Faculty advisor approval:
Davide Lazzati

May 7, 2021

OSGC affiliate approval:
Kyle Niemeyer

7 May 2021

Optimizing NASA's High-Performance Computing

Hardware Improvements and Storage Streamlining to Reduce "Time-To-Science" on Physical Science Projects and Simulations

Alexander Mote
motea@oregonstate.edu
School of Electrical & Computer Engineering
Oregon State University

7 May 2021

Supported in part through NASA and Oregon Space Grant Consortium cooperative agreement 80NSSC20M0035

Partner

The material contained in this document is based upon work supported by a National Aeronautics and Space Administration (NASA) grant or cooperative agreement. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of NASA.

Abstract:

High-performance computing has quickly become an integral part of scientific advancement, and a great deal of work has gone into making deep learning software easier to both create and deploy; however, many scientific projects have been bottlenecked by hardware limitations, rather than software ones. This research examines those limitations and investigates what changes can be made to minimize them. FPGAs (Field Programmable Gate Arrays) prove to be a promising alternative to GPUs (Graphical Processing Units) in both power consumption and interface flexibility, and storage optimization allows for these accelerators to access data more quickly and without the need for a CPU (Central Processing Unit) interface. Further research should develop models for optimizing FPGA integration, and test physical science algorithms on different processor and storage models to find optimal configurations.

Introduction:

As the questions we ask in the fields of science become better informed and more complex, science professionals increasingly turn to the use of high-performance computation and machine learning to find the answers. As a result of this increase in demand, these fields have become a booming area of research, with many new discoveries being made every day. However, a great deal of this research and its implementations are made on a small-scale, case-by-case basis, requiring the regular creation of entirely new systems and implementations which drastically increase the time until actual scientific analysis can be done. On the other hand, large-scale improvements, particularly those on the hardware level, require such a large upfront financial and developmental investment that they are often rejected in favor of more short-term projects.

This research aims to take a closer look at the long-term benefits of these hardware-level improvements and optimizations, in order to see if they offset the initial costs. Of particular interest in this area is the utility and selection of different processors, their comparative benefits, and how they can best be used together to solve the particular physical science problems encountered by NASA and its partners. In addition to the use and combination of these processors, the storage and access of data is another important area that hardware changes can improve and expedite. Through these lenses, this paper will attempt to show that these improvements may be worthy of further research; the next steps of this research will also be laid out and, hopefully, explored in future efforts.

Problem Statement:

The ongoing problem with many projects that require high-performance computing involves the “time-to-science”; the amount of time that it takes to perform mathematical and engineering operations that fall mostly outside the realm of actual scientific analysis. Often, modern physical science problems and questions are so complex that they require the creation of entirely new systems to find an answer, and the work of designing and implementing these systems on the physical level, writing and testing code on the programming level, and deploying this code on the hardware level, can quickly add up to a great deal of time and effort before scientific conclusions can be made.

In the past, this problem has often been approached with the expectation that the existing hardware is an immovable limitation. As a result of this, much of the research on this issue attempts to solve it through optimization on the software and programming level. Although a great deal of time can be saved in this way, by designing algorithms and training models that minimize computation time, the hardware of high-performance computing has seen many important advancements that this branch of research has not entirely explored. Additionally, due to the multifaceted and complex nature of these problems, software modifications and optimizations often must be done on a case-by-case basis, further adding to the time-to-science for each individual project.

Hardware improvements and optimizations, on the other hand, require a large initial investment of time, design, and budget, but can have an exponentially better return on investment in the long term if the right design decisions are made, with fewer restrictions allowing for less time to be spent on tweaking these algorithms, and more and different types of computing power allowing for easier and faster deployment of these models. Due to these advantages, the aim of this research is to find which changes and advancements can be made to high-performance computing hardware to minimize the “time-to-science” on the types of physical science problems currently investigated by NASA and its partners.

Background Information:

Although deep learning and artificial intelligence are quickly taking over the high-performance computing space, the use of CPU systems for highly parallelable problems cannot be overlooked. CPU-based supercomputers have been around for decades now, and until recently were seen as the dominant systems in terms of computing power. The easy parallelability of these systems have made them useful for the solution of partial differential equations, as well as the Monte Carlo simulations used to model interactions with many degrees of freedom. As such, they are the systems that are used to solve many astrophysics problems, such as hydrodynamic equations and gamma ray bursts (Mignone et al, 2007; Fryxell et al, 2000). CPUs are also seen as incredibly flexible in

terms of programmability, making them the ideal processor to handle many of the serial tasks required for high-performance computing projects (Jawandhiya, 2018).

However, as deep learning and neural networks have begun to reshape our understanding of how computers can solve problems, CPUs have begun to be supported of other types of processors, known generally as accelerators. In other words, while the majority of programming and interfacing is done with CPUs, and a great deal of computing is still done by these processors, accelerators offer a boost in performance at the cost of flexibility (Jawandhiya, 2018). These accelerators account for over one-third of all performance power in high-end supercomputers today (TOP500, 2020; see Figure 1 in Appendices), and have quickly become the main focus of hardware optimization research. These accelerators generally come in three varieties; namely, GPUs, FPGAs, and ASICs. Some “big data” companies like Google have developed their own proprietary processors as well; however, the nature and use of these devices are closely held by these companies, and as such will not be discussed in this paper.

