Hyperion: A Case for Unified, Self-Hosting, Zero-CPU Data-Processing Units (DPUs)

Marco Spaziani Brunella
University of Rome Tor Vergata, Axbryd

Marco Bonola
CNIT/Axbryd

Animesh Trivedi
VU, 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, self-hosting, unified CPU-free Data Processing Unit (DPU) called Hyperion. In this paper, we present the case for Hyperion, its design choices, initial work-in-progress details, and seek feedback from the systems community.

1 Introduction

Since the inception of computing, we have been designing and building computing systems around the CPU as the primary workhorse. This primary architecture has served us well. However, as the gains from Moore's and Dennard's scaling start to diminish, researchers have started to look beyond the CPU-centric designs to accelerators and domain-specific computing devices such as GPUs [26, 73, 115], FPGAs [84, 111], TPUs [72], programmable-storage [87, 116, 121], and SmartNICs [50, 128]. The use of domain-specific computing devices in wide-spread mainstream computing is heralded as the *Golden Age of Computer Architecture* by by Hennessy and Patterson in their 2017 Turning Award lecture [64].

However, even in this Golden Age, the CPU[1] remains in the critical path to manage data flows [113] (data copying, I/O buffers management [100]), accelerators (e.g. PCIe enumeration [120]), and translate between OS-level (packets, request, files) to device-level abstractions (address, locations) [14, 66, 125, 129]). Table 1 shows an overview of prior approaches (§4). Additionally, accelerator integration is always done (via virtualization or multiplexing) while keeping the CPU and accelerator view of systems resources (DRAM, memory mappings, TLBs) coherent and secure. Though necessary, such integration brings complexity to accelerator management and keeps the CPU as the final resource arbiter. In contrast to accelerators and I/O devices, the CPU performance is not expected to improve by a radical margin [101], and is even dropping with each microarchitectural attack fix [23, 81]. We are not the first one to raise issues associated with the CPU-driven computing architecture [42, 101]. Despite this awareness, CPU-driven designs and consequently, the CPU remains in the critical path of end-to-end system building, thus not escaping the dynamics of Amdahl's Law [64].

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 redesign of computing hardware (buses, interconnects, controllers,

What	Examples
Net + Accel	SmartNICs [5, 110], AcclNet [53], hXDP [35]
Net + GPU	GPUDirect [102], GPUNet [78]
Sto + GPU	Donard [22], SPIN [25], GPUfs [124], GPUDirect [103], nvidia BAM [113]
Net + Sto	iSCSI, NVMoF (offload [117], BlueField [5]), i10 [68], ReFlex [80]
Sto + Accel	ASIC/CPU [60, 83, 121], GPUs [25, 26, 124], FPGA [69, 116, 119, 143], Hayagui [15]
Hybrid System	with ARM SoC [3, 47, 90], BEE3 [44], hybrid CPU-FPGA systems [39, 41]
DPUs	Hyperion (stand-alone), Fungible (MIPS64 R6 cores) DPU processor [54], Pensando (host-attached P4 Programmable processor) [108], BlueField (host-attached, with ARM cores) [5]

Table 1: Related work (§4) in the integration of network (net), storage (sto), and accelerators (accel) devices.

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

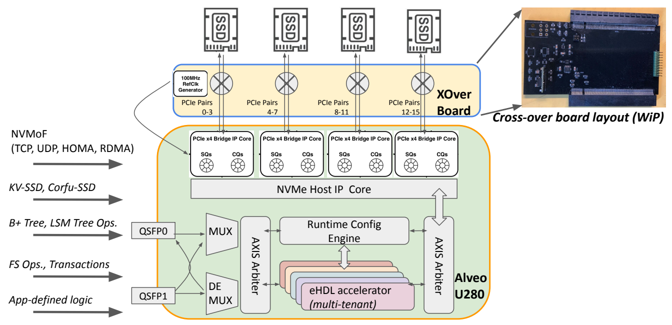

Figure 1: Hyperion architecture and layout.

DRAM, storage), systems software, and applications. A prior example of this approach is the MSR BEE3 system used for emulations [44]. A recent example is ETH's Enzian system that designs a hybrid CPU-FPGA dual socket system [41]. The Enzian paper documents the heroic engineering effort it took to design such a system where all board components need to be re-designed to integrate an FPGA as a co-processor *with a CPU*. Furthermore, such CPU-centric thinking encourages us to inherit and integrate CPU-centric hardware and software choices for an accelerator-centric design without reassessing if such choices make sense and/or can be simplified (see §2).

In this work, we take a more pragmatic approach and investigate the design of a unified NIC-FPGA-storage *Data Processing Unit* (DPU) called Hyperion (Figure 1). 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 energy cost and increasing packaging density. This directly, network-attached FPGA model has been used before as well [69, 111, 123, 133]. In this paper, we present a case for such a stand-alone DPU but without a CPU (§2), and present design choices pertaining to hardware integration (§3.1), systems software (§3.2), and client-interface and workload (§3.3).

