# A Comprehensive Study on Off-path SmartNIC

Xingda Wei, Rongxin Chen, Yuhan Yang, Rong Chen and Haibo Chen
*Institute of Parallel and Distributed Systems, Shanghai Jiao Tong University*

## Abstract

SmartNIC has recently emerged as an attractive device to accelerate distributed systems. However, there has been no comprehensive characterization of SmartNIC especially on the network part. This paper presents the first comprehensive study of off-path SmartNIC. Our experimental study uncovers the key performance characteristics of the communication among the client, SmartNIC SoC, and the host. We find without considering SmartNIC hardware architecture, communications with it can cause up to 48% bandwidth degradation due to performance anomalies. We also propose implications to address the anomalies.

## 1 Introduction

Remote Direct Memory Access (RDMA) has been widely adopted in modern datacenters [11, 37, 9], pushing the network speed (towards 400 Gbps [22]) and the distributed system performance [7, 40, 39, 33, 41] to the next level. However, the high-speed network requires more CPU resources to saturate a fast RDMA-capable NIC (RNIC) [20], which places a significant CPU burden on distributed systems [18]. One-sided RDMA can reduce the CPU pressures: the RNIC can read and write host memory in a CPU-bypass way. However, the limited offloading capabilities cause network amplifications and thus degrade the system performance [31, 15].

The continuous improvements of RDMA [34] and the essential power and memory walls of the CPU have led to the emergency of SmartNICs—the RNICs with programmable capabilities, which offer systems the opportunity to offload more complex computations to the NIC. Currently, there are two main types of SmartNICs. The first one is *on-path* SmartNIC [21], which directly exposes the processing units for handling RDMA packets (NIC cores) to the systems. Unfortunately, programming the low-level NIC cores with firmware [20, 31] and isolating the offloaded program with normal RDMA requests pose significant burdens on the developers. To ease development, *off-path* SmartNIC [27, 28, 4, 26] instead attaches a programmable multi-core SoC (with DRAM) next to the RNIC cores, which is off the critical path of RDMA. Thanks to this separation, the SoC is independent of the normal RDMA requests, which can further deploy a full-fledged OS to simplify system developments [18]. Specifically, developers can treat the SoC as a full-fledged server. Due to its generality and programmability, we focus on off-path SmartNIC[1] in this paper.

[1]We only refer to off-path SmartNIC in this paper, so we use the more generic but shorter term SmartNIC (and SNIC) throughout.

There have been some valuable studies in the literature on characterizing off-path SmartNIC [20, 19, 18, 35]. However, the extensively studied part is its offloading computation capabilities. A main finding from them is that the computing power of SmartNIC is wimpier than the host [20, 19, 18]. As a result, off-path SmartNIC does not improve the speed of a single network path, e.g., NIC to the host. For example, iPipe [20] shows that the path between host and SoC has a relative high latency due to the support for more developer-friendly RDMA.

While such studies are valuable to better utilize SmartNIC for distributed systems, they mainly focus on the computation side and essentially overlook many other impacting factors on overall performance, especially on the networking side. Networking side is important since SmartNIC essentially enables multiple communication paths for distributed systems. Besides widely-known offloading computation to the SmartNIC SoC, SmartNIC further supports a client to use RDMA for communicating with the host and the SoC, and exchanging data between the host and the SoC, respectively.

To this end, this paper conducts the first systematic study on characterizing the performance of communication paths of SmartNIC. Unlike previous studies that simply report basic performance numbers of SmartNIC [19, 18, 35], we systematically analyze the performance implications of SmartNIC architecture to different paths, including when and why one path is faster than the other, what is the bottleneck for each path, how heterogeneity of SoC brings performance anomalies to paths related to the SoC, and how paths interact with each other. The highlights of our results are:

- *Diverse performance characteristics between different paths.* RDMA path from the NIC to SoC is up to 1.48× faster than to the host.
- *SoC brings new performance anomalies to paths related to it.* The low-level hardware details of SoC, including memory access path and PCIe MTU, are different from the more powerful host CPU. Without considering such factors, RDMA requests involving SoC suffers from up to 48% bandwidth degradation.
- *Paths between SoC and host under-utilize the PCIe.* RDMA from SoC to host (and vice versa) crosses the NIC internal PCIe twice, can only utilize half of the PCIe bandwidth limit, and requires processing up to 6× PCIe packets than the others.

An important lesson learned is that when considering offloading with SmartNIC, we should not only consider its heterogeneous computation feature, but also its heterogeneous

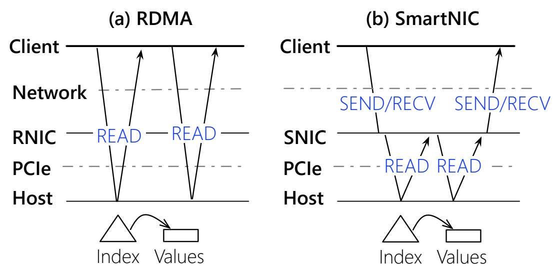

**Figure 1:** An illustration of a get request in a distributed in-memory key-value store that is accelerated by using either (a) RNICs (w/ network amplification) or (b) SNICs (w/o network amplification).

networking feature, which is the side-effect of the hardware architecture to support offloading. To begin our journal into investigations of high-performance SmartNIC, we start from a brief review of its development from RNIC.

## 2 Background and Context

### 2.1 RDMA-capable NICs (RNICs)

RDMA (Remote Direct Memory Access) is a low latency ($2\,\mu$s) and high bandwidth (200 Gbps) network protocol widely adopted in modern datacenters [11]. An intuitive way to enable RDMA is to accelerate message passing (e.g., RDMA-based RPC [14, 7, 5, 23, 17]) with *two-sided* primitives (SEND/RECV). RDMA further provides *one-sided* memory-offloading primitives—the NIC cores can READ-/WRITE[2] host memory bypassing the host CPU. Specifically, the NIC core internally uses the direct memory access (DMA) feature of the PCIe link to access the host memory.

Although RDMA has been used to boost the performance of many distributed systems [8, 40, 32, 16], usually by orders of magnitude, it still has the following two problems especially when the latest RNICs have further scaled to higher performance (e.g., 400 Gbps [22]).

**Issue #1: Host CPU occupation.** Distributed systems need non-trivial CPU resources to saturate a high-bandwidth NIC when using two-sided RDMA primitives. Our measurements show that saturating a 24-core server can only achieve 87 million packets per second (Mpps) on a 200 Gbps RNIC (ConnectX-6), while NIC cores can process more than 195 Mpps.[3] A recent work further shows that a distributed filesystem requires 2.27$\times$ CPU cores to handle network packets, when the network bandwidth scales from 25 Gbps to 100 Gbps [18]. Although deploying more powerful CPUs can alleviate this issue, RNIC bandwidth is also rapidly growing (and even faster), currently reaching up to 400 Gbps.

**Issue #2: Network amplification.** Using one-sided RDMA primitives alleviates pressure on the host CPU by allowing systems to offload memory accesses to the RNIC. However, the limited offloading capability constraints system performance, as a single request may involve multiple round trips of READs/WRITEs to complete, termed *network amplification*. Figure 1(a) exemplifies the execution of a *get* request on a distributed in-memory key-value store by using one-sided RDMA READs. The client first uses one (or multiple) READ(s) to query the index for a given key. Based on the pointer returned by the previous READs, an additional READ is used to retrieve the value.

