. In *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '10, pages 105–114, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-877-3. doi: 10.1145/1693453.1693470.

J. Cavazos, C. Dubach, F. Agakov, E. Bonilla, M. F. O'Boyle, G. Fursin, and O. Temam. Automatic performance model construction for the fast software exploration of new hardware designs. In *Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems*, pages 24–34. ACM, 2006.

J. Emer, P. Ahuja, E. Borch, A. Klauser, C.-K. Luk, S. Manne, S. S. Mukherjee, H. Patil, S. Wallace, N. Binkert, and others. Asim: A performance model framework. *Computer*, 35(2):68–76, 2002.

S. Hong and H. Kim. An Analytical Model for a GPU Architecture with Memory-level and Thread-level Parallelism Awareness. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 152–163, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-526-0. doi: 10.1145/1555754.1555775.

P. J. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. A predictive performance model for superscalar processors. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 161–170. IEEE Computer Society, 2006.

M. D. Linderman, J. D. Collins, H. Wang, and T. H. Meng. Merge: A Programming Model for Heterogeneous Multi-core Systems. In *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIII, pages 287–296, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-958-6. doi: 10.1145/1346281.1346318.