2 The case for a CPU-free DPU

The CPU-driven design has its clear merits, and its elimination is *not recommended* for every workload in general computing. However, for specialized data center workloads (data-parallel, accelerator-amenable, disaggregated), the usability of the CPU must be reassessed. There are three primary impetuses that encourage us to think about a CPU-free DPU:

First, the era of one-CPU-fits-all is over (design, manufacturing, and thermal limits [34, 51, 63]) and the way forward is *specialization* with reconfigurable hardware and accelerators. The generality of the CPU has overheads (i.e., Turing Tax) that hinder specialization for performance or efficiency. For example, calculations for the Smith Waterman algorithm in DNA sequence alignment takes 37 cycles with 40 instructions (35 arithmetic, 15 load/store) with 81 nanoJoules of energy (on a 14nm CPU). In comparison, this calculation on a specialized 40nm ASIC takes a single cycle instruction with 3.1 picoJoules of energy [131]. The generality and over-engineered design of CPUs for any workload also results in poor on-chip resource utilization [52], unused silicon [51, 63], and elevated security risks [81]. At the same time, with the availability of open-source EDA processes and projects [7, 8], exploring workload-specialized hardware designs (with or without CPU) has become more approachable and affordable.

Second, a direct consequence of keeping a CPU-driven design is to inherit its choices of memory addressing, translation, and protection mechanisms such as virtual memory, paging, and segmentation [45]. When an accelerator such as FPGA, is attached to a CPU as an external device [39] or as a co-processor [41], there is a temptation to provide/port the familiar memory abstractions like unified virtual memory [84] and/or shared memory [94]. This design necessitates a complex integration with further CPU-attached memory abstractions such as page tables and TLBs, virtualization, huge pages, IOMMU, etc., while keeping such an integration coherent with the CPU view of the system [84, 94]. Furthermore, the management of physical memory (or the illusion of a flat, uniform physical address space) on modern computing platforms with accelerators and heterogeneous CPUs is a non-trivial and complex job [10]. Hence, in this work we argue that eschewing CPU and its design baggage, we can explore new memory management designs such as compiler/language-assisted solutions even directly on physical addresses [126].

Lastly, the CPU-centric design encourages the *active* resources disaggregation where resources remain attached to a host CPU that manages the disaggregation logic. This design results in a coarser disaggregation granularity with complex and bloated software [56] and a tight integration of processor/memory [61, 122]. To achieve the vision painted by Han et al. in their seminal HotNet'13 paper [62], there is a renewed push for *passive* disaggregation where disaggregation logic/smartness lies with clients, and a remote resource only serves requests as fast as possible [12, 36, 61, 122, 130]. Passive disaggregation promotes a network-attached model, where memory, storage, DPUs, and ASICs are directly connected to a network, and offers a better match with fine-grained computing models like Serverless [37, 107]. It also enables systems designers to rethink (i) network protocols for discovery and configuration protocols (e.g., Catapult fabric [111]); (ii) work division between clients and remote servers for distributed resource allocation, and access (e.g., Clio [61], DUA [123]); and (iii) offload-friendly abstractions with isolation, multiplexing mechanisms (e.g., group offloading and memory reassignments [12, 77, 93]).

To summarize: In this work, we make a case for the elimination of the CPU and its design baggage, and argue that its elimination can bring substantially simplicity and offer performance/energy advantages. Our attempt to design and implement Hyperion is a step in this direction.

3 Hyperion

Hyperion is a standalone, network-attached DPU that unifies 100 Gbps Ethernet NIC, FPGA, and NVMe storage devices in a single DPU. Figure 1 shows the overall architecture of Hyperion with the FPGA board, and attached NVMe SSDs.

3.1 Hardware Design

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 Controller Memory Buffers (CMBs) [21]) 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, which is connected to a 100 Gbps network directly. The FPGA PCIe lanes are connected (x16) 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 FPGA to storage devices. The end-to-end hardware path can be *specialized* to a workload with an optimized network transport (TCP, UDP, RDMA, HOMA [104]), storage API (NVMoF [117], i10 [68], ReFlex [80], KV-SSDs [27]) with any arbitrary storage functions on the FPGA (compression, pointer chasing, deduplication, or application-defined codes).

Why FPGA? Three factors drive the selection of FPGA:

1. **Application-specific reconfigurability:** The use of FPGA allows us to reconfigure hardware (deep pipelines, unrolled loops, data parallelism, large caches) to the best possible implementation of an application-specific logic. ASICs offer similar benefits, but require high initial investment and manufacturing costs. Furthermore, as there is an increasing trend to pack thousands of workload-specific processing units (PU) in a close vicinity (e.g., Cerebras [2], Telsa Dojo [6]), the distance among PUs and memories (SRAM, DRAM, or HBM) is of critical importance. Here, we believe that an FPGA-based design offers the best tradeoffs.

2. **Improved FPGA systems software support:** The primary challenge for managing FPGAs comes from carefully managing the pipelined execution of the workload with Hardware Description Languages (HDLs). With the availability of high-quality DSLs [18, 75, 82, 118], OS-shells [84], and HDL compilers (hXDP [35]), it has become more affordable to generate a high-quality HDL for high data rates (100+ Gbps) [53, 92]. Overall FPGA compilation and debugging processes have also improved [95, 136].

