Understanding the Idiosyncrasies of Emerging BlueField DPUs

Arjun Kashyap
University of California, Merced
Merced, CA, USA
akashyap5@ucmerced.edu

Yuke Li
University of California, Merced
Merced, CA, USA
yli304@ucmerced.edu

Darren Ng
University of California, Merced
Merced, CA, USA
dng350@ucmerced.edu

Xiaoyi Lu
University of California, Merced
Merced, CA, USA
xiaoyi.lu@ucmerced.edu

Abstract

Data Processing Units (DPUs) are becoming available in datacenter environments to offload/accelerate workloads from the host. However, a comprehensive analysis is required to help users determine how to effectively utilize DPUs for their workloads, considering the various configurations and generations available. To fill in this gap, we conduct a fair and rigorous characterization by performing 15 benchmarking tests to demonstrate the evolution of representative SoC-based DPUs, specifically NVIDIA's BlueField-1, BlueField-2, and BlueField-3. Our work surfaces several idiosyncrasies across three key characterization dimensions—network, DMA engine, and memory. For network, we exhaustively test two major DPU modes—on-path (and five submodes) and off-path modes. We develop DPUDMABench, a microbenchmark suite to systematically analyze different data exchange primitives supported by DPU's DMA engine. We also conduct two application case studies examining the DPU mode's performance impact on TCP/IP and RDMA-based key-value stores (MICA and HERD). Based on our multi-generational DPU characterization, we identify and summarize 14 major idiosyncrasies, along with providing guidelines for optimal system and future hardware design.

CCS Concepts

• **Hardware → Networking hardware**; *Emerging technologies.*

ICS '25, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1537-2/25/06
https://doi.org/10.1145/3721145.3725780

Keywords

Data Processing Unit (DPU), BlueField DPU, Performance

ACM Reference Format:
Arjun Kashyap, Yuke Li, Darren Ng, and Xiaoyi Lu. 2025. Understanding the Idiosyncrasies of Emerging BlueField DPUs. In *2025 International Conference on Supercomputing (ICS '25), June 08–11, 2025, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3721145.3725780

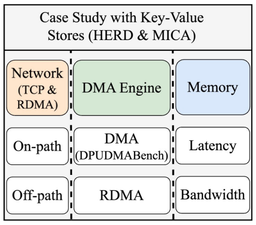
1 Introduction

A Data Processing Unit (DPU) is a device that adds computing power to a high-performance Network Interface Card (NIC). DPUs contain programmable compute units, making them suitable for offloading and/or accelerating server tasks. Recently, SoC (System on Chip)-based DPUs [7, 31, 37–39] are becoming increasingly utilized in datacenter environments. For example, they are used to accelerate deep learning training [18], data compression [24–26], molecular dynamics [21] workloads and offload MPI operations [47]. The prevalence of SoC-based DPUs is mainly due to—a) their support for both on-path and off-path modes of DPU operation (discussed in § 2), b) presence of hardware accelerators, and c) ease of programming [22] compared to their FPGA/MIPS-based counterparts [29, 50]. Thus, it is crucial to gain a comprehensive understanding of DPU characteristics to guide researchers and system/hardware designers in leveraging DPUs for optimal workload performance.

SOTA Related Work. Figure 1(a) summarizes and compares prior SoC-based DPU work [33, 34, 49, 51] with characterization done in this work. Xing et al. [51] designed a framework comparing the network performance of two SoC-based DPUs on a single machine. Their study was conducted only for the kernel network stack for TCP transport. In our study, we evaluate the performance of both TCP (kernel and userspace stack) and RDMA (userspace stack) transport. Only the latency of a single mode of on-path ("traffic offload" mode

		[51]	[34]	[33]	[49]	**Ours**
DPU mode	Off-path	✓	✓	✓	✓	✓
	On-path	1	0	0	0	5
Network	RDMA	✗	✓	✓	✓	✓
	Ethernet	✗	✗	✗	✗	✓
DMA engine	DMA	0	0	0	2	4
	RDMA	✗	✓	✗	✓	✓
Memory		✗	✓	✗	✗	✓
DPU generation		BF2	BF2/3	BF2	BF2	BF1/2/3
Client DPU + Server DPU		✗	✗	✗	✗	✓
Case Study		T-KVS	P3DFFT	✗	R-KVS FS	R-KVS T-KVS

(a) Related work comparison



(b) Scope overview

Figure 1: Novelty and contribution overview. Unlike prior works, we evaluate 5 on-path submodes and 4 DMA types. “T-KVS”, “R-KVS”, and “FS” stand for TCP-based KVS, RDMA-based KVS, and file system.

based on the results reported) is evaluated. On the other hand, we perform a comprehensive characterization of a total of five different submodes within the on-path mode of DPUs and measure their bandwidth impact on hosts.

DPU-Bench [33] presented a microbenchmark to offload collective communication to DPUs. Michalowicz et al. [34] characterized off-path DPU mode for RDMA transport, DPU’s memory bandwidth, and measured DPU performance for MPI-based workloads. We characterize DPU memory both in terms of latency and bandwidth, alongside DMA and network performance, and assess the benefits of deploying DPUs on both client and server machines within a client-server architecture. Wei et al. [49] characterized off-path RDMA communication paths and conducted a case study on RDMA-based key-value store (KVS) and distributed file system. They evaluated the performance of host-DPU communication via RDMA and only conducted throughput evaluation of polling-based DPU-initiated DMA primitives. Our work is different in two ways. Firstly, we characterized both on-path and off-path DPU modes with different network transports (TCP and RDMA), while Wei et al. [49] evaluated only the off-path mode with RDMA transport. Secondly, we conduct a detailed analysis (latency and throughput) of all four low-level DMA primitives available in DPU—polling vs. event-based and host-initiated vs. DPU-initiated.

Hence, previous studies do not perform a comprehensive characterization taking into account different aspects of DPUs—modes of operation (on-path and off-path), network type (TCP and RDMA) and network stack, DMA engine, and memory. *To the best of our knowledge, this is the first comprehensive study analyzing the evolution across three BlueField DPU generations, evaluating all five on-path submodes along with in-depth analysis of four types of DPU DMA primitives.*

Motivation & Characterization. Exhaustive characterization of SoC-based DPUs is challenging due to their varied features, such as hardware components (NIC, (R)DMA) and configuration options (DPU modes). For instance, BlueField-1 [37] (BF-1), BlueField-2 [38] (BF-2), and BlueField-3 [39] (BF-3) have different generations of NIC/adapter. The DPU’s network adapters combined with their modes make their performance implications on the host and SoC unclear. Furthermore, DPUs offer different primitives of data exchange between the host and the DPU—RDMA, DMA, or both. Determining the appropriate data movement primitive to use while offloading tasks becomes non-trivial. The different DPU modes and host-DPU data exchange, along with the characteristics of DPU memory, make it challenging for users to assess how their workloads can fully leverage the advantages offered by SoC-based DPUs.

Thus, it becomes imperative to characterize and evaluate SoC-based DPUs. A comprehensive study can also help guide future DPU hardware designs. To this end, we analyze the progression across three DPU generations, BF-1, BF-2, and BF-3, across three key dimensions—networking, DMA engine, and memory. We also present the performance implications of DPU modes on Key-Value Store (KVS) application as shown in Figure 1(b). All our major findings as idiosyncrasies and advice are summarized in Table 1.

a) Network: We characterize DPU’s network performance across off-path and on-path modes (including five submodes), focusing on RDMA and TCP transports in on-path mode. As client-server architecture is common for networked applications, we evaluate network performance by deploying DPUs on both ends. Our analysis shows that the less-explored on-path DPU (sub)mode significantly impacts the host’s performance across all three DPU generations. For example, on BF-3, it leads to a 50% reduction in TCP bandwidth and a 30% increase in RDMA latency compared to off-path mode.

b) DMA engine: The DMA operations differ based on the DMA completions—event-based and polling-based—and initiator—host vs. DPU. We develop a micro-benchmark suite named **DPUDMABench** that evaluates the performance (throughput and latency) of DPU’s DMA engine and helps identify its idiosyncrasies. We also compare the performance between DPU’s DMA engine and RDMA engine. Our findings indicate that the performance of the DPU’s DMA engine is heavily influenced by the initiator for event-based DMA operations. For instance, host-initiated event-based DMA operations exhibit up to 4.8×/2.2× higher throughput than DPU-initiated DMA on BF-3/BF-2.

c) Memory: Memory characterization is crucial due to the frequent runtime memory accesses in most applications. We evaluate both latency and bandwidth of DPU’s on-board memory. Our study reveals that while BF-3 offers high memory bandwidth, it unexpectedly shows higher memory latency—up to 7.3× greater than BF-2 for block sizes ranging from 1 MB to 4 MB.

d) Case study: We perform two application case studies to analyze the performance impact of DPU modes on KVS and provide system design guidelines. We chose KVS for two main reasons—a) KVS follows a traditional client-server architecture that helps test DPU's network (DPU modes), compute, and memory, and b) KVS is widely used in many HPC and datacenter workloads [11, 17, 23, 48]. We choose two popular open-source in-memory KVS—HERD [20], an RDMA-based KVS, and MICA [27, 28], a TCP-based KVS. In our first case study, we examine how the off-path DPU mode can support various KVS client-server combinations and analyze their performance. In our second case study, we evaluate how five different submodes of the on-path DPU mode affect KVS server performance. We observe that the KVS throughput gap among five on-path submodes can reach up to 16×, even on the latest BF DPU generation.

Contributions. In summary, the following are our contributions:
❶ A rigorous multi-generational performance characterization of DPUs (i.e., BF-1, BF-2, and BF-3), covering network performance across on-path modes (with five submodes) and off-path mode, along with memory analysis.
❷ DPUDMABench, a micro-benchmark suite for quantifying the performance of DPU's DMA engine.
❸ Two application case studies evaluating the performance implication of both DPU modes on KVS application and proposing KVS design guidelines tailored to each mode.
❹ Performed 15 benchmarking tests and summarized 14 of our key findings as idiosyncrasies and advice, as shown in Table 1. We also provide recommendations for DPU hardware designers (Table 5) and discuss the generalizability of our work.

To help the DPU community, our source code and/or other artifacts have been made available at https://github.com/padsys/DPUIdioBench.

