EPiC Series in Computing

Volume 97, 2024, Pages 72–80

Proceedings of 36th International Conference on Computer Applications in Industry and Engineering

Peer-to-Peer Data Transfer Evaluation in SmartSSD-based Multi-devices System

Luka Daoud*, Gongjin Sun†, and Hingkwan Huen‡

Samsung Semiconductor, Inc., San Jose, California, USA
Data Center Device Solutions, Memory Solutions Lab
Luka.Daoud@ieee.org, Gongjin.S@samsung.com, Kwan.Huen@samsung.com

Abstract

Due to the growing demands on big data, data centers are expanding their systems to improve their capacity and capability. Common expansion techniques are: adding more memories, increasing the storage capacity, and attaching hardware accelerator devices. Benefit aside, such expansion puts a higher demand on the host system due to the increasing amount of data movements. It quickly consumes the available system bandwidth and sets a heavy burden on data transfers among the devices. To ease this situation, recent advanced solutions have appeared to optimize the data flow. These include peer-to-peer data transfer, allowing direct device-to-device data exchange without involving host memory. This paper evaluates the system performance for peer-to-peer (P2P) data transfer among connected devices. The results show that P2P data transfer between two devices in the system is 2x - 6x faster than non-P2P cases via bypassing the host memory. Not only does it reduce the memory occupancy but also the system power cost.

1 Introduction

Big data has been expanding tremendously. This demands data centers be properly supported to handle such a huge volume of data. Data centers are mainly formed of many system servers that are full of storage devices, memories, and accelerating engines. Solid-state drive (SSD) is a popular choice due to its performance, power, and lifespan potential. Modern data centers were designed to store relational data and support accelerating applications and processing stored information. Nowadays, data centers are assembled with storage elements and accelerating cards attached to nearby servers. This allows more efficient data flow for processing as the conventional CPU processing model cannot keep up with the massive data growth. This has led to the concept of near-storage computation, where data are processed close to the storage components.

Near-storage computing is often used in modern data centers to accelerate data-intensive applications, including machine learning, databases, computer vision [5, 6] data analytics, graph

*IEEE Member, PhD in Electrical and Computer Engineering.
†Senior Engineer at Samsung Semiconductor, Inc.
‡Principal Engineer at Samsung Semiconductor, Inc.

analytics, and data security [4, 3] by reducing the overhead of data movement [1, 2]. These applications are widely deployed on heterogeneous computing systems equipped with various hardware accelerators such as Field-Programmable Gate Arrays (FPGA), Graphics Processing Units (GPU), Tensor Processing Units (TPU), and others. Such accelerators are often connected to the host system as a PCIe-attached expansion card. Although they can deliver very high performance by offloading computing-intensive tasks from the host CPU, the large-scale dataset used as input often has to be read first into host-side memory from external storage devices (e.g., SSD) and then sent to on-board memory attached to the accelerators. Once the tasks are completed, the results must be returned to the external storage via the host again. As a result, significant data movement overhead degrades the performance of the whole system.

To overcome this issue, many existing systems [11, 12, 16, 15, 13, 8] use a technique called "Peer-to-Peer" (P2P) to let the external storage communicate with the accelerator directly. With P2P, a large dataset stored in the storage can be streamed into onboard memory in the accelerator without much involvement from host memory. Commands from the host trigger this transaction. P2P transfer not only eliminates the redundant data flow between the host and its attached devices but also reduces the usage of host memory, effectively freeing up host system bandwidth. This opens up the potential of accelerators further. This paper tries to evaluate the potential of P2P and provide valuable insight for state-of-the-art P2P deployment.

