A Smart Cache for a SmartNIC!

Scaling End-host Networking to 400 Gbps & Beyond

Annus Zulfiqar, Ali Imran, Venkat Kunaparaju, Ben Pfaff[†], Gianni Antichi[‡*], Muhammad Shahbaz

Purdue University [†]*Feldera* [‡]*Queen Mary University of London* [*]*Politecnico di Milano*

1 MOTIVATION & GOALS

The success of modern public/edge clouds hinges heavily on the performance of their end-host network stacks if they are to support the emerging and diverse tenants' workloads (e.g., distributed training in the cloud [22, 27, 42, 48] to fast inference at the edge [14, 47]). Unlike the core network (e.g., switches and routers with 50 Tbps+ of aggregate throughput and sub-μs latencies [19, 20]), the end-host network has struggled to keep pace with the rising performance demands [32–36, 49].

In this paper, we (1) argue that it is the past assumptions and conventional wisdom that are preventing end-host networks from realizing their full potential, and (2) show how modern programmable switch pipelines (e.g., OpenFlow [25, 39] and P4 [8]), offer a novel approach to developing fast-path caches (Figure 1)—yielding significantly higher hit rates compared to conventional caches (e.g., Megaflow [33]) while operating entirely within the limited (hardware) rule space of Modern SmartNICs [18, 28, 29, 43].

1.1 End-host Networking: Past & Present

Since the late 90s, the end-host networking stack has transformed into a switching substrate [21, 33], acting as the last-hop layer in the modern distributed computing landscape—routing traffic to/from virtual machines (VMs) and containers, connecting them to the outside world [12, 21, 33, 41]. Early incarnations of these switches primarily resulted in mimicking the functionalities of fixed-function hardware switches as hardcoded software switches (e.g., Linux Bridges and OpenFlow Switches [13, 15]). We have come a long way since then (a) through a series of software optimizations aimed at maximizing CPU performance [33, 35, 36] to (b) leveraging hardware offloads using modern SmartNICs [16, 18, 28, 29, 43]. Nevertheless, the challenge persists: *the end-host networking stack struggles to scale effectively with emerging workload and increasing link rates* [36, 49].

• ***Software Optimizations.*** Applying the entire multi-table switch pipeline (e.g., OpenFlow) on each incoming packet proved prohibitively expensive; CPUs struggled with performing multiple lookups, leading to significant performance degradation with each additional lookup [33]. The *first* optimization, therefore, involved dividing the end-host software switch into a slow-path (implementing the multi-table pipeline) and a fast-path (hosting a single-lookup cache),

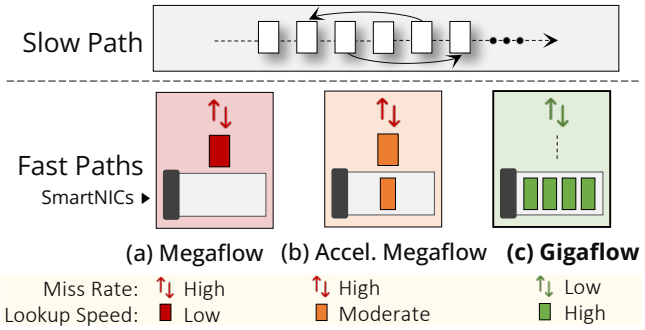

Figure 1: Comparing Megaflow, Accelerated Megaflow (with hardware cache), and Gigaflow, in terms of cache miss rates and (average) lookup speeds.

Figure 1a. The initial packet of a flow is processed by slow-path pipeline, generating a single exact-match rule stored in the fast-path cache; subsequent packets then match the cached rule. The *second* optimization involved replacing these early exact-match caches with wildcard caches (i.e., Megaflow [33]), thereby enhancing the aggregated switch throughput by handling more traffic in the fast-path. And, more recently, a *third* optimization focuses on improving the lookup speed of fast-path caches by replacing the existing Tuple Space Search (TSS) classifier [33, 38] with compact Machine Learning models (i.e., Range Query-Recursive Model Index, RQ-RMI) [35, 36]. Despite these optimizations, the inherent limitations of a CPU—declining performance due to the slowdown of Moore's Law [1]—restrict the overall performance of these switches to less than 10 Gbps per CPU.

