



Technical Perspective

hXDP: Light and Efficient Packet Processing Offload

By Noa Zilberman

FOR A VERY long time, the network was considered the slow part of a system. Even today, our CPU is running at clock speeds of several GHz, while our home network is likely to still run at tens to hundreds of megabits per second. This, however, is far from being the case in the wired systems world. Network switches process tens of terabits every second, and many network interface cards (NICs) have one or more 100G ports. Suddenly, the CPU is slow in comparison with the rate of data arriving from the network.

Unsurprisingly, solutions to bridging the performance gap between the CPU and the network are either software or hardware based. In the following paper, the authors offer an interesting solution: taking a software-based solution (Linux's eXpress Data Path—XDP) and offloading it to the hardware (Field Programmable Gate Arrays—FPGA). In that, it achieves the best of both worlds: ease of adoption and use, combined with performance benefits.

XDP provides a safe execution environment for programmable packet processing, while running within the kernel. This is in contrast with some other solutions, such as DPDK, which use kernel bypass. Running within the kernel provides mechanisms for application isolation and security, as well as better manageability. XDP runs its programs within eBPF virtual machines, with programs written in (restricted) C and compiled to eBPF.

The NIC is an ideal target for packet processing offloading: it is located between the network and the CPU, can achieve high packet processing rate, and—more importantly—it can free up cycles on the CPU. This type of a NIC is often referred to as a smart NIC. Smart NICs based on FPGA are considered more flexible than other types of smart NICs and can support a range of bespoke accelerator architectures. However, programming

FPGAs requires expertise, and packet processing solutions tend to consume significant resources on the FPGA. While existing solutions simplify FPGA development and use high-level programming languages, they tend to limit the functionality available to the user, limit the performance, or consume significant hardware resources.

The solution suggested in the paper, termed hXDP, focuses on two goals: being able to run XDP programs efficiently on FPGA-based NICs, and consuming as little hardware resources as possible.

By using XDP, the authors free programmers from learning new programming languages, design paradigms or hardware architectures, and enable using a well-known programming model. By minimizing hardware resource consumption, the authors enable using the NIC for more than just packet processing, namely for application specific accelerators.

It is also interesting to note the non-goals of hXDP. First, hXDP does not try to outperform the packet processing rate of previous NIC-based works. Instead, it focuses on providing same or better performance than running on a CPU. This distinction enables adopting an execution model that requires a fixed amount of hardware resources.

A second non-goal is providing a transparent offloading solution to the FPGA-NIC. Programmers should still be aware that their XDP program will be running on a NIC, but they maintain the XDP programming model as in Linux.


Through careful work, the authors provide interesting insights into the XDP instruction set, which saves resources and optimizes hXDP's compiler. To that end, the authors both reduce and extend the instruction set. The reduction comes from eliminating instructions for kernel verifier checks and using cheaper hardware checks

instead. Extensions leverage the FPGA programmability, for example to support three-operand operations.

hXDP is implemented as a stand-alone logical module for an FPGA, which means it can be added or removed from an FPGA-NIC design. With resource efficiency as a goal, hXDP consumes just 10% of the FPGA logic resources, on a mid-range FPGA, and less than 4% of its memory. This leaves enough resources for other accelerators to coincide within the same NIC, as hoped.

hXDP's performance is superior to some high-end CPUs, while running at 10% of the CPU frequency. Moreover, the packet forwarding latency is an order of magnitude lower than using a CPU and lower than a comparable SoC-based NIC.

Given that hXDP achieves performance superior to CPUs for real-world network intensive applications, such as a firewall or a load-balancer, while at the same time freeing CPU resources, it is easy to see why this work is so interesting. The authors explore venues for further improvements, such as a multicore design and using larger memories. Combined with migration to ASIC or higher-end FPGA, further performance gains are almost guaranteed.

The world of smart NICs is rapidly developing, with giants such as Intel and NVIDIA introducing new solutions that combine standard SoC architecture, programmable pipelines, and accelerators. It would be fascinating to look at this work in retrospect 10 years from now and see what has prevailed: ease of programming and use, raw performance, or perhaps a mix of both? 

Noa Zilberman is an associate professor and leads the Computing Infrastructure Group in the Department of Engineering Science at the University of Oxford, U.K.

Copyright held by author.