Retrospective: Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU

Victor W Lee*, Changkyu Kim†, Jatin Chhugani*, Michael Deisher§, Daehyun Kim¶, Anthony D Nguyen§, Nadathur Satish†, Mikhail Smelyanskiy‖, Srinivas Chennupaty§, Ronak Singhal§, Pradeep Dubey§
*Google, †Meta, §Intel, ¶Samsung, ‖Nvidia

I. Summary of the original paper

The original paper published in ISCA 2010 investigated multiple performance claims showing 1-2 orders of magnitude performance difference in favor of GPU over CPU running various throughput computing kernels. Our paper sought to dig deeper into these performance claims in terms of their various architectural, micro-architectural and algorithmic components. We proposed a systematic framework for analyzing the performance delta across the two parallel processors, enumerating key attributes of CPU and GPU architectures and related performance optimizations. Critical optimizations identified for CPU included multi-threading, cache blocking, and reorganization of memory accesses for SIMD-ification. Optimizations for GPU included minimizing global synchronization and using local shared buffers. Our paper also identified some fairness concerns regarding the large performance deltas such as using single thread/single core scalar code as an optimized proxy for a multi-core CPU, or comparing a high-end GPU to a mobile CPU, etc. Putting it all to practice, the paper showed how most of the CPU performance gap with respect to GPU could be recovered using properly conducted algorithm-architecture co-design, no different from what was needed and undertaken for the GPU. In summary, our paper highlighted the importance of algorithm-architecture co-design in the emerging multi/many-core era. This co-design benefits both GPUs and CPUs, and when it is leveraged to get optimal performance on GPUs, it should not be ignored for the CPU baseline.

II. Since 2010

Significant advancements have been made in CPU and GPU performance since 2010, with both reaching extraordinary levels of compute power per System-on-Chip (SoC). Parallel hardware remains the driving force behind achieving such high performance.

We take great pride in the extensive recognition and impact the paper has garnered, becoming the most cited paper in ISCA 2010 with over 1000 citations [4], and establishing a benchmark for comparing performance across both platforms; its influence has permeated numerous computer architecture studies. While the primary focus of the paper centers around comparing the performance of CPUs and GPUs in throughput computing applications, its overarching conclusions extend far beyond this domain. This work continues to get cited and inspire the research community and its thought leaders to this date. [1] [2]

At a broader level, the paper has had two lasting impacts on the architecture community:

- **Growing value-add of architecture-algorithm co-design in parallel architectures:** Given the parallel nature of both CPUs and GPUs, both are now increasingly dependent on the choice of algorithm for extracting maximal performance. And more importantly, this algorithmic choice is often different for CPU versus GPU, given the difference in their architectural tradeoffs: scalar-vs-SIMD, cache-vs-memory capacity and bandwidth, regular-vs-irregular parallelism, different set of fixed-function supports, etc. Therefore, an algorithm optimal for GPU for a certain task (such as, sorting) may be quite sub-optimal for CPU, and vice versa. Stated differently, two architectures should not be compared without offering to each its own optimal choice of algorithm to extract its full potential.
- **Importance of sanity-checking achieved performance against expected performance on a given architecture:** Any claim of performance on an architecture should be sanity-checked against its own performance model or roofline model. This should automatically render using unoptimized baselines invalid. In other words, performance claims should undergo meticulous scrutiny against fundamental resource metrics like memory bandwidth, computation capability (e.g., FLOPs), etc. to determine their validity before unquestioningly accepting them. It is imperative for the research community to hold itself accountable, demanding rigorous comparisons, and rejecting any research that fails to meet the standards of due diligence.

Furthermore, the paper delved into pivotal optimization techniques that retain their relevance in modern-day processors. These techniques encompass a range of strategies, including:

- **Maximizing data-, thread- and function-level parallelisms** is critical to fully utilize compute resources. While modern GPUs require thousands of threads to keep their processing units occupied, modern CPUs also benefit from hundreds of threads to achieve optimal performance.
- **Cache blocking** is essential for both CPUs and GPUs to maximize cache usage, as caches play a pivotal role in addressing the memory wall.
- **Recomputing data to reduce reliance on memory bandwidth** is another effective optimization technique to reduce the pressure on memory bandwidth for severely bandwidth limited algorithms. However, recomputing could require more effort in maintaining the application.
- **Minimizing synchronization or utilizing low cost synchronization** whenever possible.

