A DBMS-centric Evaluation of BlueField DPUs on Fast Networks

Lasse Thostrup
Technical University of Darmstadt

Daniel Failing
Technical University of Darmstadt

Tobias Ziegler
Technical University of Darmstadt

Carsten Binnig
Technical University of Darmstadt & DFKI

ABSTRACT

Modern networks have evolved significantly in the last years. First, network speed has increased considerably and thus the use of low-overhead techniques such as RDMA has become more and more important to design efficient distributed DBMSs. Second, a recent trend in modern networks is that in addition to high-speed data transfer using RDMA, network components such as switches and NICs become programmable by providing additional computation on the device, such as DPUs (Data Processing Units). Such devices enable processing or manipulation of data as it is traversing the network and that way allow distributed systems to offload computation. While for the recent generation of RDMA-based DPU cards, there is no study that shows the offloading capabilities of DBMS tasks to such RDMA-enabled DPUs.

Therefore, in this paper, we aim to provide a first systematic study to evaluate the basic performance characteristics of the BlueField network cards in the context of typical DBMS operations. For the evaluation, we analyze the offload potential of using BlueField as a RDMA-enabled DPU for two important use cases: (1) a remote B-tree and (2) an end-host sequencer (i.e., remote counter). We chose these two use cases since they represent core tasks where RDMA has shown benefits. As a result, in our evaluation, we show that the recent generation of RDMA-based Bluefield DPUs can provide several benefits and can not only reduce access latencies but also improve the throughput. However, offloading computation to the DPU needs a careful design and naively offloading all computation to the DPU often leads to performance degradation.

1 INTRODUCTION

Motivation. In-memory DBMSs have become ubiquitous in academia and industry and many commercial offers are available today [3, 6, 7]. This is not surprising as the performance they offer is still unmatched compared to disk-based systems. However, a major challenge of in-memory DBMSs is that large data sets often do not fit into the memory of a single machine or the processing capabilities of one machine are insufficient. To that end, scale-out DBMSs are becoming the predominant solution to handle ever-increasing data set sizes and leverage more processing by utilizing more networked machines. Especially, DBMSs which are purpose-built for the cloud and can scale out or in on-demand are highly requested. As a core architecture for modern scale-out cloud DBMSs, disaggregated architectures have been crystallized which separate compute and storage. A prominent example of this architecture is Snowflake [5] but there are also other DBMS such as FoundationDB which take a more radical approach to separate not only storage and compute but also other database components in the pursuit to scale them independently [30].

However, with such disaggregated solutions, the network is increasingly on the hot path since data more often has to be fetched over the network. To keep up with this paradigm shift, networks are becoming faster with link speeds in the hundreds of gigabits, but also bringing rise to more powerful low-overhead networking technologies such as RDMA (Remote Direct Memory Access) which has influenced DBMS developments in academia and industry alike [11, 15, 17, 18]. The key driver for RDMA's success is that it offers low latencies in the single digit microsecond range as well as high throughput and thus has shown major gains for very different DBMS workloads.

A recent trend in modern networks is that in addition to high-speed data transfer using RDMA, network components such as switches and NICs become programmable by providing additional computation on the device, such as DPUs (Data Processing Units). Such devices thus enable processing or manipulation of data as it is traversing the network and that way allow distributed systems to offload computation [9, 10, 31]. The DPUs that are available today span far in terms of compute architectures, such as ASICs, FPGAs and general CPU cores, and therefore come with trade-offs in programmability and performance.

For the recent generation of RDMA-based network cards, DPUs are also becoming available. One of these devices which can be combined with RDMA is the BlueField Network Interface Card [21]. The BlueField card provides very flexible programmability due to its general-purpose ARM CPU cores that are available as compute resources on the DPU. Moreover, specialized ASICs for tasks such as data encryption and decryption are available. While evaluations have shown that security-related tasks or tasks to provide tenant-isolation in data centers [4] can be provided in an efficient manner, there is no study that shows the offloading capabilities of DBMS tasks to such RDMA-enables DPUs.

Contributions. Therefore, in this paper we aim to provide a first systematic study to evaluate the basic performance characteristics of the BlueField network cards in the context of typical DBMS operations. For the evaluation, we analyze the offload potential of using BlueField as an RDMA-enabled DPU for two important use cases: (1) a remote B-tree and (2) end-host sequencer (i.e., remote counter). We chose the remote B-tree because it is a frequently used data structure for DBMSs and it is also used in disaggregated architectures to avoid transferring all data across the network [23, 33]. On the other hand, the end-host sequencer is a commonly used building block for many distributed system tasks such as global ordering [16], for coordinating write access to shared memory [2], or to implement optimistic concurrency control [26].

For these two scenarios, we compare baselines that rely on existing one-sided and two-sided RDMA primitives with a solution that can use the DPU as an offload engine. For example, for an RDMA-based B-tree such as [33], the DPU could implement the tree traversal natively on the DPU. As such B-tree operations can be implemented in one round-trip from the compute layer to the storage layer without involving the CPU of the storage nodes. Moreover, in addition to compute resources DPUs often come with their own memory which allows them to store the B-tree in the DPU instead of using the CPU memory. Based on these observations, we thus aim to analyze whether a better performance can be observed for the two use cases above.

As a result, in our evaluation, we show that the recent generation of RDMA-based BlueField DPUs can provide several benefits. First, we show that we can reduce the network latency by avoiding the PCIe path from the NIC to the host which adds a non-negligible overhead to the overall latency [19]. Second, if DPUs are used as additional compute resources to the remote CPUs, we show that also overall gains in throughput can be achieved. However, offloading computation to the DPU needs a careful design and naively offloading all computation to the DPU often leads to performance degradations since the computational resources on the DPUs are often less powerful than the CPU in the remote host.

Outline. We first cover the background of the BlueField DPU and RDMA in Section 2 and next provide an overview of the experimental setup in Section 3. We then present the benchmarking of the before-mentioned use cases in Sections 4 to 6. Finally, we then present our conclusion on the evaluation to outline the pros and cons of offloading typical DBMS tasks to the BlueField DPU.

2 BACKGROUND

In this section, we provide the relevant background on RDMA and the BlueField DPU.

2.1 Remote Direct Memory Access (RDMA)

RDMA has come to be an established state-of-the-art communication method for distributed data-processing systems [2, 13, 25–27, 33], since it overcomes the overhead of traditional kernel-space network stacks such as TCP/IP. To leverage RDMA, an application can make use of different communication schemes that can be categorized as one-sided (READ / WRITE) or two-sided (SEND / RECEIVE) operations, which refers to the involvement of the sender- & receiver-CPU in the communication. For one-sided operations, only the sender-CPU is actively involved, but as a consequence, the sender also has to decide where on the remote node the data should be written or read. With two-sided operations, the receiver-CPU is also actively involved in the communication since it needs to issue RECEIVE requests before SEND requests can be issued on the sender side, and it can thus also decide where to place data which simplifies the remote memory management.

