Hyperion: A Unified, Zero-CPU Data-Processing Unit (DPU)

Marco Spaziani Brunella
University of Rome Tor Vergata,
Axbryd

Marco Bonola
CNIT/Axbryd

Animesh Trivedi
Vrije Universiteit Amsterdam

ABSTRACT

Since the inception of computing, we have been reliant on CPU-powered architectures. However, today this reliance is challenged by manufacturing limitations (CMOS scaling), performance expectations (stalled clocks, Turing tax), and security concerns (microarchitectural attacks). To re-imagine our computing architecture, in this work we take a more radical but pragmatic approach and propose to eliminate the CPU with its design baggage, and integrate three primary pillars of computing, i.e., networking, storage, and computing, into a single unified CPU-free Data Processing Unit (DPU) called Hyperion. We present our vision to make the Hyperion DPU self-sufficient and self-hosting, hence not needing to attach it to any host server, thus making it a genuinely CPU-free DPU. We present our initial work-in-progress details and seek feedback from the SPMA community.

1 INTRODUCTION

As proclaimed in their 2018 Turing Award lecture by Hennessy and Patterson, we are in a New Golden Age for Computer Architecture [32], as evident from the emergence of accelerators and domain-specific computing devices in mainstream computing [15, 25, 37, 38, 47, 49, 62, 64, 65, 70, 75]. However, even in this Golden Age for domain-specific accelerators, the CPU[1] remains in the critical path to coordinate operations and manage data flow orchestration (data copying, I/O buffers management [54]), executing code for accelerator management (e.g. PCIe enumeration [69]), and translation between OS-level/POSIX abstractions [7] (network packets to application requests [34] to file names and offset [76], to device-level addresses [73]). In contrast to accelerators and I/O devices, the CPU performance is not expected to improve by a radical margin [55] (and is even dropping with the microarchitectural fixes [13, 44]). Consequently, the CPU remains in the critical path of end-to-end system building, thus not escaping the dynamics of Amdahl's Law [32]. We are not the first one to raise issues associated with the CPU-driven computing architecture [22, 55].

The first-principle reasoning suggests the solution: a system where there is no CPU, i.e., a zero-CPU or CPU-free architecture. A completely new computing architecture like zero-CPU will require a radical and destructive redesigning of computing hardware (buses, interconnects, controllers, DRAM, storage), systems software, and applications. A classic example of this approach is the MSR BEE3 system [23]. In this work, we take a more pragmatic approach and investigate the design of a self-contained, NIC-compute-storage *Data Processing Unit* (DPU) called Hyperion. Hyperion aims to establish end-to-end hardware control/data paths within the DPU without any CPU involvement. The unique design of Hyperion allows us to consider building a standalone, self-contained DPU, where no host system is needed to run it, thus reducing the cost of operation, packaging density, and energy requirements. This directly, network-attached FPGA model has been used before as well [62, 71, 77]. Figure 1 shows the overall architecture.

What	Examples
Network + Accelerator	SmartNICs [4, 61], AcclNet [26], hXDP [18]
Network + GPU	GPUDirect [56], GPUNet [42]
Storage + GPU	SPIN [14], GPUfs [72], GPUDirect [57], nvidia BAM [63]
Network + Storage	iSCSI, NVMoF (offload [66], BlueField [4]), i10 [35], ReFlex [43]
Storage + Accelerator	ASIC/CPU [31, 46, 70], GPUs [14, 15, 72], FPGA [36, 65, 68, 81], Hayagui [8]
Hybrid Systems	with ARM SoC [2, 24, 50], BEE3 [23], hybrid CPU-FPGA systems [19, 21]
DPUs	Hyperion (stand-alone), Fungible (MIPS64 R6 cores) DPU processor [27], Pensando (host-attached P4 Programmable processor) [59], BlueField (host-attached, with ARM cores) [4]

Table 1: Related work in the integration of network, storage, and accelerators (GPUs, FPGAs) devices.

