RETROSPECTIVE: An Analytical Model for a GPU Architecture with Memory-Level and Thread-Level Parallelism Awareness

Sunpyo Hong
Technology Pathfinding and Innovation
Intel Corporation
sunpyo.hong@intel.com

Hyesoon Kim
School of Computer Science
Georgia Institute of Technology
hyesoon.kim@cc.gatech.edu

I. Motivation

It is an exceptional privilege to have the opportunity to write a retrospective for ISCA, reflecting on the significance of our publication in ISCA 2009, which marked the first major contribution from HpArch Lab. At that time, Sunpyo Hong was a junior Ph.D. student and Hyesoon Kim was a new assistant professor at Georgia Tech.

The year 2007 witnessed the emergence of GPU/CUDA technology, which garnered significant attention within the computer architecture community. Notably, the CUDA and GPU architecture lecture series by Wen-mei Hwu and David Kirk from UIUC ECE 498AL served as a primary educational resource on GPU architecture. Subsequently, several universities began offering specialized courses on GPUs to fuel GPU architecture research.

Our publication was a natural extension (and in some sense a by-product) of our collaboration on the Qilin paper [4], which was started in 2007 and published later than this ISCA paper. Our joint efforts with CK Luk focused on developing auto-work distribution techniques for GPUs and CPUs, necessitating a comprehensive understanding of CUDA's performance characteristics on GPUs. During this time, Ryoo et al.'s [5] inspired us to write a deeper GPU architecture performance analysis paper.

The primary motivation behind our paper was to educate the performance advantages and behaviors of GPUs. Specifically, we aimed to debunk the common wisdom during that time of optimizing for occupancy yielded optimal results as shown in Figures 3 and 4 of the ISCA paper, illustrating the occupancy-performance trade-off. We sought to shed light on how GPUs truly performed and provided visualization of GPU performance behavior.

The challenge we faced was how to construct a high-level GPU performance model that balanced architecture details and simplifications to be useful for programmers and architecture researchers. Looking back after 14 years, the high-level modeling of GPU architecture was relatively less complex compared to contemporary GPUs. Unlike modern designs, early GPU architectures lacked global memory caches. We realized that the major performance impact factor was memory cycles and their potential to overlap across multiple GPU warps. By categorizing scenarios of memory and computation overlap and predicting high-level performance, we could identify the key contributors to GPU performance and model them accordingly. And knowing what contributes to the performance and modeling them was the paper's contribution. Because the GPU architectures had so drastically different architecture compared to CPUs and since this work had fresh contents such as the calculation of memory warps overlapping along with the computation warps, we believe that this is why the work appealed to academic researchers thereafter. Thanks to the reviewers of this work who valued this new research attempt and later with the support of John Nickolls at NVidia, we are honored to have this opportunity to write a retrospective.

II. What the paper showed

The paper's significance lies in its ability to establish a connection between CUDA programs and their performance implications. Programmers must consider the performance impact when writing code. Techniques such as reducing program lines, minimizing memory footprints/array sizes, and increasing parallelism through the use of parallel-for constructs are commonly employed for program optimizations. These techniques are relatively easy to visualize. However, optimizing for GPUs introduces the need to consider hardware utilization, which was a novel concept at the time. In retrospect, the distinction between optimization for latency and parallelism versus optimization for throughput drove these changes. The aim of this ISCA paper was to bridge the gap between optimization differences and program-level considerations.

To achieve this, we introduced two metrics: memory warp parallelism (MWP), representing the number of concurrent memory requests that can be executed, and computation warp parallelism (CWP), representing the amount of computation that can be performed by other warps while one warp awaits memory values. By utilizing both MWP and CWP metrics, we classified the three cases: MWP $>$ CWP, CWP $>$ MWP, and insufficient parallelism to estimate the overall execution time of a program. Later the roofline model [7] provides clear

explanations of compute-limited (MWP>CWP) and memory-limited (CWP>MWP) cases but not the third case, which is not-enough parallelism. While the insufficient parallelism appeared to be an exceptional case during the development of the performance model, it turned out to be a critical component for programmers to consider. The analytical model converts the compute execution time from a thread to the unit of warp, which is the execution unit, and takes into account of the available number of GPU cores with respect to CUDA number of threads and blocks. Because the model connects the CUDA software with the GPU hardware parameters, the analytical model might represent high complexity to use, therefore, further contents are discussed in the Hong's Ph.D. dissertation.