Especially the one-sided RDMA operations have seen high adoption in distributed data processing systems since they allow sender nodes to write into remote memory directly without involving additional CPU cores of the receiving nodes. In a distributed DBMS, this reduces the overall CPU resources involved in data transfer and can thus lead to less resource consumption as well as overall lower latencies in many cases [8, 32]. Moreover, one-sided RDMA

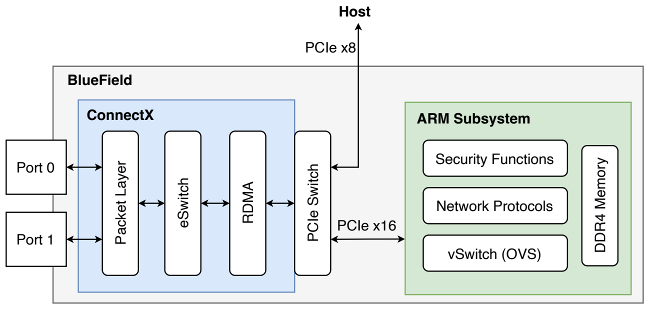

Figure 1: Functional diagram of the BlueField DPU [20].

is thus, in particular, beneficial for disaggregated DBMS architectures in which the storage servers have limited or sometimes even near-zero computational resources [24, 28, 29]. On the other hand, the applications of two-sided RDMA is typically within RPC-style applications where the receiver node has to be actively involved in the communication.

2.2 Data Processing Units

We next provide the relevant background on the Data Processing Units (DPUs) that we use for our evaluation. Many of the major network hardware vendors are including DPUs to their product offerings, with examples like the Intel IPU, Broadcom Stingray, AMD Pensando or the Nvidia BlueField. DPUs come with different compute architectures ranging from P4 programmable ASICs to general-purpose CPU cores. In our evaluation we focus on the BlueField cards from Nvidia. The BlueField cards are equipped with general-purpose ARM cores which are capable of executing any program logic in contrast to the more rigid ASIC architectures. The DPU runs its own OS (e.g., Ubuntu) and as such resembles another independent server with an added set of networking features. Internally, as illustrated in Figure 1, the DPU consists of the networking component (ConnectX) which provides hardware-offload of the network stack for more efficient (i.e., less CPU intensive) networking. Moreover, the ARM cores which act as computational resources on the DPU are equipped with DDR4 memory and are connected to the ConnectX over an internal PCIe switch. The BlueField (from the second generation on) is additionally equipped with hardware accelerators for compression/decompression, encryption and regex pattern matching.

The BlueField DPUs themselves are attached to the host CPU via PCIe. For controlling and routing the network traffic to the host, the DPUs have different modes of operation. There is a *Separated Host* mode, where the ARM subsystem will appear as an additional computer on the network. When the remote machine uses this setup, a client can decide where data should be transferred to, either the ARM subsystem or the host. Another mode is the *Embedded Mode*, where the ARM subsystem manages the network on behalf of the host and controls the physical ports of the network card. With that, the traffic from or to the host always goes through the ARM subsystem. The Embedded Mode can be further specialized, such that the host does not have access to the DPU directly. This mode allows cloud providers to provide isolation and offloading of cloud management tasks to the DPU.

To control the network communication in the Embedded Mode and forward traffic with low latency, the ConnectX subsystem of the BlueField offers a component called eSwitch. While the DPU controls the physical network ports in the Embedded Mode, the host as well as the DPU can have additional virtual interfaces which are then available in the DPU for further network flow control. By default, an OpenVSwitch process is running on the BlueField, which controls the communication between physical and virtual network ports. Flow rules controlling the flow of packets will be configured in the OpenVSwitch, and, if possible, offloaded to the eSwitch. In a typical network flow for RoCE (RDMA over Converged Ethernet), the network packets are processed by the eSwitch, which forwards the packet to the host or DPU based on the MAC addresses. However, the eSwitch is limited to specific network headers, like MAC or IP addresses, VLAN tags or TCP/UDP ports. As a result, processing or manipulation of RDMA packet headers is not possible in the eSwitch. Controlling traffic unsupported by the eSwitch means that the packets need to be processed by the BlueField's ARM cores.

Overall, the host CPUs, the BlueField DPU and the ConnectX subsystem are connected by a PCIe switch. In contrast to other vendors, BlueField DPUs do not have a dedicated DMA engine to access the memory of the host. However, RDMA operations can be used for the communication between DPU and host CPU. The RDMA API then generates normal requests through the ConnectX subsystem, which can then access both the host's as well as the DPU's memory. In this way, not only can the DPU make DMA requests to the host's memory, but the host can also access the DPU's memory.

3 USE CASES AND EXPERIMENTAL SETUP

Use Cases. As mentioned before, in our evaluation we consider two use cases to evaluate the performance of the BlueField cards. In the first use case, we focus on a remote B-tree, which has already been studied in the context of RDMA in previous work [33]. We choose to evaluate and implement two remote B-tree strategies, either with RPC over two-sided RDMA in Section 4 or completely one-sided RDMA (i.e., no RPC) in Section 5. In the last use case we evaluate an end-host sequencer, which is a common building block in distributed systems, to e.g., assert a global order or coordinate access to shared memory [2, 12]. We take a look at a sequencer implemented only with one-sided RDMA or with RPC two-sided RDMA.

Experiment Setup. We evaluate both available generations of the BlueField DPUs — the BlueField 1 and BlueField 2 — which mostly differ in their processing power; i.e., the BlueField 1 only runs at a clock frequency of 800 MHz whereas the BlueField 2 runs at 2.5 GHz (details outlined in Table 1). In our experimental setup, we are mirroring a typical disaggregated storage and compute setup. The storage node is equipped with a BlueField 1 as well as BlueField 2 card to compare both DPUs. The storage server is running with 2× Intel Xeon Gold 6326 CPUs, 512 GB DDR4 memory and PCIe 4.0. The compute node uses a non-programmable RDMA ConnectX-5 NIC and is equipped with 2× Intel Xeon Gold 5220 CPUs with 512 GB DDR4 memory (Table 2). For both servers, we only use one NUMA socket so as to not involve any cross-NUMA traffic effects. Both servers are connected with RDMA over Converged Ethernet (RoCE) v2. Note that the BlueField cards used in our setup only provide maximum 25 Gbps per network link. However, since the BlueFields NICs are equipped with two links, we can in total use a 50 Gbps connection between the compute and storage server by splitting the traffic over the two links.

We configure the BlueField cards in the *Embedded Mode*, and configure two virtual interfaces which route to either the ARM subsystem or the host. The routing is offloaded into the eSwitch and does therefore not introduce measurable overhead on the ARM subsystem.