3. **Predictable performance with energy efficiency:** Unlike the CPU and I/O devices that target fine-grained time-based statistical multiplexing (μsec to nsec) to maximize resource utilization, FPGAs target a much coarser time-scale (10-100s milliseconds), or even spatial multiplexing which commits resources to a tenant. This sharing model helps with building highly *predictable execution pipelines* where once an associated bitstream has been sent to the FPGA, the circuit runs a certain clock frequency without any outside interference [70,89]. The use of FPGAs has been shown to be energy efficient [35, 112, 116] as its energy consumption is proportional to the *active and used* programmable LUTs and the operating frequency. Unused logic elements do not consume any energy, resulting in deployments which consume 10-20 Watts, which is an order of magnitude less than a server-grade machine [70].

Apart from the choice of FPGA, Hyperion uses NVM Express (NVMe) for block SSDs, Ethernet for network, and the PCIe between FPGA and SSDs. These choices are dictated by practicality and the engineering efforts required. For example, the choice of PCIe over other high-performance local interconnects (CXL, CAPI) or networks (TrueFabric [55]), can be revised as workload demands increase.

3.2 Software and Programming

Due to the absence of the CPU and conventional operating system, doing the classical resource management with elevated privileges to mediate accesses to a shared resource in Hyperion would be challenging. Hence, we must re-negotiate the work division among hardware, compiler, and application with the compiler taking a leading role. The role of compiler is not unusual here. It has been shown that compiler-assisted designs can help with the traditional OS roles such as for context switching [48, 88, 97], memory virtualization [126], single-level memory/storage [30, 67], extraction of parallelism [35], virtualization and multi-tenancy [75, 138].

With this compiler-centric approach we run the risk of repeating the failure of the VLIW processors[2]. However, we argue that there are two fundamental shifts that work in our favor. First, domain/workload-specific architectures are common, and associated languages (e.g., OpenCL, Chisel [18]) and compilers are used extensively as the norm. There are significant research and commercial interests in co-designing domain- or workload-specific hardware/software. Second, unlike VLIW processors, a DPU (specifically FPGA driven) is

[2]VLIW compilers were left responsible for parallelism extraction in general workloads, which lead Donald Knuth to comment that *"... the "Itanium" approach that was supposed to be so terrific—until it turned out that the wished-for compilers were basically impossible to write"* [29].

not aimed to deliver performance for all/any workload, hence, restricting the optimization design space. For example, hXDP has demonstrated that compile-time heuristics (the Bernstein conditions) with a simple language (eBPF) can lend itself to automatic parallelism extraction for packet processing workloads with a VLIW softcore processor [35].

Inspired by the LLVM project, in this work we argue that FPGA programming needs to decouple the frontend (application logic) and backend (HDL codes) with an accelerator-independent, intermediate representation (IR) language. The IR can be used to reason about correctness and safety properties of the program, with compiler-assisted transformations for pointer swizzling and privilege calls. We make a case that the extended Berkeley Packet Filter (eBPF) [40, 99] language is a suitable match for such an IR for three key reasons. First, eBPF is not tied to a specific application-domain and it is used in networking [65, 135], tracing [59], caching [58], security [74], and storage [20, 28, 85, 141]. It is also supported by healthy, growing communities (Cilium, the ebpf foundation), thus, establishing expertise and a knowledge base. Second, due to the simplified nature of the eBPF instruction set, it is possible to verify and reason about its execution. The Linux kernel already ships with an eBPF verifier [127] (with simplified symbolic execution checks). Lastly, eBPF supports efficient code generation (via JITing) for multiple hardware devices such as x86, ARM, or FPGAs, thus solidifying its position as an accelerator-independent, unifying IR for heterogenous computing [76]. Bear in mind, here we take a broader position regarding eBPF where the Linux kernel implement is one of many possible implementation of an eBPF execution environment. For example, there are userspace BPF VMs [9], checkers [57], and application-specific ISA extensions [35]. Apart from eBPF, we also consider P4, another popular programming language for in-network acceleration (NICs and switches). However, P4 programs are designed around packet processing and network abstractions. In restricted capabilities (with only filtering and forwarding) there are P4 to eBPF compilers available, though the generality of P4 for general data processing is yet to be explored.

Hyperion supports any eBPF-supporting programming language as a frontend. It then uses `clang/LLVM` to generate eBPF IR from the frontend. The eBPF IR is then passed to a two stage compilation process. In the first stage, the eBPF IR is passed through the open-source hXDP compiler for parallelism extraction and optimized VLIW transformations [17, 35]. In the second stage, the optimized eBPF IR is passed through an eBPF-to-HDL compiler for the final HDL code generation. Unlike hXDP, Hyperion runs HDL codes directly, not as a VLIW softcore processor on the FPGA.

Beyond the basic compilation of application-provided code to HDL, there are challenges associated with (i) secure multi-tenant execution; and (ii) FPGA configuration, management, accessibility of data-center resources [123]. Many past design choices here can be simplified as there are no host system resources (on the CPU or OS) that need to be kept coherent and secure while doing execution in the FPGA. We propose to leverage the slot-style slicing of FPGA resources [75] with a compiler to do workload partitioning [138]. Hyperion runs a configuration kernel that can receive authorized FPGA bitstream over the network and assign slices to it.