In this work, we target a Samsung-pioneered product near-storage accelerator, Samsung's SmartSSD®, and evaluate its P2P performance with various transfer patterns. Unlike traditional SSDs, Samsung's SmartSSD is a computational storage device (CSD) that contains two discrete functions: a high-performance NVMe SSD and Xilinx's Ultrascale FPGA used for hardware acceleration. An internal DDR memory buffer is also available and shared between the SSD and FPGA, allowing data exchange between the two functions. This way FPGA can process the dataset directly under the P2P mode and avoid the bottlenecks that are often caused by heavy data movement between the SmartSSD and the host, thus achieving higher performance and energy efficiency.

In this paper, we studied the P2P data transfer among devices connected through the common PCIe tree. A simulation environment was built to assess the influence of data transfer in the system among the attached devices with respect to their size. Additionally, the impact of the location of the devices in the PCIe bus topology of the system on the data transfer bandwidth is demonstrated. The rest of this paper is organized as follows: Section 2 discusses several types of data communication. In Section 3, the SmartSSD and its P2P architecture are demonstrated. The evaluation of data transfer in different communication conditions is analyzed in Section 4. Finally, Section 5 concludes the paper and presents future work.

2 Background and Related work

Hardware accelerator-based heterogeneous computing systems have been widely deployed and typically contain high-performance hardware accelerators for offloading tasks from the host and large capacity and low latency storage devices such as NVMe-based SSD for storing large datasets. Although the computing performance of accelerators keeps increasing to benefit more and more applications, migrating data between the storage and accelerators is increasingly becoming a bottleneck for improving the overall system performance further, especially for those data-intensive applications. Traditionally, SSDs and accelerators are connected to the host as peripheral devices. Before the accelerators are able to work, the data set as input has to be loaded into host memory from the storage first and then routed to the onboard memory attached to the accelerators. If DMA is not used, the CPU will be much more involved in this

process and significant OS overhead such as context switch, kernel software stack activities, and cache pollution is caused.

Although DMA can be used to transfer data between SSD and host memory without much involvement of CPU, intensive data transfer is still bottlenecked by available bandwidth between host and peripheral devices. The overhead of data movement still cannot be ignored. To reduce the overhead of data movement, researchers proposed various solutions ranging from optimizing the data path to designing new computer architectures [16, 15]. One of them, Peer-to-peer (P2P) based DMA transfer has been widely explored for direct communication between SSD and accelerators.

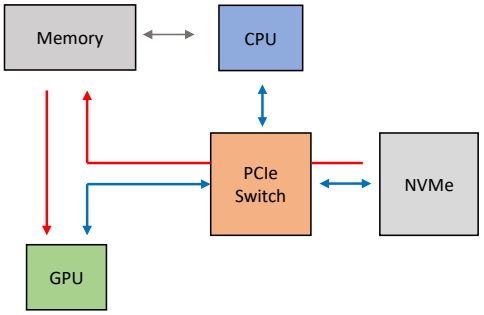
2.1 Peer to Peer transfer

PCIe P2P communication is a PCIe feature that enables two PCIe devices to directly talk to each other without using host memory as intermediate storage. Therefore, many existing works explored applying P2P to various heterogeneous computing systems. For example, P2P can be used between SSD and FPGA, SSD and GPU, or GPU and GPU.

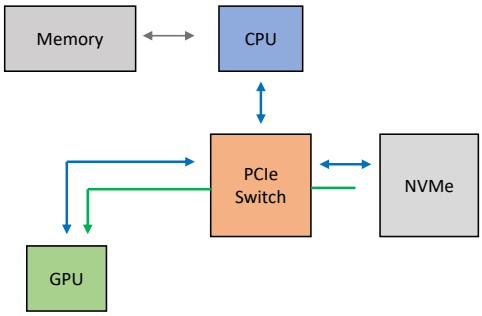
Authors in [16] proposed a novel nonvolatile memory management unit that reduces the overhead of data movement by directly connecting the SSD and the GPU being used as an accelerator. Compared with traditional IOMMU-based transfer that makes multiple data copies and causes OS-related overhead, NVMMU unifies the stacks of the SSD and the GPU and enables forwarding data between SSD and GPU without severe CPU intervention. NVMMU provides an easily-used programming model for applications. Authors in [15] proposed Morpheus, a model that allows applications to move computations to a storage device. With this model, application objects can be sent directly from a storage device to a coprocessor (e.g., a GPU) by P2P transfer. They implemented an SSD supporting this model called Morpheus-SSD that improves the performance of object deserialization, reduces power consumption, and speeds up the total execution significantly in a heterogeneous computing platform.