[2]We use READ/WRITE to indicate RDMA READ/WRITE in this paper.
[3]Detailed hardware setup is described in §2.4.

### 2.2 From RNICs to SmartNICs

To address the issues of RNICs, SmartNIC adds the on-board memory (usually 4–64 GB) and exposes them to the systems, which enables offloading customized computations onto it. SmartNICs can be categorized as follows.

**On-path SmartNIC.** As shown in Figure 2(b), the on-path SmartNIC exposes the NIC cores to the systems with low-level programmable interfaces, allowing them to directly manipulate the raw packets. As its name implies, the offloaded code is *on* the critical path of the network processing pipeline. Example NICs include Marvell LiquidIO [21] and Netronome Agilio [24]. The benefit is that the offloaded code is closer to the network packets. Therefore, inline requests that only interact with the NIC, such as writing to on-board memory (②), are extremely efficient [20, 31].

However, the on-path SmartNIC has two limitations. First, the offloaded code competes for NIC cores with the network requests sent to the host (①). Further, if too much computation is offloaded onto it (④), the network performance of the host also suffers a significant degradation [20]. Second, programming on-path NICs is hard because they typically only expose a low-level interface for raw packet manipulation.

**Off-path SmartNIC.** As shown in Figure 2(c), the off-path SmartNIC offers an alternative by packaging additional compute cores and memory in a separate SoC next to the NIC cores, where the offloaded code is *off* the critical path of the network processing pipeline. From the NIC perspective, the SoC can be viewed as a second full-fledged host with an exclusive network interface. To bridge the NIC cores, SoC and host together, a PCIe switch is integrated inside the SmartNIC to dispatch network packets. Example NICs include Mellanox Bluefield [27, 28] and Broadcom Stingray [4].

Compared to on-path counterparts, the offloaded code does not affect the network performance of the host as long as it does not involve network communications (②). Thanks to this clear separation, the SoC can run a full-fledged kernel (e.g., Linux) with a full network stack (i.e., RDMA), which further simplifies system development and allows for offloading more complex tasks [18]. However, leveraging off-path SmartNICs to accelerate distributed systems is often more challenging than using on-path counterparts. This is because the PCIe switch prolongs all communication paths (i.e., ①, ②, and ③), causing potential performance degradation.

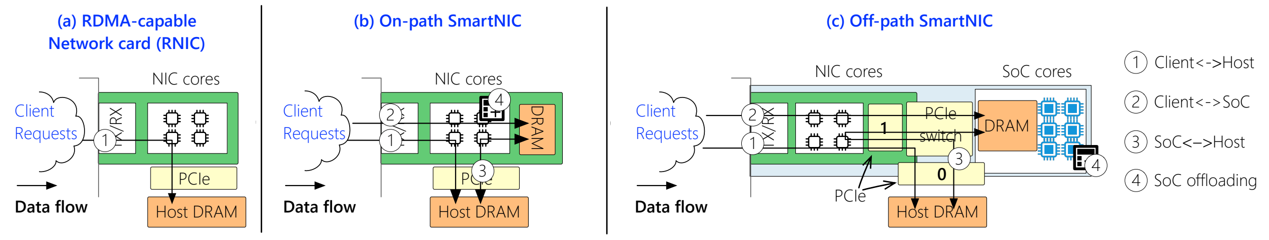

**Figure 2:** Architecture overview of different NICs: (a) RDMA-capable NIC (RNIC), (b) on-path SmartNIC, and (c) off-path SmartNIC.

**Table 1:** Hardware description of Bluefield-2 [27].

| Component | Description |
|---|---|
| NIC cores | ConnectX-6 (2× 100 Gbps RDMA ports) |
| SoC cores | ARM Cortex-A72 processor (8 cores, 2.75 GHz) |
| SoC memory | 1× 16 GB of DDR4-1600 DRAM |
| PCIe1 | PCIe 4.0 ×16 (256 Gbps bandwidth) |

## 2.3 Target SmartNIC: Mellanox Bluefield-2

We conduct our study on Bluefield-2, a typical off-path SmartNIC optimized for offloading general-purpose computations. Figure 2(c) illustrates the overall architecture of Bluefield, and Table 1 lists its detailed hardware configuration.

**Hardware.** Bluefield equips a mature RNIC (ConnectX-6) as its NIC cores for high-speed networking, which supports all RDMA operations. The programmability comes from an integrated on-borad SoC, which has an 16 GB DRAM and an ARM Cortex-A72 (8 cores, 2.75 GHz). A PCIe 4.0 switch bridges the NIC cores, SoC and the host together, which supports a bi-direction 256 Gbps bandwidth. Note that the SoC is directly linked to the switch, not via PCIe. The Bluefield hardware counters [29] suggest that Bluefield only has two internal PCIe channels: one linking RNIC with the switch (PCIe1) and the other linking the switch with the host (PCIe0). To the best of our knowledge, how the PCIe switch interacts with the SoC is undocumented.

**Software.** The on-board SoC runs a full-fledged Linux such that developers can treat it as a normal ARM server. The kernel further hosts a full network stack for RDMA, allowing the offloaded tasks interacting with others via RDMA.

**Communication primitive: RDMA.** All SoC-related communications are done using RDMA to simplify system development. Specifically, as shown in Figure 2(c) the clients can issue one-sided or two-sided RDMA requests to the SoC, just like a twin server on the host (②). Meanwhile, the SoC can also interact with the host via RDMA and vice verse (③). However, exchanging data between the SoC and the host is tricky because it must pass through the RNIC (PCIe1 and NIC cores) for RDMA support. We observe that such a restriction adds a hidden bottleneck to this path (③).

**Existing states of exploring Bluefield.** Previous studies [20, 19, 18, 35] focus on the computing power of Bluefield (④ in Figure 2), revealing its SoC cores are relatively weak—both in terms of performing offloaded tasks and sending network requests—because the frequency and number of cores are inferior to the host CPU. The relative performance between the SoC and the host is unlikely to change due to power constraints of SmartNIC, so we take it as a premise during our investigation.

In contrast, few studies have considered various communication patterns in Bluefield (i.e., ①, ②, and ③), which is the focus of this paper. Thostrup *et al.* [35] found that the latency of accessing the SoC memory (②) by using READ is shorter than that of accessing the host memory (①). iPipe [20] shows that using RDMA to communicate between the host and SoC (③) has higher latency than using DMA due to the software overhead of supporting RDMA. This paper systematically explores the performance characteristics of Bluefield and summarizes insightful lessons and advices for improvement.

## 2.4 Notation and testbed

**Notations.** When describing hardware details related to Bluefield, we follow its hardware specification and use PCIe1 to denote the PCIe channel that links NIC cores with the PCIe switch, PCIe0 as the channel linking the switch and host's PCIe controller (see Figure 2(c)). We also refer the ARM cores along with the on-chip memory of Bluefield as SoC and, without loss of generality, its hosting server as host. Besides, we call the machine issuing the RDMA requests requester and the destination hardware component responder. Using Figure 2(c) as an example, the requesters of ① and ② are any RDMA-capable machines (also called clients), and the responders are the host and SoC, respectively. For ③, the requester and responder are the host and SoC, respectively, and vice verse.