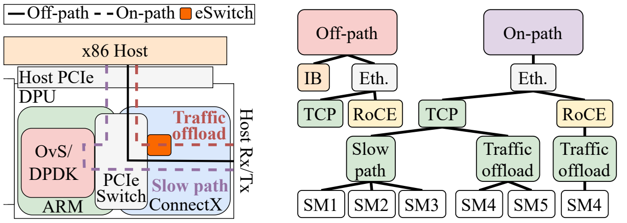
2 Background

DPUs typically consist of processing cores, memory, network adapter (e.g., ConnectX), and hardware accelerators, functioning as a standalone service. The SoC cores run their own operating system. DPUs also possess hardware accelerators like the DMA engine that help move/copy the memory buffers between the host and the DPU across the PCIe bus. NVIDIA's BlueField DPUs majorly operate under two modes [42]—on-path (DPU or embedded CPU function) mode and off-path (separated host) mode, as shown in Figure 2(a). The important abbreviations used in the rest of the paper are defined in Table 2.

On-path mode: It allows offloading custom packet operations (e.g., packet filtering). The on-path mode further supports two paths based on forwarding of network traffic to host—"slow path" and "traffic offload" (Figure 2(a)). In the "slow path", the DPU's ARM cores sit on the critical network path between the host and the NIC ports. The "slow path" is further subdivided into three categories based on how SoC handles network traffic. The first category is via Open vSwitch [44] (OvS) running on the DPU OS where the packet flows through the OvS's kernel data path (`OvS-Kernel` or **submode 1**). The second category is via

Table 1: Major idiosyncrasies and advice. "DPU" indicates BF generation idiosyncrasy/advice apply to.

Charac.	Major Idiosyncrasies/Advice	DPU
Network (§ 3.2)	**N1.** Traversing ARM cores in on-path submodes incurs 1.3×–2× overhead on the host (§ 3.2.1)	All
	N2. On-path for RoCE exhibits 22%–37% higher latency across all RDMA verbs than off-path (§ 3.2.2)	All
	N3. DPU-remote host RDMA latencies are similar for BF-2 and BF-3 for 256 KB–2 MB sizes (§ 3.2.2)	BF-3
DMA engine (§ 3.3)	**D1.** Event-based DMA performance depends on which device (host or DPU) is the initiator (§ 3.3.1)	BF-2 BF-3
	D2. Polling DMA operations on BF-3 exhibit lower performance (upto 39% higher latency and 20% lower throughput) than BF-2 (§ 3.3.1)	BF-3
	D3. Host/DPU core utilization for event-based DMA is dependent upon message size (§ 3.3.1)	BF-2 BF-3
	D4. DMA/RDMA Read performance depends on which device (host or DPU) is the initiator (§ 3.3.2)	BF-2
	D5. Latency of DMA Write > RDMA Write for small messages due to lack of inline support (§ 3.3.2)	BF-2 BF-3
Memory (§ 3.4)	**M1.** Latency of BF-3 is up to 7.3× higher than BF-2 for medium-sized memory blocks	BF-3
	M2. BF-1 and BF-2 have comparable memory bandwidth, with significant gap compared to BF-3	BF-1 BF-2
App. (§ 4)	**A1.** In off-path mode, KVS client and KVS server placement among hosts/DPUs for best performance depends on DPU generation and host vs. DPU performance gap (§ 4.1)	All
	A2. In off-path mode, user can save host resources by running KVS server on DPU when KVS client is on another DPU without performance degradation (§ 4.1)	All
	A3. Dedicated hash accelerator is needed on DPUs to bridge the performance gap with host (§ 4.1.3)	All
	A4. In on-path mode, KVS server should use submodes 4–5 for better performance than submodes 1–3 (§ 4.2)	All



(a) DPU architecture and modes of operation

(b) Network transport support for various modes and submodes (SM) of DPU

Figure 2: DPU architecture, modes, and submodes.

Table 2: Important abbreviations used.

Acronym	Definition	Acronym	Definition
SM	Submode	CH-SH	Client (Host) - Server (Host)
CX	ConnectX	CD-SH	Client (DPU) - Server (Host)
D-to-H	DPU to Host	CD-SD	Client (DPU) - Server (DPU)
H-to-D	Host to DPU	CH-SD	Client (Host) - Server (DPU)
Rx/Tx	Receive/Transmit	CDH-SH	Client (DPU+Host) - Server (Host)
DPDK	Data Plane Development Kit	L-CD-SH	Local Client (DPU) - Server (Host)

OvS with DPDK [14, 45] (`OvS-DPDK` or **submode 2**) where the OvS data path is moved from kernel space to userspace. The third category utilizes DPDK directly for both control and data path via poll mode drivers [15] (`DPDK-PMD` or **submode 3**) ensuring packet forwarding/manipulation entirely in userspace. The "traffic offload" avoids software overheads by offloading packet processing rules to the embedded switch (eSwitch) inside the DPU. "Traffic offload" is further subdivided into two categories based on whether the kernel or userspace data path is offloaded to the eSwitch—`OvS-Kernel (HW offload)` or **submode 4** and `OvS-DPDK (HW offload)` or **submode 5**. Thus, DPUs support a total of 5 different on-path submodes.

Off-path mode: The network traffic to/from the host bypasses the ARM subsystem. The DPU acts like an independent server with its own networking ports separate from that of the host. The DPU's SoC cores have no visibility/control over the host network traffic.

BF-2/3 additionally has a NIC mode, which turns it into a traditional NIC. Our characterization focuses solely on the on-path and off-path modes, as these are utilized in common off-path (e.g., Xilinx Alveo [50]) and on-path (e.g., AMD Pensando [3] and Marvel Octeon [30]) SmartNICs/DPUs.

3 Idiosyncrasies of DPUs

3.1 Methodology & Testbed

In this study, we uncover several idiosyncrasies of BF-1, BF-2, and BF-3 DPUs across three dimensions—network, DMA engine, and memory. For network characterization, we investigate the influence of off-path and on-path DPU modes and submodes on network transport performance. We use perftest [43], iperf3 [10], and DPDK pktgen [13] tools to evaluate network performance. For DMA characterization, we exhaustively evaluate the DMA engine using our proposed DPUDMABench (§ 3.3) and compare the performance between DPU's DMA engine and the RDMA engine. We measure DPU's DRAM performance using well-known microbenchmarks—STREAM [32] and tinymembench [2]. Lastly, we conduct two in-depth case studies to demonstrate how DPU's on-path and off-path modes affect the performance of KVS (i.e., MICA and HERD). We adapt both KVSs to run on BF DPUs' ARM-based architecture.

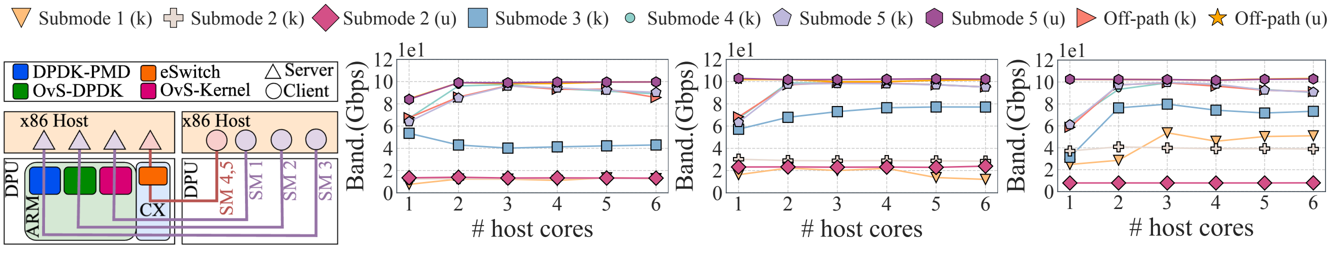
Testbed. We run our experiments on two Dell Precision 3630 machines, each equipped with a 12-core 3.3 GHz Intel Xeon E-2136 CPU (with hyperthreading enabled), 3200 MHz 32 GB DDR4 memory, and 12 MiB L3 cache. The machines run AlmaLinux 8.5 (Linux 4.18.0), and each machine contains either NVIDIA's BF-1, BF-2, or BF-3 DPUs. This allows us to keep the rest of the hardware and software setup the same except for the choice of DPU based on the experiment. The BF-1, BF-2, and BF-3 are connected back-to-back via an InfiniBand (IB) EDR (100 Gbps), HDR (200 Gbps), and NDR (400 Gbps) cable, respectively. Our evaluation is limited to two nodes because of the lack of availability of the same large-scale clusters that consist of all three BF generations since BlueField DPUs are new and emerging.

DPU hardware. BF-1/BF-2 have 16 GB DDR4 memory, while BF-3 has 16 GB DDR5 memory. BF-1 has 16 ARM cores (Cortex-A72) and one 100 Gbps IB/Ethernet port (ConnectX-5). BF-2 and BF-3 feature 8 cores (Cortex-A72 @ 2.5 GHz vs. Cortex-A78 @ 3 GHz) with single IB/Ethernet port NICs: ConnectX-6 (200 Gbps) and ConnectX-7 (400 Gbps). DPUs are configured either in on-path or off-path modes based on experiment. BF-1/BF-2 and BF-3 run Ubuntu 20.04.5 (Linux 5.4) and Ubuntu 22.04 (Linux 5.15). The OFED versions used are v5.5 for BF-1, v5.8 for BF-2, and v23.1 for BF-3. BF-2 and BF-3 use NVIDIA DOCA SDK [40] (v1.5.0 and v2.5.0) for host-DPU communication.

3.2 Networking

The network transport available to host in different DPU modes is shown in Figure 2(b). In off-path DPU mode, the host supports both Ethernet (Eth.) and InfiniBand (IB). In on-path mode, the host supports RDMA over Converged Ethernet (RoCE) with OvS hardware offloading [36] and TCP, featuring five submodes for TCP and one for RoCE. For Ethernet transport in on-path mode, the packet handling operations for host traffic are either performed on the SoC cores or the embedded switch. However, the performance overheads of different on-path modes are unknown. In this section, we systematically examine the impact of DPU modes, particularly the on-path mode, on network transport (TCP and RDMA) performance for BF DPUs. For TCP, we evaluate host performance across various on-path DPU submodes. For RDMA, we analyze latency differences between on-path and off-path modes across BF-1/2/3.