Graphical Processing Units, or GPUs, have become the “poster child” of modern deep learning systems. Originally designed for the kinds of vector operations used in graphical rendering, their processing speeds eventually began to exceed that of the average CPU with respect to certain mathematical operations. Conveniently, these operations are also used in the deployment of many types of neural networks and parallel computation problems, making them an ideal candidate for use in both high-performance computing and deep learning systems. In particular, deep learning systems have a wide range of applications beyond traditional computing, from image recognition and signal processing, to identifying new planets and phases of matter. These processors exhibit far better performance than CPUs with respect to both cost and power consumption, but are not as versatile as CPUs and are more difficult to program for, and thus are generally used through a CPU-based interface, with CPUs offloading certain operations to GPUs for faster processing (Jawandhiya, 2018). Figure 2 in Appendices (Galloy, 2013) shows how the optimal processing power of GPUs has grown to vastly outperform that of CPUs.

One of the biggest issues currently affecting GPU viability is power consumption. Although GPUs have higher optimal processing power than CPUs per unit of power consumed, the amount of power required to reach this optimal level is still quite vast, especially at a high-performance level with dozens or hundreds of GPUs. Figure 3 in Appendices shows how high-end GPUs consume an average of 100 more Watts than high-end CPUs, a power spike that would actually increase if not for the fact that GPUs begin to overheat at this level (Rupp, 2016). The waste heat generated by these units also creates a size constraint for designers, forcing them to build larger computation units that must be carefully cooled in order to achieve as close to optimal performance

as possible (Berten, 2016). Additionally, the nature of pushing these processors to perform duties outside their original graphical purpose means that optimizing their vast parallelization capabilities is often a huge design challenge (Bakhoda et al, 2009). In other words, some problems are too big or strange for GPUs to easily solve with the cores and instruction sets they have, and this often causes a great deal of design time on the software level to adjust the solution approach in a way that fits the parameters a GPU is equipped to solve.

Field Programmable Gate Arrays, or FPGAs, are perhaps the most exciting possible solution being explored to address some of the shortcomings of the GPU. Although previously overlooked due to low performance compared to GPUs, FPGAs have quickly become a contender in this field due to their high processing ability per unit of power consumed (Berten, 2016). The main benefit of FPGAs, aside from their low power consumption, is their ease of reconfiguration; specifically, FPGAs essentially allow for the “rewiring” of logic gates and memory elements based on parameters set by the programmer. This allows for the rapid prototyping of hardware configurations, and optimization of circuit functionality per unit area (Jawandhiya, 2018). This also allows FPGAs to perform certain types of operations more efficiently than GPUs, even outperforming them on some neural networks (Nurvitadhi et al., 2016).

Another small benefit to FPGAs is their higher resistance to radiation compared to other processors (Richter, 2021), which may be an advantage when building high-performance computing systems for deep space deployment. Their interfacing capabilities also far exceed GPUs, allowing further versatility and more optimal access to things like memory (Berten, 2016). However, their low performance in traditional deep learning operations does bring up some concerns with regards to the high-level computations required for physical science simulations and problem-solving. Figure 4 in Appendices (Berten, 2016) shows a few of the benefits and tradeoffs of using FPGA accelerators versus GPUs.

On the extreme end of the efficiency scale are Application-Specific Integrated Circuits, or ASICs. These devices are similar to FPGAs, only they cannot be reconfigured and are essentially hardwired to perform a specific set of functions. While the performance power of ASICs cannot be beat by any of our other accelerator options, their absolute lack of flexibility and high engineering costs make them less than ideal for the kinds of computing projects done by NASA and its partners (Jawandhiva, 2018). Although not widely available or feasible for large-scale computational physics at the time of this writing, a great deal of work is also being done in the realm of neuromorphic computing, with new microchips and circuits being developed for the express purpose of simulating the human brain. Although the majority of these advances and implementations are designed for the medical field (Berggren et al, 2020), these advancements could bring about a new, more powerful generation of

neural networks in the near future, and their development should be watched closely by those in the computer hardware industry.

The storage and access of the data used by these processors is another area of interest for optimization. CPU access to a system's memory is relatively quick and easy to manage, with high-end CPUs managing memory bandwidths as high as 50 gigabytes per second (Jawandhiva, 2018). GPUs have a harder time managing the serialized filesystems of most storage architecture, and as a result typically access this data by interfacing with the CPU system, often causing the CPU to interrupt its computations to handle the data transfer. The time it takes for the CPU to access and send this data also means that the GPU is "waiting" for this data to arrive, and is unable to perform calculations until the transfer is complete. All of this stalling and waiting adds up to a great deal of wasted time in the pipelining of these memory operations. However, some computation systems have experimented with a centralized storage system, such as NASA's Center for Climate Simulation. Their Transiting Exoplanet Survey Satellite project implemented a system where several processor clusters were connected to a single shared memory space (Carriere, 2020). This storage system allows for these multiple high-performance computing systems to access large amounts of data concurrently, significantly reducing computing time that is otherwise lost in data transfer.

Another recent breakthrough in high-performance computing storage comes in the form of SHIP, a system of Storage for Hybrid Interconnected Processors. Similar in concept to the centralized storage system, this method of storage creates a "wrapper" around a solid-state hard drive that performs the necessary file system operations for any and all types of processors described above. Using remote direct memory access (RDMA) protocols, the same ones currently used on many network-based computing systems, the SHIP system essentially creates read/write drivers for different processors to use as necessary, and deploys these drivers on FPGA processors at the memory level, allowing the co-processors on a high-performance computing network to access data storage directly. This eliminates the need for CPUs to handle these transfer requests, and increases data transfer speed. Figure 5 shows an example implementation of this storage system (Vega, 2020).