In [13], the authors investigated P2P DMA's potential on Ethernet NICs and NVMe SSDs. They developed a library called Libpop to manipulate memory on devices for invoking P2P DMA. In [8], a model of mapping the coprocessor memory to a system memory block of the host memory was proposed. The CPU can move data from non-volatile memory to the coprocessor without redundant copies. The authors in [10] applied P2PDMA to TCP-based applications. The authors proposed IO-TCP that targets I/O-intensive applications and offloads disk I/O and TCP packet transfer to SmartNIC from the CPU. With IO-TCP, SmartNIC can access disks via P2PDMA.

Besides the above academic work, P2P transfer is also implemented successfully in product-level acceleration platforms. Typical examples include Samsung's SmartSSD® [11], Nvidia's GPU Direct Storage (GDS) [12], and Xilinx's P2P FPGA. Figure 1 shows the architecture of GPUDirect Storage that is similar to SmartSSD's one. The left and the right show the data paths with and without using GPUDirect storage, respectively. The blue paths represent PCIe transfer and the green ones represent GPUDirect transfer between GPU and NVMe devices. For both SmartSSD and GPUDirect Storage, besides the different accelerator types (GPU) used, GPUDirect Storage supports network data from remote memory using RDMA (Remote DMA). That being said, the idea of using P2P transfer to reduce the overhead of data movement is the same.



(a) Without GPUDirect Storage



(b) With GPUDirect Storage

Figure 1: GPUDirect Storage

2.2 Why SmartSSD?

This work focuses on the evaluation of P2P performance on SmartSSD Drive. SmartSSD® is a product-level implementation of a computational storage drive from Samsung. It has been widely used for extensive research on data-intensive application acceleration and new heterogeneous computing systems [11, 14, 9]. We believe the evaluation targeting it will bring valuable insight for future P2P solutions.

3 SmartSSD Architecture

SmartSSD® is a computational storage device composed primarily of two components: an SSD storage system, and an FPGA accelerating system. Figure 2 shows the internal architecture of the SmartSSD. It contains a NAND-based NVMe SSD, Xilinx FPGA accelerator (KU15P) with an attached DRAM and a PCIe switch. The SmartSSD is connected to the host through PCIe Gen3 x4. The NAND's controller and the accelerator are connected to the integrated PCIe switch.

The FPGA DRAM as a common memory area (CMA) is exposed to both the FPGA kernels and the host memory address space as a PCIe BAR. With the assistance of the PCIe switch, the internal NVMe SSD can access the CMA for P2P transfer which reduces both Host-SSD and host-FPGA PCIe traffic significantly. That said, CMA PCIe BAR allows other PCIe devices under the same root complex to exchange data P2P. For the SmartSSD, the CPU orchestrates

