A Smart Cache for a SmartNIC!

Rethinking Caching, Locality, & Revalidation for Modern Virtual Switches

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

*University of Michigan †Purdue University ‡Feldera *Politecnico di Milano — •Student*

1 MOTIVATION & GOALS

Virtual Switches (vSwitches) are vital components in modern data center networks, providing a unified interface to enforce high-level policies on incoming packets and route them to physical interfaces, containers, or virtual machines. As performance demands escalate, there has been a shift toward offloading vSwitch processing to SmartNICs to alleviate CPU load and improve efficiency. However, existing solutions struggle to handle the growing flow rule space within the NIC, leading to high miss rates and poor scalability.

We introduce GIGAFLOW, a novel caching system designed for SmartNICs to accelerate vSwitch packet processing. Our core insight is that by harnessing the inherent pipeline-aware locality within programmable vSwitch pipelines—defining policies (e.g., L2, L3, and ACL) and their execution order (e.g., using P4 and OpenFlow)—we can create cache rules for shared segments (sub-traversals) within the pipeline, rather than caching entire flows (Figure 1). These shared segments can be reused across multiple flows, resulting in higher cache hit rate (up to 51%) and improved rule-space coverage (up to 450×) over traditional caches (i.e., Megaflow) using the limited memory of today's SmartNICs—all while operating at line speed. We also discuss open problems in cache revalidation mechanisms and call on the networking community to creatively address the emerging challenges.

1.1 End-host Networking: Past & Present

Since the late 90s, the end-host networking stack has transformed into a switching substrate built around virtual switches (vSwitches) [1]. Early incarnations of these vSwitches primarily mimicked fixed-function hardware switches as hardcoded software switches (e.g., Linux Bridge and iptables). We have come a long way since then (a) through a series of software optimizations aimed at maximizing CPU performance to (b) leveraging hardware offloads using modern SmartNICs. Nevertheless, the challenge persists: *these vSwitches struggle to scale effectively with emerging workloads and increasing link rates.*

• ***Software Optimizations.*** In software, applying the entire multi-table pipeline—comprising a sequence of network policies (e.g., L2, L3, or ACL)—within a vSwitch for each incoming packet is prohibitively expensive. Early optimizations introduced single-lookup caches, namely Microflow and Megaflow caches [1], to reduce this overhead. Microflow caching installs an exact-match rule after processing the first packet in a flow, while Megaflow uses wildcard rules to handle broader traffic patterns more efficiently. Recent efforts have improved Megaflow lookup speed using compact ML models like RQ-RMI [2]. Still, CPU limitations—and the slowdown of Moore's Law—cap vSwitch throughput at under 10 Gbps per core.

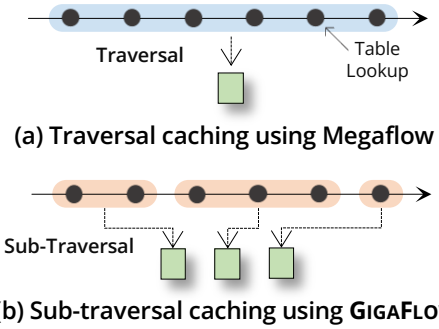

Figure 1: (a) A *traversal* is a complete sequence of table lookups through the vSwitch pipeline that generates a Megaflow rule. (b) A *sub-traversal* is a subset of these lookups within a traversal, capturing smaller, reusable segments shared across multiple flows.

• ***SmartNIC Offloads.*** There is an urgent push within the networking industry to shift from CPU-based virtual switching to SmartNICs. Equipped with a hardware (HW) cache, these NICs can process and route traffic directly to and from the virtual endpoints (e.g., using SR-IOV), effectively bypassing software-based processing. These NICs can reach link speeds of 400 Gbps and higher when the matching rule is present in the HW cache. However, the main challenge is the limited size of these HW caches, typically holding between 10–50K wildcard rules—far fewer than the software-based caches. This limitation is attributed to the restricted power budget of SmartNICs, typically around 75 W, and the complexity of integrating large TCAMs on-chip. As a result, despite their performance advantages, high miss rates within these caches result in significantly lower overall aggregate throughput. Thus, most implementations today use HW caches to handle only a small subset of traffic—primarily long and bursty flows—while directing the bulk of other traffic to software for processing.