• ***SmartNIC Offloads.*** There is an urgent push within the networking industry to shift from CPU-based end-host switching to SmartNICs, as indicated by the numerous new NIC products introduced by key players such as Amazon (e.g., Nitro [2]), Nvidia (e.g., Connect X6 [28], Bluefield DPU [29]), AMD (e.g., Pensando DPU [3, 4]), Intel (e.g., IPU [17]), Marvell (e.g., LiquidIO [23]), and Microsoft (e.g., Fungible DPU [26]). Equipped with a hardware cache (Figure 1b), these NICs can process and route traffic directly to/from the virtual endpoints (e.g., using SR-IOV [30]), thereby bypassing the software fast-path and the slow-path. These NICs can reach link speeds of 400 Gbps and higher [29] when the matching

rule is present in the hardware cache.[1] However, the main challenge with these NICs is the size of these hardware wildcard caches, which typically hold in the order of 10K rules—much smaller than the software fast-path caches [24, 33, 40]. This limitation is attributed to the restricted power budget of these SmartNICs, typically around 75 W [44], and the hardware complexity associated with integrating a large TCAM on-chip [9, 19, 20]. As a result, despite their performance advantages, the high miss rate of these caches results in significantly lower overall aggregate throughput. Therefore, these hardware caches today handle only a small subset of traffic, typically long flows, while the remainder is directed to the software fast-/slow-path.

1.2 Towards Smart Pipeline-Aware Caching

In this paper, we offer a fresh perspective on storing rules within the SmartNIC hardware. Until now, two assumptions have guided the design of fast-path caching: (1) multi-table lookups are expensive, thus necessitating the need for a single-lookup cache (e.g., Megaflow); and (2) the only available locality information for guiding cache-rule generation is derived from the traffic alone. However, these assumptions no longer hold true today.

First, unlike CPUs, the modern SmartNICs can perform multi-table lookups in the hardware at link speeds, similar to network switches (e.g., PISA [9, 19]). A pipeline of smaller TCAMs can operate at higher clock speeds compared to a single large TCAM, thus, enabling NICs to sustain even higher link rates. (This is particularly relevant as discussions around the 800 G Ethernet standard are underway [11].)

Second, modern switch (slow-path) pipelines are programmable (e.g., P4 and OpenFlow), letting the operators specify which policies to apply (e.g., L2, L3, or ACL) and in what order. We can leverage this pipeline-aware locality to further inform how we generate the cached rules. For instance, an exact-match rule captures the temporal locality of traffic (e.g., packets from a particular flow arriving frequently), while a Megaflow rule exploits the spatial locality of traffic (e.g., packets from flows with matching prefixes arriving closer in time). A Megaflow rule is, thus, an aggregate of multiple exact-match rules.

Extending this further, we observe that a slow-path pipeline is an aggregate of many Megaflow rules—each complete *traversal* of the slow-path pipeline yields a Megaflow rule. Conversely, a Megaflow rule is a composition of multiple *sub-traversals* of a slow-path pipeline, with different Megaflow rules sharing a sub-traversal.

[1]Note that the on-NIC ARM/RISC-V cores are typically less powerful and would likely result in even poorer performance compared to the server cores [10].

Pipeline	Rulespace Coverage		Increase
	Megaflow	*Gigaflow*	
OFD [31]	32.0K	14,674.6K	459×
PSC [37]	32.0K	4,977.4K	155×
ANT [5–7]	32.0K	1,283.4K	40×

Table 1: Rule space (#flow rules) coverage in Megaflow versus Gigaflow for a high-locality environment.