Beyond these points, other important insights that have been gained over time include:

- Achieving a fully cache coherent view of data between CPUs and GPUs continues to be impractical. Techniques which were suggested then are still as relevant as before.
  - Singular view of memory (amongst CPU and GPU)
  - Amortize cost of synchronization with lots of compute per synchronization
- Programmability is just as significant as performance. For instance, achieving high program efficiency for irregular programs can be challenging with SIMD. Similarly, SIMT without underlying hardware support is less efficient for programs with complex control flows. Trading compute for memory bandwidth is another example where programmability can play an important role in deciding whether that optimization makes sense or not. The programmability argument was further emphasized by the follow-up work on whether compiler technology can close the performance gap from carefully hand-tuned workloads. [3]
- Power has increasingly become the primary constraint for processor architecture. To optimize power efficiency, modern processors incorporate different types of cores and numerous accelerators. This trend is expected to continue in the future.
- Designing and utilizing specialized functional blocks in System-on-Chip (SoC), such as nonlinear math functions, is key to achieving optimal power-performance. However, identifying the specific hardware accelerators required for this purpose remains quite challenging. Workload analysis will be even more important in future, especially in the era of AI.
- While chip-level performance matters, it is important to further optimize performance of an end-to-end system, including network, storage and supporting infrastructure. If left unoptimized, these components can become Amdahl's bottleneck and waste CPU or GPU optimization effort.

III. Conclusion

We, as authors of the paper, are pleased to note that over the last 12 or more years since this work was published both CPUs and GPUs have grown many fold in terms of the architectural parallelism they support. In fact, the need as well as the practice of algorithm-architecture co-design called out by the paper has now become widespread. This has in turn resulted in a large collection of architecture-aware, novel performance optimization techniques for both CPUs and GPUs. Furthermore, these architectures have increasingly adapted and evolved along similar lines with support for gather-scatter, large caches, and now systolic engines for matrix arithmetic. As a result, there is a larger set of common optimization techniques now applicable to both CPUs and GPUs. Consequently, reported performance speedup claims have become better and fairer in terms of extracting the most from each architecture's roofline potential, as well as easier to understand and explain. As the programming community of these parallel processors continues to attract more domain experts and data scientists; whereas, processor complexity is on the rise, the architecture community needs to turn its attention towards increased automation of such algorithm-architecture co-design to reduce the growing industry reliance on its performance-ninjas.

Acknowledgements

We would like to acknowledge the researchers and students who created the benchmarks and worked on the optimizations we used in the paper. We also would like to acknowledge the feedback from reviewers and colleagues to make this a great paper.

References

[1] C. Gregg and K. Hazelwood, "Where is the data? why you cannot debate cpu vs. gpu performance without the answer," in *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*, 2011, pp. 134–144.

[2] S. Matsuoka, J. Domke, M. Wahib, A. Drozd, and T. Hoefler, "Myths and legends in high-performance computing," 2023.

[3] N. Satish, C. Kim, J. Chhugani, H. Saito, R. Krishnaiyer, M. Smelyanskiy, M. Girkar, and P. Dubey, "Can traditional programming bridge the ninja performance gap for parallel computing applications?" in *39th International Symposium on Computer Architecture (ISCA 2012), June 9-13, 2012, Portland, OR, USA*. IEEE Computer Society, 2012, pp. 440–451. [Online]. Available: https://doi.org/10.1109/ISCA.2012.6237038

[4] G. Upasani, M. D. Sinclair, A. Sampson, P. Ranganathan, D. Patterson, S. Shah, N. Parthasarathy, and R. Jain, "Fifty years of isca: A data-driven retrospective on key trends," 2023.