Current Trends in Computer Sciences & Applications

DOI: 10.32474/CTCSA.2024.03.000159

ISSN: 2643-6744

Research Article

New Research Opportunities with Modern Accelerators

Cristóbal A. Navarro*

Associate Professor Computer Sciences, Austral University, Chile

***Corresponding author:** Cristóbal A. Navarro, Associate Professor Computer Sciences, Austral University, Chile

Received: 📅 December 19, 2023 **Published:** 📅 January 11, 2024

Abstract

Parallel processors have undergone a profound transformation in recent years, transitioning from homogeneous general-purpose units to a heterogeneous ecosystem comprising a mix of general and specific-purpose cores on a single chip. This shift, driven by the demands of Artificial Intelligence (AI) and computer graphics applications, has not only altered the architecture of processors but has also introduced novel challenges in optimizing algorithms for parallel execution. In this brief review, we delve into the evolution of parallel processors and explore the research challenges arising from this shift. We will be focusing on the particular case of GPUs, where tensor cores and ray tracing cores have created new research opportunities on finding what other applications, different from AI and graphics, could be reformulated as a series of tensor/ray-tracing core operations and further accelerate their performance compared to their regular GPU implementation.

From general purpose to specific purpose

Parallel computing gained a strong relevance with the introduction of the first dual-core CPU in the early 2000s. From there, parallel architectures as well as research in parallel computing achieved significant milestones, such as the possibility to pack dozens of CPU cores in a single chip, new parallel algorithms and the development of parallel programming languages and tools [1]. Today we have a large ecosystem of parallel processors sitting in many of the devices we use every day; from laptops and cellphones to TVs and Cars. In the last couple of years, with the surge of artificial intelligence and videogames, parallel computing has become even more relevant, as it is the technological bed for many states of the art applications that require high performance. During these last 5-6 years, the computing community has witnessed how parallel processors have evolved from being an homogenous set of general purpose cores, to an heterogenous set that now includes specific- purpose cores. A notable case study in this technological transformation is the Graphics Processing Unit (GPU). Around 2006, when NVIDIA announced the CUDA programming platform [2], GPUs transitioned from being specialized hardware for graphics rendering to general purpose accelerators. From that moment GPUs became an attractive device for doing very fast scientific computations. Nearly a decade later, with the surge of Artificial Intelligence (AI), the community realized that the performance of GPUs was not high enough to properly handle the new Deep Learning models being developed. For this reason, near 2017, NVIDIA introduced tensor cores [3-12] inside the chip to further accelerate the performance of all AI applications. GPU Tensor cores are Application Specific Integrated Circuits (ASICs), or simply specific-purpose cores that perform fast matrix multiply accumulate (MMA) operations. With Tensor cores, AI applications can further accelerate their performance by an extra order of magnitude, allowing the training of large models to go down from months to a few days. As of 2024, tensor cores are present in NVIDIA [13], AMD [6] and Intel GPUs [14], and are slowly becoming part of the CPUs as well. The videogame industry also had its revolution recently. In 2018 (one year after the inclusion of tensor cores) NVIDIA designed the Ray Tracing (RT) core to be inside their GPU chips. RT cores enable the processing of the ray tracing algorithm in real-time, bringing the possibility for interactive 3D applications to feature photorealistic lighting. Ray tracing [7] is one of the most computationally demanding tasks in 3D rendering as it requires thousands of rays to be traced and checked in order to find which triangles they hit. The difficulty comes because it is a search problem; for each ray, one needs to find which triangle has been hit by it. Doing it by brute force would mean checking all triangles of the scene for each ray, making it very inefficient. Space partitioning trees [9] and other variants of trees have been implemented in GPU [8], although the nature of trees introduce a difficult irregular memory accesses for the GPU architecture which is limited in this aspect. As a solution to this problem, an RT core offers a hardware implemented Bounding Volumne Hierarchy (BVH) tree data structure [15], allowing a ray to find ray/triangle intersections (other custom primitives as well) overall significantly faster than the software-implemented alternatives. Due to the success of the Ray Tracing core, as of 2024, all major GPU companies include them in one or other equivalent form.