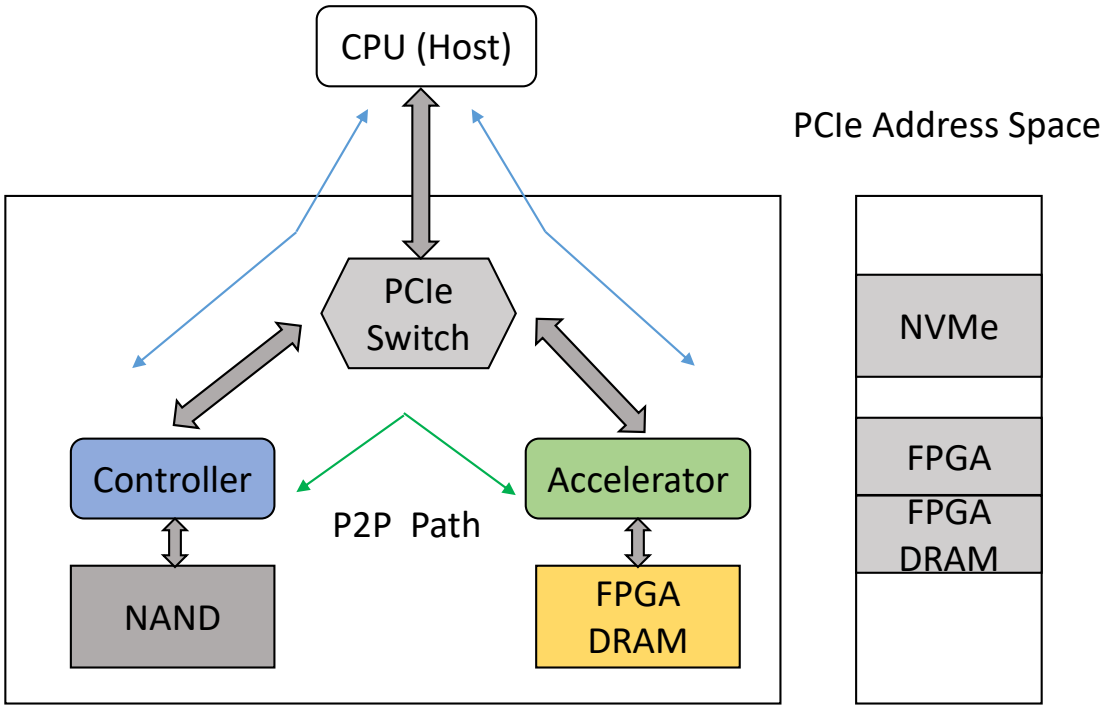


Figure 2: SmartSSD's Architecture

the system application running on the device and data movement between the SSD and the FPGA. It initializes the SmartSSD with the computation process and allocates buffer objects in the CMA. The host not only issues SSD "read" and "write" commands to the SSD controller, but it triggers the FPGA computation as well.

The SmartSSD device supports data movement between the NVMe SSD (storage system) and the FPGA (computing engine) DRAM through the in-between internal data path. By utilizing the onboard PCIe switch, the SSD and the FPGA DRAM are effectively connected allowing peer-to-peer data movement, completely eliminating the data flowing in and out of the device, and bypassing the host memory. P2P enables near-storage data processing, which thereby diminishes or eliminates data traffic between host-SSD and host-FPGA. As a result, the PCIe traffic decreases due to direct data transfer trips from host to device, leading to high performance.

4 Performance Evaluation

This section exhibits the performance study of the P2P data transfer between devices in the system. The essential objectives of this study are to illustrate the throughput of P2P data transfer and to show that data movement bypasses the host memory.

4.1 Simulation Environment

To precisely evaluate the impact of P2P data transfer among devices in the machine, we set a server attached with multiple U.2 SmartSSDs. In this simulation environment, we created two identical applications to transmit data from one device to another for P2P and non-P2P options. The application selects the source and destination devices, the block size to move, and the type of communication. The data transfer performance was evaluated for different data sizes and for different zones where the devices are connected to the system. The measurements were collected on a Dell R7515 server with an AMD EPYC processor running Ubuntu 18.04 LTS with a 4.15.0-55-generic kernel.

4.2 Evaluation Results

In this experiment, we performed data transfer from one device-SSD to another device computing engine memory in the server connected through a PCIe bus. The host code application was developed in C++ language using the Xilinx runtime (XRT) API and OpenCL APIs [7]. A buffer object was allocated in one SmartSSD device to receive data from another device's SSD connected to the same server. We measured the time it takes a block size to move from one device to another. The experiment was replicated for multiple block sizes and the transfer time was captured each time. For every test, the transfer process was repeated 100 times and the average time was calculated. Figure 3 shows the data transfer bandwidth of moving data from one device to another for both P2P and non-P2P data transfer for different block sizes.