	BlueField-1 BF1M332A	**BlueField-2** BF2H332A
CPU	8× ARMv8 A-72 model 0 @ 800MHz	8× ARMv8 A-72 model 3 @ 2.5GHz
Memory	1× 16 GB DDR4-2400	1× 16GB DDR4-3200
Caches	L1: 32 KB / core L2: 1 MB / 2 cores L3: 12 MB (shared)	L1: 32 KB / core L2: 1 MB / 2 cores L3: 6 MB (shared)
Network	ConnectX-5	ConnectX-6
Interfaces	2× 25 Gbps	2× 25Gbps
PCIe Host	PCIe 4.0 ×8	PCIe 4.0 ×8
PCIe ARM	PCIe 4.0 ×16	PCIe 4.0 ×16
OS	Ubuntu 18.04	Ubuntu 20.04

Table 1: Comparison of BlueField-1 and BlueField-2.

	Storage node	**Compute node**
CPU	2× Intel Xeon Gold 6326 @ 2.9 - 3.5 GHz	2× Intel Xeon Gold 5220 @ 2.2 - 3.9 GHz
Caches	L1: 48 KB / core L2: 1.25 MB / core L3: 24 MB (shared)	L1: 32 KB / core L2: 1 MB / core L3: 24.75 MB (shared)
Network	BlueField-1 BlueField-2	ConnectX-5
Memory	16× 32GB DDR4-3200	8× 64GB DDR4-2666
OS	Ubuntu 20.04	Ubuntu 18.04

Table 2: Server nodes used in the experimental setup.

4 USE CASE 1: REMOTE B-TREE WITH RPC

For the first use case we use a remote B-tree which is accessed via RPC calls. To realize the RPC framework, we use two-sided RDMA SEND/RECEIVE verbs with existing optimization such as door-bell batching on both client- and server-side and inlining for reducing PCIe overhead [12]. For our B-tree, we use an OLC (optimistic lock-coupling) synchronization protocol to allow scalable reads [14]. For keys and values, we use 8 byte integers. We use either a mix of 50/50 read-write or read-only workloads as evaluation.

For all experiments in this use case (unless otherwise stated) we use 8 threads (maximum number) on the BlueFields and on the CPUs of the storage server respectively to achieve a comparable setup. This is important for Section 4.1.2 when we show the offload potential by varying the percentage of requests handled by the NIC. Additionally, using 8 threads on the storage better reflects a typical storage node with weaker CPUs than compute nodes [29, 34]. On the compute server, we use 16 threads to attribute to the fact that compute servers are typically equipped with more computational resources and thus be able to generate sufficient workload in our evaluation.

The evaluation of this use case is structured as follows: we first report the throughput characteristics by first contrasting the two BlueField generations and next dive further into using the BlueField-2 in union with the host server. Subsequently, we evaluate the latency characteristics to determine whether any latency benefits can be observed for the BlueField DPU.

4.1 Throughput Characteristics

4.1.1 BlueField-1 vs BlueField-2. The BlueField-1 was already launched in 2017, and has in recent years been preceded by the BlueField-2. We first aim to evaluate the performance improvement between the two generations and their individual ability to handle B-tree RPC requests. We initialize B-trees of different sizes in the local memory of either the BlueField-1 or 2. Due to the very low clock frequency of the BlueField-1 ARM cores (over 3× less than the BlueField-2) and the fact that the RPC handling is typically very CPU intensive, we expect to see a significant performance difference between the two generations.

In Figure 2 we observe the achieved throughput for a (a) read-only or (b) 50/50 read-write workload. For the smallest tree with 1 million (M) keys (tree size between 16 & 32 MB), the BlueField-2 outperforms the first generation by 2.4× for 50/50 read-write which can be attributed to the difference in clock frequency (800 MHz & 2.5 GHz). Coupled with the fact that most of the tree can reside in the CPU caches, this indicates they are both CPU bound. However, with larger tree sizes, the BlueField-2 becomes increasingly memory bound and the relative performance difference is decreasing.

Between the two workloads (read-only and 50/50) we can only observe a slight difference in performance for the small tree with 1 M keys. The reason why barely any difference in performance is observed for 16 M and 256 M is due to more sparse reads and writes, and the reduction in throughput (i.e., a read has a smaller chance of conflicting with a write).

To isolate the performance difference between the two BlueField generations further, we now take a look at the pure RPC handling capabilities. We issue RPC calls from the compute node which does not contain any operation to be performed, i.e., a No-Op. In Figure 3 we scale the number of threads on the BlueFields that handle the RPC requests. We observe that the BlueField-2 has very good scalability, i.e., almost linear, whereas the BlueField-1 only increases the throughput with 1.9× from 2 to 8 threads. Note, since we use both available interfaces of the BlueFields, we run with a minimum of 2 threads.

Based on these findings it is clear that the second BlueField generation already provides a substantial performance boost over the first generation. The suboptimal performance of the BlueField-1 renders it hard to integrate into any performance-critical data-intensive use cases. We argue that the performance difference mainly stems from the very low clock frequency of the BlueField-1 ARM cores, as this is the main differentiator. In the remainder of the paper, we thus focus on the performance of the BlueField-2.

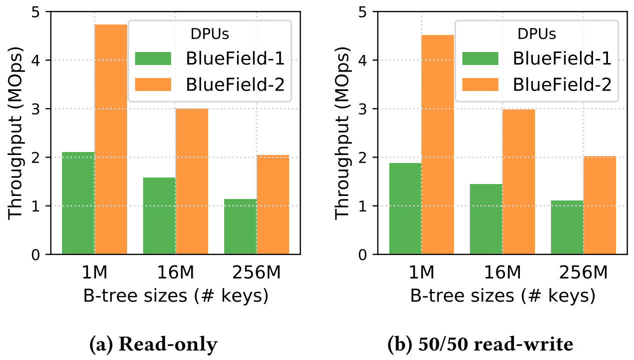

Figure 2: Remote B-tree on BlueField-1 or BlueField-2 with RPC.

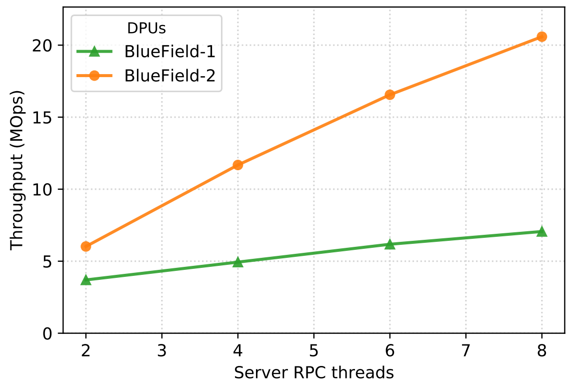

Figure 3: No-Op RPC BlueField-1 vs BlueField-2.