2 THE DESIGN OF HYPERION

Commercially, NICs and storage devices (e.g. NVM Express) are available as separate PCIe devices. Communication between the two requires control coordination with P2P DMA from the CPU (if supported, e.g., NVMe CMB [12]) via the PCIe root complex, which typically resides on the CPU complex (keeping it in the loop). To make the DPU self-sufficient, Hyperion runs a PCIe root complex with an NVMe controller on the FPGA board and connects its PCIe lanes to off-the-shelf NVMe storage devices via a PCIe bifurcation. Hence, all access to the storage is funneled through the FPGA. With such a design, Hyperion now has an *end-to-end hardware path* from network to storage devices without involving the CPU. The end-to-end hardware path can be leveraged to optimize the network transport (TCP, UDP, RDMA, HOMA [58]), storage accesses (NVMoF, i10 [35], ReFlex [43], or customized interface like KV-SSDs [16]) with any arbitrary storage functions on the FPGA (compression, encryption, pointer chasing, deduplication, I/O scheduling, or application-defined codes). Closest to Hyperion's design is LeapIO [50], which integrates an ARM SoC with NVMe storage and RDMA NIC as a single DPU, which is still attached to a host x86 CPU.

Why FGPA: Two key factors drive the selection of FPGAs. First, the use of FPGAs has been shown to be highly energy and performance efficient (not necessarily for a serialized single flow execution) [18, 50, 65]. The primary challenge for managing FPGAs comes from carefully managing the pipelined execution of the

[1] referring to the CPU from the host (e.g. x86) as well as smart accelerators like ARM SoC.

SPMA '22, April 5th, 2022, Rennes, France

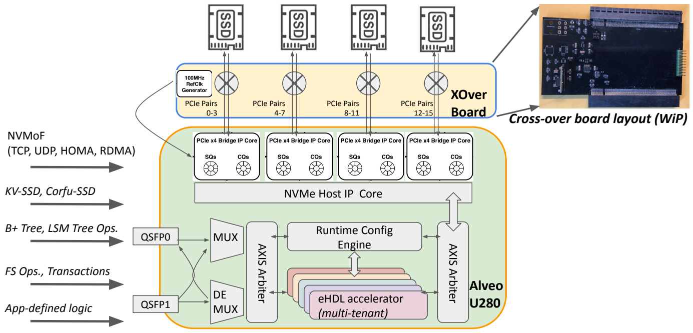

Figure 1: Hyperion architecture and layout.

workload. With the availability of high-quality DSLs [9, 40, 45, 67], OS-shells [47], and HDL compilers (hXDP [18]), it has become more affordable to generate a high-quality ISA for code execution at high data rates (100+ Gbps) [26, 51]. Second, the use of FPGA allows us to reconfigure hardware (deep pipelines, unrolled loops, heavy parallelism, large caches) to the best possible implementation of application-specific logic. Application-specific ASIC can also offer the same benefits but require high initial investment and manufacturing costs. Furthermore, as there is an increasing trend in compacting and packing thousands of processing elements in close vicinity (e.g., Cerebras core [1], Telsa Dojo [5]), the distance between the processing units (PUs) and memories (SRAM, DRAM, or HBM) is of critical importance. Here, an FPGA-based design offers the best tradeoff in packaging density and hardware reconfigurability.

Programming of FPGA: One of the primary challenges with FPGA is programming it. While the standard for FPGA programming has always been Hardware Description Languages, recently, there has been a series of efforts to raise the hardware programming abstractions [9, 40, 45, 67]. In this work, we make a case that the extended Berkeley Packet Filter (eBPF) language and toolchain [20, 53] can be used as the language of choice for the Hyperion DPU for three key reasons. First, despite it name, the eBPF instruction set is not tied to a specific application-domain and it has already been used in networking [33, 78], tracing [30], security [39], and storage [11, 17, 48, 80]. Second, due to the simplified nature of the eBPF instruction set, it is possible to verify the correctness and bounded execution of programs. The Linux kernel already ships with an eBPF verifier [74] (with simplified symbolic execution checks), and multiple other prototypes are available [6, 28]. Lastly, eBPF can support highly efficient code generation (via JITing) for multiple hardware devices such as x86, ARM, or FPGAs, potentially positioning it as the unified ISA for heterogenous computing [18, 41]. We propose to use an eBPF-supported programming language such as C that can be compiled to eBPF bytecode using the LLVM compiler, and then eBPF to VHDL using a novel compiler, built on top of hXDP [18]. The use of eBPF for more complex applications can be challenging [29], however, here we take a broader position regarding eBPF where the Linux kernel implement is one of many possible implementations of the BPF execution environment. For example, there are userspace BPF VMs [6], checkers [28], and application-specific ISA extensions [18]. Beyond the basic compilation of application-provided code to HDL, there are challenges associated with (i) secure multi-tenant execution [40, 79]; (ii) basic OS-level services on the FPGA [47]; (iii) FPGA configuration, management, accessibility of data-center resources [71], which we plan to tackle.

