Lauberhorn: a Smart NIC that is part of the OS

Pengcheng Xu
ETH Zurich
Switzerland
pengcheng.xu@inf.ethz.ch

Timothy Roscoe
ETH Zurich
Switzerland
troscoe@inf.ethz.ch

Remote Procedure Calls (RPC) are essential for various data center workloads, including micro-services, serverless, networked file systems, and many more. Previous work on characterizing RPC workloads [10] demonstrated that short RPC invocations in the ballpark of 1 microsecond make up a significant portion of all RPC workloads.

Despite the high frequency of short RPC workloads, traditional data center RPC architecture using PCIe DMA-based NICs incur high latency and CPU overhead. This is due to a deliberate *split of networking states* between the CPU and NIC. This historical and stable split comes from numerous factors. Firstly, there is a fundamental distrust between the OS and NIC, largely coming from the perception that the NIC never does what the OS wants. The introduction of various IOMMUs has further strengthened this view. Secondly, architecture researchers are well-known to ignore the OS, attempting to bypass it by assigning applications to fixed CPU cores to minimize scheduling disruptions. Last but not least, there is a common misconception that fine-grained interaction between the OS and NIC is slow; this might be true in traditional descriptor-ring-based PCIe DMA, but not so much in PCIe MMIO, and even less so in modern cache-coherent interconnects.

Recent works explore alternative packet delivery mechanisms to the host CPU. Many of them embody kernel bypass mechanisms to remove the OS kernel from the critical path, making trade-offs between energy-efficiency, flexibility, and throughput [1, 3, 4, 6, 7, 11]. Others integrate the NIC tightly with the operating system through various on-chip mechanisms and similarly sacrifice flexibility [5, 9]. Tightly integrating the CPU and NIC on the same die also poses challenges in practicality, since compute power and networking speeds do not evolve in lock-step.

We argue that kernel-bypass approaches are at their limit due to various advancements in the data center architecture. Servers can come with over a hundred cores, making obsolete mechanisms for multiplexing CPU time between multiple tasks. Furthermore, cache-coherent interconnects allow for communication between the CPU and NIC lower latency and higher throughput, without the need for huge batch sizes.

Such advancements call for a radically different, OS-centric approach based on new interconnects and sharing of state between the OS and NIC. In our ongoing project, Lauberhorn, we propose an accelerated communication architecture that *fuses the NIC* and the OS with coherently attached FPGA, effectively freeing the CPU from all overheads in RPC processing to achieve high efficiency and performance.

We implement a variant of the protocol proposed by Ruzhanskaia, et. al. [8] on top of cache-coherence protocols like CXL to achieve efficient fine-grained communication between the CPU and NIC. We then implement all but handler execution in RPC processing on the NIC, offloading operations such as compression, encryption, and arguments marshaling to the FPGA as hardware accelerators. Finally, a tight integration between the NIC and the OS scheduling system allow for performance as good as kernel-bypass approaches, but with far more flexibility.

A key novelty and contribution of Lauberhorn is how it uses precise scheduling states to dispatch RPC requests. We implement a *multi-level scheduling* approach, putting the NIC completely in charge of deciding where to dispatch an RPC request. The NIC holds authority of which user space RPC application runs on a CPU core. The CPU starts in a kernel thread and polls for a user space process to dispatch on the core. After a process is dispatched on the core, the CPU serves the request and continues polling *in user space* on a different pair of cache lines. The NIC can then dispatch another request to the same process with *zero software overhead* (a simple jump on the CPU core). Alternatively, it can issue a command asking the CPU core to go back to kernel mode to dispatch a different user process. Such an approach allows maximal performance without software overhead when one service is hot, while retaining flexibility switching between processes when necessary.

We set an ambitious target for an end-to-end RPC latency overhead of *1 microsecond*, accounting for everything in the request processing path other than executing the user handler function on CPU. Preliminary results on Enzian [2] show that we can send and receive Ethernet frames from the CPU over the ECI cache coherence protocol in around 800 nanoseconds, making a promising case for our latency target.

References

[1] Belay, A., Prekas, G., Klimovic, A., Grossman, S., Kozyrakis, C., and Bugnion, E. Ix: a protected dataplane operating system for high throughput and low latency. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation* (USA, 2014), OSDI'14, USENIX Association, p. 49–65.

[2] Cock, D., Ramdas, A., Schwyn, D., Giardino, M., Turowski, A., He, Z., Hossle, N., Korolija, D., Licciardello, M., Martsenko, K., Achermann, R., Alonso, G., and Roscoe, T. Enzian: an open, general, CPU/FPGA platform for systems software. In *ASPLOS '22: Proceedings*

of the Twenty-Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (February 2022).

[3] Fried, J., Ruan, Z., Ousterhout, A., and Belay, A. Caladan: Mitigating Interference at Microsecond Timescales. pp. 281–297.

[4] Humphries, J. T., Natu, N., Chaugule, A., Weisse, O., Rhoden, B., Don, J., Rizzo, L., Rombakh, O., Turner, P., and Kozyrakis, C. ghOSt: Fast & Flexible User-Space Delegation of Linux Scheduling. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (Virtual Event Germany, Oct. 2021), ACM, pp. 588–604.

[5] Ibanez, S., Mallery, A., Arslan, S., Jepsen, T., Shahbaz, M., Kim, C., and McKeown, N. The nanoPU: A Nanosecond Network Stack for Datacenters. pp. 239–256.

[6] Kaffes, K., Chong, T., Humphries, J. T., Belay, A., Mazières, D., and Kozyrakis, C. Shinjuku: Preemptive Scheduling for {usecond-scale} Tail Latency. pp. 345–360.

[7] Peter, S., Li, J., Zhang, I., Ports, D. R. K., Woos, D., Krishnamurthy, A., Anderson, T., and Roscoe, T. Arrakis: The Operating System is the Control Plane. In *11th Symposium on Operating Systems Design and Implementation (OSDI'14)* (Broomfield, Colorado, USA, October 2014).

[8] Ruzhanskaia, A., Xu, P., Cock, D., and Roscoe, T. Rethinking Programmed I/O for Fast Devices, Cheap Cores, and Coherent Interconnects, Sept. 2024. arXiv:2409.08141 [cs].

[9] Schuh, H. N., Krishnamurthy, A., Culler, D., Levy, H. M., Rizzo, L., Khan, S., and Stephens, B. E. CC-NIC: a Cache-Coherent Interface to the NIC. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1* (La Jolla CA USA, Apr. 2024), ACM, pp. 52–68.

[10] Seemakhupt, K., Stephens, B. E., Khan, S., Liu, S., Wassel, H., Yeganeh, S. H., Snoeren, A. C., Krishnamurthy, A., Culler, D. E., and Levy, H. M. A Cloud-Scale Characterization of Remote Procedure Calls. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz Germany, Oct. 2023), ACM, pp. 498–514.

[11] Zhang, I., Raybuck, A., Patel, P., Olynyk, K., Nelson, J., Leija, O. S. N., Martinez, A., Liu, J., Simpson, A. K., Jayakar, S., Penna, P. H., Demoulin, M., Choudhury, P., and Badam, A. The Demikernel Datapath OS Architecture for Microsecond-scale Datacenter Systems. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles* (Virtual Event Germany, Oct. 2021), ACM, pp. 195–211.