3.2.1 ***TCP***. We evaluate the performance of five different on-path submodes by measuring the bandwidth between two hosts using kernel/userspace TCP stack as shown in Figure 3(a). We accomplish this by configuring the DPU on one host in various on-path submodes while the DPU on the other host is set to off-path mode. This setting affirms that any observed overheads will result from the on-path mode,



(a) On-path submodes **(b) BF-1** **(c) BF-2** **(d) BF-3**

Figure 3: Host-to-Host bandwidth with increasing number of cores in different on-path DPU modes for BF-1, BF-2, and BF-3. Kernel (k) and userspace (u) stack use 128 KB and 1025 B payloads for maximum bandwidth, respectively.

as the DPU on the other host functions as a conventional NIC in the off-path mode. Additionally, the DPU in on-path mode is set to only forward packets from the NIC port to the host without any custom operation, indicating the peak on-path performance perceived on the host side.

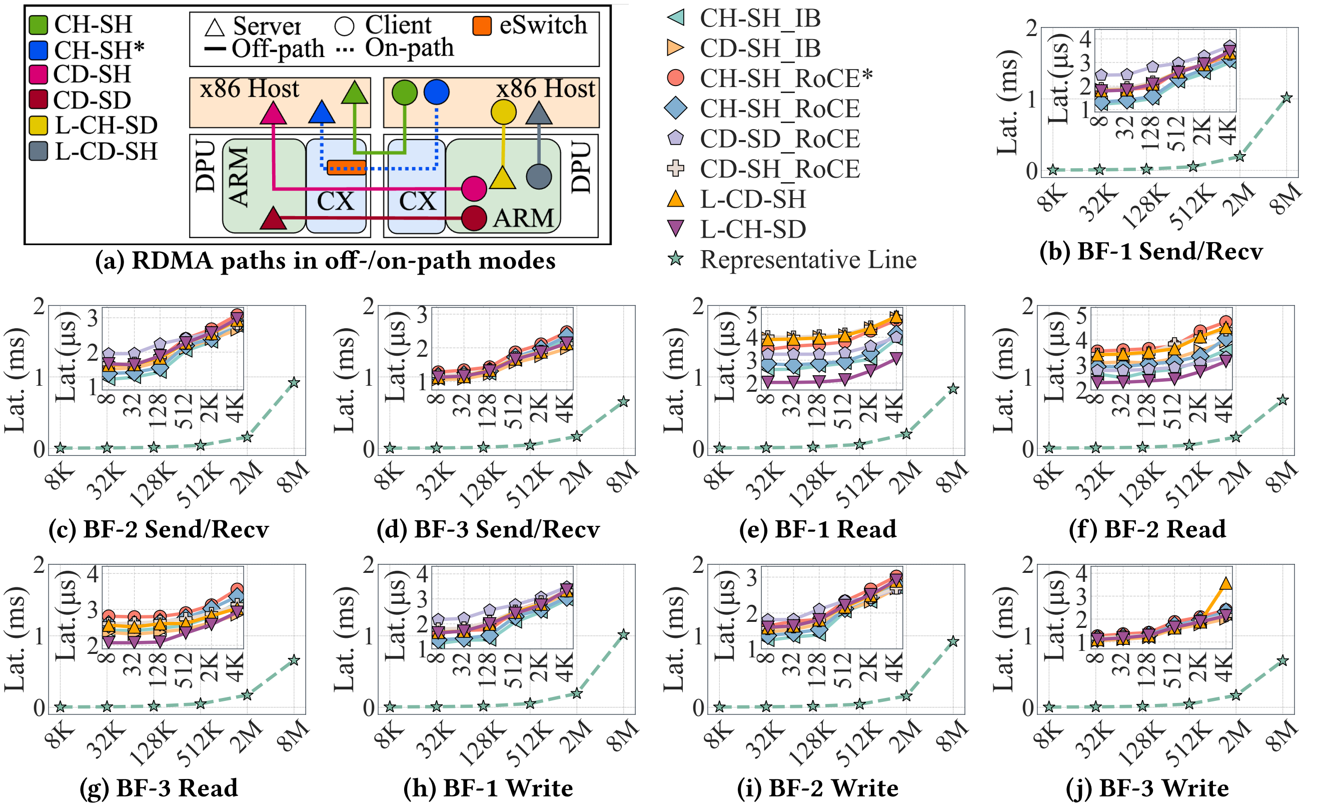
Idiosyncrasy N1. *Traversing the ARM cores in on-path submodes incurs 1.3×–2× overhead on the host across BF DPUs.* Figures 3(b)– 3(d) depict the bandwidth achieved between hosts as a function of host cores for TCP (kernel and userspace) for BF-1, BF-2, and BF-3. The submodes 1–2, using the kernel stack, exhibit bandwidth improvements with newer DPU generations. For example, the peak bandwidth improvement from BF-1 to BF-2 is under 2% for submode 1 and 2.2× for submode 2, while BF-2 to BF-3 sees gains of 3.9× and 1.4×, respectively. However, irrespective of using kernel/userspace network stack, submode 1 and submode 2 deliver 87%, 71.5%, and 50% lower bandwidth compared to submodes 4–5 on BF-1, BF-2, and BF-3, respectively. This is due to DPU ARM cores being in the critical path of network traffic from the host and unable to forward packets at high speed.

The submode 3 achieves higher performance when compared to submode 2, despite utilizing ARM cores. For instance, submode 3 achieves up to 3.9×, 2.7×, and 1.9× higher host bandwidth for BF-1, BF-2, and BF-3, respectively. The higher packet forwarding capability of submode 3 is due to the lack of OvS overheads when compared to submode 1 and submode 2. We also measure the instructions per cycle (IPC) to assess whether ARM cores act as a bottleneck. For example, in submode 2, BF-3's ARM core achieves an IPC of 3.31 with 40 Gbps peak host bandwidth, while BF-2's ARM core has an IPC of 1.42 with 28.5 Gbps peak host bandwidth. This confirms that the bottleneck in submode 1–3 is due to core frequency/IPC limitations. On the other hand, submodes 4–5 show comparable performance to off-path mode since ARM cores are no longer in the data path. Submode 3 is still 2×, 1.3×, and 1.3× slower than submodes 4–5 in BF-1, BF-2, and BF-3, respectively.

> **Takeaway-1:** Designers must offload network operations cautiously to avoid performance loss. Submode 1–3 may not be useful for host applications requiring high bandwidth across BF-1/2/3. Performing any custom operations on the ARM cores in these submodes would further degrade host performance. Submodes 4–5 provide high bandwidth, but performing custom packet processing operations might alter this. A limitation of submode 4–5 is that it supports basic packet processing operations (for example, forward/drop/tunneling) as it runs on the hardware switch when compared to submode 1-3.

3.2.2 ***RDMA.*** Next, we report the latency achieved by RDMA over IB and RoCE transports in both DPU modes. Here, we use small and large message sizes to represent buffer sizes <8 KB and 8 KB–8 MB, respectively. The different RDMA paths between the server, client, and their DPUs in the two DPU modes are illustrated in Figure 4(a). Figures 4(b) – 4(j) show the latency of different RDMA verbs—Send/Recv, Read, and Write for different RDMA paths. We do not report bandwidth results as no notable idiosyncrasies are observed across the different RDMA paths. We observe obvious trends with successive DPU generations for CH-SH, including lower IB latency compared to RoCE and decreasing RDMA latencies for both IB and RoCE with each generation, owing to improvements in network adapters. Next, we delve into the idiosyncrasies and their underlying causes.

Idiosyncrasy N2. *On-path mode for RoCE incurs non-negligible overheads on message latency for RDMA verbs across BF DPUs.* Successive BF generations show reduced RDMA verbs latency in both on-path and off-path modes. However, when comparing on-path (CH-SH_RoCE*) to off-path (CH-SH_RoCE) within a generation, on-path mode consistently exhibits higher latency. On-path latency exceeds off-path latency by up to 37%/30%/30% for Read, 22%/23%/28% for Write, and 33%/24%/26% for Send/Recv on BF-1/BF-2/BF-3,



(a) RDMA paths in off-/on-path modes

(b) BF-1 Send/Recv

(c) BF-2 Send/Recv

(d) BF-3 Send/Recv

(e) BF-1 Read

(f) BF-2 Read

(g) BF-3 Read

(h) BF-1 Write

(i) BF-2 Write

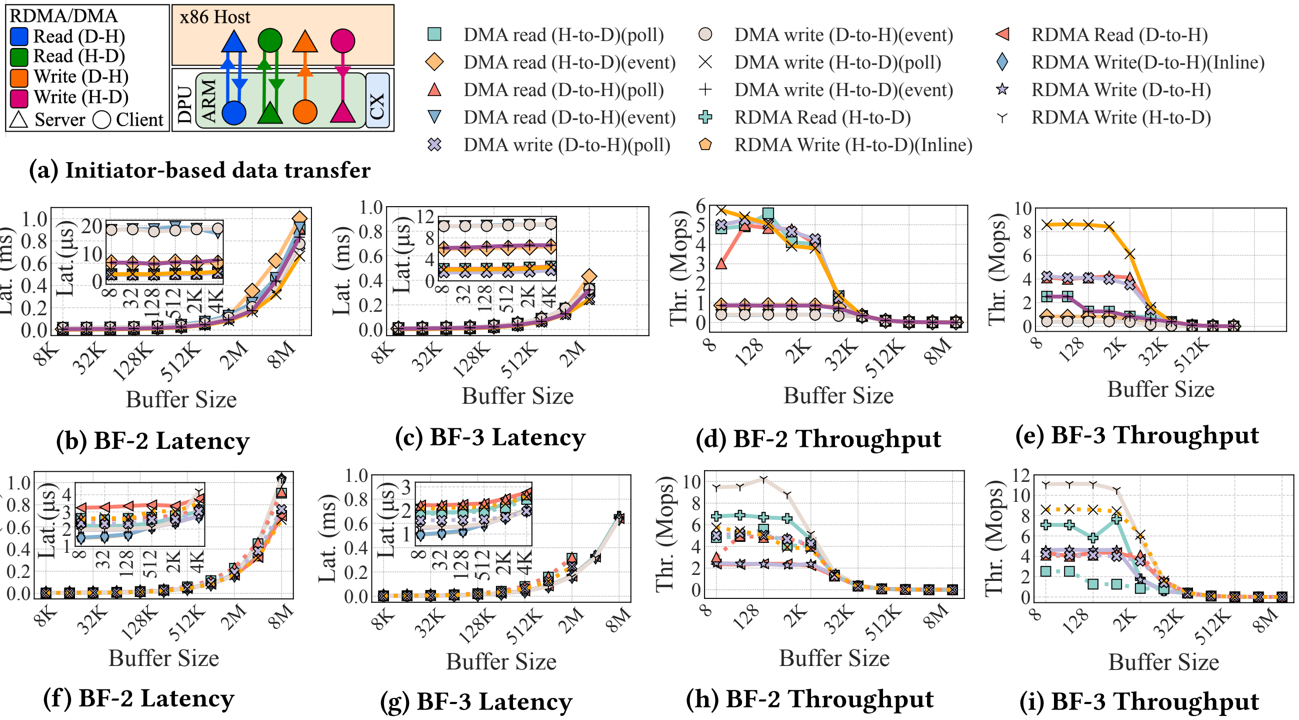
(j) BF-3 Write

Figure 4: RDMA latency with DPU modes on different client-server combinations among host and BF-1/BF-2/BF-3. "IB/RoCE" suffix indicates link type. (*) suffix represents on-path DPU mode while lack of it represents off-path mode. To simplify the graphs, representative data points are shown when payload is >4 KB due to similar and overlapped data points among different configurations. "CD-SD_IB" and "CH-SD_IB/RoCE" are not shown because the former is not supported [36] and for the latter BF-1 lacks out-of-band connection. Key findings are N2–N3.