In this Figure, both P2P and non-P2P data transfer bandwidths are monotonically increasing with the block size until they are saturated with the system capability. P2P data transfer meets the maximum possible PCIe bus bandwidth, within 4 GB/s. As shown in the figure, P2P bandwidth is always higher than the case for non-P2P no matter what the data size is. For transferring small data sizes, P2P is 6x faster compared to the non-P2P method. This is due to the time overhead to initiate the DMA with the transfer process. Though the transfer speed decreases with the data size, it saturates at 2x faster speed for the P2P mode.

The reason for this performance ratio is that the transfer throughput is associated with the data movement trip. In regular non-P2P data movement, the system allocates a subset of the main host memory, where data is moved up from the sender device to this memory subset, and

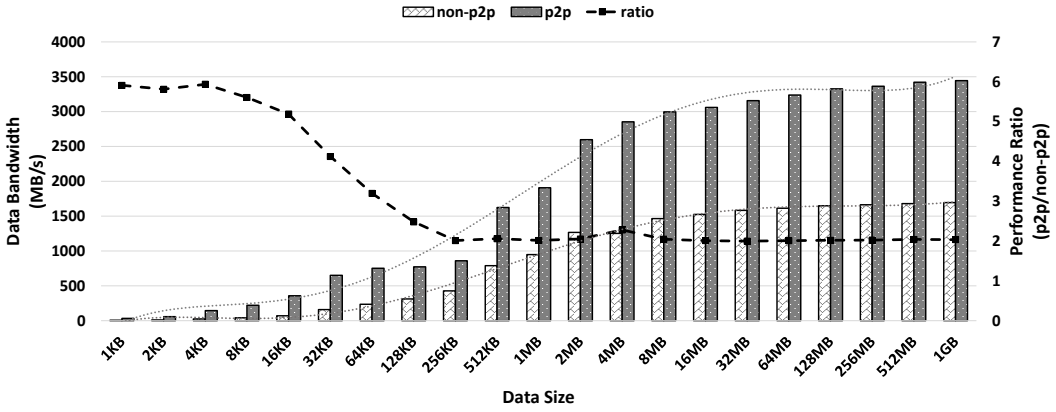
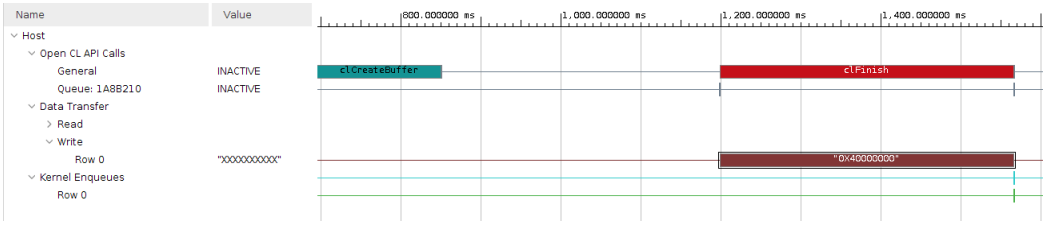
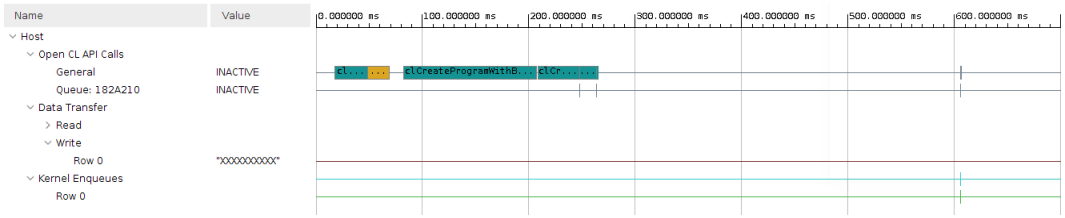


Figure 3: Average Bandwidth of P2P and non-P2P data transfer for different block sizes.