Another major advancement in big data storage that occurred during this research period is computational storage drives, which attach FPGAs or other co-processors to a solid-state drive. This allows for certain operations to be performed at the memory level, with minimal data transfer distance. These co-processors can be reprogrammed to handle a number of different jobs as required for the project or data being processed, essentially allowing for computational acceleration directly at the storage level of the system architecture (Salamat et al, 2021). This method still allows a CPU-based system to offload computational work to a co-processor, and can lead to immense reductions in computing time for the right operations.

Solution:

Unsurprisingly, when you combine a field as complex as high-performance computing with the enormously complicated questions of physical science, it is difficult to find a single satisfying answer. However, if one were to begin designing a new high-performance computing system for NASA and its partners today, there are several useful tools that would allow a designer to make smart decisions and get the system as close to optimal for the greatest possible number of problems that may be tackled by NASA and its partners. Ultimately, these choices come down to a few key factors: computing power, memory access, and power consumption.

Computing power is definitely the hardest factor to generalize in this design process. As previously mentioned, GPUs have long been considered the ideal balance of flexibility and efficiency, but FPGAs have begun to show that they can keep up in certain scenarios. For example, binarized and recurrent neural networks have both been shown to run efficiently on FPGAs (Nurvitadhi et al., 2016), and even convolutional neural networks can be accelerated by FPGA systems, albeit at a slight loss of precision (Richter, 2021). Unfortunately, the many possible processes and applications of high-performance computing systems means that there is no singular clear answer for which processor to use in all cases. Overall, if one were designing a single high-performance computing system for use on multiple physical science problems, a combination of CPU, GPU, and FPGA processors may be optimal for reducing overall time-to-science on future projects, as each of these processors will be either faster at performing the necessary operations, able to consume less power for an acceptable loss in speed, or easier to design deep learning algorithms necessary for each individual project.

Memory access is perhaps the area with the most immediate available options for improvement. The centralized storage system described above is a huge step forward in high-performance computing, and the shared data structure allows for much faster GPU processing (Carriere, 2020). The SHIP architecture is another large advancement, which would allow for FPGAs to access data an order of magnitude faster than current methods (Vega, 2020). In general, some combination of these systems would likely be ideal for NASA and its partners, with computational storage like the “SmartSSDs” accessible by CPU, GPU, and FPGA systems alike through RDMA protocols, allowing all computing systems to access data concurrently and “pass off” specific computations and processes to the memory-level FPGAs.

Power consumption has also become a hotspot for high-performance computing research. As previously mentioned, this is another area where FPGAs have an advantage over traditional high-performance computing methods, with far better power efficiency than both CPUs and GPUs (Richter, 2020). Additionally, if FPGAs are not a viable option due to their lower computing power, there are tools to model and optimize

the power consumption of GPU systems, such as the NeuralPower framework (Marculescu et al, 2018). This is another area that is hard to design for on a general basis; however, a high-performance computing system with a combination of all three processors, as described above, would be able to provide the most options to reduce overall power consumption across multiple types of simulations and deployments.

However, if one is designing for a single specific project with a known algorithm design in mind, there are tools to more effectively determine the type of processor or processors that will be best suited for the job, and even allow the designer to optimize the ways in which these processors are configured. McPAT, a tool developed by Hewlett-Packard, is able to model CPU architectures with multiple different configurations, and offer feedback on speed, size, and power consumption of each configuration. Figure 6 shows the block diagram of this framework (Li et al, 2009). GPGPU-Sim, a tool based on NVIDIA's CUDA programming model, performs similar modeling and offers similar results for GPU-based computations (Bakhoda et al, 2009). Deep neural network accelerator designs can also be simulated using MIT's Timeloop infrastructure, again modeling a great deal of hardware options and finding ideal configurations. Figure 7 shows the tool flow of this framework (Prashar et al, 2019). Currently, FPGAs do not have an ideal method of simulation; however, they can be quickly prototyped and modified as necessary to find an optimal configuration.

Conclusion:

High-performance computing is an incredibly fast-moving field with a seemingly endless string of potential changes on the horizon, but there are improvements that can be made to our current computing systems right now. GPUs have long been implemented as accelerators for these complex computations, but they often wind up being underutilized and consume a great deal of power. Comparatively, FPGAs have begun to show promise as a low-power alternative, with better speeds per unit of power compared to GPUs in a large number of deep learning environments. FPGAs may also be a favorable co-processor in deep space computation systems, due to their higher resistance to radiation. Additionally, changes in data storage can allow for multiple computing clusters to access relevant sections of data simultaneously and without passing through multiple layers of drivers or other processors, further parallelizing the necessary computations.

Due to the level of familiarity and understanding researchers currently have with CPUs and GPUs, several tools exist to model these systems and optimize for speed and power consumption. Further research into hardware optimizations of these high-performance computing systems should focus on creating similar models for FPGA deployments, and analyzing the benefits of modified storage access, such as centralized storage systems or SHIP architecture, with multiple types of processors.