Application interfaces: The next big question in the design of Hyperion is the choice of the application interface. Here we take inspiration from Willow [70], which pioneered an RPC-powered, programmable SSD interface where a user provides application- and SSD-side RPC stubs. This design supports any desired combinations of network transport and storage interfaces. For example, we can build network-attached SSDs that can support Corfu-style consensus protocol [10], block-level NVMoF accesses, NFS acceleration, or bump-in-the-wire/near-data execution of application-provided codes (B+/LSM tree search and insertions, FS walks, transactions). We focus on three application classes for Hyperion. First, high volume applications such as fail2Ban [3], inspecting and writing network traffic and logs authentication/malicious data to attached SSDs. Such applications must handle high volumes of packet data under a tight time budget (100s of millions of packets/sec). Second, a latency-sensitive application such as network pointer-chasing. In a disaggregated storage, pointer chasing over B+ trees, extent trees, LSM trees (used in many databases, file systems, and key-value stores [60]) results in multiple network RTTs with significant performance degradation [48]. Lastly, network-attached SSDs that can export application-defined, high-level, fault-tolerant abstractions such as trees, lookup-tables, distributed/shared logs [10], and transactional interfaces (similar to Boxwood [52]).

3 STATUS & SPMA FEEDBACK

We are prototyping Hyperion with a Xilinx Alveo U280 board (which has 2x 100 Gbps QSFP onboard). We have designed a PCIe cross overboard to attach NVMe devices to the Alevo U280 board with power[2]. The board implementation is delayed due to the COVID-19 related component sourcing issues. Meanwhile, we have developed an XDP-compatible, eBPF B+ tree lookup implementation that can be run with the in-kernel Linux eBPF implementation. We are in the process of compiling it to U280, and completing the more complex tree insertion logic as well.

From the SPMA community, we seek feedback on issues such as: (i) Is network-attached Hyperion-style DPU a pragmatic choice for CPU-free designs? (ii) Can the additional complexity arisen from the elimination of the CPU be justified by the potential performance gains? (iii) Are FPGA-related toolchains ready to compile and run complex application-defined workloads in a secure, high-performance manner without significant manual effort? (iv) Looking beyond hardware, what kind of application-level interfaces/abstractions required for building *distributed* CPU-free applications that can be executed over multiple DPUs? (v) How to ensure secure, third-party, multi-tenant execution in the FPGA enclaves?

ACKNOWLEDGMENTS

This work is generously supported by the NWO grant number OCENW.XS3.030, Project Zero: Imagining a Brave CPU-free World!, and the Xilinx University Donation Program.

[2] All Hyperion artifacts, including HDL codes, will be open-sourced.

REFERENCES

[1] Cerebras Systems: Achieving Industry Best AI Performance Through A Systems Approach. https://f.hubspotusercontent30.net/hubfs/8968533/Cerebras-CS-2-Whitepaper.pdf. Accessed: 2022-Feb-02.

[2] DFC Open Source Community. https://github.com/DFC-OpenSource. Accessed: 2022-Feb-01.

[3] Fail2ban. https://www.fail2ban.org/wiki/index.php/Main_Page. Accessed: 2022-Feb-02.

[4] Mellanox BlueField SmartNIC for Ethernet. https://www.mellanox.com/files/doc-2020/pb-bluefield-smart-nic.pdf. Accessed: 2022-Feb-02.

[5] Tesla details Dojo supercomputer, reveals Dojo D1 chip and training tile module. https://www.datacenterdynamics.com/en/news/tesla-details-dojo-supercomputer-reveals-dojo-d1-chip-and-training-tile-module/. Accessed: 2022-Feb-02.

[6] Userspace ebpf vm, 2022. Accessed: 2022-Feb-02, https://github.com/iovisor/ubpf.