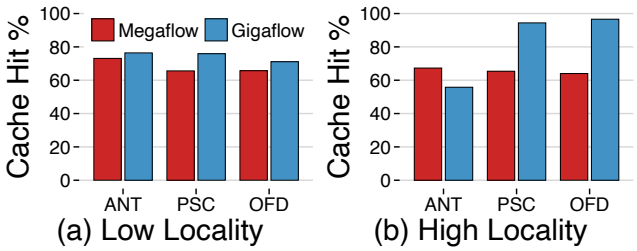

Figure 2: Comparing cache hits (%) of Megaflow versus Gigaflow using real-world vSwitch pipelines: Antrea (ANT) [5–7], Pisces (PSC) [37], and OFDPA (OFD) [31].

Gigaflow: A Sub-Traversal Cache. Based on the previous two insights: (a) line-rate multi-table lookups inside SmartNICs and (b) Megaflows with overlapping traversal, we develop a new fast-path cache, Gigaflow, that stores sub-traversals of a slow-path pipeline (Figure 1c). Upon a miss in the NIC, a mapper in the slow-path computes a set of candidate sub-traversals to install in the NIC hardware tables. The candidates are selected such that the cross-product of the new sub-traversals and the existing ones across all tables in the NIC yields the most rule-space coverage.

2 PRELIMINARY RESULTS

Our preliminary results show that Gigaflow can achieve up to 20% higher hit rate than a hardware-accelerated Megaflow cache (Figure 2) while capturing 200× more rule space on average (Table 1).

We implement Gigaflow as a 4-table P4 pipeline with 8K entries each (32K entries in total), and Megaflow as a single P4 table with 32K entries using the Xilinx's P4-SDNet compiler [45] on an Alveo U250 FPGA [46]. We generate traffic with 100K unique flows and test it against three real-world slow-path pipelines: Antrea OVS (`ANT`, 22 tables) [5–7], Pisces L2L3-ACL (`PSC`, 7 tables) [37], and Cord OFDPA (`OFD`, 10 tables) [31].

On average, in low-locality environments (Figure 2a), Gigaflow yields an 18.4% increase in hit rates compared to Megaflow across all three pipelines; and in high-locality environments (Figure 2b), the hit rate is even higher with an average increase of 46.8%. Moreover, Gigaflow is able to capture a rule space of up to 14.6M for `OFD`, whereas Megaflow was limited to only 32K rules across all pipelines.

REFERENCES

[1] MIT CSAIL Alliances. last accessed: 04/19/2024. The Death of Moore's Law: What it means and what might fill the gap going forward. https://cap.csail.mit.edu/death-moores-law-what-it-means-and-what-might-fill-gap-going-forward.

[2] Amazon. last accessed: 04/19/2024. AWS Nitro System. https://aws.amazon.com/ec2/nitro/.

[3] AMD. last accessed: 04/19/2024. Pensando. https://www.amd.com/en/accelerators/pensando.

[4] AMD. last accessed: 04/19/2024. Pensando DSC-200 Distributed Services Card. https://www.amd.com/system/files/documents/pensando-dsc-200-product-brief.pdf.

[5] Antrea. last accessed: 04/19/2024. Antrea: Enhance pod networking and enforce network policies for Kubernetes clusters. https://antrea.io/.

[6] Antrea. last accessed: 04/19/2024. Antrea OVS Pipeline. https://antrea.io/docs/main/docs/design/ovs-pipeline/.

[7] Antrea-IO. last accessed: 04/19/2024. Antrea OVS Pipeline. https://github.com/antrea-io/antrea/blob/main/docs/design/ovs-pipeline.md.

[8] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. In *ACM SIGCOMM CCR*.

[9] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. 2013. Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN. In *ACM SIGCOMM*.