respectively, as seen in Figures 4(b) – 4(j). The latency difference arises from messages traversing the DPU's eSwitch in the on-path submode 4 for RoCE, unlike off-path mode. The on-path vs. off-path latency gap decreases with successive BF DPUs for small messages. However, BF-3 exhibits a larger latency gap than BF-1/BF-2 for large messages. For large-sized messages, BF-3's on-path mode shows 12%–30%, 20%–28%, and 19%–26% higher latency for Read, Write, and Send/Receive, respectively, compared to off-path. In contrast, BF-1/BF-2 shows lower latency gaps, ranging from 11%–19%, 4%–15%, and 2%–11%, respectively.

Idiosyncrasy N3. *RDMA latencies between DPU and remote host are comparable for BF-2 and BF-3 for 256 KB–2 MB messages.* Successive DPU generations are expected to deliver lower RDMA latency due to advancements in network adapters within the DPU. This holds true for the CH-SH_IB/RoCE configuration. In contrast, RDMA latencies for CD-SH with BF-3 and BF-2 are similar across all RDMA verbs for 256 KB–2 MB messages, although BF-3 demonstrates lower latencies for other message sizes. We observe this phenomenon on both RoCE (CD-SH_RoCE) and IB (CD-SH_IB). We attribute this behavior to BF-3's higher DRAM latency compared to BF-2. For instance, BF-3's memory latency for 1 MB blocks is up to 7.3× higher compared to BF-2 (explained in § 3.4). The latency improvements from the network adapter (ConnectX-6 vs. ConnectX-7) are offset by BF-3's higher memory access latency. BF-3 does demonstrate lower latency than BF-2 for messages <256 KB and >2 MB in the CD-SH_RoCE/IB. This aligns with BF-3's memory latency being lower than BF-2 for block sizes in these ranges, further validating that BF-3's unique memory characteristics drive this peculiar behavior.

> **Takeaway-2:** System designers need to consider the overheads of RoCE in on-path submode 4. BF-3 in on-path shows up to a 30% increase in RDMA latency compared to off-path mode.



(a) Initiator-based data transfer

(b) BF-2 Latency

(c) BF-3 Latency

(d) BF-2 Throughput

(e) BF-3 Throughput

(f) BF-2 Latency

(g) BF-3 Latency

(h) BF-2 Throughput

(i) BF-3 Throughput

Figure 5: DMA (row 1) and RDMA vs. DMA (row 2) comparison on BF-2/3 for varying sizes. BF-2/BF-3 latency (b)/(c) and throughput (d)/(e) for DMA operations initiated by host/DPU for different DMA completions. BF-2/BF-3 RDMA vs. DMA latency (f)/(g) and throughput (h)/(i). "Inline" refers to data and control messages in the same packet. Dotted and solid lines represent DMA and RDMA, respectively. BF-3's DMA lacks results for 4 MB–8 MB as it supports a maximum size of 2 MB. Key findings are D1–D2 and D4–D5.

3.3 DMA engine

Here, we analyze and compare the performance of two host-DPU data exchange primitives—DMA and RDMA. BF-2 introduced a DMA engine that allows one to copy memory buffers between the host and the SoC via NVIDIA's DOCA SDK [40]. We design a microbenchmark, **DPUDMABench**, to exhaustively evaluate the various DMA operations supported by BF-2's and BF-3's DMA engines. DPUDMABench comprises two components—one on the host and one on the DPU—allowing it to evaluate DMA engine performance for operations initiated by either side. The benchmark allows checking for DMA completions in two ways—polling (busy-wait) and event-based (wait for a signal). It initializes and pre-registers DMA buffers on both the host and DPU. The benchmark reports average latency over multiple iterations, with each iteration issuing a single DMA operation and waiting for its completion. Throughput is evaluated by enqueuing 1024 DMA requests in a batch per iteration and waiting for the batch's completion.

DPUDMABench helps in generating different DMA operations—a) DPU-initiated vs. host-initiated, and b) event vs. polling-based. It leverages the DMA APIs from the respective DOCA SDKs of BF-2 and BF-3, ensuring uniform generation of diverse DMA operations across both DPU generations. We also compare the data movement performance of the DMA engine with that of the RDMA engine. We evaluate BF-2 and BF-3 here since BF-1 does not expose the DMA engine. The RDMA performance for host-to-BF-1 data exchange was discussed in § 3.2.2. We use small, medium, and large message sizes to represent buffer sizes <8 KB, 8 KB–512 KB, and >512 KB, respectively. Figure 5(a) depicts the data exchange for different DMA/RDMA operations based on the device initiator (host or DPU).

3.3.1 ***DMA***.

We analyze the performance and core utilization of different DMA operations supported by DPU's DMA engine. Figures 5(b) – 5(i) depict the latency and throughput of both DOCA DMA Read and Write operations for various payload sizes.

Idiosyncrasy D1. *Event-based DMA performance depends on which device (host or DPU) is the initiator.* The latency of event-based DMA operations initiated by the host shows an average of 3×/1.74× lower latency than DPU-initiated DMA operations, as shown in Figure 5(b) and Figure 5(c) for BF-2 and BF-3, respectively. For throughput, Figures 5(d) – 5(e) indicate that host-initiated DMA operations have 2.2×/4.8× higher throughput than DPU-initiated DMA operations on BF-2/BF-3. The notable disparity in latency and bandwidth between event-based DMA completions on DPUs, as opposed to that on the host, underscores the increased overhead involved in transitioning between kernel and userspace in DPUs. On the contrary, polling-based DMA performance does not depend on the initiator. Polling for small message sizes shows 3×–8×/2.5×–6.5× lower latency compared to event-based operations, irrespective of whether the host or the BF-2/BF-3 initiates the operation. For small message sizes, polling shows similar throughput irrespective of the initiator of operation and is up to 5.8×/3.5× faster than event-based counterparts as seen in Figure 5(d) and Figure 5(e).

Idiosyncrasy D2. *Polling DMA operations on BF-3 exhibit lower performance than BF-2.* DPU-initiated DMA on BF-3 shows 21%–39%/6%–28% higher latency than BF-2 for 64 KB–2 MB payloads for DMA Reads/Writes, as shown in Figure 5(b) and Figure 5(c). Similarly, DPU-initiated DMA Writes and Reads on BF-3 achieve 20% (4.2 Mops vs. 5.24 Mops) and 17% lower throughput, respectively, compared to BF-2 for payloads <4 KB, as seen from Figure 5(d) and Figure 5(e). Host-initiated DMA on BF-3 has 22%–38%/10%–23% higher latency than BF-2 for 64 KB–2 MB payloads for DMA Reads/Writes. Higher BF-3 DMA latency is due to the DMA engine's maximum payload capacity across PCIe. The maximum PCIe payload size (MPPS) determines the PCIe Maximum Transfer Unit (MTU) where larger PCIe transactions are broken into MTU-sized packets. DMA engine's MPPS is 512 B and 256 B for our BF-2 and BF-3 DPUs, which is auto-negotiated during PCIe device initialization [35]. For a 2 MB payload, BF-2's and BF-3's DMA engine issue $\lceil 2MB/512B \rceil$=4096 and $\lceil 2MB/256B \rceil$=8192 PCIe packets. Hence, BF-3's DMA engine requires twice as many PCIe packets to complete a transaction, leading to increased latency and lower throughput.

Idiosyncrasy D3. *Host/DPU core utilization for event-based DMA depends upon message size.* Figure 6(a) shows DPU and host core utilization for polling-/event-based DMA for both small and large-sized payloads. As expected, polling consumes 100% of the host/DPU core regardless of the buffer size. Upon further investigation, we found that ~75% of the cycles were spent checking the DMA completion status. In contrast, core utilization for event-based methods depends upon the payload size. For example, event-based completion methods consume 53.3%/45% and 43.8%–49% core cycles for 2 B payload on BF-2/BF-3 and the host, respectively. For large payloads (8 MB), the core utilization for both DPU and host is less than 1.5%.

> **Takeaway-3:** BF-3's DMA engine shows lower performance than BF-2's for polling DMAs, depending on the payload size. Designers also need to consider the trade-off between performance and core utilization when using different DMA operations.

3.3.2 ***DMA vs. RDMA.*** Next, we compare polling-based DMA operations with one-sided RDMA operations between the host and the DPUs, as shown in Figures 5(f) – 5(i). Event-based DMA operations are not compared as they have lower performance when compared to polling-based DMA as discussed previously in § 3.3.1.