4.1.2 BlueField-2 Offloading. We now evaluate the offloading potential of the BlueField-2 DPU by utilizing the DPU together with the host CPUs of the storage server. We use all 8 CPU cores on the DPU and also use 8 CPU cores of the host, which we keep fixed regardless of the partition sizes to the host or DPU. To use both compute resources, we range-partition the B-tree between the host memory and the DPU memory and observe the achieved overall throughput. For partitioning, we use different setups ranging from 0-100% of the B-tree being stored on the DPU. Moreover, we create the index requests uniformly in the whole key range such that the relative partition sizes of the tree also match the workload generated to each compute device (i.e., the host CPU or the BlueField-2 DPU).

In Figure 4, we see the results when we gradually increase the range partition of the B-tree on the DPU and decrease it on the host, indicated by the x-axis. With 0% DPU offload, the host contains the full B-tree and as such all requests are handled by the host and the DPU is not processing any requests. Instead, with 25% offload, 3/4 of the B-tree is on the host and 1/4 is on the DPU. Overall, in Figure 4 we initially see a steady increase in throughput as more requests are routed to the DPU up until around 25% whereas for

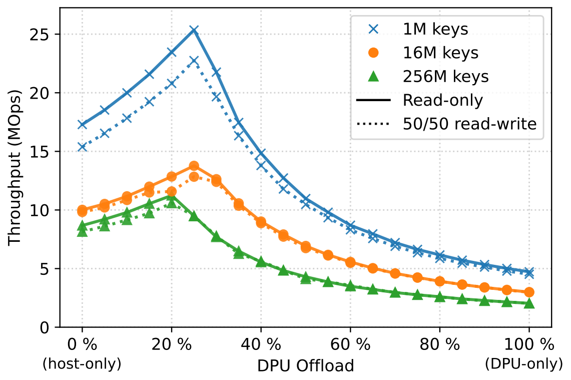

Figure 4: Remote B-tree with increasing offload on BlueField-2 for various B-tree sizes. RPC requests with read-only or 50/50 read-write.

the tree with 1M keys, the overall throughput increases by 47%. For larger tree sizes, the observed throughput increase is slightly less, with around 30% for the B-tree with 256 M keys.

However, offloading more than about 25% of the B-tree to the DPU is detrimental to the throughput since the DPU is then overloaded and the performance degrades to DPU-only throughput as reported in Figure 2 already. This degradation of throughput is not surprising as both the CPU on the DPU is weaker and the main-memory is slower than that of the storage host as reported in Table 1 and Table 2. Moreover, the read-only and 50/50 read-write workload only differs slightly for the higher throughput and smaller B-tree cases for the same reasons as discussed before.

These results indicate that while the BlueField-2 is not powerful enough to achieve high throughput in comparison to the host, it yields a significant speedup by using the DPU resources in addition to the host CPU. However, this imposes challenges for real-world use cases, as the optimal partitioning of the B-tree is dependent on the workload (i.e., potential access skew) and the performance of the host server in relation to the DPU. As such, more sophisticated adaptive solutions could come into play, which re-balances and re-partitions the B-tree between host and DPU based on utilization metrics, to automatically adapt to the most optimal partitioning between the host and DPU. Such a design is, however, out of scope of this paper.

4.1.3 Local B-tree - Throughput. To compare the achieved performance for the remote B-tree with RPC, we locally execute lookup and update operations on the B-tree on either the storage host or BlueField-2, ultimately determining whether the RPC-handling or the B-tree is the bottleneck.

In Figure 5 we execute local updates (a) or lookups (b) on different sized trees. If we compare the lookup performance to the 100% offloaded B-tree scenario in Figure 4, for the smallest tree the RPC lookup (read-only) throughput is 4.7 MOps and 9.9 MOps for the local B-tree. This indicates that half of the throughput is lost to RPC networking overhead for the smaller tree. For the largest tree of 256M keys, the difference is much less pronounced where the RPC lookup throughput on the DPU is 2 MOps and 2.5 MOps without RPC overhead. The reason behind this is that for the largest tree,

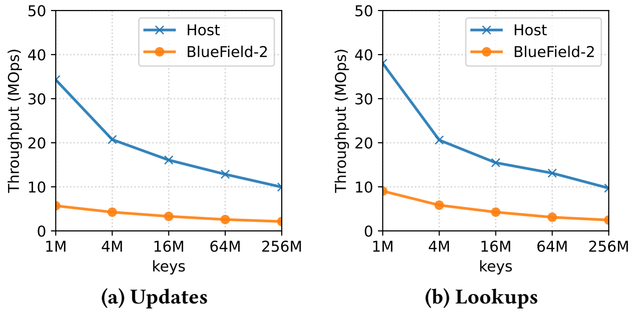

Figure 5: Throughput of local B-tree update and lookup operations on 8 threads.

the bottleneck is increasingly the cache misses going to the main-memory. The CPU cache and instruction overhead introduced by the RPC handling does therefore not affect the performance as much relative to the smaller tree. The same trend is also observed for the host.

Overall, these findings also confirm our previous results in Figure 4 since the benefits of the offloading stem from the lookup requests that the DPU can provide additionally to the host CPUs.

4.2 B-tree Latency Characteristics

We now evaluate the latency characteristics of the B-tree with RPC.

4.2.1 BlueField-2 Offloading. Since the CPU cores of the DPU are co-located with the ConnectX networking chip on the same device, we want to evaluate whether any latency improvements can be observed by using the DPU in contrast to the host. We again evaluate the performance by offloading different partition sizes of the B-tree to the DPU. To not overload the host or DPU we let only 1 client thread on the compute node issue requests.

In Figure 6 we report the median latency for read-only RPC requests. We observe that the best latency is achieved with the complete B-tree located on the host (i.e., 0% offload). The latency increases slightly (between 14 and 20%) with more RPC requests being routed to the DPU. This is in contrast to our initial expectations, as having the CPU cores and memory situated on the same device should result in lower network latency. However, two factors might be at play here which render the achieved latency of the BlueField-2 DPU worse than the host; (a) since the DPU CPU cores are in fact also separated from the networking hardware (ConnectX) over PCIe, this potentially diminishes the latency benefit and (b) memory access latencies might be worse on the DPU for B-tree lookup in comparison to the host.

In the following, we evaluate these two aspects independently to account for the latency increase observed in Figure 6.

4.2.2 RDMA Send Latency. To isolate the latency characteristics of the network we execute the PerfTest[1], which is a standard RDMA benchmarking tool.

