Accelerators

Steve Keckler, NVIDIA
Dejan Milojicic, Hewlett Packard Labs

Driven by artificial intelligence (AI), accelerators are taking away the "central" aspect of CPUs to become dominant processors of the vast amounts of generated data today. This transition happened in your phones and embedded devices a decade ago and, because of AI, is now taking place in servers and in the cloud.

To better understand the technical and business complexity of the disruption that accelerators are making in computer architecture and across industries, we welcome Steve Keckler, vice president of Architecture Research at NVIDIA, to this forum. Keckler has an architectural background including Massachusetts Institute of Technology's (MIT's) multithreaded M-Machine, the TRIPS multicore processor architecture at the University of Texas at Austin, and NVIDIA GPUs. We have asked Keckler a series of questions about the history of accelerators, their adoption, benefits and challenges, use cases, programming models, how they influence other architectural components, and about their future. We hope that you will enjoy Keckler's skillful analysis of the accelerator landscape and learn as much as we have.

DEJAN MILOJICIC: Accelerators have been gaining extraordinary adoption in the last few years across many industries. While this technology innovation is relatively recent, many concepts have been developed over decades. Could you point out some of the key turning points (technical or business) in the history of computing that have enabled recent accelerator success?

STEVE KECKLER: Accelerators have been around for a very long time. You may recall that floating-point hardware was not originally incorporated into mass market microprocessors until the late 1980s; instead, a system could include a floating-point coprocessor (also known as accelerator) in a separate chip. Likewise, early PCs had a separate chip

Digital Object Identifier 10.1109/MC.2021.3125157
Date of current version: 12 January 2022

0018-9162/22©2022IEEE

EDITOR **DEJAN MILOJICIC**
Hewlett Packard Labs;
dejan.milojicic@hpe.com

to accelerate audio processing, and of course graphics cards to accelerate 3D graphics and gaming have been popular since the early 1990s. Some of these accelerators ultimately were absorbed into a PC chipset, or their algorithms became fast enough executing in software on a CPU. In fact, domain-specific hardware in the 1980s and 1990s was fighting an uphill battle because Moore's law and Dennard scaling were driving rapid acceleration of CPU performance. Why build a special-purpose hardware accelerator when the CPU could do the job just fine in 18 months? Graphics was a notable outlier because the demand for higher performance by a very large gaming market far outstripped what could be incorporated into CPUs.

Now that CPU performance improvements are limited to a few percent per year, the computing ecosystem has sufficient motivation to develop special-purpose accelerators for domains that have sufficient critical mass, such as machine learning. The same influence of the end of Moore's law scaling is seen in systems on chip, which today include CPUs, GPUs, programmable digital signal processors, video processing accelerators, and neural network accelerators, as well as many other types of accelerators. Here the motivation is the greater energy efficiency on well-defined tasks that a domain-specific accelerator can provide.

The success of graphics accelerators for a wider range of applications came from two key factors. First was the desire for programmable hardware (rather than a fixed graphics pipeline) by graphics programmers, which motivated programmable shading architectures and led to the first GPU. Second was the overlapping demands of graphics and parallel computing algorithms for high compute throughput and memory bandwidth. The graphics market funded the R&D for powerful programmable parallel processors, giving general-purpose parallel computing processors (for example, GPU computing) time to mature. The parallel computing world is littered with companies that had good technology but died because they had an insufficient market to fund the necessary R&D.

I think it is also important to realize that the landscape of computing systems has changed. With the advent of generally programmable accelerators such as GPUs, these processors are now behaving as peers in a heterogeneous system rather than accelerators attached to CPUs. In accelerated data centers, far more transistor and dollar budgets are devoted to accelerators than to CPUs.

MILOJICIC: Artificial intelligence/machine learning/deep learning (AI/ML/DL) drives most recent adoption of accelerators. AI is known to have its own alternating summers and winters of popularity and adoption. If the next winter arrives, how will it affect accelerators' adoption? Will accelerators still be successful in other areas aside from AI/ML/DL, and what are these areas?