[7] Vaggelis Atlidakis, Jeremy Andrus, Roxana Geambasu, Dimitris Mitropoulos, and Jason Nieh. Posix abstractions in modern operating systems: The old, the new, and the missing. In *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, New York, NY, USA, 2016.

[8] Shinichi Awamoto, Erich Focht, and Michio Honda. Designing a storage software stack for accelerators. In *12th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 20)*, July 2020.

[9] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. Chisel: Constructing hardware in a scala embedded language. In *DAC Design Automation Conference 2012*, pages 1212–1221, 2012.

[10] Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, Ted Wobbler, Michael Wei, and John D. Davis. CORFU: A shared log design for flash clusters. In *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 1–14, San Jose, CA, April 2012.

[11] Antonio Barbalace, Martin Decky, Javier Picorel, and Pramod Bhatotia. Blockndp: Block-storage near data processing. In *Proceedings of the 21st International Middleware Conference Industrial Track*, Middleware '20, page 8–15, New York, NY, USA, 2020.

[12] Stephen Bates. Enabling the NVMe™ CMB and PMR Ecosystem. https://nvmexpress.org/wp-content/uploads/Session-2-Enabling-the-NVMe-CMB-and-PMR-Ecosystem-Eideticom-and-Mell....pdf. Accessed: 2022-Feb-02.

[13] Jonathan Behrens, Anton Cao, Cel Skeggs, Adam Belay, M. Frans Kaashoek, and Nickolai Zeldovich. *Efficiently Mitigating Transient Execution Attacks Using the Unmapped Speculation Contract*. USA, 2020.

[14] Shai Bergman, Tanya Brokhman, Tzachi Cohen, and Mark Silberstein. Spin: Seamless operating system integration of peer-to-peer dma between ssds and gpus. *ACM Trans. Comput. Syst.*, 36(2), apr 2019.

[15] Pramod Bhatotia, Rodrigo Rodrigues, and Akshat Verma. Shredder: Gpu-accelerated incremental storage and computation. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, page 14, USA, 2012.

[16] Janki Bhimani, Jingpei Yang, Ningfang Mi, Changho Choi, Manoj Saha, and Adnan Maruf. Fine-grained control of concurrency within kv-ssds. In *Proceedings of the 14th ACM International Conference on Systems and Storage*, SYSTOR '21, New York, NY, USA, 2021.

[17] Ashish Bijlani and Umakishore Ramachandran. Extension framework for file systems in user space. In *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, pages 121–134, 2019.

[18] Marco Spaziani Brunella, Giacomo Belocchi, Marco Bonola, Salvatore Pontarelli, Giuseppe Siracusano, Giuseppe Bianchi, Aniello Cammarano, Alessandro Palumbo, Luca Petrucci, and Roberto Bifulco. hXDP: Efficient software packet processing on FPGA NICs. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 973–990, November 2020.

[19] Young-Kyu Choi, Jason Cong, Zhenman Fang, Yuchen Hao, Glenn Reinman, and Peng Wei. In-depth analysis on microarchitectures of modern heterogeneous cpu-fpga platforms. *ACM Trans. Reconfigurable Technol. Syst.*, 12(1), feb 2019.

[20] Cilium. https://ebpf.io/. Accessed: 2022-Feb-02.

[21] David Cock, Abishek Ramdas, Daniel Schwyn, Michael Giardino, Adam Turowski, Zhenhao He, Nora Hossle, Dario Korolija, Melissa Licciardello, Kristina Martsenko, Reto Achermann, Gustavo Alonso, and Timothy Roscoe. Enzian: An open, general, cpu/fpga platform for systems software research. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 434–451, New York, NY, USA, 2022.

[22] William J. Dally, Yatish Turakhia, and Song Han. Domain-specific hardware accelerators. *Commun. ACM*, 63(7):48–57, jun 2020.

[23] John Davis, Chuck Thacker, and Chen Chang. Bee3: Revitalizing computer architecture research. Technical Report MSR-TR-2009-45, Microsoft, April 2009.

[24] Jaeyoung Do, Sudipta Sengupta, and Steven Swanson. Programmable Solid-state Storage in Future Cloud Datacenters. *Commun. ACM*, 62(6):54–62, May 2019.