In this experiment, the compute server issues SEND requests to either the storage host or the BlueField-2 DPU using PerfTest's *ib_send_lat*. It is important to note that the *ib_send_lat* test from

[1]https://github.com/linux-rdma/perftest

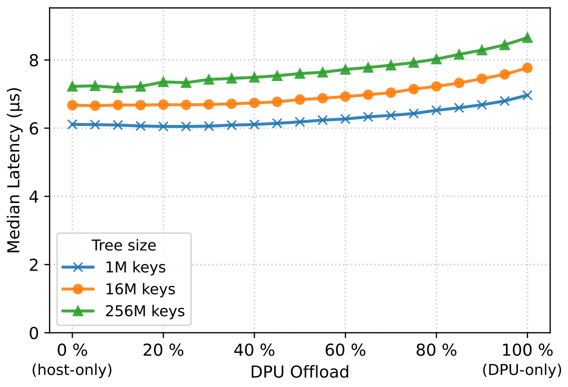

Figure 6: Remote B-tree latency with increasing offload on BlueField-2. RPC requests with read-only.

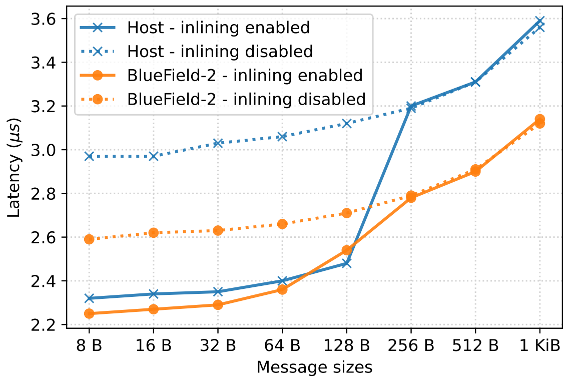

Figure 7: RDMA Send latency measured with *ib_send_lat* from the compute server to the storage server (host) or BlueField-2, with or without inlining of data up to 236 B.

PerfTest implements a ping-pong and reports half of the round-trip as latency. In this benchmark, the SEND performance of the storage server's host CPU or DPU is therefore also included. We execute the benchmark both with and without data inlining. If the payload of an RDMA send operation is less than the maximum possible inlinable size, a PCIe round-trip to the NIC can be saved. The inlining size limit for our hardware is 236 bytes and enabled by default.

Running the *ib_send_lat* benchmark for different message sizes in Figure 7 shows that with inlining enabled (up to 237 bytes), the latency is almost the same. However, without inlining (dotted lines or above 237 bytes) we see a latency improvement for the DPU around 0.4 μs. The experiment shows that while there can be a real latency improvement for the DPU, for small message sizes (e.g., 32 byte as used in our RPC framework), inlining reduces the additional PCIe overhead present for the host versus the DPU.

4.2.3 Local Memory Latency. As such, we next speculate that the reason why the observed RPC B-tree latency is not lower on the DPU versus the storage host might be due to worse memory access latency. We test this by measuring the latency for random memory access over different memory block sizes. With smaller memory sizes, the CPU will be able to cache most requests, but with larger sizes, more cache-misses will occur. We compare the latencies of the storage host and the DPU to evaluate both their ability to cache reads and the cost of cache-misses.

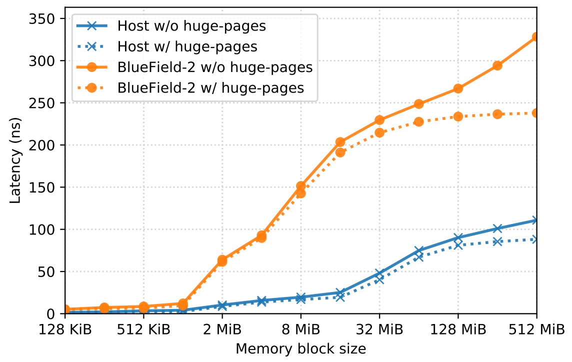

Figure 8: Memory read latencies for increasing memory block sizes. Data obtained through tinymembench[2].

In Figure 8, we report the local memory access latency for the host and BlueField-2. Already at around 1 MiB, the BlueField-2 memory accesses become relatively more expensive which can be attributed to the smaller cache sizes for the DPU. As the last-level cache of the BlueField-2 is only 6 MiB, the data spills out of the caches and memory accesses can to a smaller degree be cached. In comparison to the host with 48 MiB last-level cache, access latencies are much smaller and stable. We also report the effect of huge-pages where the increasing cost of TLB-misses already gets visible for the BlueField at around 64 MiB. For the B-tree RPC experiments, we do not utilize huge-pages.

Overall, the memory access latency is substantially higher on the DPU which contributes to the fact that we do not experience any latency benefit for the RPC B-tree use case.

4.3 Discussion

We now summarize the main findings for evaluating the BlueFields in a remote RPC B-tree use case. In conclusion, there is both potential for integrating the DPU, but also downsides.

A goal of our evaluation of the BlueField is to test whether any latency improvements can be observed. While there is in fact a detectable latency increase for networking messages on the BlueField, several factors render the remote B-tree access slightly slower than the host. These factors are slow local memory, smaller caches and smaller benefit of inlining small message sizes.

The CPU cores and the local memory on the DPU are substantially slower than a typical server-grade machine and as such blindly offloading data-intensive operations onto the DPU severely impacts the achievable performance. We instead argue that the DPU must be carefully integrated such that the workload offloaded to the DPU corresponds to the processing capabilities.

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

5 USE CASE 1: REMOTE B-TREE WITH ONE-SIDED RDMA

More and more designs are utilizing one-sided RDMA to access remote data structures in disaggregated memory setups [1, 24, 33, 34]. The reason for this is that one-sided operations help to remove the load on the (potentially weak) remote memory servers. Since we already saw that the BlueField-2 struggles to achieve good RPC performance due to the relatively slow memory and CPU cores, one-sided access is a promising use case as it does not incur any CPU overhead on the DPU.

In this use case, we therefore evaluate a remote B-tree accessed only over one-sided RDMA read operations. The way a remote B-tree lookup works for one-sided operations is that clients are first issuing a read on the root node and locally performing a binary search to determine the next child node. This is repeated until the leaf level. As such, an RDMA read request is issued for each level of the B-tree and since the reads are interdependent (i.e., the location of one read depends on the previous) they cannot be overlapped.

We first evaluate the throughput characteristics and subsequently look at the latency. Last, we discuss the findings holistically.

5.1 Throughput Characteristics

One-sided remote data structures often come with a lower throughput than their RPC counterparts since a network round-trip is necessary for each data-dependent read. However, a unique possibility with one-sided accesses is that the data structure can easily be distributed out on multiple storage nodes to spread out load and achieve higher throughput [24, 33]. This is possible since each read is anyway a remote access and can therefore be directed to any storage server.