Furthermore, while this research was built on the tests and modeling done by other professionals in the field, testing the types of algorithms commonly used in physical science projects on a number of different hardware configuration models would allow a deeper investigation into the benefits of these proposed changes.

Overall, the biggest changes that could be made to the development of a high-performance computing system today are:

- Consider implementing FPGAs where they are currently viable to decrease power consumption
- Continue to implement centralized storage wherever possible, to minimize data transfer time between processors
- Consider the addition of SmartSSDs to allow co-processors direct memory access and/or perform specific calculations at the memory level

Future research into the hardware optimization of high-performance computing systems would benefit by examining:

- Simulation models to iterate and optimize FPGA hardware configuration for more high-performance computing problems
- Further development time with FPGAs to further optimize their use in high-performance computing systems
- The deployment and testing of physical science algorithms and code bases on these systems to verify their increases in speed, efficiency, and power consumption

References:

Bakhoda, A., Yuan, G. L., Fung, W. W. L., Wong, H., & Aamodt, T. M. (2009). Analyzing CUDA workloads using a detailed GPU simulator. *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, 163–174. IEEE Xplore. https://doi.org/10.1109/ispass.2009.4919648

Berggren, K., Xia, Q., Likharev, K. K., Strukov, D. B., Jiang, H., Mikolajick, T., Querlioz, D., Salinga, M., Erickson, J. R., Pi, S., Xiong, F., Lin, P., Li, C., Chen, Y., Xiong, S., Hoskins, B. D., Daniels, M. W., Madhavan, A., Liddle, J. A., & McClelland, J. J. (2020). Roadmap on emerging hardware and technology for machine learning. *Nanotechnology*, *32*(1), 012002. https://doi.org/10.1088/1361-6528/aba70f

Berten. (2016). GPU vs. FPGA performance comparison. In *Berten DSP S.L.* http://www.bertendsp.com/pdf/whitepaper/BWP001_GPU_vs_FPGA_Performance_Comparison_v1.0.pdf

Carriere, L. (2020, November 6). *NASA@SC20: Enabling AI/ML in an HPC environment*. Www.nas.nasa.gov; NASA. https://www.nas.nasa.gov/SC20/demos/demo21.html

Fryxell, B., Olson, K., Ricker, P., Timmes, F. X., Zingale, M., Lamb, D. Q., MacNeice, P., Rosner, R., Truran, J. W., & Tufo, H. (2000). FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, *131*(1), 273–334. https://doi.org/10.1086/317361

Galloy, M. (2013, June 11). *CPU vs. GPU performance*. https://michaelgalloy.com/2013/06/11/cpu-vs-gpu-performance.html

Jawandhiya, P. (2018). Hardware design for machine learning. *International Journal of Artificial Intelligence & Applications*, *9*(1), 63–84. https://doi.org/10.5121/ijaia.2018.9105

Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., & Jouppi, N. P. (2009). McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture - Micro-42*. https://doi.org/10.1145/1669112.1669172

Marculescu, D., Stamoulis, D., & Cai, E. (2018). Hardware-aware machine learning. *Proceedings of the International Conference on Computer-Aided Design*, 1–8. ACM Digital Library. https://doi.org/10.1145/3240765.3243479

Mignone, A., Bodo, G., Massaglia, S., Matsakos, T., Tesileanu, O., Zanni, C., & Ferrari, A. (2007). PLUTO: A numerical code for computational astrophysics. *The Astrophysical Journal Supplement Series*, *170*(1), 228–242. https://doi.org/10.1086/513316

Nurvitadhi, E., Sheffield, D., Sim, J., Mishra, A., Venkatesh, G., & Marr, D. (2016). Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. *2016 International Conference on Field-Programmable Technology (FPT)*, 77–84. IEEE Xplore. https://doi.org/10.1109/fpt.2016.7929192

Nurvitadhi, E., Sim, J., Sheffield, D., Mishra, A., Krishnan, S., & Marr, D. (2016). Accelerating recurrent neural networks in analytics servers: Comparison of FPGA, CPU, GPU, and ASIC. *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. https://doi.org/10.1109/fpl.2016.7577314

Parashar, A., Raina, P., Shao, Y. S., Chen, Y.-H., Ying, V. A., Mukkara, A., Venkatesan, R., Khailany, B., Keckler, S. W., & Emer, J. (2019). Timeloop: A systematic approach to DNN accelerator evaluation. *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 304–315. IEEE Xplore. https://doi.org/10.1109/ispass.2019.00042

Richter, H. (2021, February 19). *FPGA acceleration of convolutional neural networks*. Core Deep Learning; ASIC Design Services. https://register.gotowebinar.com/recording/8979723971769866243

Rupp, K. (2016, August 18). CPU, GPU and MIC hardware characteristics over time. . https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/

Salamat, S., Haj Aboutalebi, A., Khaleghi, B., Lee, J. H., Ki, Y. S., & Rosing, T. (2021). NASCENT: Near-storage acceleration of database sort on SmartSSD. *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. https://doi.org/10.1145/3431920.3439298

TOP500. (2020, November). *Development over time*. Top500.org; TOP500 Supercomputer Sites. https://www.top500.org/statistics/overtime/

Vega, J. C. (2020). *SHIP: A storage system for hybrid interconnected processors* [MAS Thesis].