[25] Haggai Eran, Lior Zeno, Maroun Tork, Gabi Malka, and Mark Silberstein. NICA: An infrastructure for inline acceleration of network applications. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 345–362, Renton, WA, July 2019.

[26] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. Azure accelerated networking: SmartNICs in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 51–66, Renton, WA, April 2018.

[27] Fungible. Fungible F1 Data Processing Unit. https://www.fungible.com/wp-content/uploads/2021/09/PB0028.02.12020914-Fungible-F1-Data-Processing-Unit.pdf. Accessed: 2022-Feb-02.

[28] Elazar Gershuni, Nadav Amit, Arie Gurfinkel, Nina Narodytska, Jorge A. Navas, Noam Rinetzky, Leonid Ryzhyk, and Mooly Sagiv. Simple and precise static analysis of untrusted linux kernel extensions. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 1069–1084, New York, NY, USA, 2019.

[29] Yoann Ghigoff, Julien Sopena, Kahina Lazri, Antoine Blin, and Gilles Muller. BMC: Accelerating memcached using safe in-kernel caching and pre-stack processing. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 487–501, April 2021.

[30] Brendan D. Gregg. Linux Enhanced BPF (eBPF) Tracing Tools. Accessed: 2022-Feb-02, http://www.brendangregg.com/ebpf.html.

[31] Boncheol Gu, Andre S. Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Moonsang Kwon, Chanho Yoon, Sangyeun Cho, Jaeheon Jeong, and Duckhyun Chang. Biscuit: A framework for near-data processing of big data workloads. In *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16, page 153–165. IEEE Press, 2016.

[32] John L. Hennessy and David A. Patterson. A New Golden Age for Computer Architecture. *Commun. ACM*, 62(2):48–60, January 2019.

[33] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller. The express data path: Fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '18, page 54–66, New York, NY, USA, 2018.

[34] Michio Honda. Packets as persistent in-memory data structures. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, HotNets '21, page 31–37, New York, NY, USA, 2021.

[35] Jaehyun Hwang, Qizhe Cai, Ao Tang, and Rachit Agarwal. TCP == RDMA: CPU-efficient remote storage access with i10. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 127–140, Santa Clara, CA, February 2020.

[36] Zsolt István, David Sidler, and Gustavo Alonso. Caribou: Intelligent distributed storage. *Proc. VLDB Endow.*, 10(11):1202–1213, aug 2017.

[37] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, page 1–12, New York, NY, USA, 2017.

[38] Anuj Kalia, Dong Zhou, Michael Kaminsky, and David G. Andersen. Raising the bar for using gpus in software packet processing. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 409–423, Oakland, CA, May 2015. USENIX Association.

[39] Michael Kerrisk. Using seccomp to limit the kernel attack surface. Linux Plumbers Conference, 2015. Accessed: 2022-Feb-02, https://man7.org/conf/lpc2015/limiting_kernel_attack_surface_with_seccomp-LPC_2015-Kerrisk.pdf.

[40] Ahmed Khawaja, Joshua Landgraf, Rohith Prakash, Michael Wei, Eric Schkufza, and Christopher J. Rossbach. Sharing, protection, and compatibility for reconfigurable fabric with AmorphOS. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 107–127, Carlsbad, CA, October 2018.

[41] Jakub Kicinski. Using ebpf as a heterogeneous processing abi. Linux Plumbers Conference, 2018. Accessed: 2022-Feb-02, http://vger.kernel.org/lpc_bpf2018_talks/Using_eBPF_as_a_heterogeneous_processing_ABI_LPC_2018.pdf.

[42] Sangman Kim, Seonggu Huh, Xinya Zhang, Yige Hu, Amir Wated, Emmett Witchel, and Mark Silberstein. GPUnet: Networking abstractions for GPU programs. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 201–216, Broomfield, CO, October 2014.

[43] Ana Klimovic, Heiner Litz, and Christos Kozyrakis. ReFlex: Remote Flash = Local Flash. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 345–359, Xi'an, China, 2017. ACM.

[44] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *Commun. ACM*, 63(7):93–101, jun 2020.