In our B-tree benchmark, we however focus on just one storage server and compare the throughput while gradually offloading the B-tree to the DPU. We see in Figure 9 that the throughput stays stable with more of the B-tree being partitioned to the DPU. As such, there is no real difference in terms of throughput performance whether the B-tree is offloaded to the DPU or not. The DPU is therefore a good candidate for offloading one-sided accesses since it alleviates load on the main-memory of the host system of the storage server.

We also evaluate the impact on different node sizes of the tree, i.e., the size of each RDMA read, and observe that for the tested tree sizes, 2048 B node sizes provide the best throughput at 0.8 MOps. The reason behind this is that even though an RDMA read of 2048 B is more expensive in terms of throughput and latency than 512 B, larger nodes result in a bigger fanout and a shallower tree requiring fewer network round-trips. As previously mentioned, while the absolute throughput is much lower compared to an RPC solution, the contention on the remote NIC can be spread out to achieve higher throughput.

5.2 Latency Characteristics

While we did not see any latency improvement for offloading RPC B-tree lookups to the DPU due to the smaller caches and slower CPU, these factors are not as influential for one-sided accesses. We, therefore, expect to see a reduced latency as more lookups are offloaded to the DPU, due to the close proximity of the DPU cores to the network. In Figure 10 we execute the B-tree lookups with different partition sizes offloaded to the DPU. A clear trend is shown here that the DPU provides faster lookups than the host. For the different tree sizes and node sizes, the improvement is around 11-13%.

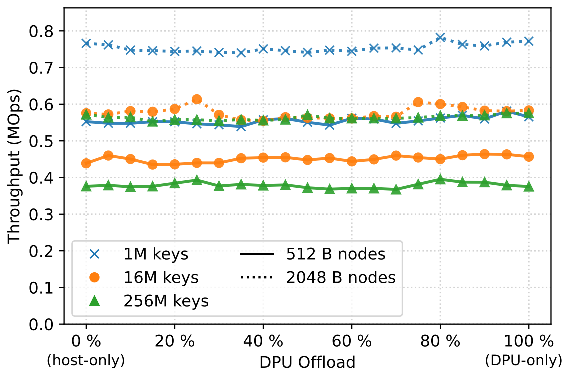

Figure 9: Remote B-tree throughput with increasing offload on BlueField-2. Read-only with one-sided RDMA.

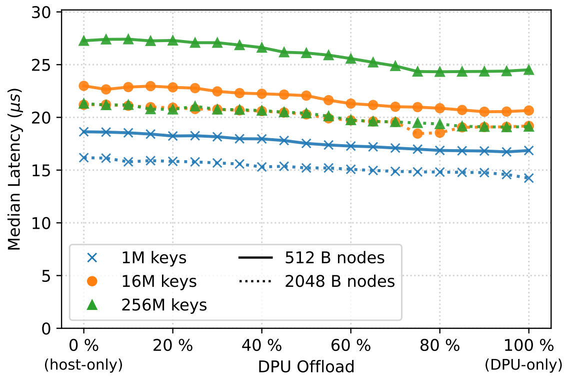

Figure 10: Remote B-tree latency with increasing offload on BlueField-2. Read-only with one-sided RDMA.

The difference in latency observed for the evaluated tree sizes and node sizes is due to the different latency of the RDMA read and the depth of the tree. As an example, a node size of 512 B has a fanout of $512B/16B = 32$ (with 16 B used as key and child pointer), so for a tree with 256 M keys, the depth will be $\lceil log_{32}(256M) \rceil = 6$, whereas a node size of 2048 B only has a depth of 5 and therefore one less RDMA read. The experiment shows that the lower latency of a 512 B read with respect to a 2048 B read does not amortize the cost of an extra read in the B-tree.

5.3 Discussion

In conclusion, since the relatively weak CPU cores of the BlueField-2 are not engaged with one-sided access, the DPU has better offloading potential. The strongest benefit comes with lower access latency due to the co-location of the CPU cores and the network on the same physical board.

Another interesting benefit given by offloading the one-sided accesses to the DPU is that read or write pressure on the local main-memory of the host is alleviated, which might benefit concurrent

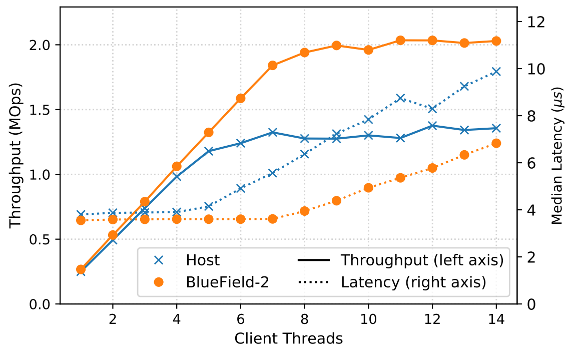

Figure 11: Throughput and latency of one-sided RDMA fetch-and-add on either the storage host or the BlueField-2 DPU.

memory-intensive applications. This is even more noticeable with faster networks such as the BlueField-2 model with 200 Gbps.

6 USE CASE 2: REMOTE SEQUENCER

A common building block in distributed systems is global counters. They are among others used for global timestamps, asserting message ordering or coordinating access to shared memory [12]. There has already been work that incorporates counters in the network such as a programmable switch [16] or directly in the SRAM of an RDMA-NIC [24], we, however, aim to evaluate the more traditional setup of placing a counter in remote DRAM main-memory. We evaluate the performance differences between the storage host and the BlueField-2 DPU with both one-sided RDMA atomic operations and with RPC with local atomic operations.

6.1 One-sided RDMA Atomics

Atomic operations are already provided in the collection of RDMA primitives such that multiple clients can perform fetch-and-add or compare-and-swap operations over the network without any additional coordination. In general, this can facilitate one-sided access to remote data structures without any locking. For this use case, we evaluate a remote counter (i.e., sequencer) accessed with one-sided RDMA fetch-and-add operations.

In Figure 11 we report the throughput and latency with an increasing number of client threads on the compute node accessing the same counter. In this use case we see a substantial benefit of the DPU in terms of achieved throughput. Placing the atomic counter on the DPU achieves an almost 50% throughput speedup. This is also reflected in the number of clients needed to saturate the remote server where the throughput of the DPU is saturated by around 8 clients whereas the storage host only can scale up to around 6 clients.

The latency is almost identical with a slight benefit on the DPU. Beyond the saturation point, the latencies increase linearly with more clients added as contention is created and requests are increasingly queued.

6.2 RPC with Local Atomics

The alternative to realizing a global sequencer with one-sided primitives is to use a two-sided approach with RPC requests which access the counter with local atomic operations. Such a design involves the remote side and therefore the CPU and memory resources have a bigger impact on the performance as already established previously. For completeness and point-of-comparison we include the achieved performance with a global sequencer realized on either the host CPU or the DPU.