**Testbed.** Table 2 presents the machine configurations on our testbed. To best utilize SmartNIC, we deploy Bluefield on the servers (SRV) with matching PCIe link (PCIe 4.0) by default. These machines can also replace Bluefield with 200 Gbps ConnectX-6 (RNIC) for comparisons. Other machines (CLI) serve as clients that issue RDMA requests to the servers. All machines in SRV and CLI are connected through one Mellanox SB7890 100 Gbps InfiniBand Switch. Note that this does not limit the network performance of 200 Gbps NICs since they both connect to the switch with two 100 Gbps ports.

**Table 2:** Machine configurations in our local rack-scale testbed.

| Name | Nodes | RDMA-capable NIC | Host PCIe (PCIe0) | Host CPU | Host Memory |
|---|---|---|---|---|---|
| SRV | 3 | 1× ConnectX-6 (200 Gbps)<br>1× Bluefield-2 (200 Gbps) | PCIe 4.0 ×16 (256 Gbps) | 2× Gold 5317 v4 (12 cores, 3.6 GHz) | 128 GB DDR4-2933 |
| CLI | 20 | 1× ConnectX-4 (100 Gbps) | PCIe 3.0 ×16 (128 Gbps) | 2× E5-2650 v4 (12 cores, 2.2GHz) | 96 GB DDR4-1600 |

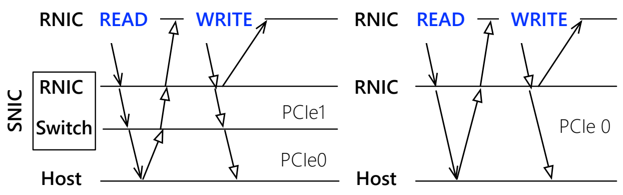

**Figure 3:** The exec. flow of READ/WRITE on SNIC and RNIC.

## 3 Characterizing SmartNIC Performance

As mentioned in §2.3, it is well-known that NIC's *computing power* is wimpier than the host CPU. Therefore, we focus on the analysis of the *communication efficiency* of SmartNIC. Figure 4 shows the end-to-end latency and peak throughput of sending different RDMA requests (READ, WRTIE, and SEND/RECV) by using RNIC and SmartNIC with different communication paths.

**Evaluation setup.** We perform our evaluations on the cluster described in Table 2 using a state-of-the-art RDMA communication framework [40]. For one-sided operations, the requester communicates with one responder with RDMA's reliable connection queue pairs (QPs). The responder addresses are randomly selected from a 10 GB address space by default. For two-sided operations, the responder implements an echo server that uses all available cores to handle messages, and the requester communicates with it using unreliable datagram QPs (UD) for better two-sided performance [16, 40, 17]. For end-to-end latency, one requester machine is used to prevent interferences from queuing effects, while for peak throughput, up to eleven requester machines are used to saturate the responder machine. Furthermore, we apply known RDMA-aware optimizations, including address alignment [42], unsignaled requests [14] and huge pages [7], to prevent interference from misuse of RDMA.

### 3.1 Communication from Client to Host (①)

**Latency.** For communication with the host, we focus on an apple-to-apple comparison between Bluefield-2 (SNIC ①) with ConnectX-6 (RNIC ①), because they share the same NIC cores [27]. Their performance gap can best illustrate the "performance tax" paid by the SmartNIC architecture. As shown in Figure 4, for READ, WRITE, and SEND/RECV, SNIC ① has 15–30%, 15–21%, and 6–9% higher latency than RNIC ①, respectively. The increased latency of SNIC is mainly due to the PCIe switch and PCIe1 added between the host and NIC cores, with a one-way overhead of 150–200 ns [36], which is non-trivial for small RDMA requests (1–2 $\mu$s). READ has a higher absolute latency increase than WRITE (0.6 $\mu$s vs. 0.4 $\mu$s) because READ passes the PCIe twice (once for request and once for completion), while WRITE omits the completion [25], as shown in Figure 3. The latency increase of SEND/RECV comes mainly from additional CPU costs at the responder and therefore is not significant compared to READ/WRITE.

**Throughput.** A direct impact of the increased latency is a considerable drop in peak throughput for small requests, because NIC cores stall longer in its pipeline to wait for PCIe requests to complete. As shown in Figure 4, for READ, WRITE, and SEND/RECV, SNIC ① has 19–26%, 15–22%, and 3–36% lower throughput than RNIC ① for payloads less than 512 bytes, respectively. The result of larger requests is similar to using RNIC since the network bandwidth becomes the bottleneck.

**Bottleneck.** The lowest bandwidth limit of NIC, PCIe1, and PCIe0 will first become the bottleneck for communication from client to host. On our testbed, Bluefield-2's 200 Gbps NIC is the performance bottleneck. Moreover, we find an interesting phenomenon that the total throughput of requester can approach *twice* the bandwidth limit of the host because the links are *bi-directional* [30]. Specifically, if packets flow in opposite directions, e.g., the READ and WRITE packets in Figure 5(a), they can be multiplexed on the same link. As an example, we dedicate two requesters (each with 12 threads to saturate the bandwidth) to issue 4 KB packets. As shown in Figure 5(b), if two clients send READ and WRITE requests separately, a total of 364 Gbps bandwidth is reached on a 200 Gbps NIC (see READ+WRITE of SNIC ①). In contrast, if both clients send the same type of requests (either READ or WRITE), only about 190 Gbps is reached.

**Takeaways.** Being "smart" pays a non-trivial performance degradation for communicating with the host for small requests. We demonstrate that extending RNIC (ConnectX-6) to SNIC (Bluefield-2) causes performance degradation by up to 36% and 30% in throughput and latency, respectively, for requests with small payloads. Therefore, we need to utilize the alternative communication paths provided by SmartNICs.

### 3.2 Communication from Client to SoC (②)

**Latency.** For sending requests from the client to SoC (SNIC ② in Figure 4), instead of the host (SNIC ①), the latency of READ decreases by up to 14% due to without passing through PCIe0, but is still 4–15% higher than RNIC since it still needs to pass through the PCIe switch. For WRITE,

