2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS) | 979-8-3503-7638-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISPASS61541.2024.00038

Accel-Bench: Exploring the Potential of Programming using Hardware-Accelerated Functions

Abenezer Wudenhe
University of California, Riverside
Rivsrside, California, USA
awude001@ucr.edu

Yu-Chia Liu
University of California, Riverside
Rivsrside, California, USA
yliu719@ucr.edu

Chris Chen
University of California, San Diego
San Diego, California, USA
chc033@ucsd.edu

Hung-Wei Tseng
University of California, Riverside
Rivsrside, California, USA
htseng@ucr.edu

***Abstract*—This paper presents Accel-Bench, a benchmark suite that aims to capture the performance of accelerator-intensive programming. To the best of our knowledge, Accel-Bench is the first benchmark suite that utilizes applications that can invoke different domain kernels in their algorithm and quantifies the potential performance gain of using hardware-accelerated functions to compose programs agnostic to their domain.**

I. Introduction

The introduction of hardware accelerators has brought exotic flavors of computing models into computer systems. Instead of implementing a rich set of fine-grained mathematical or logical operations, each operation in a hardware accelerator can implement a complete compute kernel in the accelerator's target application domain. As each operation covers a coarse-grained computation and each hardware accelerator has a limited target application domain, the design of hardware accelerators can use transistors more efficiently and deliver better performance or energy consumption than general-purpose processors when accomplishing the same task.

The integration of hardware accelerators also significantly shifted the programming paradigm. Programmers no longer describe the exact algorithm using the authoring programming language but only need to invoke a function/method that maps to an algorithm in an application domain that an underlying hardware accelerator implements. In addition to addressing the demands in accelerators' original target domains, recent research projects have successfully demonstrated the potential of accelerating a broader spectrum of problems using these domain-specific functions. Examples include using AI/ML accelerators for database queries [6], [10], [12], fast Fourier transforms [13], or scientific applications [7], [8], [11], [14], [15], and using ray tracing accelerators for data analytics [20].

This paper presents Accel-Bench, a benchmark suite targeting the future world of accelerator-intensive programming. Accel-Bench contains applications from various domains that can leverage the most promising hardware accelerators, including tensor processors, digital signal processors, and ray tracing accelerators. In contrast to conventional programs, Accel-Bench provides alternative implementations that invoke hardware-accelerated functions whenever possible. To increase the opportunities for using hardware accelerators, we carefully re-engineered the algorithms or used different approaches in several application kernels to map their core computation into accelerated, domain-specific functions. In addition to running applications on real hardware, the resulting applications can use simulators like Accel-Sim as long as the framework provides the required hardware-accelerated functions.

II. The Accel-Bench Benchmark Suite

Accel-Bench aims to allow the community to access the potential of programming using hardware-accelerated functions, evaluate performance scaling with emerging hardware accelerators, and assist the architecture design of more general hardware accelerators. Accel-Bench identifies a broad spectrum of applications that cover several vital dwarfs while their algorithmic problems can map to hardware-accelerated functions. Accel-Bench revisited these applications and revised their state-of-the-art implementations to leverage existing/potential hardware-accelerated functions. This section will describe the goals and the detailed implementations of Accel-Bench.

A. Workloads

Table I lists the applications that Accel-Bench includes. These applications cover dwarfs, including dense linear algebra, Structured Grid, Spectral Methods, and Graph Traversal. These applications fall into image processing, data mining, physics simulations, genomics, signal processing, web mining, and social network analysis beyond the domains of the hardware accelerators that modern computers provide.

For implementation on the CPU, we used a matrix multiplication and an optimized implementation from the GAP Benchmark suite [4] as a baseline. For the GPU, we utilized

2766-0486/24/$31.00 ©2024 IEEE
DOI 10.1109/ISPASS61541.2024.00038

Benchmark	Dwarf	Application Domain	Hardware accelerated function(s)	Baseline Implementation
Canny Edge Detection (CED)	Dense Linear Algebra	Image Processing	CONV	RosettaCode [2], Chai [9]
K Nearest Neighbor (KNN)	Dense Linear Algebra	Data Mining	GEMM	KNNCUDA [19], RTNN [20]
Heat (Heat2D/Head3D)	Structured Grid	Physics Simulation	CONV/FFT	TEAlab [1], FDTD3D [16]
KMeans (KM)	Dense Linear Algebra	Data Mining	GEMM	Rodinia [5]
Genomic Relationship Matrix (GRM)	Dense Linear Algebra	Bioinformatics / Genomics	GEMM	GenomicsBench [18]
Short-Time Fourier Transform (STFT)	Spectral Methods	Digital Signals Processing	FFT	RosettaCode [3]
PageRank (PR)	Graph Traversal	Web Minning	GEMV	GAP [4]
Triangle Counting (TC)	Graph Traversal	Social Network Analysis	GEMM	GAP [4]