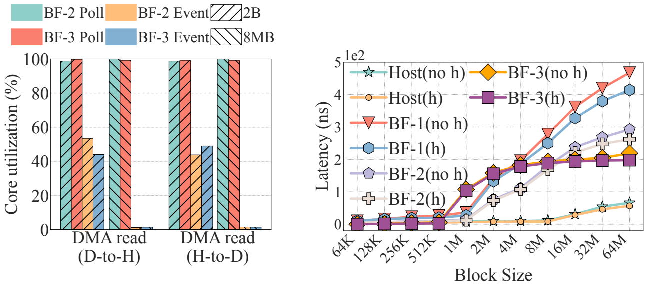
Idiosyncrasy D4. *DMA/RDMA Read performance depends on initiator (host or DPU).* For DPU-initiated operations, DMA Reads for small messages have 20% lower latency than RDMA Reads on BF-2, as seen in Figure 5(f). This is due to ARM cores taking longer to issue an RDMA Read request, which was also observed by authors in [49]. This behavior is absent on BF-3 in Figure 5(g) due to the stronger ARM cores on BF-3 compared to BF-2. However, for large messages, DMA Reads are up to 1.2×/1.96× slower than RDMA Reads on BF-2/3 as seen in Figure 5(f) – 5(g), indicating that DPU's DMA engine processes requests at a slower rate compared to the RDMA engine for large-sized Reads. For host-initiated operations, RDMA Reads consistently provide lower latency than DMA Reads by up to 25% and 47% on both BF-2 and BF-3.

Idiosyncrasy D5. *Latency of DMA Write > RDMA Write for small messages due to lack of inline support.* For small messages, the DMA Write has 15%/20% and 32%/47% higher latency on average when compared to RDMA Writes on BF-2/BF-3 for both DPU-initiated and host-initiated operations, respectively, as seen in Figure 5(f) and Figure 5(g). For RDMA Writes, inlining (combining metadata and data in a single request) further reduces the latency, increasing the latency gap between RDMA Write and DMA Write by 1.5×/1.7× and 1.6×1.9× on average for message sizes lower than 128 B for DPU-/host-initiated on BF-2 and BF-3, respectively. We find that DOCA DMA currently does not support inlining messages. However, for large message sizes (e.g., at 8 MB), DMA Writes are up to 1.5× faster than RDMA Writes on BF-2, whereas on BF-3, the opposite is true for large payload (2 MB) as seen in Figure 5(f) and Figure 5(g).

> **Takeaway-4:** Choosing between DMA/RDMA for host-DPU communication is complex due to their varied performance. For small data exchanges with host—a) on BF-2, use DMA for reading and RDMA for writing due to lower latency; b) on BF-3, use RDMA for writing and either DMA or RDMA for reading as their performance is comparable.

3.4 Memory

Here, we compare the memory performance (latency and bandwidth) of host, BF-1, BF-2, and BF-3 DPUs.



(a) Core Utilization **(b) Memory Latencies**

Figure 6: (a): Host/DPU core utilization for different DMA completions for 2 B/8 MB payloads. (b): Host and DPU memory latencies for varying memory block sizes with hugepages enabled (h) and disabled (no h).

Idiosyncrasy M1. *Memory access latency of BF-3 is higher than BF-2 for medium-sized memory blocks.* Figure 6(b) compares the memory latency of host and BF-1/2/3 DPU with/without hugepages enabled. We also test hugepage-enabled memory accesses because of its widespread use in high-performance system designs [19, 20, 28]. Surprisingly, BF-3 exhibits 7.3×, 2.1×, and 1.6× higher latency than BF-2 for 1 MB, 2 MB, and 4 MB memory block sizes, respectively. We further analyze this by examining low-level hardware cache events in BF-2/3, as shown in Table 3. We find that the higher latency of BF-3 vs. BF-2 for 1 MB–4 MB memory accesses is due to two reasons. First, during memory load operations, BF-3 has higher misses in the data Translation Lookaside Buffer (dTLB). The dTLB caches virtual-to-physical (V2P) memory address translations to avoid slower page table lookups. We observe that memory accesses for 1 MB, 2 MB, and 4 MB blocks on BF-3 result in 60%, 69%, and 42% more dTLB misses compared to similar accesses on BF-2 when hugepages are disabled. A higher dTLB miss rate results in increased latency for memory accesses, as the CPU core must frequently access the slower page table to perform V2P address translation. For smaller blocks (64 KB), BF-2 and BF-3 show similar dTLB miss rates. Second, BF-3 experiences a higher L2 cache miss ratio than BF-2, regardless of whether hugepages are enabled or disabled, resulting in lower L2 cache efficiency for BF-3. For example, for 1 MB block, the L2 cache miss ratio is 28% for BF-3 compared to 3% for BF-2.

For large block sizes (e.g., 64 MB), BF-2 memory access latency is 4.4× higher than the host but 1.6× lower than BF-1, showcasing improvements despite having the same memory type (DDR4 DRAM). BF-3 further reduces latency with 3.29× higher than the host and 1.34× lower than BF-2. Furthermore, hugepage reduces latency by only 1.13×, 1.12×, and 1.11× for BF-1/2/3, respectively.

Table 3: DPU SoC core's cache characteristics during 1 MB and 4 MB memory accesses. (h) and (no h) indicate hugepages enabled and disabled.

	dTLB misses (no h)		dTLB misses (h)		L1d cache util.		L2 cache miss	
	1MB	4MB	1MB	4MB	1MB	4MB	1MB	4MB
BF-2	87M	97M	80K	218K	44.7%	43.1%	3%	43%
BF-3	141M	138M	108K	282K	44.6%	43.1%	28%	48%

Table 4: Memory bandwidth of STREAM functions across single-/multi-core on host and DPU SoC cores.

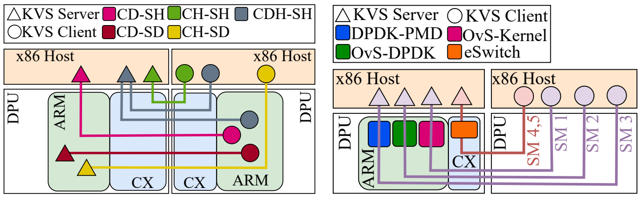
	Host		**BF-1**		**BF-2**		**BF-3**	
Cores	1	12	1	16	1	8	1	8
Copy (GB/s)	8.7	22.3	2.25	6.06	2.09	6.8	4.98	25.77
Scale (GB/s)	9.82	21.81	2.02	6.05	2.32	6.79	4.97	26.13
Add (GB/s)	12.24	24.28	2.84	7.78	2.9	7.7	6.91	27.08
Triad (GB/s)	11.86	24.37	2.84	7.79	2.98	7.7	7.20	27.13

Idiosyncrasy M2. *BF-1 and BF-2 have comparable memory bandwidth, with a significant gap compared to BF-3.* Table 4 shows the memory bandwidth of different STREAM functions on the host, BF-1, BF-2, and BF-3. Accessing memory via multiple cores vs. single core improves BF-1, BF-2, and BF-3 bandwidth by up to 2.74×, 2.58×, and 5.25×, respectively. Though the host exhibits up to 1.98× higher single-core bandwidth than BF-3, BF-3 exhibits up to 19.8% higher multi-core memory bandwidth than the host. Similarly, BF-3's single-/multi-core memory bandwidth is up to 2.41×/3.84× higher than BF-2 DPU. The superior memory bandwidth of BF-3 can be attributed to its dual-channel onboard memory and the higher data transfer rates of DDR5 compared to DDR4.

> **Takeaway-5:** Though BF-3's on-board memory can attain high bandwidth, its memory access latency demonstrates unusual characteristics. For medium-sized blocks, BF-3 has 7.3× higher memory latency than BF-2 due to higher dTLB miss rate and low L2 cache utilization. However, the trend reverses for large-sized memory blocks, with BF-3 showing 1.34× lower latency than BF-2.

4 Case Studies with Key-Value Stores

We present two case studies examining how DPU modes affect key-value store (KVS) performance. The first case study focuses on the off-path DPU mode when offloading the entire KVS client/server to DPUs. This helps determine the optimal placement of the KVS client and server across hosts and DPUs. The offloaded packet handling in on-path submodes displays varying impacts on host performance as observed in § 3.2.1. Our second case study examines the effect of offloaded packet handling when the KVS server runs on the host across different on-path submodes. Based on these studies, we propose four KVS design guidelines.



(a) KVS client-server pairings **(b) KVS server in 5 submodes**

Figure 7: KVS case studies on different DPU modes: a) off-path, b) on-path.

Figure 7 shows the KVS studies across different DPU modes, with BFs across read-/write-heavy workloads (100% GETs, 95% GETs, and 50% GETs) with uniform key distribution. The testbed is described in § 3.1. For off-path mode evaluation, we consider both HERD [20] and MICA KVS [27, 28]. We omit the "CDH-SH" configuration from 95%/50% GETs plots in the off-path evaluation as it shows a similar trend to 100% GETs. We choose MICA KVS for the on-path case study as TCP supports all the different on-path submodes. Unless otherwise stated, HERD KVS uses 16 B keys, 32 B values, and a request batch size of 32. MICA KVS uses 8 B keys, 8 B values, and a batch size of 64.

4.1 Off-path KVS study

4.1.1 RDMA-based KVS (HERD). Figure 8 shows the latency as a function of throughput for three workloads when KVS client/server either runs on the host or is offloaded to DPUs. In "CH-SH", a no-offload scenario, the KVS performance is not significantly affected by different DPU generations. In "CH-SH", the KVS latency remains consistent while using BF-1, BF-2, and BF-3, with the peak throughput improvement between BF-1/2 and BF-3 staying within 15%/19.5%/15.2% for 100%/95%/50% GETs. Compared to offloading KVS client/server to DPUs, "CH-SH" achieves optimal performance with throughput being up to 2.6×/2.9×/3.1× higher and latency being up to 3×/1.77×/1.5× lower when compared to "CD-SD" and "CD-SH". The superior performance of "CH-SH" is due to the host's higher processing capabilities over DPUs. Surprisingly, when offloading the KVS client to DPU, the KVS performance is similar whether the KVS server is offloaded to another DPU or on the host. This is because the KVS client on DPU is unable to generate sufficient KVS requests to saturate the KVS server, indicating that one could offload the entire KVS server to DPU in such cases.