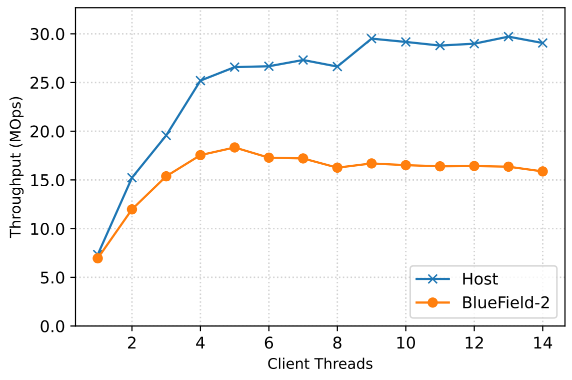

Figure 12: Throughput of RPC sequencer on either the storage host or the BlueField-2 DPU.

In Figure 12 we report the achieved throughput with an RPC implementation. Similar behavior as the B-tree RPC experiment can also be observed here, namely the relatively powerful CPU of the host results in substantially higher throughput than the DPU. The performance difference is however only up to 2× in this use case versus roughly 3× in the B-tree. The reason for this is that the counter is less memory-intensive and despite the smaller caches, the DPU can cache the atomic counter.

6.3 Discussion

Based on our experiments, the sequencer shows itself as an interesting use case to offload to the DPU, especially when utilizing one-sided atomic accesses. The closer proximity of the memory of the DPU to the network in comparison to the host results in a 50% throughput increase. While one-sided accesses achieve worse throughput than an RPC implementation, they are still often applied due to the fact that no CPU load is introduced on the storage server and that a throughput of 2 MOps is sufficient for many use cases [22].

7 CONCLUSION AND FUTURE WORK

We now conclude our findings by summarizing in which scenarios the BlueField DPUs show an acceleration potential and in which scenarios they do not. Moreover, we discuss some avenues for future work.

Summary. The BlueField-2 DPU is naturally no replacement for a general server-grade CPU and memory. It is instead marketed as an accelerator for networking-related tasks, due to its set of network-oriented hardware accelerators. The majority of these accelerators are however not applicable to our evaluated use cases as they are mostly concerned with security tasks or virtualization.

In conclusion, we evaluated the BlueField DPUs with respect to typical DBMS tasks such as remote B-trees or a global sequencer.

The main findings are that an acceleration potential exists for one-sided accesses both in terms of latency and throughput whereas two-sided accesses easily overload the DPU. However, using the DPU in combination with the host CPU of the storage server can yield promising performance benefits provided that the workload is carefully distributed with regard to the relative performance difference.

Future Work. We see that BlueField DPUs are under active development with a new generation BlueField-3 around the corner which shows even more and faster CPU cores and memory. We speculate that the BlueField-3 is therefore a promising platform for also accelerating two-sided data-intensive tasks.

Moreover, we did not cover tasks where BlueField cards provide ASIC-based accelerators such as engines for compression/decompression, encryption/decryption, regex pattern matching and NVMe. While we did not cover such engines in our evaluation, they are also interesting for DBMS workloads and are thus worth to be studied in future work.

Finally, while B-trees and global sequencers are important and interesting use cases for DBMS, there are many other use cases that are worth studying in the future. For example, if the DPUs become more powerful, we can think of implementing a full storage engine on the DPU including a concurrency scheme implementation that allows clients to access and modify data concurrently while coordinating these accesses.

ACKNOWLEDGMENTS

We thank the reviewers for their feedback. This work was partially funded by the German Research Foundation (DFG) under the grants BI2011/1 & BI2011/2 (DFG priority program 2037) and the DFG Collaborative Research Center 1053 (MAKI). Finally, we want to thank hessian.AI at TU Darmstadt as well as DFKI Darmstadt for the support.

REFERENCES

[1] Marcos K. Aguilera, Kimberly Keeton, Stanko Novakovic, and Sharad Singhal. 2019. Designing Far Memory Data Structures: Think Outside the Box. In *Proceedings of the Workshop on Hot Topics in Operating Systems, HotOS 2019, Bertinoro, Italy, May 13-15, 2019*. ACM, 120–126. https://doi.org/10.1145/3317550.3321433

[2] Carsten Binnig, Andrew Crotty, Alex Galakatos, Tim Kraska, and Erfan Zamanian. 2016. The End of Slow Networks: It's Time for a Redesign. *Proc. VLDB Endow.* 9, 7 (2016), 528–539. https://doi.org/10.14778/2904483.2904485

[3] Peter A. Boncz, Martin L. Kersten, and Stefan Manegold. 2008. Breaking the memory wall in MonetDB. *Commun. ACM* 51, 12 (2008), 77–85. https://doi.org/10.1145/1409360.1409380

[4] Scott Ciccone and John F. Kim. 2022. *NVIDIA Introduces BlueField DPU as a Platform for Zero Trust Security with DOCA 1.2.* NVIDIA. https://developer.nvidia.com/blog/nvidia-introduces-bluefield-dpu-as-a-platform-for-zero-trust-security-with-doca-1-2/

[5] Benoît Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, and Philipp Unterbrunner. 2016. The Snowflake Elastic Data Warehouse. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, Fatma Özcan, Georgia Koutrika, and Sam Madden (Eds.). ACM, 215–226. https://doi.org/10.1145/2882903.2903741

[6] Franz Faerber, Alfons Kemper, Per-Åke Larson, Justin J. Levandoski, Thomas Neumann, and Andrew Pavlo. 2017. Main Memory Database Systems. *Found. Trends Databases* 8, 1-2 (2017), 1–130. https://doi.org/10.1561/1900000058

[7] Franz Färber, Sang Kyun Cha, Jürgen Primsch, Christof Bornhövd, Stefan Sigg, and Wolfgang Lehner. 2011. SAP HANA database: data management for modern business applications. *SIGMOD Rec.* 40, 4 (2011), 45–51. https://doi.org/10.1145/2094114.2094126

[8] Philipp Fent, Alexander van Renen, Andreas Kipf, Viktor Leis, Thomas Neumann, and Alfons Kemper. 2020. Low-Latency Communication for Fast DBMS Using RDMA and Shared Memory. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1477–1488. https://doi.org/10.1109/ICDE48307.2020.00131

[9] Daniel Firestone, Andrew Putnam, Sambrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian M. Caulfield, Eric S. Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert G. Greenberg. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018*, Sujata Banerjee and Srinivasan Seshan (Eds.). USENIX Association, 51–66. https://www.usenix.org/conference/nsdi18/presentation/firestone

[10] Matthias Jasny, Lasse Thostrup, Tobias Ziegler, and Carsten Binnig. 2022. P4DB - The Case for In-Network OLTP. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1375–1389. https://doi.org/10.1145/3514221.3517825

[11] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2014. Using RDMA efficiently for key-value services. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*, Fabián E. Bustamante, Y. Charlie Hu, Arvind Krishnamurthy, and Sylvia Ratnasamy (Eds.). ACM, 295–306. https://doi.org/10.1145/2619239.2626299

[12] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2016. Design Guidelines for High Performance RDMA Systems. In *2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016*, Ajay Gulati and Hakim Weatherspoon (Eds.). USENIX Association, 437–450. https://www.usenix.org/conference/atc16/technical-sessions/presentation/kalia

[13] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2016. FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation* (Savannah, GA, USA) *(OSDI'16)*. USENIX Association, USA, 185–201.