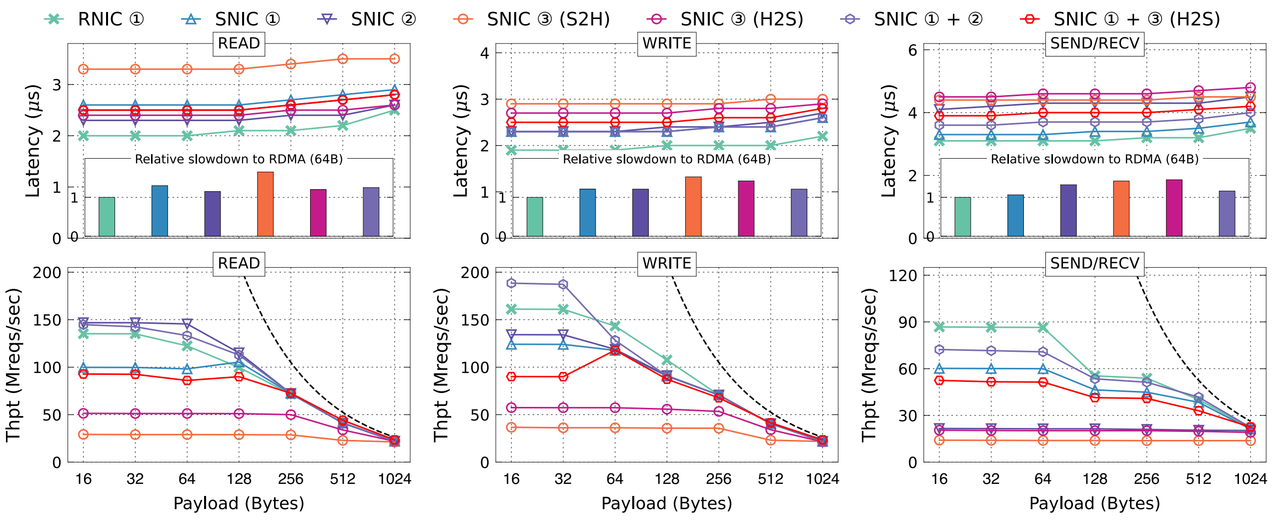

**Figure 4:** The end-to-end latency (upper) and peak throughput (lower) of random inbound RDMA requests on different NICs. The symbols ①, ②, and ③ in the legend correspond to the communication paths listed in Figure 2.

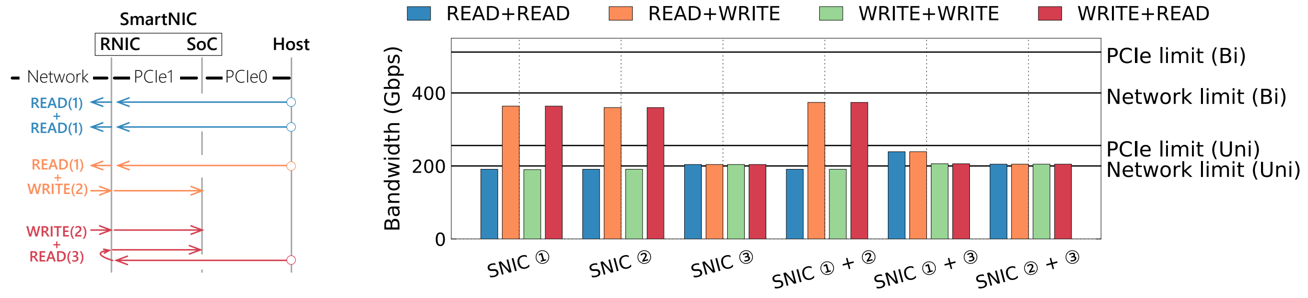

**Figure 5:** (a) An illustration of three examples of different combinations of data flows on different paths, and (b) the peak throughput of different combinations of data flows on different communications paths

SNIC ② provides similar performance to SNIC ① due to asynchronously sending completion by NIC cores (see Figure 3). For SEND/RECV, SNIC ② has 21–30% higher latency than SNIC ①, due to the lower computing power of SoC.

**Throughput.** For READ and WRITE, due to less cross-PCIe penalties, SNIC ② has better throughput than SNIC ①, reaching 1.08–1.48× for payloads less than 512 bytes. Interestingly, for READ, SNIC ② is even observably higher than RNIC ① before saturating the network bandwidth. For this undocumented results, we suspect that it is due to the closer packaging of SoC memory and the PCIe switch. Note that a confident analysis relies on the hardware details of Bluefield, which unfortunately are not available to us. For WRITE, SNIC ② is still lower than that of RNIC ①. We attribute this to two reasons. First, SoC has fewer DRAM channels compared to the host (1 vs. 8), limiting the concurrency of write accesses. However, READ is not affected because read accesses on DRAM are faster than write accesses [12, 38]. Second, SoC can only utilize a portion of NIC cores. Finally, for SEND/RECV, the throughput drops by up to 64%, again due to the wimpy SoC cores.

**Bottleneck.** As shown in Figure 2(c), since SNIC ② only flows through NIC and PCIe1, the bottleneck is their lower bandwidth limit, which is still Bluefield-2's 200 Gbps NIC. Therefore, as shown in Figure 5(b), the performance of SNIC ② is also similar to that of SNIC ①, namely the total of 400 Gbps and 200 Gbps bandwidth for opposite direction and same direction communication, respectively.

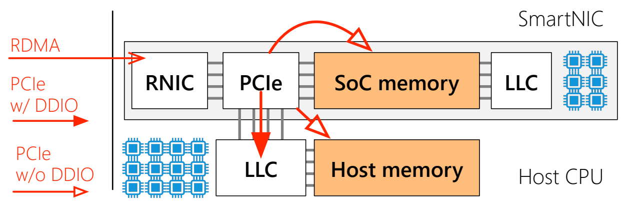

**Figure 6:** Different paths to access host memory and SoC memory.

**Advice #1: Avoid skewed memory accesses.** The wimpy SoC cores may impact the memory access behavior of one-sided RDMA primitives, because it usually supports fewer features compared to the more powerful host CPU cores. Specifically, Data Direct I/O (DDIO) [13] is widely supported by the host CPUs, which allows the NIC to directly read-/write data from/to its last level cache (LLC), as shown in Figure 6. SoC cores may also equip with similar features

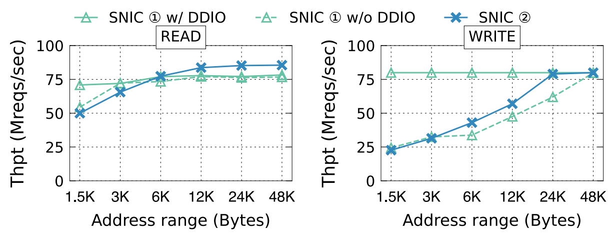

**Figure 7:** The peak throughput of accessing host memory and SoC memory via SNIC, READ (a) and WRITE (b).

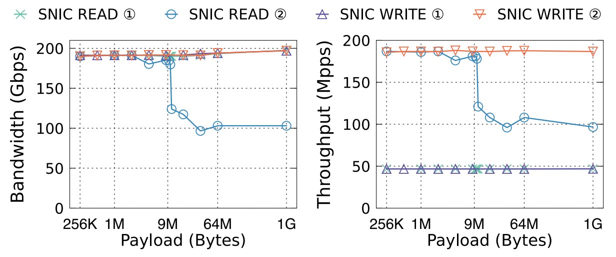

**Figure 8:** The bandwidth (a) and PCIe packet throughput (b) for accessing (READ and WRITE) the host (SNIC ①) and SoC (SNIC ②) via SmartNIC. Note that, for brevity, we omit the result of SEND/RECV since it is the same as WRITE for large payloads [14].