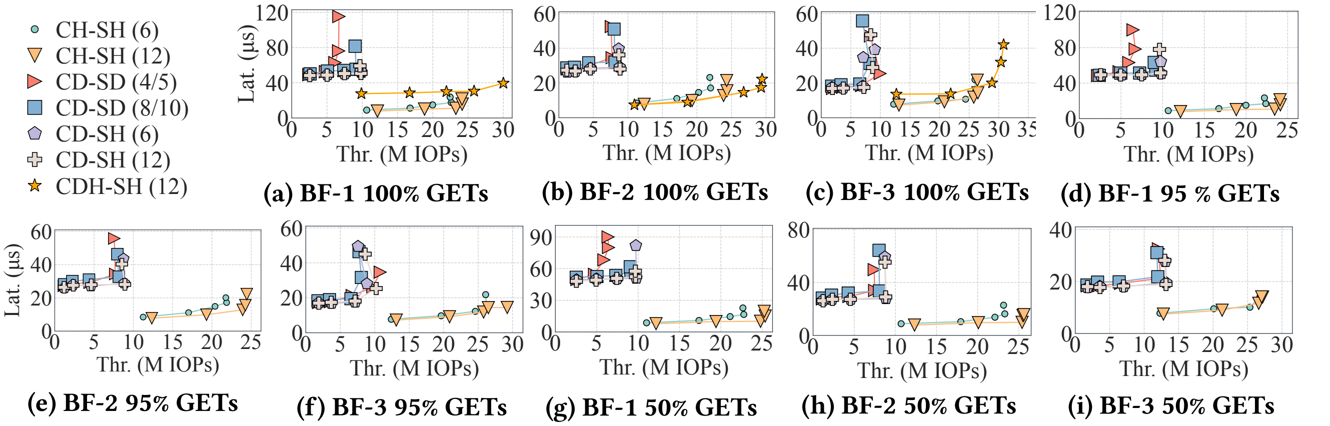
Figure 8(a) – Figure 8(c) additionally compare the performance of "CDH-SH" for 100% GETs. We observe that two KVS clients can improve the throughput of the HERD KVS by 1.24× compared to a single KVS client. Upon monitoring the network bandwidth of the KVS server, we observe that the "CDH-SH" configuration receives and transmits 1.2× more KVS request/response traffic. The increase in the number of KVS requests processed by clients contributes to higher throughput for the "CDH-SH" configuration.

4.1.2 TCP-based KVS (MICA). Figure 9 shows the latency vs. throughput for three workloads when KVS client/server either runs on the host or is offloaded to DPU. For BF-1 and BF-2, "CH-SH" achieves optimal performance with throughput being up to 3× higher and latency being up to 2.27× lower when compared to "CD-SD" and "CD-SH". The performance of "CD-SH" and "CD-SD" is similar, indicating that one could offload the entire KVS server to DPU when the client is offloaded to another DPU. This observation is similar to what we observed for HERD KVS previously in § 4.1.1.

"CH-SD" shows interesting and diverse performance trends on BF-2 and BF-3. On BF-2, the KVS server's performance remains unaffected by changes in the number of KVS client threads, indicating that the KVS server on BF-2 is the bottleneck. On the other hand, on BF-3, "CH-SD" achieves up to 2.3×/2.2×/2.5× higher throughput and 32%/34%/37.5% lower latency for 100% GETs, 95% GETs, and 50% GETs, respectively, compared to "CH-SD" on BF-2. "CH-SD" emerges as the best configuration for achieving optimal KVS performance with BF-3 compared to other setups. For example, on BF-3, "CH-SD" delivers 1.3× higher throughput and 40% lower latency than "CH-SH", benefitting from BF-3's higher multi-core memory bandwidth relative to the host and BF-2, as shown earlier in Table 4.

For "CD-SH", the throughput increases linearly with KVS client threads for all BFs. Offloading the KVS client to BF-1/2 fails to saturate the host KVS server, unlike BF-3. This is due to the weaker ARM cores on BF-1/2 compared to those on BF-3. With BF-2, "CH-SH" achieves 1.47× higher peak throughput for 100% GETs than when the KVS client is offloaded to DPU. Further analysis reveals that host-based KVS client generates 1.33× higher requests than client on DPU, resulting in higher throughput. Additionally, KVS performance in "CH-SH" using BF-3 delivers 3× and 2× lower latency for 100% GETs and 95% GETs, respectively, while achieving similar throughput across all workloads compared to BF-2. We find that the higher latency of "CH-SH" using BF-2 is due to the request queueing at the server-side NIC.

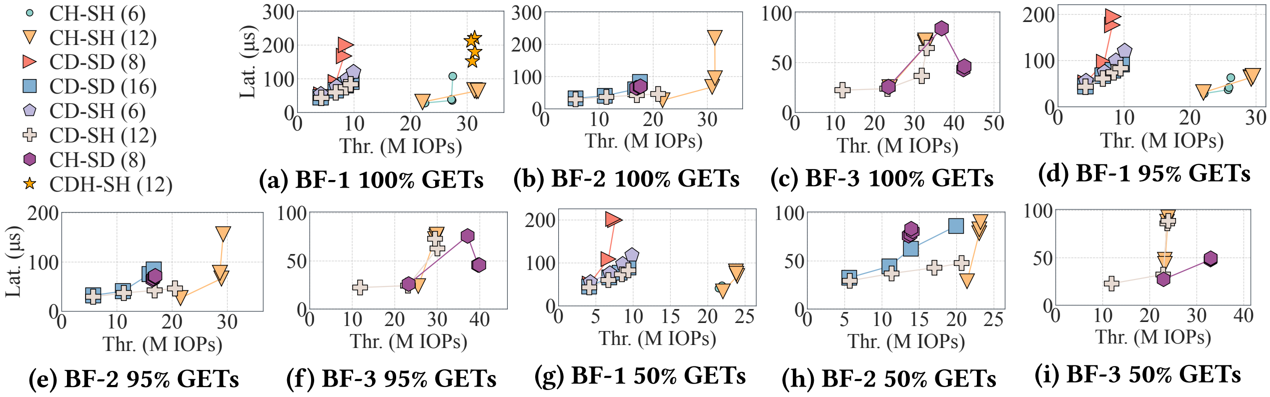
4.1.3 Hashing Overheads on DPU. Hash generation plays a vital role for KVS [6, 9, 20, 28] as it helps locate/store the key into in-memory data structures. In MICA and HERD, key hash generation occurs on the KVS client. Upon measuring its overhead on MICA, we observe that the `CityHash64` (64-bit hash) hash used consumes up to 1.6% on the host and up to 7.17% on the DPU. HERD's key hash generation uses the `CityHash128` hash function, consuming up to 22.83% host



(a) BF-1 100% GETs (b) BF-2 100% GETs (c) BF-3 100% GETs (d) BF-1 95 % GETs

(e) BF-2 95% GETs (f) BF-3 95% GETs (g) BF-1 50% GETs (h) BF-2 50% GETs (i) BF-3 50% GETs

Figure 8: HERD KVS performance with BF-1, BF-2, and BF-3 DPUs. Number in '()' represents KVS server threads. The KVS server on BF-1 uses 5/10 threads, and on BF-2/BF-3, it uses 4/8 threads.



(a) BF-1 100% GETs (b) BF-2 100% GETs (c) BF-3 100% GETs (d) BF-1 95% GETs

(e) BF-2 95% GETs (f) BF-3 95% GETs (g) BF-1 50% GETs (h) BF-2 50% GETs (i) BF-3 50% GETs

Figure 9: MICA KVS performance across BF-1, BF-2, and BF-3. Number in '()' represents KVS server threads.

cycles and 32% of DPU cycles, 1.4× more than on the host, due to the wimpy nature of DPU ARM cores. Thus, hashing on DPU requires more compute cycles than on the host. Next, we systematically evaluate different hash function single-core performance on the host and DPUs. We use SMhasher [1] to run six common hash functions [52].

Figure 10 shows the hash latency for small-sized input and hash bandwidth for large-sized input for different hash widths. We evaluate different hash widths to reflect their common use cases: 32-bit hashes in database systems and 64-bit or 128-bit hashes in key-value stores [20, 28], file systems, and file checksums [1]. Among the DPUs, BF-3 exhibits better hash performance than BF-2. For instance, BF-3 exhibits up to 1.97×, 2.71×, 9× higher bandwidth than BF-2 across `Murmur2`, `xxHash`, and `FNV`, respectively. The hash latency on BF-3 is 9.2%–69.42% lower than BF-2 across different hashing schemes. BF-2 achieves 1.84×–3× higher bandwidth than BF-1 for most hash functions. The enhanced performance of BF-3/2 can be attributed to its powerful SoC core.

Hashing performance on the x86-based host surpasses BF-1/2/3, with host cores consistently achieving higher bandwidth than DPU cores. SMhasher utilizes Single Instruction Multiple Data (SIMD) instructions like AVX [16]/CLMUL [12]/AES-NI [46] on x86 and NEON [4] on ARM to accelerate hashing. AVX and CLMUL instructions can accelerate cryptographic (`MD5`) and polynomial-based (`City` and `Murmur`) hashes, respectively. Compared to ARM's NEON, x86's AVX supports wider vector units [5] and can provide better hash performance. For instance, `City64` on host outperforms DPUs by 23×/11×/10× on BF-1/2/3. Hash latency is in the tens of nanoseconds for host, hundreds of nanoseconds for BF-3/2, and thousands of nanoseconds for BF-1 except for `FNV` and `MD5` hashes. Both `FNV` and `MD5` belong to the class of cryptographic hashes and, hence, have lower performance than other hash functions.