(a) Time Trace of data movement in non-P2P architecture.



(b) Time Trace of data movement in P2P architecture.

Figure 4: Time Trace of data transfer in non-P2P (a) and P2P (b) architectures.

then it is moved down to the receiver device. However, for P2P communication, data is directly moved from one device to another bypassing the main host memory.

In order to indicate the memory usage in the system during the transfer process, we provoked the XRT to capture the data profiling of the application. The application was designed with OpenCL events to trace various phases of the application layer. Vitis Analyzer was used to view and analyze trace data. Figure 4 visualizes the event trace of data transfer for P2P and non-P2P communication modes. As it is indicated in Figure 4(a) data is migrated from the memory to the device as in the write event. However, for the same application with the P2P technique, the memory is not involved in the transaction process, as shown in Figure 4(b). Therefore, transferring big chunks of data in non-P2P mode requires enough space in the memory to smooth the data communication out.

With respect to the physical place of the pair devices in the system and its influences on the p2p data movement, different PCIe slot connections in the server were tested with p2p communication. In this experiment, ten SmartSSDs were connected in ten different PCIe slots, and 100 GB of data was p2p transferred between each pair of devices. The average bandwidth was reported. Figure 5 demonstrates the P2P data transfer bandwidth from one device to another in different slots in the server. This Figure shows that the P2P data transfer bandwidth is not affected by the physical location of the devices in the system. It reaches the maximum limit of the PCIe bus bandwidth, $3 \sim 4$ GB/s. This experiment was performed on a Dell R7515 server with an AMD EPYC processor. This system is a highly scalable single-socket 2U rack and all the PCIe slots are connected to the same root complex. Therefore, each pair of devices share the same bus emphasizing the direct P2P connection.

5 Conclusions and Future Work

In this paper, we demonstrated the benefit of the P2P data transfer between devices including storage elements and accelerating engines over the non-P2P approach. The results showed that

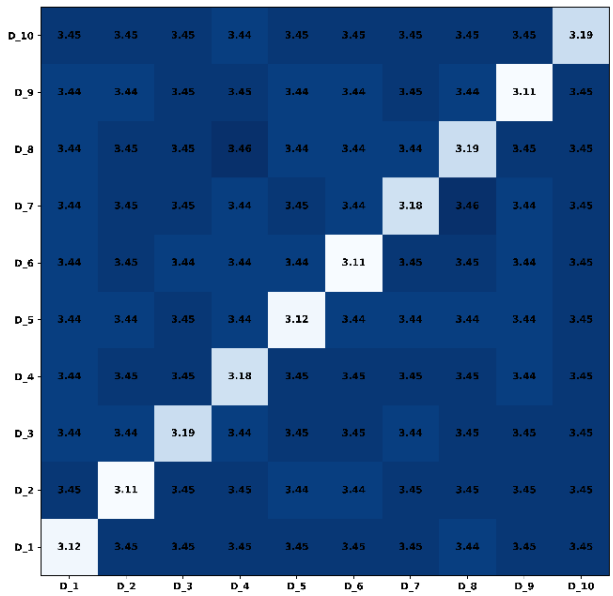


Figure 5: P2P data transfer bandwidth between multiple devices (GB/s).

P2P data transfer bandwidth is always higher than the case for the non-P2P method. The P2P to non-P2P performance ratio is double for transferring large block sizes and up to 6 times for small block sizes. In P2P architecture, the data directly moves between devices skipping the system memory, which reduces the trip time of the transferred data. In non-P2P cases, read and write operations share the physical PCIe v3 x4 interface with the host. This is the real reason that the P2P is 2X the improvement as it eliminates host reads/writes to SSD and, at the same time eliminates host reads/writes to CMA, too. In addition, the P2P data transfer bandwidth is independent of the physical location of the paired devices in the system as long as they are connected to the same root complex of the PCIe bus. For future work, moving data between multiple devices at the same time will be considered and the congestion of the PCIe bus will be evaluated.