3.3 Client Interface and Workloads

To provide a client interface that can be *specalized*, Hyperion takes inspiration from Willow [121], which pioneered an RPC-backed programmable SSD interface where a user provides application- and SSD-side RPC stubs. Such a flexible design can support any desired specialization of both network as well as storage interfaces. For example, we can build network-attached SSDs that can support Corfu consensus protocol [19, 134], block-level NVMoF accesses, NFS acceleration, or the bump-in-the-wire/near-data execution of application-provided codes (B+/LSM tree search, compaction and insertions, file system walks, transactions) [116, 139]. Here, we can leverage client-driven request routing [91] with a shared-nothing, run-to-completion datapath [24] for performance.

We focus on three application classes for Hyperion. First, high volume applications such as fail2Ban [4], 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 [109]) results in multiple network RTTs with significant performance degradation [85]. Lastly, network-attached SSDs that can export application-defined, high-level, fault-tolerant abstractions such as trees, lookup-tables [27], distributed/shared logs [19, 134], atomic writes [105], concurrent appends [31], caches [58], and concurrent data structures and transactional interfaces (similar to Boxwood [96]).

One primary challenge here is the composability of multiple functionalities and the state management on FPGA during processing. Often storage integration with FPGA is done at the block-level for state-less data processing on data streams (such as grep). Hence, appropriate APIs and abstractions are needed to integrate high(er)-level storage abstractions with efficient state management on FPGA/BPF such as file systems [28, 116, 119], file/data format integration [86, 106], data caching [58], OoS scheduling (priority sharing of storage/network resources), checkpointing, deduplication, encryption, etc. We are in the process of building such modules as shared libraries for FPGA codes.

3.4 Current Status

We are prototyping Hyperion with a Xilinx Alveo U280 board which has 2x100 Gbps QSFP [1]. We have designed a PCIe cross overboard [46] to attach 4x NVMe devices to the U280 with power[3].

The current system boots in a *stand-alone mode* without any CPU when power is applied and FPGA JTAG self-tests are passed. The board is currently attached to a host-system via USB for programming, however, we are in the process of developing an OS-shell and control path over the network that can program the FPGA completely independently as well, leveraging Partial Dynamic Reconfiguration through the Internal Configuration Access Port (ICAP) of the FPGA. We have chosen to use a B+ tree key-value store as one of the first applications for Hyperion. We have written an XDP-compatible B+ tree that runs on the in-kernel XDP path (in-memory). On Hyperion, the tree will store all its data on NVMe devices directly, and will serve get/put/delete requests over the network.

Raw latencies of our hardware are: L2 network RTT ~$1\mu s$, NVMe latency is $[5-8]\mu s$. Currently we do not do any caching, hence, all tree access results in an access to the storage device. With this setup, the average (expected) lookup latencies are: $O(1+(tree\_height \times 8)\mu sec))$. From the past experience with network packet processing pipelines, we expect Hyperion to support 1 million lookup operations/sec, although the peak performance depends on how many PCIe lanes, NVMe devices, and FPGA kernels are running in parallel. In our current compilation process, the B+ tree implementation generates 1000+ pipelines stages. This is one of the largest designs we have tested with our toolchain, which challenges the resource availability on the FPGA. Although we are confident that even this unoptimized B+ tree implementation can fit on the FPGA and there is plenty of room for optimization to achieve real multi-tenancy.

See Appendix-A after the references for images.

4 Related Work

Nider and Fedorova also question of the utility of "the Last CPU" in the system and investigate the design of a *system management bus* to take over the OS/CPU responsibilities [101]. Table 1 shows efforts for *pair-wise device* interactions such as GPU-with-storage [22, 25, 26, 113, 124], GPU-with-network [43, 78, 102], accelerator-to/from-storage [13, 15, 16, 90], SmartNICs [50, 110, 128], and networked storage accesses [79, 117]. FPGA are explored with (1) networks [35, 53, 132, 142]; and (2) storage [116, 119, 121]. BPF offloading to NIC/FPGA for processing are done with Endance DAG cards [49], Netronome [71], Combo6 [98], but mostly limited to monitoring and traffic shaping. FPGA-assisted KV stores have considered a close integration of network and KV processing (in-memory) [32, 38, 69, 89] and selective integration of NAND flash (e.g., BlueDB and Xilinx-KV [33, 137]). One of the closest design inspirations to Hyperion is LeapIO [90] that integrates NVMe flash with RDMA NIC and ARM SoC. Hyperion and LeapIO share the similar motivations (cost, energy, and performance efficiency), however, Hyperion could eschew much of design complexity of LeapIO (interaction of host x86 CPU and ARM SoC). Hyperion targets a broader design space, where we consider unification of reconfigurable hardware (here FPGA), network transport (100 Gbps Ethernet) and storage (NVMe flash). This unification offers multiple hardware/software specializations to support multiple workload needs.