> **Off-path KVS Design Guidelines: A1.** The optimal placement of KVS client/server depends on the DPU generation and the performance gap between the DPU and the host. In our setting, with BF-3, HERD performs best with

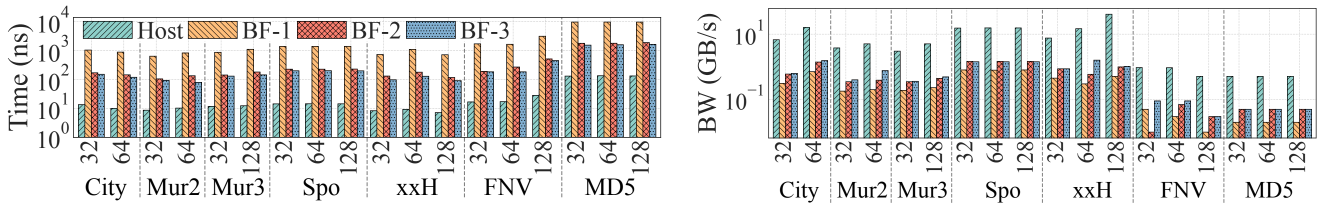
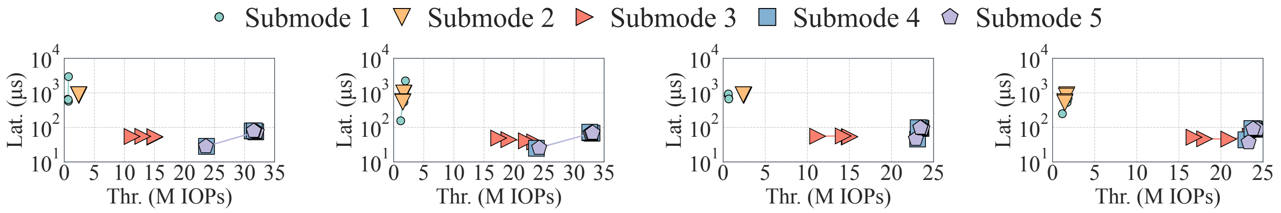


Figure 10: Hash performance on host and BF-1/2/3. X-ticks denote hash size (in bits) for an input. Hashing time and bandwidth are shown for 16 B and 256 KB-sized input, respectively. "Mur2", "Mur3", "Spo", and "xxH" denote Murmur2, Murmur3, Spooky, and xxHash, respectively.



(a) BF-2 100% GETs **(b) BF-3 100% GETs** **(c) BF-2 50% GETs** **(d) BF-3 50% GETs**

Figure 11: Performance of on-path DPU submodes on MICA KVS server. Note that the Y-axis is in log scale. We omit the graph for 95% GETs as performance trends were similar to 100% and 50% GETs workloads.

> both client and server on the host, while MICA achieves optimal performance with the client on the host and server offloaded to DPU. **A2.** Offloading the KVS server to DPU saves host resources and can enhance performance. In our setting, for HERD (BF-1/BF-2/BF-3) and MICA (BF-1/BF-2), server performance is similar whether hosted on DPU/host and when the client runs on another DPU. For MICA with BF-3, offloading the KVS server delivers the best performance. **A3.** A dedicated hash accelerator is needed on DPUs to bridge the performance gap with the host. On BF-3, `Murmur` hash has the lowest latency, while `xxHash` has the highest bandwidth.

4.2 On-path KVS study

In on-path mode, packet handling/forwarding are offloaded to either the DPU's SoC cores (`submodes 1–3`) or the eSwitch (`submodes 4–5`). We explore the impact of KVS packet handling in on-path DPU mode on TCP-based MICA KVS by evaluating the KVS server performance across five submodes. The KVS client runs on another host with its DPU configured in off-path mode. This ensures that the KVS server performance characteristics solely rely on the different on-path submodes of the DPU on the KVS server. The evaluations are conducted on BF-2 and BF-3 since they have superior on-path mode performance than BF-1, as discussed in § 3.2.1.

Figure 11 shows the KVS server performance of different on-path DPU submodes with BF-3 and BF-2 as we vary the KVS client threads for different workloads. In all cases, the KVS server utilizes all the available host cores (12 in our system). `Submode 1–2` exhibit poor performance on BF-2 and BF-3, achieving a maximum throughput of only 1.95 Mops and 2.45 Mops, with latencies reaching up to 2952 μs and 992 μs, respectively. The suboptimal performance in `submodes 1–2` stems from the SoC cores' inability to handle high-speed packet transfers, with `submode 2` slightly outperforming `submode 1` due to its userspace datapath.

`Submode 3` delivers up to 6.14×/9.8× higher throughput and significantly lower latency than `submode 2` across various workloads for BF-2/BF-3, due to its efficient packet forwarding via `DPDK-PMD` without OvS overheads. `Submodes 4–5` deliver the best KVS performance, with lower latency and high throughput, as the datapath is fully offloaded to the DPU's hardware switch, eliminating DPU ARM cores from the critical path. For example, `submodes 4–5` achieve up to 12×/16× and 2.1×/1.4× higher throughput when compared to `submodes 1–2` and `submode 3` on BF-2/BF-3 across all workloads, respectively. When comparing the KVS performance of `submodes 4–5` between BF-3 and BF-2, these submodes exhibit up to 12% lower latency on BF-3 compared to BF-2. Thus, the KVS packet forwarding in each on-path submode imposes different overheads on KVS, with BF-3's on-path `submodes 3–5` demonstrating a modest performance improvement over BF-2.

Table 5: Hardware design recommendations for future DPU generation based on observed idiosyncrasies. "DPU gen." indicates the generation where idiosyncrasy is observed.

Category	Recommendation	Idiosyncrasy	DPU gen.	Impact
Accelerator	Dedicated hashing accelerator	A3	BF-1/2/3	Accelerates hashing-dominated workloads
DMA	Enhancement of DMA engine	D2, D4, D5	BF-2/3	Improves DMA Read/Write performance
Network	Powerful on-path cores or compute units	N1 & A4	BF-1/2/3	Mitigates on-path submodes 1−3 overheads
Memory	Improve dTLB & L2 cache utilization	M1	BF-3	Reduces latency for 1 MB–4 MB blocks

> **On-path KVS Guidelines: A4.** KVS server should utilize on-path submodes 4−5 as it offers higher performance than submodes 1−3 and is comparable to KVS performance in off-path.

5 Discussion

Generalizability and transferability of our work. NVIDIA's BF DPUs are considered representative of other SoC-based DPUs like Marvell's OCTEON 10 DPU [31] and Broadcom's Stingray [7]. Other SoC-based DPUs can directly utilize our benchmarks. DPUDMABench is designed for NVIDIA BF-2/3 DPUs leveraging NVIDIA's DOCA SDK [40]. However, our approach of generating various DMA operations based on initiators and completions could be used to characterize DMA engines on other DPUs, like Alveo [50]. For Stingray, we believe on-path vs. off-path (N1-N2, A1-A2, A4) and memory (M2) idiosyncrasies would apply, while DMA-related ones (D1, D3, D5) would depend on the presence of the DMA engine. For firmware-based [30] and pipeline-based [3] DPUs, some of our findings may not be applicable due to architectural changes. However, our methodology for network and memory characterization can be applied to them, as it relies on widely available tools/software.

DPU mode/submode selection. On-path mode is typically used for offloading packet manipulation operations to DPU. For offloading performance-critical operations, using on-path submodes 4−5 is beneficial due to lower overheads than on-path submodes 1−3. We recommend offloading complex packet manipulation operations only if supported by submodes 4−5, given the hardware limitations of the eSwitch. Off-path mode could be used when offloading tasks other than packet manipulation to DPUs (e.g., KVS client/server).

Evolution of BF-1/2/3. Our study reveals that while certain aspects of BF have improved, others have regressed. For instance, the DMA engine's performance regressed, with BF-3 displaying higher latency and lower throughput compared to BF-2. Although BF-3 significantly improves memory bandwidth over BF-1/2, its memory latency remains high for medium block sizes. On-path submodes leveraging ARM cores, such as submode 3, show performance improvement over generations but still incur higher overhead compared to other submodes. BF-3 additionally contains a many-core RISC-V processor [41] in the network path, but its performance is significantly worse than the on-path ARM cores [8].

Future hardware design recommendations. Table 5 provides recommendations to DPU hardware designers based on our study. For accelerators, we find that hash computations on DPU cores are much slower than on host and recommend adding a dedicated hashing accelerator. For DMA engine, we find that DMA reads on BF-2/3 are slower than RDMA reads. Additionally, DMA operations on BF-3 show lower performance than BF-2 across various payloads. Thus, we advocate for enhancing the DMA engine and/or its driver. For network, we find that DPU's SoC cores (in on-path submodes 1−3) expose high overheads on the host for simple packet forwarding and recommend introducing powerful on-path cores or compute units. Finally, for memory, we observe the memory latency of BF-3 is 1.6×–7.3× higher than BF-2 due to a higher miss rate in dTLB and poor L2 cache utilization. We recommend improving the dTLB and L2 caching policy/mechanism to reduce memory access latency.

6 Conclusion and Future Work

This paper conducts 15 comprehensive benchmarking tests to uncover several nonintuitive idiosyncrasies across three generations of BlueField devices. We assess the network transport performance of on-path (and five submodes) and off-path DPU modes, along with memory characterization. We also propose DPUDMABench to evaluate DPU's DMA engine and its various APIs thoroughly. We perform two application case studies with KVS—for on-path and off-path DPU modes. Our findings on the evolution of DPU generations lead to the identification of 14 key idiosyncrasies and insights for the community. In the future, we plan to characterize additional applications on DPUs.

Acknowledgments

We would like to thank all the anonymous reviewers for their valuable feedback. This work was partly supported by NSF research grants OAC #2321123 and #2340982 and a DOE research grant DE-SC0024207.

References

[1] 2025. SMhasher. https://github.com/rurban/smhasher.

[2] 2025. Tinymembench. https://github.com/ssvb/tinymembench.

[3] AMD. 2025. AMD Pensando. https://www.amd.com/en/products/accelerators/pensando.html.

[4] ARM. 2025. Neon. https://developer.arm.com/Architectures/Neon.

[5] Emily Blem, Jaikrishnan Menon, and Karthikeyan Sankaralingam. 2013. Power struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and x86 Architectures. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*. 1–12. doi:10.1109/HPCA.2013.6522302

[6] Alex D. Breslow, Dong Ping Zhang, Joseph L. Greathouse, Nuwan Jayasena, and Dean M. Tullsen. 2016. Horton Tables: Fast Hash Tables for In-Memory Data-Intensive Computing. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. USENIX Association, Denver, CO, 281–294. https://www.usenix.org/conference/atc16/technical-sessions/presentation/breslow

[7] Broadcom. 2018. Stingray™ PS225 SmartNIC Adapters. https://datasheet.octopart.com/PS225---DUAL-PORT-25GBE-PCIE-ETHERNET-SMARTNIC-Avago-datasheet-116220963.pdf.

[8] Xuzheng Chen, Jie Zhang, Ting Fu, Yifan Shen, Shu Ma, Kun Qian, Lingjun Zhu, Chao Shi, Yin Zhang, Ming Liu, and Zeke Wang. 2024. Demystifying Datapath Accelerator Enhanced Off-path SmartNIC. arXiv:2402.03041 [cs.NI] https://arxiv.org/abs/2402.03041

[9] Biplob Debnath, Sudipta Sengupta, and Jin Li. 2010. FlashStore: High Throughput Persistent Key-Value Store. *Proc. VLDB Endow.* 3, 1–2 (sep 2010), 1414–1425. doi:10.14778/1920841.1921015

[10] Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, and Kaustubh Prabhu. [n. d.]. iPerf - The Ultimate Speed Test Tool for TCP, UDP and SCTP. https://iperf.fr/.