New Research Opportunities

The recent inclusion of specific purpose cores in parallel accelerators created the research question; is it possible that other applications, different from AI and Graphics, could also benefit from the new tensor cores and RT cores? The answer is yes and this has opened a whole new research field in GPU Computing; to find ways to reformulate common computational patterns, even ones already adapted for traditional GPU Computing, now as a series of tensor/ray-tracing operations and obtain an additional performance lift. When programming tensor or RT cores, a great part of the pipeline is a black box, which is where the hardware-implemented functionality takes part. Therefore, adapting a computational pattern to tensor/RT cores greatly involves coming up with a new statement of the computational pattern, now formulated as a series of tensor/RT operations. Successful research has been done in the recent years. In the case of tensor cores, new ways have been proposed to further accelerate arithmetic reductions [16, 13, 5-12, 17-21] prefix sum [4-12, 17-21, 22-29] Fast Fourier Transform [22], [10], [23], [5], stencil computations for PDE simulations [11] and even fractals [14, 25-24]. In general, all of these works achieve significant higher performance when compared to doing it traditionally in GPU. Moreover, many times this benefit in performance also comes with less energy consumption, making it a more energy efficient approach as well. In the case of Ray Tracing cores, a significant amount of works can also be found. One of the most relevant research topics have been on finding ways to compute the nearest neighbors of many particles in parallel, using the high search speed of RT cores [20, 26-28]. Other works include a fully RT core approach for answering the Range Minimum Query (RMQ) problem [17], which consists of finding the minimum in a given interval [i,j] of an unordered array. In the case of geometry, point location has been solved with RT cores as well [18]. More recently, a clustering approach has been proposed that leverages RT cores [19]. There are still several candidate open problems for being adapted to tensor or RT cores. The key for bringing new ideas to tensor cores is find ways to group the arithmetic operations of a process as a series of matrix multiply accumulate (MMA) operations. If this can be done, and the matrices involved can be populated almost entirely with useful data, then there is a strong chance that the tensor cores can provide a performance boost. There are technical limitations though, for example the MMA operation offers several data types such as FP16, TF32, BF16 and INT8, among others. The less the precision, the faster the performance, therefore one should be cautious on what datatype to work with, ensuring both correctness and speed. In the case of RT cores, the key is to realize that the ray-triangle intersection is actually a search tool that when properly used, can solve non-graphical problems. The two major challenges when adapting a computation to RT core are i) to find a proper 3D geometrical representation of the input data, and ii) to find a ray launch scheme such that when colliding with the input data (triangles), it answers the intended search query. If these two challenges can be overcomed, then the problem may be computed with RT cores. Future parallel processors may keep bringing new specific-purpose cores to the table, creating new research challenges. Moreover, this research is not only limited to GPUs, but to all the current processors that are adding specific-purpose units in their chip, including embedded devices as well.

References

1. Cristobal A Navarro, Nancy Hitschfeld Kahler, Luis Mateu (2014) A survey on parallel computing and its applications in data-parallel problems using gpu architectures. Communications in Computational Physics, 15(2): 285-329.
2. Jason Sanders, Edward Kandrot (2010) CUDA by example: an introduction to general-purpose GPU pro- gramming. Addison-Wesley Professional.
3. Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, Ronny Krashinsky (2021) Nvidia a100 tensor core gpu: Performance and innovation. IEEE Micro, 41(2): 29-35.
4. Abdul Dakkak, Cheng Li, Jinjun Xiong, Isaac Gelado, Wen mei Hwu (2019) Accelerating reduction and scan using tensor core units. In Proceedings of the ACM International Conference on Supercomputing, p. 46-57.
5. Sultan Durrani, Muhammad Saad Chughtai, Abdul Dakkak, Wen mei Hwu, Lawrence Rauchwerger (2021) Fft blitz: the tensor cores strike back. In Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, p. 488-489.
6. Massimiliano Fasi, Nicholas J Higham, Florent Lopez, Theo Mary, Mantas Mikaitis (2023) Matrix multi- plication in multiword arithmetic: error analysis and application to gpu tensor cores. SIAM Journal on Scientific Computing, 45(1): 1-19.
7. Andrew S Glassner (1989) An introduction to ray tracing. Morgan Kaufmann.
8. Patrick Gralka, Ingo Wald, Sergej Geringer, Guido Reina, Thomas Ertl (2020) Spatial partitioning strategies for memory-efficient ray tracing of particles. In 2020 IEEE 10th Symposium on Large Data Analysis and Visualization (LDAV), p. 42-52.
9. Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, Dinesh Manocha (2009) Fast bvh construction on gpus. In Computer Graphics Forum, 28(2): 375-384.
10. Binrui Li, Shenggan Cheng, James Lin (2021) tcfft: A fast half-precision fft library for nvidia tensor cores. In 2021 IEEE International Conference on Cluster Computing (CLUSTER), pp. 1-11.