[45] David Koeplinger, Matthew Feldman, Raghu Prabhakar, Yaqi Zhang, Stefan Hadjis, Ruben Fiszel, Tian Zhao, Luigi Nardi, Ardavan Pedram, Christos Kozyrakis, and Kunle Olukotun. Spatial: A language and compiler for application accelerators. *SIGPLAN Not.*, 53(4):296–311, jun 2018.

[46] Gunjae Koo, Kiran Kumar Matam, Te I, H. V. Krishna Giri Narra, Jing Li, Hung-Wei Tseng, Steven Swanson, and Murali Annavaram. Summarizer: Trading communication with computing near storage. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, page 219–231, New York, NY, USA, 2017.

[47] Dario Korolija, Timothy Roscoe, and Gustavo Alonso. Do OS abstractions make sense on FPGAs? In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 991–1010, November 2020.

[48] Kornilios Kourtis, Animesh Trivedi, and Nikolas Ioannou. Safe and efficient remote application code execution on disaggregated NVM storage with ebpf. *CoRR*, abs/2002.11528, 2020.

[49] Dongup Kwon, Dongryeong Kim, Junehyuk Boo, Wonsik Lee, and Jangwoo Kim. A fast and flexible hardware-based virtualization mechanism for computational storage devices. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 729–743, July 2021.

[50] Huaicheng Li, Mingzhe Hao, Stanko Novakovic, Vaibhav Gogte, Sriram Govindan, Dan R. K. Ports, Irene Zhang, Ricardo Bianchini, Haryadi S. Gunawi, and Anirudh Badam. *LeapIO: Efficient and Portable Virtual NVMe Storage on ARM SoCs*, page 591–605. New York, NY, USA, 2020.

[51] Jiaxin Lin, Kiran Patel, Brent E. Stephens, Anirudh Sivaraman, and Aditya Akella. PANIC: A High-Performance programmable NIC for multi-tenant networks. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 243–259, November 2020.

[52] John MacCormick, Nick Murphy, Marc Najork, Chandramohan A. Thekkath, and Lidong Zhou. Boxwood: Abstractions as the foundation for storage infrastructure. In *6th Symposium on Operating Systems Design & Implementation (OSDI 04)*, San Francisco, CA, December 2004.

[53] Steven McCanne and Van Jacobson. The bsd packet filter: A new architecture for user-level packet capture. In *Proceedings of the USENIX Winter 1993 Conference Proceedings on USENIX Winter 1993 Conference Proceedings*, USENIX'93, page 2, USA, 1993.

[54] Ryo Nakamura, Yohei Kuga, and Kunio Akashi. How beneficial is peer-to-peer dma? In *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys '20, page 25–32, New York, NY, USA, 2020.

[55] Joel Nider and Alexandra (Sasha) Fedorova. The last cpu. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '21, page 1–8, New York, NY, USA, 2021.

[56] NVIDIA. Developing a Linux Kernel Module using GPUDirect RDMA. https://docs.nvidia.com/cuda/gpudirect-rdma/index.html. Accessed: 2022-Feb-02.

[57] NVIDIA. GPUDirect Storage: A Direct Path Between Storage and GPU Memory. https://developer.nvidia.com/blog/gpudirect-storage/. Accessed: 2022-Feb-02.

[58] John Ousterhout. A linux kernel implementation of the homa transport protocol. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 99–115, July 2021.

[59] Pensando. The Pensando Distributed Services Card (DSC). https://pensando.io/products/dsc/. Accessed: 2022-Feb-02.

[60] Alex Petrov. Algorithms behind modern storage systems: Different uses for read-optimized b-trees and write-optimized lsm-trees. *Queue*, 16(2):31–51, apr 2018.

[61] Boris Pismenny, Haggai Eran, Aviad Yehezkel, Liran Liss, Adam Morrison, and Dan Tsafrir. Autonomous nic offloads. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2021, page 18–35, New York, NY, USA, 2021.

[62] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. A Reconfigurable Fabric for Accelerating Large-scale Datacenter Services. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 13–24, Minneapolis, Minnesota, USA, 2014. IEEE Press.

[63] Zaid Qureshi, Vikram Sharma Mailthody, Isaac Gelado, Seung Won Min, Amna Masood, Jeongmin Park, Jinjun Xiong, CJ Newburn, Dmitri Vainbrand, I Chung, et al. Bam: A case for enabling fine-grain high throughput gpu-orchestrated access to storage. *arXiv preprint arXiv:2203.04910*, 2022.