[10] Xuzheng Chen, Jie Zhang, Ting Fu, Yifan Shen, Shu Ma, Kun Qian, Lingjun Zhu, Chao Shi, Ming Liu, and Zeke Wang. 2024. Demystifying Datapath Accelerator Enhanced Off-path SmartNIC. *arXiv preprint arXiv:2402.03041* (2024).

[11] Ethernet Technology Consortium. last accessed: 04/19/2024. Ethernet 800G Specification. https://ethernettechnologyconsortium.org/wp-content/uploads/2021/10/Ethernet-Technology-Consortium_800G-Specification_r1.1.pdf.

[12] Daniel Firestone. 2017. VFP: A Virtual Switch Platform for Host SDN in the Public Cloud. In *USENIX NSDI*.

[13] The Linux Foundation. last accessed: 04/19/2024. Linux Bridge. https://wiki.linuxfoundation.org/networking/bridge.

[14] Graham Gobieski, Brandon Lucia, and Nathan Beckmann. 2019. Intelligence Beyond the Edge: Inference on Intermittent Embedded Systems. In *ACM ASPLOS*.

[15] Natasha Gude, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown, and Scott Shenker. 2008. NOX: towards an operating system for networks. In *ACM SIGCOMM CCR*.

[16] Malvika Gupta. last accessed: 04/19/2024. Open vSwitch Offload by SmartNICs on Arm. https://community.arm.com/arm-community-blogs/b/tools-software-ides-blog/posts/open-vswitch-offload-by-smartnics-on-arm.

[17] Intel. last accessed: 03/12/2024. Intel Infrastructure Processing Units (IPUs) and Smart-NICs. https://www.intel.com/content/www/us/en/products/details/network-io/ipu.html.

[18] Intel. last accessed: 04/19/2024. Intel Ethernet Controller 700 Series - Open vSwitch Hardware Acceleration Application Note. https://builders.intel.com/docs/networkbuilders/intel-ethernet-controller-700-series-open-vswitch-hardware-acceleration-application-note.pdf.

[19] Intel. last accessed: 04/19/2024. Tofino: P4-programmable Ethernet switch ASIC that delivers better performance at lower power. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html.

[20] Intel. last accessed: 04/19/2024. Tofino2: Second-generation P4-programmable Ethernet Switch ASIC that Continues to Deliver Programmability without Compromise. https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-2-series.html.

[21] Teemu Koponen, Keith Amidon, Peter Balland, Martin Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, Rajiv Ramanathan, Scott Shenker, Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang. 2014. Network Virtualization in Multi-tenant Datacenters. In *USENIX NSDI*.

[22] ChonLam Lao, Yanfang Le, Kshiteej Mahajan, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. 2021. ATP: In-network Aggregation for Multi-tenant Learning. In *USENIX NSDI*.

[23] Marvell. last accessed: 04/19/2024. Data Processing Units (DPU). https://www.marvell.com/products/data-processing-units.html.

[24] Marvell. last accessed: 04/19/2024. Marvell LiquidIO III. https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-liquidio-III-solutions-brief.pdf.

[25] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. In *ACM SIGCOMM CCR*.

[26] Microsoft. last accessed: 04/19/2024. Microsoft announces acquisition of Fungible to accelerate datacenter innovation. https://blogs.microsoft.com/blog/2023/01/09/microsoft-announces-acquisition-of-fungible-to-accelerate-datacenter-innovation/.

[27] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. PipeDream: generalized pipeline parallelism for DNN training. In *ACM SOSP*.

[28] Nvidia. last accessed: 04/19/2024. CONNECTX-6 DX. https://www.nvidia.com/en-us/networking/ethernet/connectx-6-dx/.

[29] Nvidia. last accessed: 04/19/2024. NVIDIA BLUEFIELD DATA PROCESSING UNITS. https://www.nvidia.com/en-us/networking/products/data-processing-unit/.

[30] Nvidia. last accessed: 04/19/2024. Single Root IO Virtualization - SR-IOV. https://docs.nvidia.com/networking/display/mlnxofedv461000/single+root+io+virtualization+(sr-iov).

