Accelerating the HPC I/O for Low Latency and High Throughput with 16-nanometer FPGA-based Hardware Accelerators

Babar Khan
Computer Science, TU Darmstadt, Germany

***Abstract*—The I/O stack in data center struggles with the growing demands of diverse workloads. Our novel hardware/-software framework, named DeLiBA, aims to bridge this gap by leveraging a FPGA framework to quickly deploy the FPGA-based I/O accelerators. While the current version of DeLiBA is focused on Linux block I/O layer of the data center, its architecture is equally applicable to HPC I/O stack. Our initial results achieve a 10% increase in throughput and demonstrates an overall speedup of 2.3x times compared to conventional methods.**

***Index Terms*—HPC, I/O, storage, network, Linux, FPGA**

I. Introduction

The Linux block layer is a kernel subsystem that is responsible for handling block devices, e.g., hard disk drives (HDDs), solid state disks (SSDs), and remote storage (SAN). This layered architecture adds a significant overhead along the entire request path. Measurements have shown that it takes between 18,000 and 20,000 instructions to send and receive a single fundamental $4\,\mathrm{kB}$ I/O request [1]. In x86 systems, around 50% of the total execution time of a single $4\,\mathrm{kB}$ I/O request is spent in submitting and completing I/O requests at the kernel stage [2]. While considering 32-bit ARM Cortex A9 processors, it is important to note that the overhead of a $4\,\mathrm{kB}$ I/O accounts for 90% of the total execution time, with storage device latency contributing only 10% [3]. Similarly, positioned between the applications and the underlying storage hardware, the HPC I/O stack also yields huge performance bottleneck due to its multi-layered complex I/O architecture. Our open-source hardware/software framework, Development of Linux Block I/O Accelerators (DeLiBA), introduced here, is a proposal to alleviate the complexity in the multi-layered Linux block I/O layer. It lifts key functionality of the kernel-level libraries of Linux I/O stack up into user-space, enabling the use of a wide spectrum of programming tools and techniques equally applicable to kernel-level libraries of HPC I/O stack [4]. As a first use-case for DeLiBA, we have implemented a proof-of-concept of an I/O accelerator for the client side of an open-source software-defined distributed storage protocol called Ceph. With an increasingly diverse HPC I/O workload, Ceph is a good choice for HPC as it provides all three major storage interfaces, i.e., object, block and file. [7], [8].

II. Methodology to Accelerate I/O

The focus of Figure 1 is an initial overview of DeLiBA. In the example shown, a client application generates a read operation ❶, which would pass through filesystem layers, and end up at the driver responsible for the device the data is stored on. For our purposes, that physical storage is represented by the nbd driver [5], [6], which redirects the I/O requests back up into user-space ❷ using a netlink interface. NBD is a Linux based block device protocol that allows to export a block device to a client application. As mentioned in Section I, DeLiBA uses Ceph as a use-case. Therefore, Ceph then provides a number of software support functions (`librados`,`librbd`) to perform the initial steps of I/O processing ❸, similar to the approach that would be used in kernel-level. After passing through these Block Layer Libraries, requests would continue to be processed in software. However, with DeLiBA, we open up an alternative, namely the choice to route the requests toward an Field Programmable Gate Arrays (FPGA) accelerator ❹ for further processing. FPGAs are now standard in HPC and cloud datacenters, providing more flexibility at lower power than ASICs or GPUs. And unlike CPUs, which execute instructions sequentially or in SIMD pattern, FPGAs can effectively perform specific I/O tasks in parallel. To this end, the nature of block I/O operations fits well to the task-based computational model in DeLiBA, where each I/O request can be mapped to a task to be executed on the FPGA accelerator. As Ceph is a distributed filesystem, we have to communicate over the network ❺ after completing client-side processing. Currently, for both software and hardware-based I/O processing, we rely on Linux kernel network stack for this operation, which in turn interacts with actual NIC to communicate with the remote Ceph storage ❻.

III. Hardware Implementation and Evaluation

DeLiBA is used to integrate a Xilinx Alveo U280 FPGA card providing accelerators for performing key computations of the Ceph protocol stack into the block I/O stack. The client side host uses an AMD EPYC Rome 7302P 16-core CPU with $128\,\mathrm{GB}$ of memory, attached by $10\,\mathrm{Gb/s}$ Ethernet to the Ceph server. The FPGA card is attached to the client host by PCIe Gen3 x8 and uses a system clock of $200\,\mathrm{MHz}$. A key aspect is the communication overhead required for interacting with the host. These switches between software-hardware-software execution from the DeLiBA I/O pool, to the Ceph kernels on the FPGA, and back to the host-side network stack, carry significant overhead. Using the software-side C++ timing library chrono library, we measured a full roundtrip, of

