# Toward Hardware-Assisted Kernel-Bypass Data Movement and Transfer

Keisuke Iida*
*The University of Tokyo*
*iida@os.is.s.u-tokyo.ac.jp*

Takahiro Shinagawa
*The University of Tokyo*
*shina@is.s.u-tokyo.ac.jp*

## Abstract

Datacenters play a central role in modern computing infrastructure, supporting a wide range of applications from web services to data processing. Despite this diversity, a significant portion of CPU cycles is spent on simple data movement and transfer operations, such as memory filling and copying, inter-process communication (IPC), and device I/O, along with common processing like compression and encryption. Although these operations are relatively simple and repetitive, they consume valuable CPU time and pollute the CPU cache. These costs, referred to as *datacenter taxes* [1], present considerable opportunities for optimization using accelerators. Several prior studies have explored offloading data movement and transfer to DMA hardware. Recent CPUs are also equipped with sophisticated data movement accelerators [2].

However, conventional DMA-based data operations face three key challenges. First, DMA hardware cannot be directly accessed from user space. It is typically managed by privileged software such as OS kernels or hypervisors for security reasons, requiring user-space programs to invoke privileged software for each data copy, which increases startup latency and diminishes the benefit of offloading. Second, DMA hardware cannot handle virtual addresses. It typically operates only on physical addresses and is unaware of the address translation performed by the CPU using page tables. Third, DMA hardware is typically tied to specific devices. Device-to-device transfer requires kernel intervention, first copying data to a kernel buffer before sending it to the destination device.

We propose a hardware-assisted data movement and transfer scheme that offloads various memory-related operations to hardware without involving the OS kernel. Our scheme provides a dedicated hardware engine to accelerate memory operations, tightly integrated with the OS or hypervisor to enable the management of multiple address spaces, device topologies, and device memory maps. Our goal is to support the following three features: (G1) user-space DMA, (G2) virtual address support, and (G3) peer-to-peer (P2P) DMA.

To achieve G1, our hardware engine supports descriptor-based memory operations. By mapping these descriptors into user space, user processes can bypass the kernel and access the memory operation engine directly. Each process is given its own descriptors isolated from others, and the hardware engine enforces access control in coordination with the OS.

To achieve G2, our hardware engine is equipped with an address translation mechanism. The OS or hypervisor preconfigures the page tables of target processes in the hardware engine, which then performs page walks as needed to carry out address translation. This enables memory movement across domains, including within user space, between processes, between processes and the kernel, and between VMs, without intervention from kernels and hypervisors. Access controls are enforced based on capabilities.

To achieve G3, our hardware engine interprets the descriptors of various I/O devices. By initiating device memory operations itself, it enables data transfer between devices (e.g., between storage and network, or storage and GPUs) directly from user space, without relying on kernels or hypervisors.

We implemented a prototype hardware engine using the Alveo U50 FPGA board. We confirmed that offloading `memset()` and `memcpy()` in the Linux kernel significantly improves performance. We are currently implementing an address space manager to handle multiple address spaces.

In the future, we plan to implement key exchange mechanisms to enable data transfer between confidential virtual machines. We also plan to implement a user-defined code execution environment, such as eBPF, within the FPGA to provide flexibility and extensibility, allowing users to define custom processing logic in hardware from user space.

## References

[1] S. Kanev et al. Profiling a warehouse-scale computer. In *Proc. ISCA 2015*, page 158–169, 2015.

[2] R. Kuper et al. A quantitative analysis and guidelines of data streaming accelerator in modern intel xeon scalable processors. In *Proc. ASPLOS 2024*, pages 37–54, 2024.

*Keisuke Iida is a student.