Appendices:

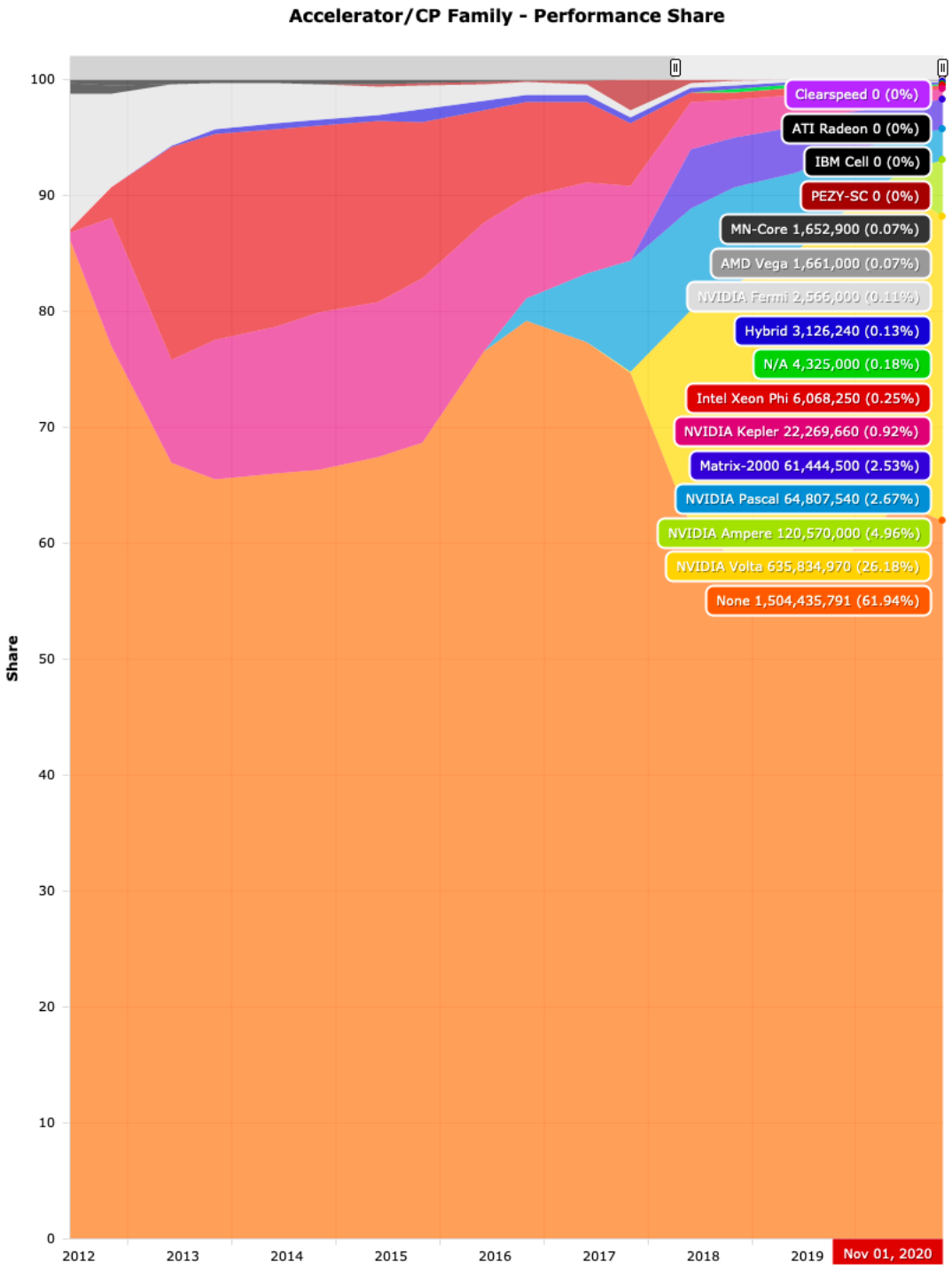

Figure 1: Share of co-processor computing performance in the top 500 fastest supercomputers over time; as we can see, accelerators account for over 38 percent of all performance in these computers as of November 2020. (Top500, 2020)