(e.g., ARM CCI [1]), but it is still vendor-specific. The SoC cores on our testbed (ARM Cortex-A72 in Bluefield-2) do not support DDIO. It makes one-sided RDMA requests more vulnerable to skewed memory accesses—the requested memory addresses fall into a small range. This is because DRAM requires to utilize all memory modules concurrently, while LLC can tolerate skewed access better by accessing DRAM more quickly.

Figure 7 shows the peak throughput of accessing host memory and SoC memory via SNIC with the increase of address ranges.[4] For WRITE, the throughput of SNIC ② using SoC cores drops to 22.7 M reqs/s (rom 77.9 M reqs/s) when address range decreases to 48 KB (from 1.5 KB). In contrast, the performance of SNIC ② using Host CPU cores is hardly affected when DDIO is enabled. For READ, the degradation is relatively smaller. the throughput of SNIC ② drops from 85 M reqs/s to 50 M reqs/s when decreasing the range from 48 KB to 1.5 KB. This is because DRAM can serve reads faster than writes [12, 38].

**Advice #2: Avoid large READ requests.** It is common practice to fully exploit network bandwidth by using large requests. For example, using requests with payloads larger than 16 KB is enough to saturate a 200 Gbps RNIC. However, we observe that the READ performance of SNIC ② collapses at request payload size above 9 MB, as shown in Figure 8(a). The reason is that NIC cores suffer from head-of-line blocking when processing large READ requests. For a READ request, the NIC issues a PCIe read transaction to fetch the data, which is further segmented into multiple PCIe packets. The maximal size of a PCIe packet is determined by the PCIe Maximum Transfer Unit (MTU), which is negotiated by the linked hardware devices during bootstrap [25]. Table 3 lists the PCIe MTU on our testbed. SoC cores (the endpoint of SNIC ②) use a smaller PCIe MTU (128 B) due to its lower computing power. Therefore, processing a large READ request sent to SoC memory (SNIC ②) must wait for more PCIe packets to arrive, resulting in lengthy processing stalls. As shown in Figure 8(b), for READ requests with payloads larger than 9 MB payloads, the throughput of issuing PCIe packets to SoC (SNIC ②) drops dramatically to less than 120 Mpps (from 186 Mpps). Note that WRITE requests are not affected since DMA does not wait for the completion [43, 25].

**Table 3:** PCIe Maximum Transfer Unit (MTU) on our testbed, and the number of PCIe packets required to transfer $N$ bytes through different communication paths of Bluefield. Our simplified model omits control path packets (e.g., two-sided message arrival notification).

| | Host CPU cores ($H_{\text{MTU}}$) | SoC cores ($S_{\text{MTU}}$) |
|---|---|---|
| **PCIe MTU** | 512 B | 128 B |

| | SNIC ① | SNIC ② | SNIC ③ |
|---|---|---|---|
| **PCIe1** | $\lceil N/H_{\text{MTU}} \rceil$ | – | $\lceil N/H_{\text{MTU}} \rceil + \lceil N/S_{\text{MTU}} \rceil$ |
| **PCIe0** | $\lceil N/H_{\text{MTU}} \rceil$ | $\lceil N/S_{\text{MTU}} \rceil$ | $\lceil N/H_{\text{MTU}} \rceil$ |

On the contrary, the host uses a larger PCIe MTU (512 B), which avoids this issue for large requests over SNIC ①. As shown in Figure 8(b), the NIC can issue 46.7 million PCIe packets per second to the host (SNIC (①). The aggregated bandwidth reaches 191 Gbps, bottlenecked by the network.

**Takeaways.** For READ and WRITE, sending requests to SoC is typically faster than that to the host (or even faster than via RNIC) because SoC is “closer” to the NIC (without PCIe0). In contrast, using SEND/RECV to communicate with SoC is slower due to wimpy SoC cores. Furthermore, designers still need to carefully consider the differences between host CPU cores and SoC cores to avoid performance anomalies, especially for one-sided RDMA primitives. Specifically, skewed memory accesses may degrade the performance due to the lack of DDIO support in SoC cores. In addition, sending large READ requests to SoC can cause head-of-line blocking and collapses the performance, so the request should be proactively segmented into smaller ones.

## 3.3 Communication between SoC and Host (③)

**Latency.** As shown in Figure 10(a)), the latency of sending requests from SoC to the host (SNIC ③ S2H) is very high, especially for READ, since the requester (SoC) takes longer to issue an RDMA request to the NIC. The latency in the opposite direction (from the host ot SoC, SNIC ③ H2S) is reduced but still 4–17% higher than SNIC ②. Although the intra-machine communication saves one network round-trip, it adds additional PCIe transfers. Specifically, the request on SNIC ② flows the requester-side PCIe (not shown in Fig-

[4]Note that we attach Bluefield to CLI machines for the evaluation because we are unable to disable DDIO on the SRV machines.

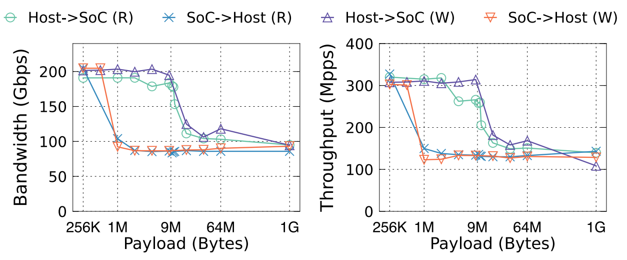

**Figure 9:** The bandwidth (a) and PCIe packets throughput (b) for sending (R)EAD/(W)RITE requests between the host and SoC.

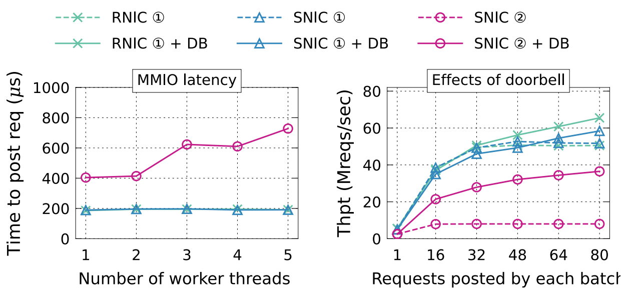

**Figure 10:** The latency of posting requests to NICs (a) and the impact of doorbell batching (DB) on the requester.

ure 2(c)), the network, PCIe1, and the PCIe switch, while the request on SNIC ③ (H2S) flows PCIe0, the PCIe switch, PCIe1 twice (in and out), and the PCI switch (again).

**Throughput.** For requests with payloads less than 512 bytes, the throughput of SNIC ③ (both S2H and H2S) is bottlenecked by the requester's processor. Since a single requester machine (either SoC or the host) cannot saturate the NIC with small requests[5], the READ throughput of SNIC ③ only reaches 29 M reqs/s and 51.2 M reqs/s for S2H and H2S, respectively, which is still far from its limit. For WRITE and SEND/RECV, the results are similar. For larger requests, the throughput is restricted by the PCIe bandwidth.