11. Xiaoyan Liu, Yi Liu, Hailong Yang, Jianjin Liao, Mingzhen Li, et al. (2022) Toward accelerated stencil computation by adapting tensor core unit on gpu. In Proceedings of the 36th ACM International Conference on Supercomputing, p. 1-12.

12. Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, Jeffrey S Vetter (2018) Nvidia tensor core programmability, performance & precision. In 2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW), pages 522-531.

13. Jack Choquette (2023) Nvidia hopper h100 gpu: Scaling performance. IEEE Micro, 43(3): 9-17.

14. Jon Peddie (2023) The sixth era gpus: Ray tracing and mesh shaders. In The History of the GPU-New Developments, pp. 323-360.

15. John Burgess (2020) Rtx on-the nvidia turing gpu. IEEE Micro, 40(2): 36-44.

16. Roberto Carrasco, Raimundo Vega, and Cristobal A Navarro (2018) Analyzing gpu tensor core potential for fast reductions. In 2018 37th International Conference of the Chilean Computer Science Society (SCCC), pages 1-6.

17. Enzo Meneses, Cristobal A Navarro, Hector Ferrada, Felipe A Quezada (2023) Accelerating range minimum queries with ray tracing cores.

18. Nate Morrical, Ingo Wald, Will Usher, Valerio Pascucci (2020) Accelerating unstructured mesh point location with rt cores. IEEE transactions on visualization and computer graphics, 28(8): 2852-2866.

19. Vani Nagarajan, Milind Kulkarni (2023) Rt-dbscan: Accelerating dbscan using ray tracing hardware, pp. 963-973.

20. Vani Nagarajan, Durga Mandarapu, Milind Kulkarni (2023) Rt-knns unbound: Using rt cores to accelerate unrestricted neighbor search. In Proceedings of the 37th International Conference on Supercomputing, pages 289-300.

21. Cristobal A Navarro, Roberto Carrasco, Ricardo J Barrientos, Javier A Riquelme, Raimundo Vega (2020) Gpu tensor cores for fast arithmetic reductions. IEEE Transactions on Parallel and Distributed Systems, 32(1): 72-84.

22. Anumeena Sorna, Xiaohe Cheng, Eduardo Dazevedo, Kwai Won, Stanimire Tomov (2018) Optimizing the fast fourier transform using mixed precision on tensor core hardware. In 2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW), pages 3-7.

23. Xiaohe Cheng, Anumeena Sorna, Eduardo D Azevedo, Kwai Wong, Stanimire Tomov (2018) Accelerating 2d fft: Exploit gpu tensor cores through mixed-precision. In The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'18), ACM Student Research Poster, Dallas, TX pp. 1-2.

24. Cristobal A Navarro, Felipe A Quezada, Nancy Hitschfeld, Raimundo Vega, Benjamin Bustos (2020) Efficient gpu thread mapping on embedded 2d fractals. Future Generation Computer Systems, 113(1): 158-169.

25. Felipe A Quezada, Cristobal A Navarro, Nancy Hitschfeld, Benjamin Bustos (2022) Squeeze: Efficient compact fractals for tensor core gpus. Future Generation Computer Systems, 135(1): 10-19.

26. Stefan Zellmann, Martin Weier, Ingo Wald (2020) Accelerating force-directed graph drawing with rt cores. In 2020 IEEE Visualization Conference (VIS), pp. 96-100.

27. Shiwei Zhao, Zhengshou Lai, Jidong Zhao (2023) Leveraging ray tracing cores for particle-based simulations on gpus. International Journal for Numerical Methods in Engineering, 124(3): 696-713.

28. Yuhao Zhu (2022) Rtnn: accelerating neighbor search using hardware ray tracing. In Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 76-89.

29. Anastasios Zouzias, William F McColl (2023) A parallel scan algorithm in the tensor core unit model. In European Conference on Parallel Processing, pp. 489-502.

DOI: 10.32474/CTCSA.2023.03.000159