[64] Christopher J. Rossbach, Jon Currey, Mark Silberstein, Baishakhi Ray, and Emmett Witchel. PTask: Operating System Abstractions to Manage GPUs As Compute Devices. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 233–248, Cascais, Portugal, 2011. ACM.

[65] Zhenyuan Ruan, Tong He, and Jason Cong. INSIDER: Designing In-Storage Computing System for Emerging High-Performance Drive. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 379–394, Renton, WA, 2019.

[66] Deboleena Sakalley. Using FPGAs to accelerate NVMe-oF based Storage Networks, 2022. Accessed: 2022-Feb-02, https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2017/20170810_FW32_Sakalley.pdf.

[67] Eric Schkufza, Michael Wei, and Christopher J. Rossbach. Just-in-time compilation for verilog: A new technique for improving the fpga programming experience. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 271–286, New York, NY, USA, 2019.

[68] Robert Schmid, Max Plauth, Lukas Wenzel, Felix Eberhardt, and Andreas Polze. Accessible near-storage computing with fpgas. In *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, New York, NY, USA, 2020.

[69] Adrian Schüpbach, Andrew Baumann, Timothy Roscoe, and Simon Peter. A declarative language approach to device configuration. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, page 119–132, New York, NY, USA, 2011.

[70] Sudharsan Seshadri, Mark Gahagan, Sundaram Bhaskaran, Trevor Bunker, Arup De, Yanqin Jin, Yang Liu, and Steven Swanson. Willow: A user-programmable ssd. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, page 67–80, USA, 2014.

[71] Ran Shu, Peng Cheng, Guo Chen, Zhiyuan Guo, Lei Qu, Yongqiang Xiong, Derek Chiou, and Thomas Moscibroda. Direct Universal Access: Making Data Center Resources Available to FPGA. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 127–140, Boston, MA, 2019.

[72] Mark Silberstein, Bryan Ford, Idit Keidar, and Emmett Witchel. Gpufs: Integrating a file system with gpus. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '13, page 485–498, New York, NY, USA, 2013.

[73] Theano Stavrinos, Daniel S. Berger, Ethan Katz-Bassett, and Wyatt Lloyd. Don't be a blockhead: Zoned namespaces make work on conventional ssds obsolete. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '21, page 144–151, New York, NY, USA, 2021.

[74] Daniel Thompson and Leo Yan. Kernel analysis using ebpf, 2018. Accessed: 2022-Feb-02, https://elinux.org/images/d/dc/Kernel-Analysis-Using-eBPF-Daniel-Thompson-Linaro.pdf.

[75] Maroun Tork, Lina Maudlej, and Mark Silberstein. Lynx: A smartnic-driven accelerator-centric architecture for network servers. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, page 117–131, New York, NY, USA, 2020.

[76] Animesh Trivedi, Nikolas Ioannou, Bernard Metzler, Patrick Stuedi, Jonas Pfefferle, Ioannis Koltsidas, Kornilios Kourtis, and Thomas R. Gross. Flashnet: Flash/network stack co-design. In *Proceedings of the 10th ACM International Systems and Storage Conference*, SYSTOR '17, New York, NY, USA, 2017.

[77] Jagath Weerasinghe, Raphael Polig, Francois Abel, and Christoph Hagleitner. Network-attached fpgas for data center applications. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 36–43, 2016.

[78] XDP: eXpress Data Path. https://www.iovisor.org/technology/xdp.

[79] Yue Zha and Jing Li. *Virtualizing FPGAs in the Cloud*, page 845–858. New York, NY, USA, 2020.

[80] Yuhong Zhong, Hongyi Wang, Yu Jian Wu, Asaf Cidon, Ryan Stutsman, Amy Tai, and Junfeng Yang. Bpf for storage: An exokernel-inspired approach. In *Proceedings of the Workshop on Hot Topics in Operating Systems*, HotOS '21, page 128–135, New York, NY, USA, 2021.

[81] Yu Zou and Mingjie Lin. FERMAT: fpga-accelerated heterogeneous computing platform near nvme storage. In *29th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2021, Orlando, FL, USA, May 9-12, 2021*, page 262. IEEE, 2021.