KECKLER: One of my AI colleagues really despises the characterization of AI of having summers and winters. He argues that AI has in fact delivered many successes over the years, but the goalposts keep getting moved with every success, resulting in the perception of the hype and failure cycle. I am sympathetic to this view. I would further argue that the confluence of relatively inexpensive high-performance computing (HPC) and readily available large data sets has fueled the latest surge in neural networks to the extent that the technology is now being used commercially to solve new problems or solve old problems faster and more cheaply across many industries. So, it is hard to imagine demand for data-driven automated learning to go away.

Even without ML, no one who has predicted that a plateau in the demand for computational performance has ever been right. I believe that society has an insatiable demand for computation, and without a new type of Moore's law there will be a need for domain-specific acceleration. At NVIDIA, we recently added hardware support to accelerate ray-tracing algorithms to produce even better images (and of course interactive and immersive visual effects). We have barely scratched the surface of the capabilities of virtual and augmented reality. Scientific discoveries that transform our world are driven by simulation and modeling made possible by HPC. While ML is being applied to these domains, if better algorithms or approaches emerge, there will be a demand for hardware to process them more quickly and more efficiently.

MILOJICIC: The wide proliferation of GPUs has resulted in many competitors attempting to replicate this success. As a result, we have ended up with a lot of heterogeneity. There are good sides of heterogeneity and challenging sides. Heterogeneity enables optimization of hardware to specific needs to provide more performance at less power. At the same time, it could be challenging to system designers, systems integrators, programmers, and users. How can we amplify benefits and minimize challenges of heterogeneity?

KECKLER: I believe that system designers and programmers should embrace heterogeneity, recognizing that different types of processors are suited to different types of tasks. For example, CPUs excel at single-threaded latency-sensitive code, while GPUs excel at parallel throughput-oriented code. Because applications consist of both types of code, a GPU-accelerated

program employs both types of processors for the tasks at which each are best. The key to exploiting this type of heterogeneity is a programming model that enables a programmer to express both the serial and parallel portions of an algorithm. While programming languages like CUDA and OpenCL help a programmer to specify explicitly the serial and parallel portions of a program, there is a lot of innovation in programming systems that employ higher levels of abstraction. Such systems enable a programmer to describe the algorithm and then employ compiler and/or runtime-system software to partition and schedule the work. MIT's Tensor Algebra Compiler is an example of such a system in the domain of sparse and dense tensor algebra. There are other examples in the domains of image processing and graph analytics.

MILOJICIC: Many use cases today are the result of the creation, collection, and processing of massive amounts of data. As a result, terms such as memory-driven architectures have been introduced. Accelerators have proven to be extremely efficient in computing of data. Can accelerators also be effective in serving data, for example, in data-driven operations such as data stores, databases, and file systems?

KECKLER: Accelerators can absolutely be designed to serve data. We already see the demand for this today with programmable processors connected directly to storage devices such as nonvolatile memory to perform tasks such as compression and encryption. Likewise, software-defined networking is driving greater compute capability into network controllers and ultimately switches. NVIDIA's BlueField line of networking products [data processing units (DPUs)] serve the high-performance networking space with capabilities for compression, data de-duplication, and security applications, among others. The BlueField-3 DPU includes up to 16 Arm cores, a programmable datapath accelerator, and hardware accelerators for encryption, compression, and network processing. Broadly speaking, providing acceleration throughout the system requires a decentralization of not just the hardware but also the software, and to date legacy operating system designs are holding back innovation because they are not ready for such decentralization. Reinventing operating system services in a decentralized manner to match the underlying system hardware architecture is going to be a huge challenge over the next decade.

MILOJICIC: How are accelerators influencing interconnects and memory designs? Traditionally, designs of interconnects were influenced by CPUs within coherency domains and then by performance requirements (bandwidth, latency, tail-latency) in larger domains. Are accelerator interconnect requirements any different than CPU requirements, and how? Similarly, hierarchies of caches and memory designs were influenced by CPUs. How is it different for accelerators?