TABLE I
TABLE OF BENCHMARKS

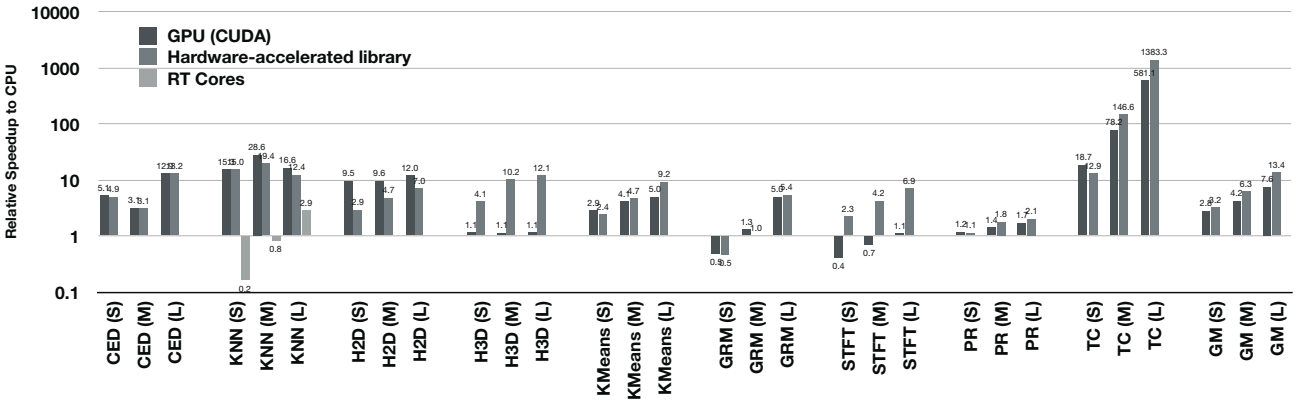

Fig. 1. The performance result on the default machine

cuBLAS [17] with and without Tensor Cores to accelerate matrix multiplication, along with CUDA kernels to calculate the Hamming product and summarize the final count.

III. Results

This section summarizes our evaluation of Accel-Bench. The result shows that the hardware-accelerated API-based paradigm that Accel-Bench promotes is on par with or outperforms the state-of-the-art implementation's performance.

A. Performance

On average, using hardware accelerated API as the programming paradigm can speed Accel-Bench applications by 1.13× to 1.77× compared to the state-of-the-art GPU implementations, despite that GPU implementations are already 2.8× to 7.6× faster than CPU implementations. Figure 1 details the relative speedup of running Accel-Bench on the default server, compared to the CPU baseline.

In general, we found the performance gain of using hardware-accelerated APIs is more significant when dataset sizes become more extensive. As the data structures that hardware-accelerated APIs accept do not always fit the original application's data structures, Accel-Bench must contain code to explicitly convert and prepare data structures for the inputs and outputs of hardware-accelerated APIs. When the dataset becomes larger, the increased complexity in the accelerated function's counterpart helps mitigate the overhead of such a process.

H2D and H3D provide another aspect in showing the strength of hardware-accelerated APIs. Despite simulating the thermal states using stencil operations, H3D additionally considers twice as many diagonals as H2D. Therefore, the computation of H3D becomes significantly higher than H2D. The current Accel-Bench implementation maps stencil operations into convolution and the rest in FFT. Using the convolution function for stencil will result in zero elements in computation and negate the effect of hardware acceleration to a certain degree. FFT requires high overhead in data conversion but is also less optimized due to the absence of FFT accelerators on NVIDIA GPUs. Therefore, the amount of computation in functions that hardware-accelerated APIs can accelerate does not allow hardware-accelerated APIs to gain performance. In contrast, hardware-accelerated APIs become more effective as these functions take more computation in H3D.