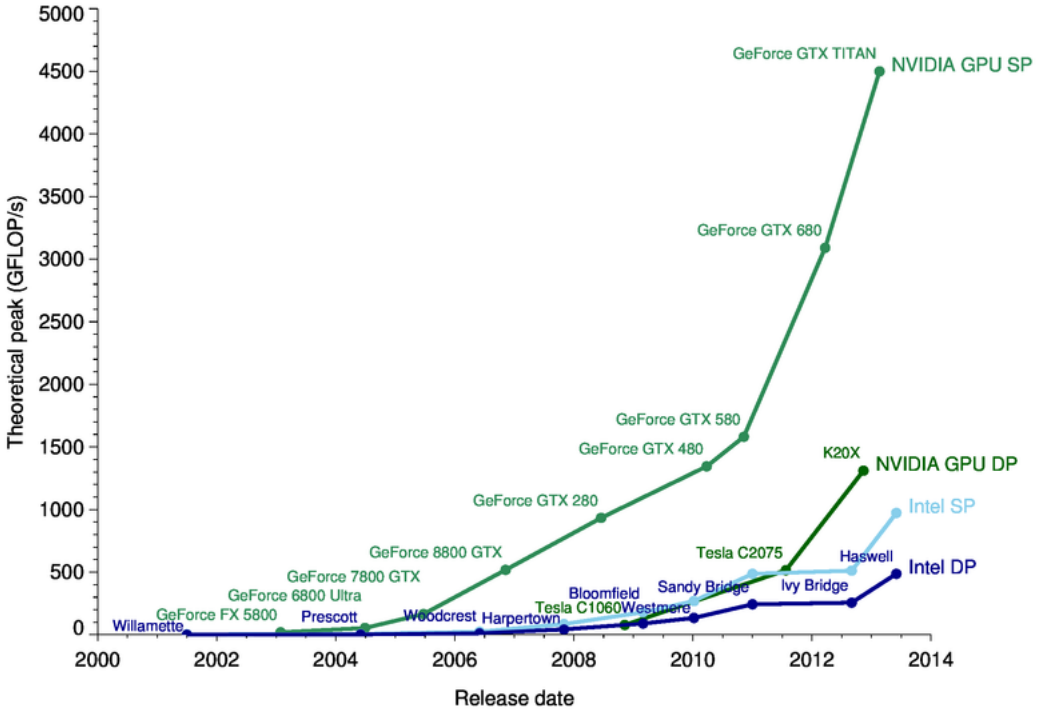

Figure 2: Comparison of optimal processing power for single- and double-precision floating point operations (FLOPS) in CPUs (blue) and GPUs (green) over time. (Galloy, 2013)

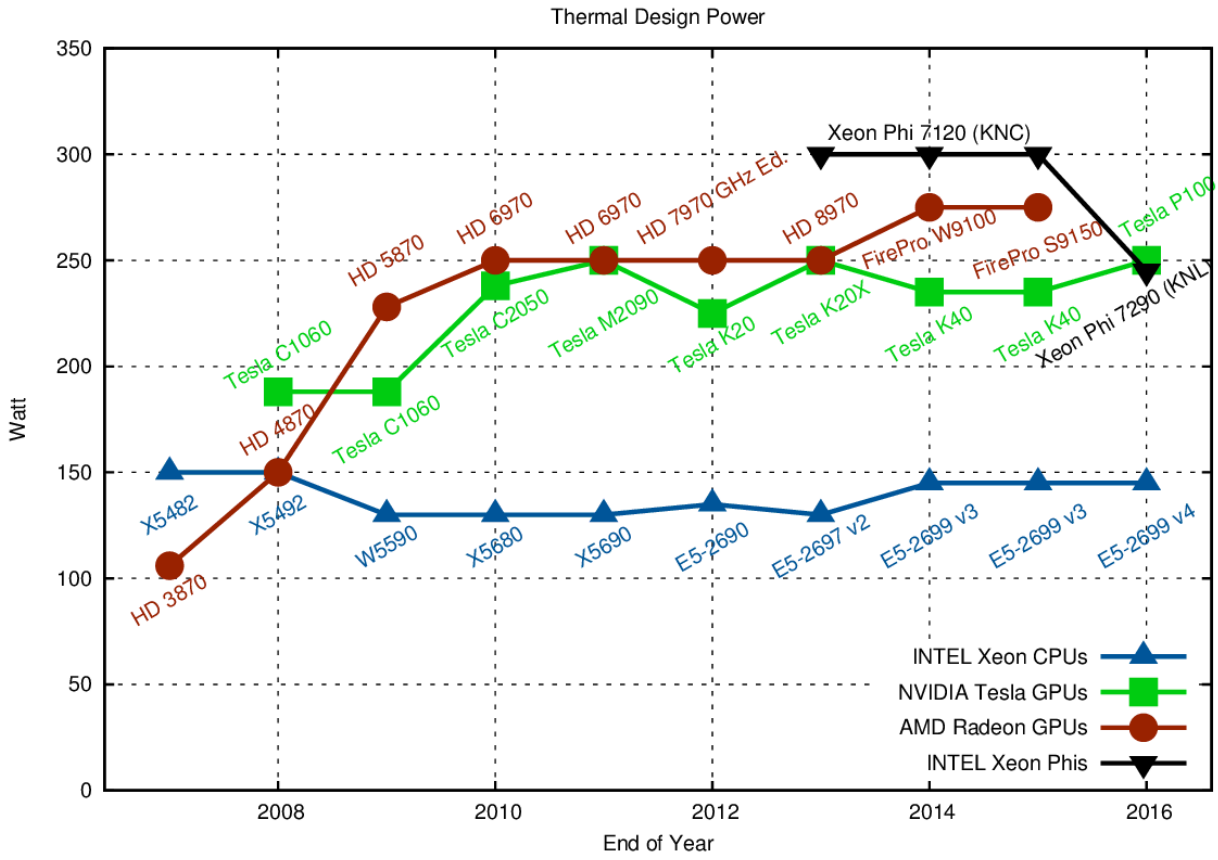

Figure 3: Comparison of power consumption between high-end CPUs and GPUs over time. The Xeon Phis seen in black are a kind of “hybrid model”, but generally behave similarly to GPUs in high-performance computing. (Rupp, 2016)