KECKLER: The extremely high computational throughput of accelerators has demanded innovations in memory and interconnect designs. For example, GPUs have driven high-bandwidth memory through the generations of graphics double data rate dynamic random-access memory and more recently on-package high-bandwidth memory. Because traditional CPU networks were not suited to the bandwidth demands of GPUs, NVIDIA developed NVLink and NVSwitch architectures to provide a dedicated high-bandwidth network to connect many GPUs. Likewise, Google's TPU architecture deploys a custom network aimed to meet the needs of their applications and compute hardware. As GPUs have become more general purpose and deployed in scalable multi-GPU systems, a formal memory model has been developed to describe the memory semantics. The GPU memory consistency model is intentionally weak, allowing programmers the freedom to exploit massive parallelism without unnecessary synchronization. Other loosely coupled accelerators are likely to require a weak memory model for the same reason: to deliver systemwide performance. Because there is currently no systemwide memory model for GPUs and CPUs, let alone a system with a wider range of accelerators, there is an opportunity for innovative research to fill this gap.

MILOJICIC: Programming models have been instrumental in driving the design of CPU-based systems, for example, shared memory or message passing in HPC and traditional distributed systems. How are programming models influencing accelerators? Are there new features or even types of accelerators that may result in new programming models?

KECKLER: If we look narrowly at the ML accelerator domain, frameworks such as PyTorch, TensorFlow, Caffe, and ONNX were developed to provide portable programming interfaces to different types of hardware. These frameworks are effectively domain-specific languages for ML, which can then target CPUs, GPUs, or more specialized accelerators. More broadly, new programming models and languages such as CUDA and OpenCL were developed for general-purpose programmable accelerators such as GPUs. These languages have flourished because they provided a relatively simple method of expressing data parallelism so that it can be easily exploited by data parallel hardware.

I would argue that programming models and accelerators are best codesigned, which is true for both GPUs and ML accelerators and their programming models. In the future, I expect that we will see 1) a continued evolution of programming models for general-purpose accelerator architectures and 2) new programming models and mechanisms for emerging algorithmic domains. We will also see capabilities incorporated into existing programming models to better enable

programmers to exploit accelerators. One example of this is NVIDIA's Legate, which is a programming system that allows NumPy programs to seamlessly exploit scalable multiple-GPU systems by translating the NumPy application interface into the Legion distributed programming model.

MILOJICIC: The availability and maturity of tools are frequently essential for the adoption of technologies. Is the same true for accelerators, and can you provide some examples?

KECKLER: I would argue that ML accelerators, for example, have been catalyzed by ML frameworks. Without these tools, many more programmers would have spent many more hours working unnecessarily close to the metal. More general-purpose accelerators such as GPUs have full software stacks with compilers, profiling tools, debuggers, and a whole host of application acceleration libraries such as cuBLAS, cuSPARSE, and cuDNN. Without these types of tools, such accelerator systems would be far less attractive to program and far less popular with application developers. I think you can see this effect in the published MLPerf benchmark results, in which very few of the ML startups have submitted any data. Developing a software ecosystem for an accelerator is difficult and requires a substantial investment in both money and time. Only those companies that provide robust hardware and software will have their products accepted in the marketplace.

MILOJICIC: The security of CPUs and CPU-based systems has always been an important topic, and it is ever more so nowadays. Accelerators took a back seat in security in the past. As accelerators are increasingly becoming first-class processing elements and not just CPU-attached subsidiary processors, security will become equally important for accelerators. Is security any different for accelerators than for CPUs? A similar question can be applied to failure tolerance. For example, could CPU techniques for preventing propagation of memory failures across the system be applied to accelerators?

KECKLER: You draw an important distinction here between first-class programmable accelerators and subsidiary accelerators. The security requirements for the former are similar to those of CPUs, although the vulnerability surface may be different. For example, as accelerators generally do not employ speculative execution, they are not subject to the types of attacks exemplified by Spectre and Meltdown. Accelerators may also be able to be provisioned with more inherent isolation than a multitasking CPU. For example, NVIDIA's multi-instance GPU (MIG) subdivides a single GPU into multiple isolated sub-GPUs, each of which can be allocated to a different process. Nonetheless, the security aspects and requirements for confidentiality, integrity, and availability are effectively the same. The methods of meeting those requirements such as encryption, isolation, and attestation are generally similar across CPUs and first-class accelerators. The same logic applies to fault tolerance, where principles of error coding and redundancy are used in both CPUs and first-class accelerators. How CPUs and accelerators actually deploy these principles may differ due to the nature of the processors.