**Bottleneck.** As shown in Figure 5(b) for packets flowing in a single direction, communication between host and SoC is bottlenecked by PCIe bandwidth (256 Gbps) rather than the uninvolved NIC (200 Gbps). Therefore, the peak bandwidth of SNIC ③ is slightly higher than SNIC ① and ② (204 Gbps vs. 191 Gbps). Readers might be interested in why the results of SNIC ③ cannot be close to 256 Gbps. We suspect that it requires much more PCIe packets than the others. For packets flowing in opposite directions, SNIC ③ can only achieve the same results instead of reaching twice the limit as the other paths (i.e., SNIC ① and ②), since each request passes through PCIe1 twice (in and out), exhausting the bi-directional link.

**Advice #3: Avoid large READ/WRTIE requests.** Communications between the host and SoC (SNIC ③) also suffers from head-of-line blocking for large READ requests like SNIC ②, and this issue further appears with large WRITE requests because the SmartNIC must first read data from the requester and then write it to the responder. As shown in Figure 9(a), the READ/WRITE performance of SNIC ③ collapses to about 100 Gbps for large requests. Table 3 shows the number of PCIe packets required to transfer N bytes via different communication paths. For SNIC ③, the NIC would generates more packets due to passing through PCIe1 twice. Further, the performance of S2H collapses earlier than H2S as it will passes through PCIe1 first. Suppose we transfer data at 200 Gbps from SoC to the host. The SoC cores first transfer 195 M PCIe packets per second (pps) to the NIC (PCIe1), then the NIC forwards data back to the PCIe switch via PCIe1 again with 49 Mpps (the host supports 512 B MTU), and finally, the switch forwards 49 Mpps through PCIe0. Therefore, SmartNIC should process at least 293 Mpps for transferring data at 200 Gbps, which is 6× and 1.5× higher than SNIC ① and SNIC ②, respectively. This is further confirmed by our empirical measurements using hardware counters. As shown in Figure 9(b), for sending 256 KB READ requests from SoC to the host, the bandwith reaches 204 Gbps, and the NIC transfers about 320 M PCIe packets per second.

[5]We use up to eleven requester machines for SNIC ① and SNIC ②.

**Advice #4: Enable doorbell batching carefully.** The time of posting each request to the NIC is dominated by Memory-Mapped IO (MMIO) [40]. The SoC suffers a high MMIO latency when communicating with the host (see Figure 10(a)). A known optimization is doorbell batching (DB) [15]: to send a batch of $N$ requests, the requester first links them together in memory, then use one MMIO to ask the NIC to read these requests with DMA in a CPU-bypass way. DB reduces the number of MMIOs required from $N$ to 1. Thus, for RNIC ① and SNIC ①, DB is always helpful and can bring 2–30% performance improvement (see Figure 10(b)). For the communication between host and SoC (SNIC ③), DB is still helpful at the SoC-side. As shown in Figure 10(b), when sending a batch of READs to the host, DB improves the SoC performance by 2.7–4.6× for batch sizes 16–80. The huge improvement is partly due to the CPU-bypass feature of DMA, and also because the NIC is faster in using DMA to read requests stored on SoC memory (see §3.2). However, DB is not always helpful at the host-side, as it is slower to read host memory using NIC DMA (see §3.1). For batch sizes of 16, 32, and 48, DB decreases the throughput of host-SoC communication by 9%, 7%, and 6%, respectively.

**Takeaways.** First, enabling doorbell batching is critical for SNIC ③ at the SoC side, but is negative at the host side for small batch sizes. Second, SNIC ③ has different bottlenecks than SNIC ① and SNIC ②. It is always bottlenecked by the uni-directional bandwidth of PCIe, while others are limited by the minimal bi-directional bandwidth of PCIe and the NIC. If this factor is not adequately considered, distributed systems are likely to underutilize the NIC bandwidth. Finally, we should avoid transferring large requests, either READ or WRITE, between the host and SoC.

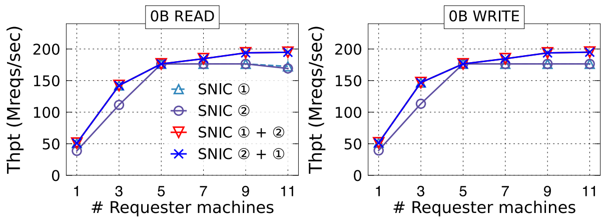

**Figure 11:** PCIe packets throughput for (a) READ and (b) WRITE with the increase of requester machines.

## 4 Concurrent communication paths

**Concurrent communication with the host and the SoC (①+②).** We focus on the throughput results (see the lower part of Figure 4) since the latency results are roughly the average of the two paths. We evaluate the peak throughput by assigning half of the clients to send request to the host and the SoC separately. We can see that the total peak throughput of concurrently using them (SNIC ①+②) is typically faster than each of them. For READ, WRITE, and SEND/RECV, SNIC ①+② outperforms the lower of them by up to 1.45×, 1.50×, and 3.3×, respectively. Specifically, for SEND/RECV, two responders (host and SoC) have more CPU cores.

However, the READ/WRITE performance improvement is non-intuitive and undocumented, since two paths should compete for NIC cores. This may be because the SmartNIC internally reserves some NIC cores for each endpoint. Therefore, sending requests to the host and the SoC concurrently can further increase *peak* throughput by enabling more NIC cores. We design a microbenchmark to confirm this assumption, which first increases the requester machines to saturate the NIC and then changes the responder. All requests use 0 B payload to avoid issuing any DMA [2] and thus return before reaching PCIe1. For READ, five requester machines are sufficient to saturate NIC cores when using SNIC ① or SNIC ② alone. Therefore, for concurrently using SNIC ① and SNIC ②, we first dedicate five requester machines for one responder, and then add requester machines for the other responder. Both cases (SNIC ①+② and SNIC ②+①) offer similar performance, with 4–13% and 5–10% higher throughput than using SNIC ① or SNIC ② alone. For WRTIE, all results are almost the same.

Finally, as expected, the aggregated throughput of the two paths (SNIC ① and SNIC ②) is much higher than concurrently using them (352 Mpps vs. 195 Mpps), indicating that most NIC cores are still shared, i.e., each can communicate with two endpoints, and only a few is dedicated. This also implies that concurrently using multiple resources of SmartNIC is non-trivial.

**Concurrent inter- and intra-machine communication (①/②+③).** There exist four concurrent combinations of inter- and intra-machine communication. Due to space limit, we only report the results of SNIC ①+③/H2S here. For the other combinations, similar behaviors are observed. To study concurrently using two paths, we first use enough clients (five requester machines) to saturate the NIC for SNIC ①, and then start requester at the host (24 requester threads) to send RDMA requests to the SoC (SNIC ③/H2S). We observe that concurrently enabling intra-machine communication degrades the performance of inter-machine communication. As shown in Figure 4, for READ, WRITE, and SEND/RECV, the throughput of small requests (less than 512 bytes) drops 7–15%, 4–27%, and 9–14%, by comparing SNIC ① and SNIC ①+③(H2S). For large requests, the performance is always bottlenecked by the network bandwidth, resulting in similar results. The host-SoC communication (SNIC ③) affects other communication paths on the SmartNIC, since it relies on the NIC (PCIe1 and the PCIe switch) to bridge host and SoC and causes more PCIe packets for each RDMA request.