5 Discussion and Feedback

Hyperion is still in its early prototyping phase. From the systems community, we seek feedback on issues like:
(1) Is eliminating the CPU a worthy pursuit? In this paper we made a controversial case for removing the CPU, and we believe that with the recent hardware and software advancements it is the right time to re-evaluate the role of the CPU and the design baggage that it brings. However, we are interested in hearing counter-arguments. We understand that beyond technology, operational costs and complexities might put limits to the realization of this idea. At what levels of performance, energy, and packaging efficiency gains from a CPU-free design will be worth it? The elimination of the CPU-side mediation also necessitates a bigger supporting role from the FPGA toolchains, languages, and compilers, a role which was previously split between the host CPU and OS. Are FPGA toolchains ready?
(2) What is the right client-interface to build *distributed* Hyperion applications? Looking beyond hardware and a single DPU, what kind of application-level interfaces/abstractions are required for building *distributed* CPU-free applications that can be executed over multiple DPUs? A passive resource disaggregation puts the responsibility of control coordination on the client-side. Multiple clients either have to coordinate themselves or use an external service [11, 70]. However, in order to realize the full potential of Hyperion, applications should also reduce the client-side CPU/OS involvement (e.g., use RDMA or DPDK) while interacting with Hyperion. How should one build distributed applications and composable service ecosystems of such standalone, passively disaggregated DPUs?
(3) Operational complexity in multi-tenant clouds? In datacenters, hardware and software fail. Tenants are untrusted. The costs of inefficiency and downtime are high. Hence, how to ensure that Hyperion can offer secure, multi-tenant execution in FPGAs [140]? How to reduce microarchitectural attacks with Hyperion? Can or should micro-architectural resources of Hyperion be managed explicitly with tenants to ensure sufficient isolation with Hyperion DPUs [114]?

[3] All Hyperion artifacts (compilers, board design) will be open-sourced.

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.

References

[1] Alveo U280 Data Center Accelerator Card. https://www.xilinx.com/products/boards-and-kits/alveo/u280.html. Accessed: 2022-Feb-20.

[2] 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.

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

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

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

[6] 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.

[7] CHIPS (Common Hardware for Interfaces, Processors and Systems) Alliance. https://chipsalliance.org/, 2022. Accessed: 2022-Feb-02.

[8] The OpenRoad Project, Democratizing Hardware Design. https://theopenroadproject.org/, 2022. Accessed: 2022-Feb-02.

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

[10] Reto Achermann. *On Memory Addressing*. PhD dissertation, ETH Zurich, 2020.

[11] Marcos K. Aguilera, Naama Ben-David, Rachid Guerraoui, Virendra J. Marathe, Athanasios Xygkis, and Igor Zablotchi. Microsecond consensus for microsecond applications. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 599–616. USENIX Association, November 2020.

[12] Sebastian Angel, Mihir Nanavati, and Siddhartha Sen. Disaggregation and the application. In *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*. USENIX Association, July 2020.

[13] Nils Asmussen, Michael Roitzsch, and Hermann Härtig. M³x: Autonomous accelerators via Context-Enabled Fast-Path communication. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 617–632, Renton, WA, July 2019. USENIX Association.

[14] 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. Association for Computing Machinery.

[15] 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)*. USENIX Association, July 2020.

[16] 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)*. USENIX Association, July 2020.

[17] axbryd. hXDP repo. https://github.com/axbryd/hXDP-Artifacts. Accessed: 2022-Feb-02.

[18] 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.

[19] 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. USENIX Association.

[20] 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. Association for Computing Machinery.

[21] 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.

[22] Stephen Bates. Project Donard: NVM Express for Peer-2-Peer between SSDs and other PCIe Devices. https://www.snia.org/sites/default/files/SDC15_presentations/nvme_fab/StephenBates_Donard_NVM_Express_Peer-2_Peer.pdf. Accessed: 2022-Feb-02.

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

[24] Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. IX: A protected dataplane operating system for high throughput and low latency. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 49–65, Broomfield, CO, October 2014. USENIX Association.

[25] 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.

[26] 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. USENIX Association.

[27] 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. Association for Computing Machinery.

[28] 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.

[29] Andrew Binstock and Donald Knuth. Interview with Donald Knuth. https://www.informit.com/articles/article.aspx?p=1193856, 2008. Accessed: 2022-Feb-02.

[30] Daniel Bittman, Peter Alvaro, Pankaj Mehra, Darrell D. E. Long, and Ethan L. Miller. Twizzler: a Data-Centric OS for Non-Volatile memory. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 65–80. USENIX Association, July 2020.

[31] Matias Bjørling. Zone append: A new way of writing to zoned storage. Santa Clara, CA, February 2020. USENIX Association.

[32] Michaela Blott, Kimon Karras, Ling Liu, Kees Vissers, Jeremia Bär, and Zsolt István. Achieving 10gbps line-rate key-value stores with FPGAs. In *5th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 13)*, San Jose, CA, June 2013. USENIX Association.

[33] Michaela Blott, Ling Liu, Kimon Karras, and Kees Vissers. Scaling out to a Single-Node 80gbps memcached server with 40terabytes of memory. In *7th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 15)*, Santa Clara, CA, July 2015. USENIX Association.

[34] Shekhar Borkar and Andrew A. Chien. The future of microprocessors. *Commun. ACM*, 54(5):67–77, may 2011.

[35] 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. USENIX Association, November 2020.

[36] Irina Calciu, M. Talha Imran, Ivan Puddu, Sanidhya Kashyap, Hasan Al Maruf, Onur Mutlu, and Aasheesh Kolli. *Rethinking Software Runtimes for Disaggregated Memory*, page 79–92. Association for Computing Machinery, New York, NY, USA, 2021.