References

[1] Luka Daoud and Hingkwan Huen. Performance study of software-based encrypting data at rest. *Proceedings of 37th International Confer*, 82:122–130, 2022.

[2] Luka Daoud, Fady Hussein, and Nader Rafla. Real-time bitstream decompression scheme for fpgas reconfiguration. In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1082–1085. IEEE, 2018.

[3] Luka Daoud, Fady Hussein, and Nader Rafla. High-level synthesis optimization of aes-128/192/256 encryption algorithms. *International Journal of Computers and Their Applications*, 29:129–136, 2019.

[4] Luka Daoud, Fady Hussein, and Nader Rafla. Optimization of advanced encryption standard (aes) using vivado high level synthesis (hls). 2019.

[5] Luka Daoud, Muhammad Kamran Latif, HS Jacinto, and Nader Rafla. A fully pipelined fpga accelerator for scale invariant feature transform keypoint descriptor matching. *Microprocessors and Microsystems*, 72:102919, 2020.

[6] Luka Daoud, Muhammad Kamran Latif, and Nader Rafla. Sift keypoint descriptor matching algorithm: A fully pipelined accelerator on fpga. In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 294–294, 2018.

[7] Luka Daoud, Dawid Zydek, and Henry Selvaraj. A survey of high level synthesis languages, tools, and compilers for reconfigurable high performance computing. In *Advances in Systems Science: Proceedings of the International Conference on Systems Science 2013 (ICSS 2013)*, pages 483–492. Springer, 2014.

[8] Myoungsoo Jung. Computing device, data transfer method between coprocessor and non-volatile memory, and computer-readable recording medium, July 3 2018. US Patent 10,013,342.

[9] Ji-Hoon Kim, Yeo-Reum Park, Jaeyoung Do, Soo-Young Ji, and Joo-Young Kim. Accelerating large-scale graph-based nearest neighbor search on a computational storage platform. *IEEE Transactions on Computers*, 72(1):278–290, 2022.

[10] Taehyun Kim, Deondre Martin Ng, Junzhi Gong, Youngjin Kwon, Minlan Yu, and KyoungSoo Park. Rearchitecting the tcp stack for i/o-offloaded content delivery. In *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022*. USENIX, 2023.

[11] Joo Hwan Lee, Hui Zhang, Veronica Lagrange, Praveen Krishnamoorthy, Xiaodong Zhao, and Yang Seok Ki. Smartssd: Fpga accelerated near-storage data analytics on ssd. *IEEE Computer architecture letters*, 19(2):110–113, 2020.

[12] Mellanox. Nvidia gpudirect™ technology – accelerating gpu-based systems, 2010.

[13] Ryo Nakamura, Yohei Kuga, and Kunio Akashi. How beneficial is peer-to-peer dma? In *Proceedings of the 11th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 25–32, 2020.

[14] Weikang Qiao, Jihun Oh, Licheng Guo, Mau-Chung Frank Chang, and Jason Cong. Fans: Fpga-accelerated near-storage sorting. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 106–114. IEEE, 2021.

[15] Hung-Wei Tseng, Qianchen Zhao, Yuxiao Zhou, Mark Gahagan, and Steven Swanson. Morpheus: Creating application objects efficiently for heterogeneous computing. *ACM SIGARCH Computer Architecture News*, 44(3):53–65, 2016.

[16] Jie Zhang, David Donofrio, John Shalf, Mahmut T Kandemir, and Myoungsoo Jung. Nvmmu: A non-volatile memory management unit for heterogeneous gpu-ssd architectures. In *2015 International Conference on Parallel Architecture and Compilation (PACT)*, pages 13–24. IEEE, 2015.