Corresponding author: Annus Zulfiqar (zulfiqaa@umich.edu)

1.2 Towards Smart Virtual Switching

• **Pipeline-Aware Caching with Gigaflow.** Traditionally, vSwitch caching has been guided by two assumptions: (1) multi-table lookups are expensive, necessitating a single-lookup cache, and (2) cache-rule generation relies solely on traffic-derived locality (temporal and spatial). We argue that it is time to revisit these assumptions. Unlike CPUs, SmartNICs can perform multi-table lookups in hardware at line speeds, enabling cross-product rule combinations that greatly expand rule space beyond physical table limits. Secondly, vSwitch pipelines are programmable (e.g., using OpenFlow), letting operators configure the types and order of policies (e.g., L2, L3, ACL). This introduces a powerful pipeline-aware locality that extends beyond temporal and spatial localities.

Traditional caching captures locality in two forms: Microflow rules exploit temporal locality by caching exact-match flows, while Megaflow rules use wildcards to capture spatial locality by grouping flows with overlapping headers. Pipeline-aware locality, in contrast, leverages the vSwitch pipeline structure to cache shared sequences of table lookups. We extend the notion of a traversal [1]—a unique sequence of table lookups and matched rules that defines a Megaflow rule (Figure 1a). Each traversal is then broken into smaller, reusable sub-traversals (Figure 1b), which capture common lookup patterns across flows. By caching these sub-traversals in SmartNIC tables, flows can share pipeline segments, enabling more scalable and efficient caching.

Gigaflow: A Sub-Traversal Cache. Building on these insights—(a) line-rate multi-table lookups in SmartNICs and (b) Megaflow rules constructed from overlapping sub-traversals—we design Gigaflow, a caching system that maximizes rule-space coverage and cache hit rate while maintaining line-rate performance: upon a cache miss in the NIC, a mapper in the vSwitch computes a set of candidate sub-traversals to install in the SmartNIC Gigaflow 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 maximizes the rule-space coverage.

• **Incremental Cache Revalidation.** Gigaflow targets rule-space coverage in hardware, but software-only caches remain prevalent and suffer performance bottlenecks due to frequent revalidation. Caching performance depends on its rule-space coverage, but revalidation overhead scales with the number of cached entries, constraining practical cache sizes. Consequently, cache sizes are intentionally limited to support rapid, sub-second revalidation and maintain correctness. With increasing link speeds and diverse workloads, enhancing vSwitch performance necessitates reconsideration of cache revalidation which currently represents a computational bottleneck. We propose adopting recent advancements in Incremental View Maintenance (IVM) from the databases community to fundamentally rethink cache revalidation. Our ongoing research reframes cache revalidation as an IVM problem, significantly reducing complexity from the order of cached entries (hundreds of thousands) to the order of number of vSwitch rule updates (few dozens).

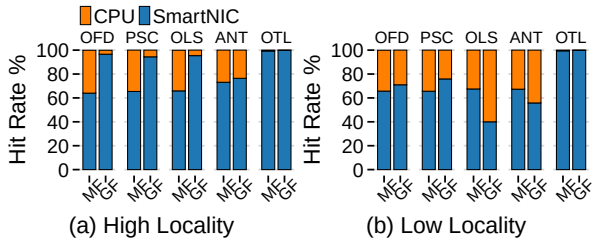

Figure 2: End-to-end cache hit rate: Gigaflow (4x8K) vs. Megaflow (32K) in high/low locality environments.

SmartNIC Cache	vSwitch Pipelines (§2)				
	OFD	PSC	OLS	ANT	OTL
Megaflow	32K	32K	32K	32K	32K
Gigaflow	14.7M	4.9M	10.8M	1.3M	48K

Table 1: Gigaflow (4x8K) vs. Megaflow (32K) maximum rule-space coverage with high-locality.

2 PRELIMINARY RESULTS

We implement Gigaflow in the Open vSwitch (OVS) [1] 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 on an Alveo U250 FPGA. We generate traffic with 100K unique flows and test it against five real-world vSwitch pipelines: Cord OFDPA (`OFD`, 10 tables), Pisces L2L3-ACL (`PSC`, 7 tables), OVN Logical Switch (`OLS`, 30 tables), Antrea OVS (`ANT`, 22 tables), and Openflow TTL (`OTL`, 8 tables).

Our results show that Gigaflow can achieve up to 51% higher hit rate than a hardware-accelerated Megaflow cache (Figure 2) while capturing up to 450× more rule space (Table 1). On average, in high-locality environments (Figure 2a), Gigaflow yields a 25% increase in hit rate compared to Megaflow across all pipelines; in low-locality environments (Figure 2b), Gigaflow performs at par with Megaflow. Moreover, Gigaflow is able to capture a rule space of up to 14.7 M for `OFD`, whereas Megaflow was limited to only 32K rules across all pipelines.

REFERENCES

[1] 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*.

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