**Bottleneck.** Assuming each path has only one type of request, e.g., READ or WRITE. For SNIC ①+②, each part has the same bottleneck (the NIC), so the bandwidth limit is 400 Gbps (bi-directional) on our testbed. For SNIC ①+③, it is bottlenecked by SNIC ③, which is limited on the uni-direction of PCIe (256 Gbps) since it occupies both directions of PCIe1 (see Figure 5(b)). Nevertheless, if SNIC ① is used in opposite directions (i.e., READ and WRITE), SNIC ①+③ can reach a higher limit. For example, the aggregated bandwidth can achieve 456 Gbps if we restrict the bandwidth of data transfer on SNIC ③ to 56 Gbps. This suggests that selectively offloading small portion of data to SoC may be optimal.

**Takeaways.** Sending requests from clients to the host and the SoC concurrently (SNIC ①+②) can best utilize NIC cores to handle small RDMA requests, especially when used in opposition directions (e.g., one for READ and one for WRITE). On the contrary, uncontrolled use of intra-machine (host-SoC) communications (SNIC ③) may harm inter-machine communications, which is the intrinsic purpose of using SmartNIC. Specifically, if the uni-directional bandwidth of PCIe is smaller than the bi-directional bandwidth of the NIC, using SNIC ③ will introduce a hidden bottleneck. Therefore, we should always consider using SNIC ③ only when spare resources are made available. For example, if the inter-machine communication saturates the NIC, the bandwidth used by SNIC ③ should no larger than $P - N$, where $P$ and $N$ are the limit of the PCIe and the network, respectively. On our testbed, it should be 56 Gbps.

Finally, in real-world distributed systems, it is common that a single communication path cannot saturate all resources of SmartNIC. For example, SNIC ② is the fastest but limited by small memory and wimpy cores on the SoC. On the other hand, only using SNIC ① as RNIC would waste all resources on the SoC. Therefore, we should concurrently use multiple paths provided by the SmartNIC, but carefully avoid interference between them.

## 5 Discussion

**Other SmartNICs.** Although this paper focuses on one particular SmartNIC (Bluefield-2 [27]), we believe that our results generalize to others, since they all share the same hardware architecture as Bluefield-2, namely extending RNIC (e.g., Stingray PS225 [4] extends NetXtreme 100 Gbps RNIC [3]), attaching a heterogeneous SoC, and bridging SoC and RNIC together with a PCIe switch. The next generation of Bluefield (Bluefield-3) also adopts the same architecture, except for using a faster RNIC (400 Gbps ConnectX-7), PCIe (5.0), and SoC (ARMv8.2+ A78). Even though other NICs may have different parameters than Bluefield-2, our methodology, analysis tools, and performance models (e.g., Table 3) also apply to them.

**Suggestions.** Several anomalies uncovered by our study can be mitigated via a combination of new techniques, which we suggest vendors consider so. For example, supporting CXL [6] can significantly improve the PCIe utilization between the host and SoC, but needs strong cooperation between SoC OS and host OS to do so in a programmer-friendly way [10]. To the best of our knowledge, no SmartNIC supports CXL yet. Meanwhile, vendors can support CCI [1] to mitigate the write skew problem mentioned in §3.2.

## 6 Conclusion

Designing high-performance distributed systems with SmartNIC requires a deep understanding of its low-level hardware details. Besides the extensively studied computation capabilities, we show the heterogeneity of computation can also significantly impact to the networking related to SmartNIC. By presenting clear advices with extensive measurements, developers can avoid networking anomalies related to SmartNIC.

## References

[1] ARM. Corelink CCI-550. https://developer.arm.com/Processors/CoreLink%20CCI-550, 2022.

[2] Association., I. T. Infiniband architecture specification. https://cw.infinibandta.org/document/dl/7859, 2022.

[3] Broadcom. Bcm57504 - 100gbe. https://en.broadcom.com/products/ethernet-connectivity/network-adapters/bcm57504-100g-ic, 2022.

[4] Broadcom. Product Brief: Stingray PS225. https://docs.broadcom.com/doc/PS225-PB, 2022.

[5] Chen, Y., Lu, Y., and Shu, J. Scalable RDMA RPC on reliable connection with efficient resource sharing. In *Proceedings of the Fourteenth EuroSys Conference 2019, Dresden, Germany, March 25-28, 2019* (2019), G. Candea, R. van Renesse, and C. Fetzer, Eds., ACM, pp. 19:1–19:14.

[6] Consortium, C. Cxl specification. https://www.computeexpresslink.org/download-the-specification, 2022.

[7] Dragojevic, A., Narayanan, D., Castro, M., and Hodson, O. FaRM: Fast remote memory. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2014, Seattle, WA, USA, April 2-4, 2014* (2014), R. Mahajan and I. Stoica, Eds., USENIX Association, pp. 401–414.

[8] Dragojevic, A., Narayanan, D., Nightingale, E. B., Renzelmann, M., Shamis, A., Badam, A., and Castro, M. No compromises: distributed transactions with consistency, availability, and performance. In *Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, October 4-7, 2015* (2015), E. L. Miller and S. Hand, Eds., ACM, pp. 54–70.

[9] Gao, Y., Li, Q., Tang, L., Xi, Y., Zhang, P., Peng, W., Li, B., Wu, Y., Liu, S., Yan, L., Feng, F., Zhuang, Y., Liu, F., Liu, P., Liu, X., Wu, Z., Wu, J., Cao, Z., Tian, C., Wu, J., Zhu, J., Wang, H., Cai, D., and Wu, J. When cloud storage meets RDMA. In *18th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2021, April 12-14, 2021* (2021), J. Mickens and R. Teixeira, Eds., USENIX Association, pp. 519–533.

[10] Gouk, D., Lee, S., Kwon, M., and Jung, M. Direct access, High-Performance memory disaggregation with DirectCXL. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)* (Carlsbad, CA, July 2022), USENIX Association, pp. 287–294.

[11] Guo, C., Wu, H., Deng, Z., Soni, G., Ye, J., Padhye, J., and Lipshteyn, M. RDMA over commodity ethernet at scale. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016* (2016), M. P. Barcellos, J. Crowcroft, A. Vahdat, and S. Katti, Eds., ACM, pp. 202–215.

[12] Hassan, H., Vijaykumar, N., Khan, S. M., Ghose, S., Chang, K. K., Pekhimenko, G., Lee, D., Ergin, O., and Mutlu, O. Softmc: A flexible and practical open-source infrastructure for enabling experimental DRAM studies. In *2017 IEEE International Symposium on High Performance Computer Architecture, HPCA 2017, Austin, TX, USA, February 4-8, 2017* (2017), IEEE Computer Society, pp. 241–252.

[13] Intel. Intel® data direct i/o technology. https://www.intel.com/content/www/us/en/io/data-direct-i-o-technology.html, 2022.

[14] Kalia, A., Kaminsky, M., and Andersen, D. G. Using RDMA efficiently for key-value services. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014* (2014), F. E. Bustamante, Y. C. Hu, A. Krishnamurthy, and S. Ratnasamy, Eds., ACM, pp. 295–306.

[15] Kalia, A., Kaminsky, M., and Andersen, D. G. Design guidelines for high performance RDMA systems. In *2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016* (2016), A. Gulati and H. Weatherspoon, Eds., USENIX Association, pp. 437–450.