IV. Conclusion

This paper generates two critical insights. First, a hardware-accelerated-API-centric programming model is evenly or more competitive than conventional performance programming methods. Second, as architectural innovations focus more on hardware accelerators, we have seen more significant gains with upgraded hardware using a hardware-accelerated-API-centric programming model.

We envision Accel-Bench will encourage more exploration of hardware-accelerated-API-centric programming models. We also anticipate Accel-Bench can help identify and design architectures to optimize the potential performance issues in such models.

REFERENCES

[1] TEAlab/FFTStencils. https://github.com/TEAlab/FFTStencils/tree/main.

[2] Canny edge detector. https://rosettacode.org/wiki/Canny_edge_detector, November 2015.

[3] Fast Fourier transform. https://rosettacode.org/wiki/Fast_Fourier_transform, December 2023.

[4] Scott Beamer, Krste Asanović, and David Patterson. The gap benchmark suite. *arXiv preprint arXiv:1508.03619*, 2015.

[5] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE international symposium on workload characterization (IISWC)*, pages 44–54. Ieee, 2009.

[6] Abdul Dakkak, Cheng Li, Jinjun Xiong, Isaac Gelado, and Wen-mei Hwu. Accelerating reduction and scan using tensor core units. In *Proceedings of the ACM International Conference on Supercomputing*, ICS '19, pages 46–57, 2019.

[7] Sultan Durrani, Muhammad Saad Chughtai, Mert Hidayetoglu, Rashid Tahir, Abdul Dakkak, Lawrence Rauchwerger, Fareed Zaffar, and Wen-mei Hwu. Accelerating fourier and number theoretic transforms using tensor cores and warp shuffles. In *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 345–355, 2021.

[8] Hadi Esmaeilzadeh, Adrian Sampson, Luis Ceze, and Doug Burger. Neural acceleration for general-purpose approximate programs. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-45, pages 449–460, Washington, DC, USA, 2012. IEEE Computer Society.

[9] Juan Gómez-Luna, Izzat El Hajj, Li-Wen Chang, Victor García-Floreszx, Simon Garcia De Gonzalo, Thomas B Jablin, Antonio J Pena, and Wen-mei Hwu. Chai: Collaborative heterogeneous applications for integrated-architectures. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 43–54. IEEE, 2017.

[10] Pedro Holanda and Hannes Mühleisen. Relational queries with a tensor processing unit. In *Proceedings of the 15th International Workshop on Data Management on New Hardware*, DaMoN'19, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Kuan-Chieh Hsu and Hung-Wei Tseng. Accelerating Applications using Edge Tensor Processing Units. In *SC: The International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC 2021, 2021.

[12] Yu-Ching Hu, Yuliang Li, and Hung-Wei Tseng. TCUDB: Accelerating Database with Tensor Processors. In *the 2022 ACM SIGMOD/PODS International Conference on Management of Data*, SIGMOD 2022, 2022.

[13] Binrui Li, Shenggan Cheng, and James Lin. tcfft: A fast half-precision fft library for nvidia tensor cores. In *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–11, 2021.

[14] Tianjian Lu, Thibault Marin, Yue Zhuo, Yi-Fan Chen, and Chao Ma. Accelerating mri reconstruction on tpus. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9, 2020.

[15] Tianjian Lu, Thibault Marin, Yue Zhuo, Yi-Fan Chen, and Chao Ma. Nonuniform fast fourier transform on tpus. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 783–787, 2021.

[16] Nvidia. Fdtd3d - cuda c 3d fdtd. https://github.com/olcf/cuda-training-series/blob/master/exercises/hw2/readme.md.

[17] NVIDIA. cuBLAS. https://docs.nvidia.com/cuda/cublas/index.html, 2019.

[18] Arun Subramaniyan, Yufeng Gu, Timothy Dunn, Somnath Paul, Md Vasimuddin, Sanchit Misra, David Blaauw, Satish Narayanasamy, and Reetuparna Das. Genomicsbench: A benchmark suite for genomics. In *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 1–12, 2021.

[19] Michel Barlaud Vincent Garcia, Éric Debreuve. kNN-CUDA. https://github.com/vincentfpgarcia/kNN-CUDA, 2018.

[20] Yuhao Zhu. RTNN: Accelerating Neighbor Search Using Hardware Ray Tracing. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '22, pages 76–89, 2022.