[14] Viktor Leis, Michael Haubenschild, and Thomas Neumann. 2019. Optimistic Lock Coupling: A Scalable and Efficient General-Purpose Synchronization Method. *IEEE Data Eng. Bull.* 42, 1 (2019), 73–84. http://sites.computer.org/debull/A19mar/p73.pdf

[15] Bojie Li, Zhenyuan Ruan, Wencong Xiao, Yuanwei Lu, Yongqiang Xiong, Andrew Putnam, Enhong Chen, and Lintao Zhang. 2017. KV-direct: high-performance in-memory key-value store with programmable NIC. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*. 137–152.

[16] Jialin Li, Ellis Michael, Naveen Kr. Sharma, Adriana Szekeres, and Dan R. K. Ports. 2016. Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, Kimberly Keeton and Timothy Roscoe (Eds.). USENIX Association, 467–483. https://www.usenix.org/conference/osdi16/technical-sessions/presentation/li

[17] Xiaoyi Lu, Dipti Shankar, Shashank Gugnani, and Dhabaleswar K. Panda. 2016. High-performance design of apache spark with RDMA and its benefits on various workloads. In *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, James Joshi, George Karypis, Ling Liu, Xiaohua Hu, Ronay Ak, Yinglong Xia, Weijia Xu, Aki-Hiro Sato, Sudarsan Rachuri, Lyle H. Ungar, Philip S. Yu, Rama Govindaraju, and Toyotaro Suzumura (Eds.). IEEE Computer Society, 253–262. https://doi.org/10.1109/BigData.2016.7840611

[18] Christopher Mitchell, Yifeng Geng, and Jinyang Li. 2013. Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store.. In *USENIX Annual Technical Conference*. 103–114.

[19] 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, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*, Sergey Gorinsky and János Tapolcai (Eds.). ACM, 327–341. https://doi.org/10.1145/3230543.3230560

[20] NVIDIA. 2022. *BlueField DPU OS - Functional Diagram.* NVIDIA. https://docs.nvidia.com/networking/display/BlueFieldDPUOSLatest/Functional+Diagram

[21] NVIDIA. 2022. *NVIDIA BLUEFIELD DATA PROCESSING UNITS.* NVIDIA. https://www.nvidia.com/en-us/networking/products/data-processing-unit/

[22] Lasse Thostrup, Jan Skrzypczak, Matthias Jasny, Tobias Ziegler, and Carsten Binnig. 2021. DFI: The Data Flow Interface for High-Speed Networks. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1825–1837. https://doi.org/10.1145/3448016.3452816

[23] Qing Wang, Youyou Lu, and Jiwu Shu. 2022. Sherman: A Write-Optimized Distributed B+Tree Index on Disaggregated Memory. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1033–1048. https://doi.org/10.1145/3514221.3517824

[24] Qing Wang, Youyou Lu, and Jiwu Shu. 2022. Sherman: A Write-Optimized Distributed B+Tree Index on Disaggregated Memory. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1033–1048. https://doi.org/10.1145/3514221.3517824

[25] Xingda Wei, Jiaxin Shi, Yanzhe Chen, Rong Chen, and Haibo Chen. 2015. 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*. 87–104.

[26] Erfan Zamanian, Carsten Binnig, Tim Harris, and Tim Kraska. 2017. The End of a Myth: Distributed Transactions Can Scale. *Proc. VLDB Endow.* 10, 6 (feb 2017), 685–696. https://doi.org/10.14778/3055330.3055335

[27] Erfan Zamanian, Julian Shun, Carsten Binnig, and Tim Kraska. 2020. Chiller: Contention-Centric Transaction Execution and Data Partitioning for Modern Networks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 511–526. https://doi.org/10.1145/3318464.3389724

[28] Ming Zhang, Yu Hua, Pengfei Zuo, and Lurong Liu. 2022. FORD: Fast One-sided RDMA-based Distributed Transactions for Disaggregated Persistent Memory. In *20th USENIX Conference on File and Storage Technologies (FAST 22)*. USENIX Association, Santa Clara, CA, 51–68. https://www.usenix.org/conference/fast22/presentation/zhang-ming

[29] Qizhen Zhang, Xinyi Chen, Sidharth Sankhe, Zhilei Zheng, Ke Zhong, Sebastian Angel, Ang Chen, Vincent Liu, and Boon Thau Loo. 2022. Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1345–1359. https://doi.org/10.1145/3514221.3517856

[30] Jingyu Zhou, Meng Xu, Alexander Shraer, Bala Namasivayam, Alex Miller, Evan Tschannen, Steve Atherton, Andrew J. Beamon, Rusty Sears, John Leach, Dave Rosenthal, Xin Dong, Will Wilson, Ben Collins, David Scherer, Alec Grieser, Young Liu, Alvin Moore, Bhaskar Muppana, Xiaoge Su, and Vishesh Yadav. 2021. FoundationDB: A Distributed Unbundled Transactional Key Value Store. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2653–2666. https://doi.org/10.1145/3448016.3457559

[31] Hang Zhu, Zhihao Bai, Jialin Li, Ellis Michael, Dan R. K. Ports, Ion Stoica, and Xin Jin. 2019. Harmonia: Near-Linear Scalability for Replicated Storage with In-Network Conflict Detection. *Proc. VLDB Endow.* 13, 3 (2019), 376–389. https://doi.org/10.14778/3368289.3368301

[32] Tobias Ziegler, Viktor Leis, and Carsten Binnig. 2020. RDMA Communciation Patterns. *Datenbank-Spektrum* 20, 3 (2020), 199–210.

[33] Tobias Ziegler, Sumukha Tumkur Vani, Carsten Binnig, Rodrigo Fonseca, and Tim Kraska. 2019. Designing Distributed Tree-based Index Structures for Fast RDMA-capable Networks. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 741–758. https://doi.org/10.1145/3299869.3300081

[34] Pengfei Zuo, Jiazhao Sun, Liu Yang, Shuangwu Zhang, and Yu Hua. 2021. One-sided RDMA-Conscious Extendible Hashing for Disaggregated Memory. In *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, Irina Calciu and Geoff Kuenning (Eds.). USENIX Association, 15–29. https://www.usenix.org/conference/atc21/presentation/zuo