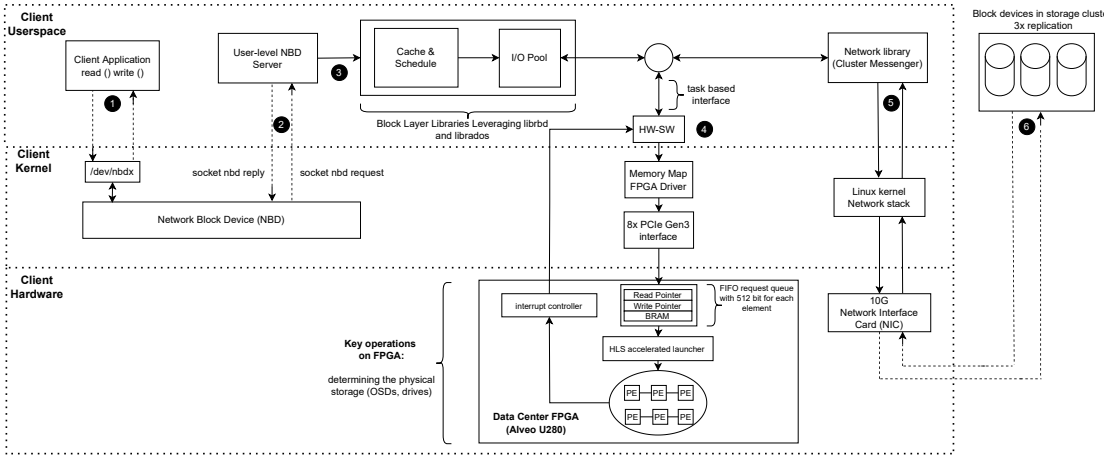

Fig. 1: Design and implementation of our framework DeLiBA to accelerate HPC I/O

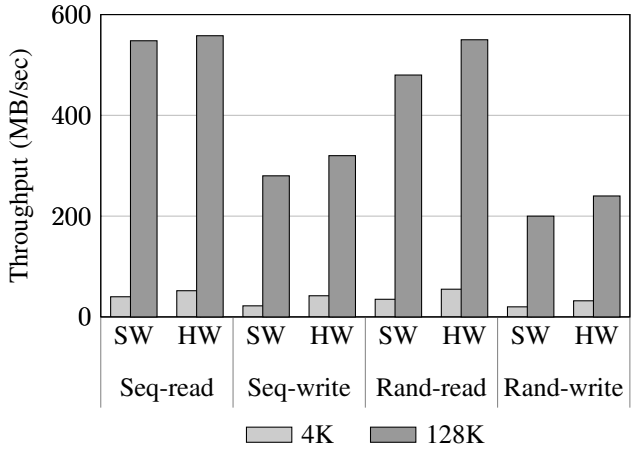

Fig. 2: Block I/O Throughput Hardware-Accelerated vs Software Baseline

around 60μs and 70μs. Even with these additional overheads due to the additional PCIe round-trips involved to the FPGA, the latency of using the hardware accelerator is similar or even better than using the pure software stack, as shown in Table I. In this manner, the proof-of-concept Ceph accelerator we use to demonstrate DeLiBA in all but one case manages to *exceed* the baseline performance of the pure software implementation. As shown in Figure 2, similar gains also apply to the throughput measurements for 4 kB and 128 kB block sizes, and sequential and random access patterns. The hardware-accelerated solution manages to speed-up throughput by up to 1.9x for sequential writes of 4 kB blocks, and by 1.2x for random writes of 128 kB blocks. The gains are even more pronounced for the rate of I/Os per second. Here, the largest gain of 2.36x for 4 kB blocks is achieved for random reads. For the larger 128 kB blocks, the I/O rate will naturally be slower. But still, the hardware accelerated stack manages a gain of 1.13x for the random-read case.

TABLE I
I/O REQUEST LATENCY ON SOFTWARE AND HARDWARE

SW vs HW (4 kB)	**Latency** [µs]			
	seq-read	***seq-write***	***rand-read***	***rand-write***
SW	65	95	130	98
HW	60	82	93	96

IV. CONCLUSION AND FUTURE WORK

While DeLiBA can already enable some performance gains over the standard I/O stack, further improvements are already being worked on. Specifically, implementing the host-side network stack also on the FPGA.

REFERENCES

[1] Caulfield, Adrian M. and De, Arup and Coburn, Joel and Mollow, Todor I. and Gupta, Rajesh K. and Swanson, Steven "Moneta: A High-Performance Storage Array Architecture for Next-Generation, Non-volatile Memories," IEEE, 385-395, 2010.

[2] Hyeong-Jun Kim and Young-Sik Lee and Jin-Soo Kim: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs, USENIX Association, 2016 pp.41-45.

[3] Stratikopoulos, Athanasios and Kotselidis, Christos and Goodacre, John and Luján, Mikel, "FastPath: Towards Wire-Speed NVMe SSDs," 2018 28th International Conference on Field Programmable Logic and Applications (FPL), 2018, pp. 170-1707.

[4] Jean Luca Bez, Suren Byna, and Shadi Ibrahim. 2023. I/O Access Patterns in HPC Applications: A 360-Degree Survey. ACM Comput. Surv. 56, 2, Article 46 (February 2024), 41 pages. https://doi.org/10.1145/3611007

[5] https://github.com/NetworkBlockDevice, "NBD Github,"

[6] Wang, Li and Wen, Yunchuan, "Design and Implementation of Ceph Block Device in Userspace for Container Scenarios," IEEE, 383-386, 2016

[7] K. Jeong, C. Duffy, J. -S. Kim and J. Lee, "Optimizing the Ceph Distributed File System for High Performance Computing," 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia, Italy, 2019, pp. 446-451, doi: 10.1109/EMPDP.2019.8671563.

[8] Alberto Chiusole, Stefano Cozzini, Daniel van der Ster, Massimo Lamanna, and Graziano Giuliani. 2019. An I/O Analysis of HPC Workloads on CephFS and Lustre. ISC High Performance 2019 International Workshops, Frankfurt, Germany, June 16-20, 2019