[37] Paul Castro, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. The rise of serverless computing. *Communications of the ACM*, 62(12):44–54, 2019.

[38] Sai Rahul Chalamalasetti, Kevin Lim, Mitch Wright, Alvin AuYoung, Parthasarathy Ranganathan, and Martin Margala. An fpga memcached appliance. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '13, page 245–254, New York, NY, USA, 2013. Association for Computing Machinery.

[39] 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.

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

[41] 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. Association for Computing Machinery.

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

[43] Feras Daoud, Amir Watad, and Mark Silberstein. Gpurdma: Gpu-side library for high performance networking from gpu kernels. In *Proceedings of the 6th International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '16, New York, NY, USA, 2016. Association for Computing Machinery.

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

[45] Peter J. Denning. Virtual memory. *ACM Comput. Surv.*, 2(3):153–189, sep 1970.

[46] Design Gateway. PCIe x16 Lanes Crossover adapter board for NVMe-IP evaluation. Accessed: 2022-Feb-02, https://eu.mouser.com/datasheet/2/854/AB18-PCIEx16-MAN-E-1594818.pdf.

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

[48] Stephen Dolan, Servesh Muralidharan, and David Gregg. Compiler support for lightweight context switching. *ACM Trans. Archit. Code Optim.*, 9(4), jan 2013.

[49] Endace. Endace dag packet capture cards: Part 1. https://tryingtokeepitsecure.bz/index.php/8-network-engineering/14-endace-dag-packet-capture-cards. Accessed: 2022-Feb-02.

[50] 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. USENIX Association.

[51] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, page 365–376, New York, NY, USA, 2011. Association for Computing Machinery.

[52] Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, pages 37–48, London, England, UK, 2012. ACM.

[53] 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. USENIX Association.

[54] 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.

[55] Fungible. TrueFabric: A Fundamental Advance to the State of the Art in Data Center Networks. https://www.fungible.com/wp-content/uploads/2020/08/WP0033.00.02020818-TrueFabric-A-Fundamental-Advance-to-the-State-of-the-Art-in-Data-Center-Networks.pdf, 2022. Accessed: 2022-Feb-02.

[56] Peter X. Gao, Akshay Narayan, Sagar Karandikar, Joao Carreira, Sangjin Han, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. Network requirements for resource disaggregation. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 249–264, Savannah, GA, November 2016. USENIX Association.

[57] 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. Association for Computing Machinery.

[58] 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. USENIX Association, April 2021.

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

[60] 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.

[61] Zhiyuan Guo, Yizhou Shan, Xuhao Luo, Yutong Huang, and Yiying Zhang. Clio: A hardware-software co-designed disaggregated memory system. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 417–433, New York, NY, USA, 2022. Association for Computing Machinery.

[62] Sangjin Han, Norbert Egi, Aurojit Panda, Sylvia Ratnasamy, Guangyu Shi, and Scott Shenker. Network support for resource disaggregation in next-generation datacenters. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, New York, NY, USA, 2013. Association for Computing Machinery.

[63] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, 2011.

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

[65] 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. Association for Computing Machinery.

[66] 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. Association for Computing Machinery.

[67] Morteza Hoseinzadeh and Steven Swanson. Corundum: Statically-enforced persistent memory safety. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2021, page 429–442, New York, NY, USA, 2021. Association for Computing Machinery.

[68] 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. USENIX Association.

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

[70] Zsolt István, David Sidler, Gustavo Alonso, and Marko Vukolic. Consensus in a box: Inexpensive coordination in hardware. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 425–438, Santa Clara, CA, March 2016. USENIX Association.

[71] Jakub Kicinski, Nicolaas Viljoen. Netronome systems, ebpf hardware offload to smartnics: cls bpf and xdp. https://www.netronome.com/media/documents/eBPF_HW_OFFLOAD_HNiMne8_2_.pdf. Accessed: 2022-Feb-02.

[72] 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. Association for Computing Machinery.

[73] 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.

[74] 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.

[75] 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. USENIX Association.

[76] 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.

[77] Daehyeok Kim, Amirsaman Memaripour, Anirudh Badam, Yibo Zhu, Hongqiang Harry Liu, Jitu Padhye, Shachar Raindel, Steven Swanson, Vyas Sekar, and Srinivasan Seshan. Hyperloop: Group-based NIC-offloading to Accelerate Replicated Transactions in Multi-tenant Storage Systems. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 297–312, Budapest, Hungary, 2018. ACM.

[78] 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. USENIX Association.

[79] Ana Klimovic, Christos Kozyrakis, Eno Thereska, Binu John, and Sanjeev Kumar. Flash storage disaggregation. In *Proceedings of the Eleventh European Conference on Computer Systems*, EuroSys '16, New York, NY, USA, 2016. Association for Computing Machinery.

[80] 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.

[81] 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.

[82] 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.

[83] 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. Association for Computing Machinery.

[84] 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. USENIX Association, November 2020.

[85] 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.