[31] Cord OF-DPA. last accessed: 04/19/2024. OpenSwitch OF-DPA User Guide. https://netbergtw.com/wp-content/uploads/Files/OPS_of_dpa.pdf.

[32] Open-vSwitch. last accessed: 04/19/2024. ofproto-dpif-upcall.c. https://github.com/openvswitch/ovs/blob/master/ofproto/ofproto-dpif-upcall.c.

[33] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. 2015. The Design and Implementation of Open vSwitch. In *USENIX NSDI*.

[34] Diana Andreea Popescu. last accessed: 04/19/2024. Latency-driven performance in data centers. https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-937.pdf.

[35] Alon Rashelbach, Ori Rottenstreich, and Mark Silberstein. 2020. A Computational Approach to Packet Classification. In *ACM SIGCOMM*.

[36] Alon Rashelbach, Ori Rottenstreich, and Mark Silberstein. 2022. Scaling Open vSwitch with a Computational Cache. In *USENIX NSDI*.

[37] Muhammad Shahbaz, Sean Choi, Ben Pfaff, Changhoon Kim, Nick Feamster, Nick McKeown, and Jennifer Rexford. 2016. PISCES: A Programmable, Protocol-Independent Software Switch. In *ACM SIGCOMM*.

[38] V. Srinivasan, S. Suri, and G. Varghese. 1999. Packet Classification Using Tuple Space Search. *ACM SIGCOMM CCR* (1999).

[39] Jean Tourrilhes, Justin Pettit, et al. last accessed: 04/19/2024. OpenFlow Switch Specification, Version 1.5.1 (Protocol version 0x06).

https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[40] Yanshu Wang, Dan Li, Yuanwei Lu, Jianping Wu, Hua Shao, and Yutian Wang. 2022. Elixir: A High-performance and Low-cost Approach to Managing Hardware/Software Hybrid Flow Tables Considering Flow Burstiness. In *USENIX NSDI*.

[41] Chengkun Wei, Xing Li, Ye Yang, Xiaochong Jiang, Tianyu Xu, Bowen Yang, Taotao Wu, Chao Xu, Yilong Lv, Haifeng Gao, Zhentao Zhang, Zikang Chen, Zeke Wang, Zihui Zhang, Shunmin Zhu, and Wenzhi Chen. 2023. Achelous: Enabling Programmability, Elasticity, and Reliability in Hyperscale Cloud Networks. In *ACM SIGCOMM*.

[42] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. TernGrad: ternary gradients to reduce communication in distributed deep learning. In *NeurIPS*.

[43] Xilinx. last accessed: 04/19/2024. Alveo SN1000 SmartNICs. https://www.xilinx.com/content/dam/xilinx/publications/product-briefs/sn1000-product-brief.pdf.

[44] Xilinx. last accessed: 04/19/2024. Alveo U25 SmartNIC. https://www.xilinx.com/products/boards-and-kits/alveo/u25.html.

[45] AMD Xilinx. last accessed: 04/11/2024. Vitis Networking P4. https://www.xilinx.com/products/intellectual-property/ef-di-vitisnetp4.html.

[46] AMD Xilinx. last accessed: 04/19/2024. Alveo U250 Data Center Accelerator Card. https://www.xilinx.com/products/boards-and-kits/alveo/u250.html.

[47] Xiaowei Xu, Yukun Ding, Sharon Xiaobo Hu, Michael Niemier, Jason Cong, Yu Hu, and Yiyu Shi. 2018. Scaling for edge inference of deep neural networks. In *Nature Electronics*.

[48] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. 2010. Parallelized Stochastic Gradient Descent. In *NeurIPS*.

[49] Annus Zulfiqar, Ben Pfaff, William Tu, Gianni Antichi, and Muhammad Shahbaz. 2023. The Slow Path Needs an Accelerator Too!. In *ACM SIGCOMM CCR*.