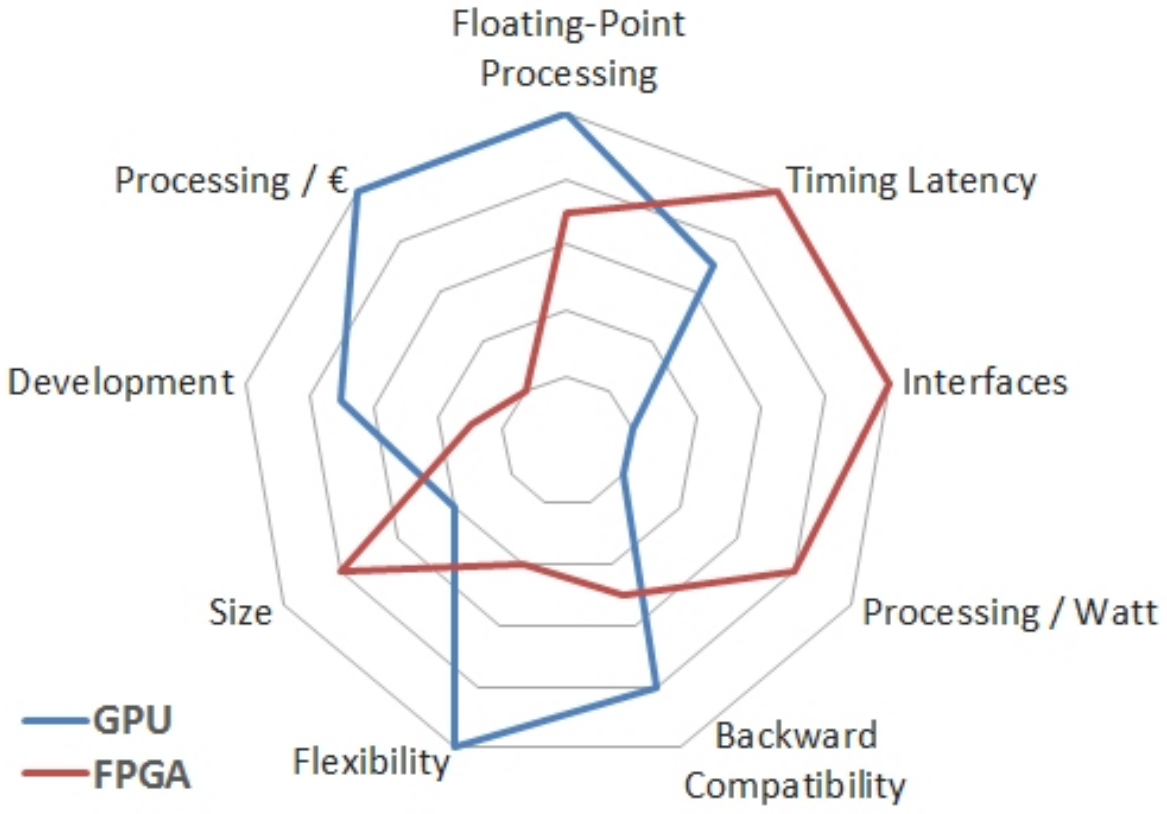

Figure 4: Comparison of GPU vs. FPGA accelerators in several categories. These primarily highlight GPUs' high processing power and lower cost, versus FPGAs' lower power consumption and ease of interfacing. (Berten, 2016)