[11] Huansong Fu, Manjunath Gorentla Venkata, Ahana Roy Choudhury, Neena Imam, and Weikuan Yu. 2017. High-Performance Key-Value Store on OpenSHMEM. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 559–568. doi:10.1109/CCGRID.2017.49

[12] Shay Gueron and Michael E Kounavis. 2010. Intel® Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode. *White Paper* (2010), 10.

[13] Intel. [n. d.]. DPDK based Packet Generator. https://github.com/pktgen/Pktgen-DPDK.

[14] Intel. 2016. Open vSwitch* with DPDK Overview. https://www.intel.com/content/www/us/en/developer/articles/technical/open-vswitch-with-dpdk-overview.html.

[15] Intel. 2024. Poll Mode Driver. https://doc.dpdk.org/guides/prog_guide/poll_mode_drv.html.

[16] Intel. 2025. Intel® Advanced Vector Extensions 512. https://www.intel.com/content/www/us/en/architecture-and-technology/avx-512-overview.html.

[17] Nusrat Sharmin Islam, Dipti Shankar, Xiaoyi Lu, Md. Wasi-Ur-Rahman, and Dhabaleswar K. Panda. 2015. Accelerating I/O Performance of Big Data Analytics on HPC Clusters through RDMA-Based Key-Value Store. In *2015 44th International Conference on Parallel Processing*. 280–289. doi:10.1109/ICPP.2015.79

[18] Arpan Jain, Nawras Alnaasan, Aamir Shafi, Hari Subramoni, and Dhabaleswar K Panda. 2021. Accelerating CPU-based Distributed DNN Training on Modern HPC Clusters using BlueField-2 DPUs. In *2021 IEEE Symposium on High-Performance Interconnects (HOTI)*. 17–24. doi:10.1109/HOTI52880.2021.00017

[19] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2014. Using RDMA Efficiently For Key-Value Services. *SIGCOMM Comput. Commun. Rev.* 44, 4 (aug 2014), 295–306. doi:10.1145/2740070.2626299

[20] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2016. Design Guidelines for High Performance RDMA Systems. In *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. USENIX Association, Denver, CO, 437–450. https://www.usenix.org/conference/atc16/technical-sessions/presentation/kalia

[21] Sara Karamati, Clayton Hughes, K. Scott Hemmert, Ryan E. Grant, W. Whit Schonbein, Scott Levy, Thomas M. Conte, Jeffrey Young, and Richard W. Vuduc. 2022. "Smarter" NICs For Faster Molecular Dynamics: A Case Study. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 583–594. doi:10.1109/IPDPS53621.2022.00063

[22] Jongyul Kim, Insu Jang, Waleed Reda, Jaeseong Im, Marco Canini, Dejan Kostić, Youngjin Kwon, Simon Peter, and Emmett Witchel. 2021. LineFS: Efficient SmartNIC Offload of a Distributed File System with Pipeline Parallelism. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (Virtual Event, Germany) *(SOSP '21)*. Association for Computing Machinery, New York, NY, USA, 756–771. doi:10.1145/3477132.3483565

[23] Tonglin Li, Xiaobing Zhou, Kevin Brandstatter, and Ioan Raicu. 2013. Distributed Key-Value Store on HPC and Cloud Systems. In *2nd Greater Chicago Area System Research Workshop (GCASR)*, Vol. 238. 12.

[24] Yuke Li, Arjun Kashyap, Weicong Chen, Yanfei Guo, and Xiaoyi Lu. 2024. Accelerating Lossy and Lossless Compression on Emerging BlueField DPU Architectures. In *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 373–385. doi:10.1109/IPDPS57955.2024.00040

[25] Yuke Li, Arjun Kashyap, Yanfei Guo, and Xiaoyi Lu. 2023. Characterizing Lossy and Lossless Compression on Emerging BlueField DPU Architectures. In *2023 IEEE Symposium on High-Performance Interconnects (HOTI)*. 33–40. doi:10.1109/HOTI59126.2023.00019

[26] Yuke Li, Arjun Kashyap, Yanfei Guo, and Xiaoyi Lu. 2024. Compression Analysis for BlueField-2/-3 Data Processing Units: Lossy and Lossless Perspectives . *IEEE Micro* 44, 02 (March 2024), 8–19. doi:10.1109/MM.2023.3343636

[27] Hyeontaek Lim. 2024. MICA2. https://github.com/efficient/mica2.

[28] Hyeontaek Lim, Dongsu Han, David G. Andersen, and Michael Kaminsky. 2014. MICA: A Holistic Approach to Fast In-Memory Key-Value Storage. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*. USENIX Association, Seattle, WA, 429–444. https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/lim

[29] Marvell. 2025. Marvell LiquidIO III. https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-liquidio-III-solutions-brief.pdf.

[30] MARVELL. 2025. Marvell® LiquidIO™ III. https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-liquidio-III-solutions-brief.pdf.

[31] MARVELL. 2025. Marvell® OCTEON 10 DPU Platform. https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-octeon-10-dpu-platform-product-brief.pdf.

[32] John D. McCalpin. 2025. STREAM: Sustainable Memory Bandwidth in High Performance Computers. https://www.cs.virginia.edu/stream/.

[33] Benjamin Michalowicz, Kaushik Kandadi Suresh, Hari Subramoni, Dhabaleswar Panda, and Steve Poole. 2023. DPU-Bench: A Micro-Benchmark Suite to Measure Offload Efficiency Of SmartNICs. In *Practice and Experience in Advanced Research Computing* (Portland, OR, USA) *(PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 94–101. doi:10.1145/3569951.3593595

[34] Benjamin Michalowicz, Kaushik Kandadi Suresh, Hari Subramoni, Dhabaleswar K. DK Panda, and Steve Poole. 2023. Battle of the Blue-Fields: An In-Depth Comparison of the BlueField-2 and BlueField-3

SmartNICs. In *2023 IEEE Symposium on High-Performance Interconnects (HOTI)*. 41–48. doi:10.1109/HOTI59126.2023.00020

[35] Rolf Neugebauer, Gianni Antichi, José Fernando Zazo, Yury Audzevich, Sergio López-Buedo, and Andrew W. Moore. 2018. Understanding PCIe Performance for End Host Networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 327–341. doi:10.1145/3230543.3230560

[36] NVDIA. 2025. RDMA Stack Support on Host and Arm System. https://docs.nvidia.com/networking/display/bluefielddpuosv393/rdma+stack+support+on+host+and+arm+system.

[37] NVIDIA. 2020. NVIDIA Mellanox BlueField Data Processing Unit (DPU. https://network.nvidia.com/sites/default/files/doc-2020/pb-bluefield-dpu.pdf.

[38] NVIDIA. 2023. NVIDIA BLUEFIELD-2 DPU. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-2-dpu.pdf.

[39] NVIDIA. 2024. NVIDIA BLUEFIELD-3 DPU. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf.

[40] NVIDIA. 2024. NVIDIA DOCA Software Framework. https://developer.nvidia.com/networking/doca.

[41] NVIDIA. 2025. DPA Subsystem. https://docs.nvidia.com/doca/sdk/dpa+subsystem/index.html.

[42] NVIDIA. 2025. NVIDIA BlueField DPU Modes of Operation. https://docs.nvidia.com/doca/sdk/bluefield+modes+of+operation/index.html.

[43] OFED. 2024. Infiniband Verbs Performance Tests. https://github.com/linux-rdma/perftest.

[44] A Linux Foundation Collaborative Project. 2016. Open vSwitch. https://www.openvswitch.org/.

[45] A Linux Foundation Collaborative Project. 2016. Open vSwitch with DPDK. https://docs.openvswitch.org/en/latest/intro/install/dpdk/.

[46] Jeffrey Keith Rott. 2012. Intel® Advanced Encryption Standard Instructions (AES-NI). https://www.intel.com/content/www/us/en/developer/articles/technical/advanced-encryption-standard-instructions-aes-ni.html.

[47] Kaushik Kandadi Suresh, Benjamin Michalowicz, Bharath Ramesh, Nick Contini, Jinghan Yao, Shulei Xu, Aamir Shafi, Hari Subramoni, and Dhabaleswar Panda. 2023. A Novel Framework for Efficient Offloading of Communication Operations to Bluefield SmartNICs. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 123–133. doi:10.1109/IPDPS54959.2023.00022

[48] Teng Wang, Adam Moody, Yue Zhu, Kathryn Mohror, Kento Sato, Tanzima Islam, and Weikuan Yu. 2017. MetaKV: A Key-Value Store for Metadata Management of Distributed Burst Buffers. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 1174–1183. doi:10.1109/IPDPS.2017.39

[49] Xingda Wei, Rongxin Cheng, Yuhan Yang, Rong Chen, and Haibo Chen. 2023. Characterizing Off-path SmartNIC for Accelerating Distributed Systems. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*. USENIX Association, Boston, MA, 987–1004. https://www.usenix.org/conference/osdi23/presentation/wei-smartnic

[50] Xilinx. [n. d.]. Alveo: Adaptable Accelerator Cards for Data Center Workloads. https://www.xilinx.com/products/boards-and-kits/alveo.html.

[51] Tong Xing, Hesam Tajbakhsh, Israat Haque, Michio Honda, and Antonio Barbalace. 2022. Towards Portable End-to-End Network Performance Characterization of SmartNICs. In *Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems* (Virtual Event, Singapore) *(APSys '22)*. Association for Computing Machinery, New York, NY, USA, 46–52. doi:10.1145/3546591.3547528

[52] Zichen Zhu, Ju Hyoung Mun, Aneesh Raman, and Manos Athanassoulis. 2021. Reducing Bloom Filter CPU Overhead in LSM-Trees on Modern Storage Devices. In *Proceedings of the 17th International Workshop on Data Management on New Hardware* (Virtual Event, China) *(DAMON '21)*. Association for Computing Machinery, New York, NY, USA, Article 1, 10 pages. doi:10.1145/3465998.3466002