An analytical model proves useful in predicting performance using a back-of-envelope computation, benefiting compilers and programmers. With GPUs providing numerous knobs for code optimization, one of the follow-ups to this paper aimed to integrate the analytical model with compilers, using it as a guide for performance optimizations. The work by [1] serves as an example. However, at the time this ISCA paper was published, the GPU compiler framework was not fully open, and it took longer to witness the integration of the analytical model and compiler.

Another application of the analytical model is providing guidance to programmers on code optimization. Furthermore, since the analytical model inherently focuses on essential performance-contributing factors, it provides researchers and programmers with insights into GPU performance. However, it is worth noting that seasoned programmers often employ the latest CUDA programming features, such as warp shuffling, which the analytical model may lack. Additionally, the paper "debunking the 100x GPU..." [3] demonstrates cases where CPUs outperform GPUs, providing concrete examples.

Due to the novelty of GPU architectures and the growing popularity of GPGPU in the computer community, this paper paved the way for numerous other GPU performance modeling studies. Subsequently, we published a paper on GPU power modeling in the follow-up ISCA [2], where we improved the GPU performance models to include caches and introduced additional metrics for programmers [6].

In today's era, GPUs are primarily employed for machine learning (ML) applications. Given the current importance of machine learning and the increased availability of open frameworks for GPU-based programs, several performance prediction studies have emerged that specialize in ML applications.

In conclusion, this ISCA paper made significant contributions by establishing a connection between CUDA programs and their performance implications. It introduced metrics, such as memory warp parallelism and computation warp parallelism, along with a categorization of memory overlapping scenarios and computation cycles, to estimate program execution time. The paper's analytical model provided valuable insights into the GPU performance and guided programmers and researchers in understanding the key performance components of GPU architecture and optimizing their code. While the programming landscape may have shifted with the emergence of machine learning applications and evolving programming paradigms, the importance of static performance prediction remains paramount for code optimization. It is worth noting that as GPUs continue to evolve and new architectural advancements are introduced, the analytical models and techniques presented in this paper may need to be adapted and expanded. Ongoing research and development in GPU performance modeling continue to refine and enhance our understanding of these architectures.

EPILOGUE

The publication of this ISCA paper provided us with the confidence and inspiration to pursue GPUs as our primary research focus. Even after 14 years since its publication, both authors have remained dedicated to advancing the field of GPUs. Sunpyo Hong, one of the authors, has continued to contribute to the field by focusing on Intel GPUs. Hyesoon Kim has been actively involved in the development of an open-source GPU.

As we reflect on the journey that began with this publication, we are grateful for the opportunities it has brought us and for the chance to be part of the ongoing advancements in GPU architecture modeling. We look forward to the future, where GPUs continue to play a pivotal role in enabling innovative and high-performance computing solutions.

REFERENCES

[1] P. Barua, J. Shirako, and V. Sarkar, "Cost-driven thread coarsening for gpu kernels," in *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT '18. New York, NY, USA: Association for Computing Machinery, 2018.

[2] S. Hong and H. Kim, "An integrated gpu power and performance model," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10, New York, NY, USA, 2010, p. 280–289.

[3] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, and P. Dubey, "Debunking the 100x gpu vs. cpu myth: An evaluation of throughput computing on cpu and gpu," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, p. 451–460, jun 2010.

[4] C.-K. Luk, S. Hong, and H. Kim, "Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42, New York, NY, USA, 2009, p. 45–55.

[5] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-m. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded gpu using cuda," in *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '08, New York, NY, USA, 2008, p. 73–82.

[6] J. Sim, A. Dasgupta, H. Kim, and R. Vuduc, "A performance analysis framework for identifying potential benefits in gpgpu applications," *SIGPLAN Not.*, vol. 47, no. 8, p. 11–22, feb 2012.

[7] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, pp. 65–76, 04 2009.