[86] Lucas Kuhring, Eva Garcia, and Zsolt István. Specialize in Moderation—Building application-aware storage services using FPGAs in the datacenter. In *11th USENIX Workshop on Hot Topics in Storage and*

File Systems (HotStorage 19), Renton, WA, July 2019. USENIX Association.

[87] 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. USENIX Association, July 2021.

[88] Joshua Landgraf, Tiffany Yang, Will Lin, Christopher J. Rossbach, and Eric Schkufza. *Compiler-Driven FPGA Virtualization with SYNERGY*, page 818–831. Association for Computing Machinery, New York, NY, USA, 2021.

[89] Bojie Li, Zhenyuan Ruan, Wencong Xiao, Yuanwei Lu, Yongqiang Xiong, Andrew Putnam, Enhong Chen, and Lintao Zhang. Kv-direct: High-performance in-memory key-value store with programmable nic. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, pages 137–152, Shanghai, China, 2017. ACM.

[90] 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. Association for Computing Machinery, New York, NY, USA, 2020.

[91] Hyeontaek Lim, Dongsu Han, David G. Andersen, and Michael Kaminsky. Mica: A holistic approach to fast in-memory key-value storage. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, pages 429–444, Seattle, WA, 2014.

[92] 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. USENIX Association, November 2020.

[93] Ming Liu, Tianyi Cui, Henry Schuh, Arvind Krishnamurthy, Simon Peter, and Karan Gupta. Offloading distributed applications onto smartnics using ipipe. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 318–333, New York, NY, USA, 2019. Association for Computing Machinery.

[94] Jiacheng Ma, Gefei Zuo, Kevin Loughlin, Xiaohe Cheng, Yanqiang Liu, Abel Mulugeta Eneyew, Zhengwei Qi, and Baris Kasikci. *A Hypervisor for Shared-Memory FPGA Platforms*, page 827–844. Association for Computing Machinery, New York, NY, USA, 2020.

[95] Jiacheng Ma, Gefei Zuo, Kevin Loughlin, Haoyang Zhang, Andrew Quinn, and Baris Kasikci. Debugging in the brave new world of reconfigurable hardware. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 946–962, New York, NY, USA, 2022. Association for Computing Machinery.

[96] 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. USENIX Association.

[97] Kiwan Maeng and Brandon Lucia. Adaptive dynamic checkpointing for safe efficient intermittent computing. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 129–144, Carlsbad, CA, October 2018. USENIX Association.

[98] Evangelos Markatos, Ji Y, Michalis Polychronakis, Vladimir Smotlacha, and Sven Ubik. Scampi - a scaleable monitoring platform for the internet. 05 2004.

[99] 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. USENIX Association.

[100] 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. Association for Computing Machinery.

[101] 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. Association for Computing Machinery.

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

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

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

[105] Xiangyong Ouyang, David Nellans, Robert Wipfel, David Flynn, and Dhabaleswar K. Panda. Beyond block i/o: Rethinking traditional storage primitives. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 301–311, 2011.

[106] Johan Peltenburg, Lars T.J. van Leeuwen, Joost Hoozemans, Jian Fang, Zaid Al-Ars, and H. Peter Hofstee. Battling the cpu bottleneck in apache parquet to arrow conversion using fpga. In *2020 International Conference on Field-Programmable Technology (ICFPT)*, pages 281–286, 2020.

[107] Nathan Pemberton and Johann Schleier-Smith. The serverless data center : Hardware disaggregation meets serverless computing. 2019.

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

[109] 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.

[110] 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. Association for Computing Machinery.

[111] 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.

[112] Murad Qasaimeh, Kristof Denolf, Jack Lo, Kees Vissers, Joseph Zambreno, and Phillip H. Jones. Comparing energy efficiency of cpu, gpu and fpga implementations for vision kernels. In *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*, pages 1–8, 2019.

[113] 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.

[114] Kaveh Razavi and Animesh Trivedi. Stratus: Clouds with microarchitectural resource management. In *12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*. USENIX Association, July 2020.

[115] 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.

[116] 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.

[117] 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.

[118] 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. Association for Computing Machinery.

[119] 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. Association for Computing Machinery.

[120] 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. Association for Computing Machinery.

[121] 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. USENIX Association.

[122] Yizhou Shan, Yutong Huang, Yilun Chen, and Yiying Zhang. LegoOS: A Disseminated, Distributed OS for Hardware Resource Disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 69–87, Carlsbad, CA, 2018.

[123] 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.

[124] 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. Association for Computing Machinery.

[125] 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. Association for Computing Machinery.

[126] Brian Suchy, Souradip Ghosh, Drew Kersnar, Siyuan Chai, Zhen Huang, Aaron Nelson, Michael Cuevas, Alex Bernat, Gaurav Chaudhary, Nikos Hardavellas, Simone Campanoni, and Peter Dinda. Carat cake: Replacing paging via compiler/kernel cooperation. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 98–114, New York, NY, USA, 2022. Association for Computing Machinery.

[127] 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`.

[128] 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. Association for Computing Machinery.

[129] 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. Association for Computing Machinery.

[130] Shin-Yeh Tsai, Yizhou Shan, and Yiying Zhang. Disaggregating persistent memory and controlling them remotely: An exploration of passive disaggregated Key-Value stores. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 33–48. USENIX Association, July 2020.