MILOJICIC: Over the decades, we have learned how to package and scale CPU-based systems. Is scaling of GPUs any different at the rack level, cluster level, or even larger scales?

KECKLER: To address this question, I would point to Selene, which is an NVIDIA supercomputer, currently number six on the Top 500 list of the most powerful supercomputers. Selene consists of 560 NVIDIA A100 DGX Servers (eight GPUs each) for a total of nearly 4,500 GPUs, connected using 850 Infiniband switches. Broadly speaking, Selene is similar in design to other rack-based supercomputers or datacenter computers, although its design is highly modular, facilitating fast system assembly. While GPUs consume more power than CPUs, they are more efficient at converting their power into performance. Selene is number 11 on the Green 500 list—and nine of the top 10 systems on the Green 500 list are powered by NVIDIA A100 GPUs. The result is the potential for higher power density in a datacenter and a potential demand for more power per rack. Looking forward, new technologies must be applied to skirt the end of semiconductor processor scaling, including packaging innovations such large multichip modules and tight integration of high-bandwidth and low-energy interconnect such as silicon photonics.

MILOJICIC: Standards and open source communities are an important factor in technology development and adoption. Are accelerators any different? There are some de facto standards, and then there are also entirely closed designs, such as those being developed by hyperscalers to sit behind a cloud delivery model. What is the right balance between openness and closed/proprietary designs?

KECKLER: I do not see a need for standardization of accelerator hardware. In fact, it is precisely the opportunity to innovate by tailoring the hardware to specific applications that delivers the performance and power efficiency of accelerators. Instead, standards are better applied at interfaces above the hardware. ML frameworks are a good example, as they allow a developer to employ the capabilities available in the framework, potentially coupled with a compilation and optimization process, to map a workflow to different ML accelerator architectures. Likewise, standard programming languages that incorporate mechanisms to express and exploit massive parallelism (such as OpenMP or the evolving C++ standards) enable programmers to employ programmable accelerator hardware. It will be incumbent on the

accelerator hardware developers to provide easy on-ramps for application developers. The wise ones will support a wide range of methods including support for many frameworks and programming languages.

MILOJICIC: Do you have any closing thoughts on the future of accelerators?

KECKLER: I am quite bullish on the future of accelerators in the computing ecosystem. I believe that, in general, we will see accelerators being tailored to more application domains. I expect that we will see such acceleration hardware manifest in a range of deployments including 1) tightly coupled instruction-based accelerators such as NVIDIA's Tensor Cores, 2) loosely coupled on-chip accelerators such as NVIDIA's ray-tracing and DL hardware, and 3) independent or peer accelerators in a heterogeneous system such as networked GPUs and CPUs in a data center. We will certainly see an expansion of hardware designs when new hot areas emerge (such as ML) and a consolidation when an area becomes overly saturated (as ML hardware accelerators are likely to experience). Regardless, the opportunities are immense but will require greater degrees of algorithm and hardware codesign than has typically been applied to traditional processor design. Furthermore, the greater proliferation of accelerators will naturally require them to become first-class citizens in a system with direct access to network and storage resources without requiring mediation by a CPU. CPUs will of course remain a vital component in a system, but we are likely to think of them as accelerators for single-threaded code rather than the center of the system.

STEVE KECKLER is vice president of Architecture Research at NVIDIA, Austin, Texas, 78717, USA. Contact him at skeckler@nvidia.com.

DEJAN MILOJICIC is a distinguished technologist at Hewlett Packard Labs, Palo Alto, California, 94306 USA. Contact him at dejan.milojicic@hpe.com.

Digital Object Identifier 10.1109/MC.2021.3134385