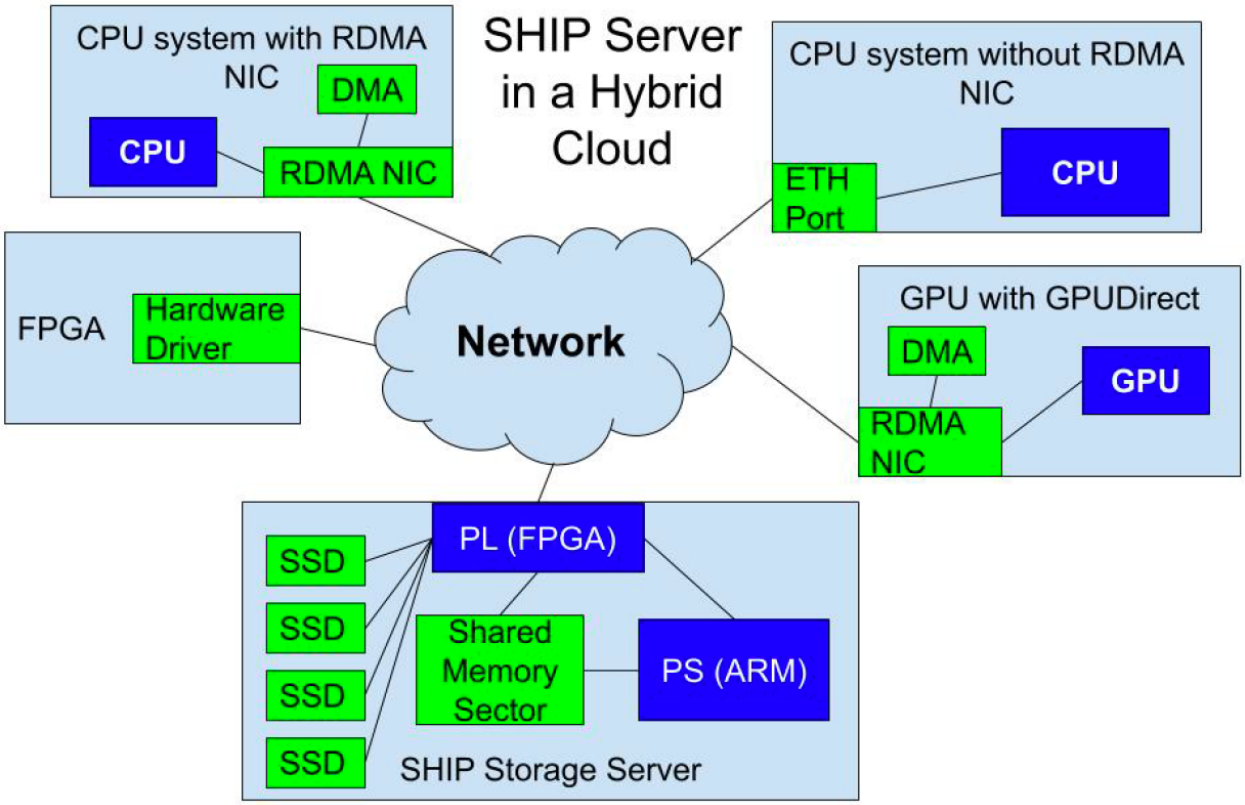

Figure 5: A basic example of SHIP memory architecture in a cloud network. Multiple different processor models can send RDMA requests to the server, which are handled via drivers at the memory level, eliminating the need for co-processors to interface with the CPU to access memory. (Vega, 2020)

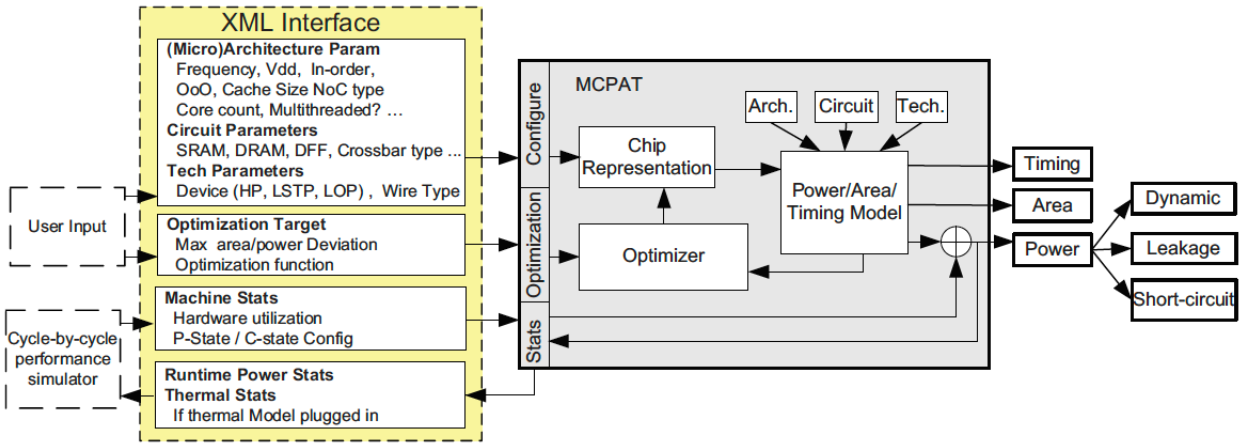

Figure 6: A block diagram of the McPAT framework used to optimize CPU parallelization. (Li et al, 2009)

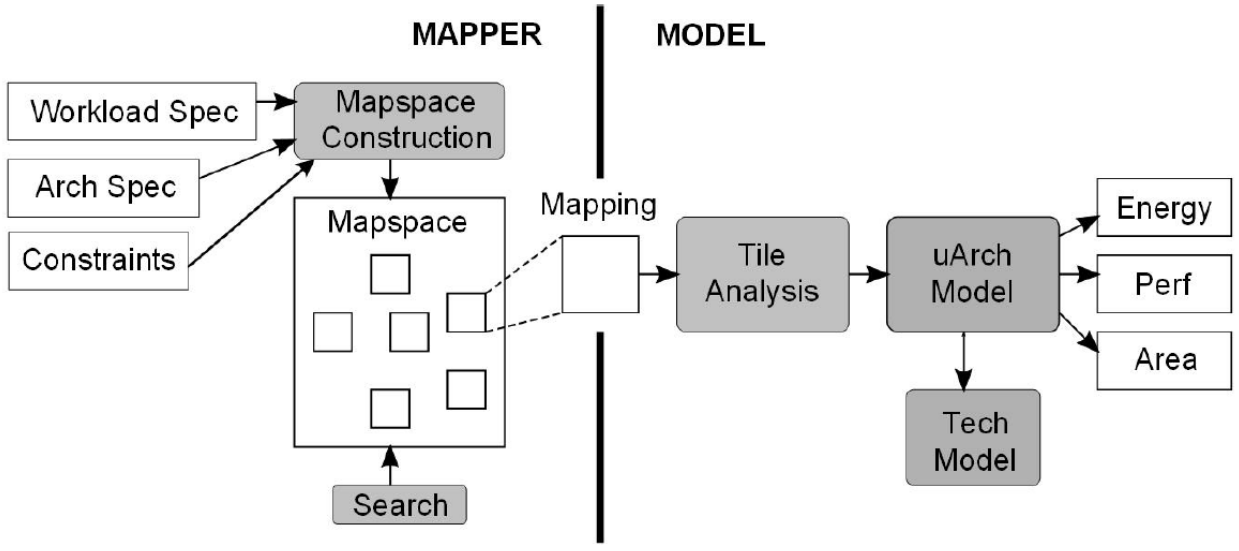

Figure 7: A diagram showing the tool flow of Timeloop's code framework, used to optimize the performance of neural networks on high-performance computing systems. (Parashar et al, 2019)