[131] Yatish Turakhia, Gill Bejerano, and William J. Dally. Darwin: A genomics co-processor provides up to 15,000x acceleration on long read assembly. In *Invited talk at the 2019 USENIX Annual Technical Conference (USENIX ATC 19)*, Renton, WA, July 2019. USENIX Association.

[132] Han Wang, Robert Soulé, Huynh Tu Dang, Ki Suh Lee, Vishal Shrivastav, Nate Foster, and Hakim Weatherspoon. P4fpga: A rapid prototyping framework for p4. In *Proceedings of the Symposium on SDN Research*, SOSR '17, page 122–135, New York, NY, USA, 2017. Association for Computing Machinery.

[133] 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.

[134] Michael Wei, John D. Davis, Ted Wobber, Mahesh Balakrishnan, and Dahlia Malkhi. Beyond block i/o: Implementing a distributed shared log in hardware. In *Proceedings of the 6th International Systems and Storage Conference*, SYSTOR '13, New York, NY, USA, 2013. Association for Computing Machinery.

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

[136] Yuanlong Xiao, Eric Micallef, Andrew Butt, Matthew Hofmann, Marc Alston, Matthew Goldsmith, Andrew Merczynski-Hait, and André DeHon. Pld: Fast fpga compilation to make reconfigurable acceleration compatible with modern incremental refinement software

development. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 933–945, New York, NY, USA, 2022. Association for Computing Machinery.

[137] Shuotao Xu, Sungjin Lee, Sang-Woo Jun, Ming Liu, Jamey Hicks, and Arvind. Bluecache: A scalable distributed flash-based key-value store. *Proc. VLDB Endow.*, 10(4):301–312, nov 2016.

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

[139] Teng Zhang, Jianying Wang, Xuntao Cheng, Hao Xu, Nanlong Yu, Gui Huang, Tieying Zhang, Dengcheng He, Feifei Li, Wei Cao, Zhongdong Huang, and Jianling Sun. FPGA-Accelerated compactions for LSM-based Key-Value store. In *18th USENIX Conference on File and Storage Technologies (FAST 20)*, pages 225–237, Santa Clara, CA, February 2020. USENIX Association.

[140] Mark Zhao, Mingyu Gao, and Christos Kozyrakis. Shef: Shielded enclaves for cloud fpgas. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2022, page 1070–1085, New York, NY, USA, 2022. Association for Computing Machinery.

[141] 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. Association for Computing Machinery.

[142] Noa Zilberman, Yury Audzevich, Georgina Kalogeridou, Neelakandan Manihatty-Bojan, Jingyun Zhang, and Andrew Moore. Netfpga: Rapid prototyping of networking devices in open source. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, page 363–364, New York, NY, USA, 2015. Association for Computing Machinery.

[143] 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.

Appendix-A: Hyperion Images

Hyperion is prototyped with a Xilinx U280 FPGA and NVMe device as shown in Figure 2 and Figure 3.

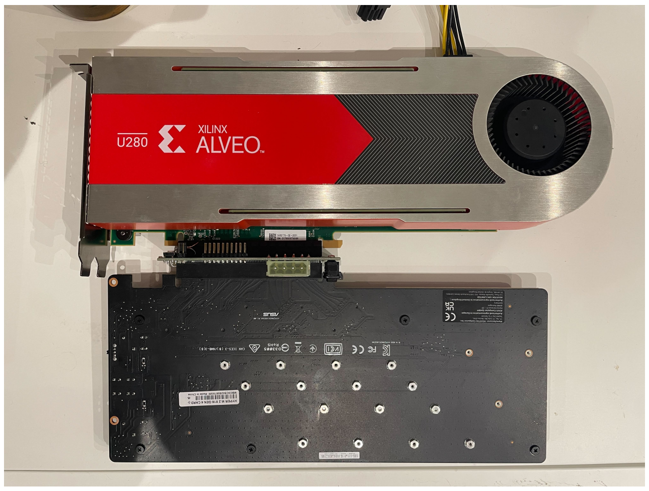

Figure 2: Hyperion: U280 FPGA-side up.

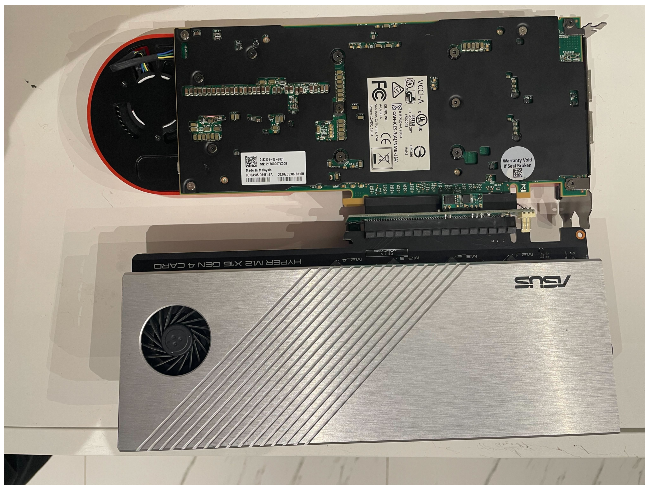

Figure 3: Hyperion: SSD-side up.