[16] Kalia, A., Kaminsky, M., and Andersen, D. G. Fasst: Fast, scalable and simple distributed transactions with two-sided (RDMA) datagram rpcs. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016* (2016), K. Keeton and T. Roscoe, Eds., USENIX Association, pp. 185–201.

[17] KALIA, A., KAMINSKY, M., AND ANDERSEN, D. G. Datacenter rpcs can be general and fast. In *16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019, Boston, MA, February 26-28, 2019* (2019), J. R. Lorch and M. Yu, Eds., USENIX Association, pp. 1–16.

[18] KIM, J., JANG, I., REDA, W., IM, J., CANINI, M., KOSTIC, D., KWON, Y., PETER, S., AND WITCHEL, E. Linefs: Efficient smartnic offload of a distributed file system with pipeline parallelism. In *SOSP '21: ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual Event / Koblenz, Germany, October 26-29, 2021* (2021), R. van Renesse and N. Zeldovich, Eds., ACM, pp. 756–771.

[19] LIU, J., MALTZAHN, C., ULMER, C. D., AND CURRY, M. L. Performance characteristics of the bluefield-2 smartnic. *CoRR abs/2105.06619* (2021).

[20] LIU, M., CUI, T., SCHUH, H., KRISHNAMURTHY, A., PETER, S., AND GUPTA, K. Offloading distributed applications onto smartnics using ipipe. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019* (2019), J. Wu and W. Hall, Eds., ACM, pp. 318–333.

[21] MARVELL. Marvell liquidio iii. https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-liquidio-III-solutions-brief.pdf, 2022.

[22] MELLANOX. ConnectX-7 product brief. https://www.nvidia.com/content/dam/en-zz/Solutions/networking/ethernet-adapters/connectx-7-datasheet-Final.pdf, 2022.

[23] MONGA, S. K., KASHYAP, S., AND MIN, C. Birds of a feather flock together: Scaling RDMA rpcs with flock. In *SOSP '21: ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual Event / Koblenz, Germany, October 26-29, 2021* (2021), R. van Renesse and N. Zeldovich, Eds., ACM, pp. 212–227.

[24] NETRONOME. Netronome agilio. NetronomeAgilioSmartNICs.https://www.netronome.com/products/smartnic/overview/, 2022.

[25] NEUGEBAUER, R., ANTICHI, G., ZAZO, J. F., AUDZEVICH, Y., LÓPEZ-BUEDO, S., AND MOORE, A. W. Understanding pcie performance for end host networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018* (2018), S. Gorinsky and J. Tapolcai, Eds., ACM, pp. 327–341.

[26] NVIDIA. Innova-2 flex. https://www.nvidia.com/en-us/networking/ethernet/innova-2-flex/, 2022.

[27] NVIDIA. Nvidia bluefield dpu-2. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-2-dpu.pdf, 2022.

[28] NVIDIA. Nvidia bluefield dpu-3. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf, 2022.

[29] NVIDIA. Performance Monitoring Counters, BlueField SW Manual v2.4.0.11082. https://docs.nvidia.com/networking/display/BlueFieldSWv24011082/Performance+Monitoring+Counters, 2022.

[30] OLUMIDE OLUSANYA AND MUNIRA HUSSAIN. Need for Speed: Comparing FDR and EDR InfiniBand (Part 1). https://dl.dell.com/manuals/all-products/esuprt_software/esuprt_it_ops_datcentr_mgmt/high-computing-solution-resources_white-papers77_en-us.pdf, 2022.

[31] SCHUH, H. N., LIANG, W., LIU, M., NELSON, J., AND KRISHNAMURTHY, A. Xenic: Smartnic-accelerated distributed transactions. In *SOSP '21: ACM SIGOPS 28th Symposium on Operating Systems Principles, Virtual Event / Koblenz, Germany, October 26-29, 2021* (2021), R. van Renesse and N. Zeldovich, Eds., ACM, pp. 740–755.

[32] SHAMIS, A., RENZELMANN, M., NOVAKOVIC, S., CHATZOPOULOS, G., DRAGOJEVIC, A., NARAYANAN, D., AND CASTRO, M. Fast general distributed transactions with opacity. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019* (2019), P. A. Boncz, S. Manegold, A. Ailamaki, A. Deshpande, and T. Kraska, Eds., ACM, pp. 433–448.

[33] SHI, J., YAO, Y., CHEN, R., CHEN, H., AND LI, F. Fast and concurrent RDF queries with rdma-based distributed graph exploration. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016* (2016), K. Keeton and T. Roscoe, Eds., USENIX Association, pp. 317–332.

[34] THOMAS, S., VOELKER, G. M., AND PORTER, G. Cachecloud: Towards speed-of-light datacenter communication. In *10th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2018, Boston, MA, USA, July 9, 2018* (2018), G. Ananthanarayanan and I. Gupta, Eds., USENIX Association.

[35] THOSTRUP, L., FAILING, D., ZIEGLER, T., AND BINNIG, C. A dbms-centric evaluation of bluefield dpus on fast networks. In *13th International Workshop on Accelerating Analytics and Data Management Systems Using Modern Processor and Storage Architectures* (2022).

[36] TIMOTHY PRICKETT MORGAN. Pushing PCI-express fabrics up to the next level. https://www.nextplatform.com/2020/03/27/pushing-pci-express-fabrics-up-to-the-next-level/, 2022.

[37] TSAI, S.-Y., AND ZHANG, Y. Lite kernel rdma support for datacenter applications. In *Proceedings of the 26th Symposium on Operating Systems Principles* (New York, NY, USA, 2017), SOSP '17, ACM, pp. 306–324.

[38] WANG, X., KOTRA, J. B., AND JIAN, X. Eager memory cryptography in caches. In *55th IEEE/ACM International Symposium on Microarchitecture, MICRO 2022, Chicago, IL, USA, October 1-5, 2022* (2022), IEEE, pp. 693–709.

[39] WEI, X., CHEN, R., AND CHEN, H. Fast rdma-based ordered key-value store using remote learned cache. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)* (Nov. 2020), USENIX Association, pp. 117–135.

[40] WEI, X., DONG, Z., CHEN, R., AND CHEN, H. Deconstructing RDMA-enabled distributed transactions: Hybrid is better! In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)* (Carlsbad, CA, Oct. 2018), USENIX Association, pp. 233–251.

[41] WEI, X., SHI, J., CHEN, Y., CHEN, R., AND CHEN, H. Fast in-memory transaction processing using RDMA and HTM. In *Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, October 4-7, 2015* (2015), E. L. Miller and S. Hand, Eds., ACM, pp. 87–104.

[42] WEI, X., XIE, X., CHEN, R., CHEN, H., AND ZANG, B. Characterizing and optimizing remote persistent memory with RDMA and NVM. In *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021* (2021), I. Calciu and G. Kuenning, Eds., USENIX Association, pp. 523–536.

[43] XILLYBUS. Down to the tlp: How pci express devices talk. http://xillybus.com/tutorials/pci-express-tlp-pcie-primer-tutorial-guide-1, 2022.