Poster: Triton: Accelerating vSwitch with Flexibility through Hardware Assisting not Bypassing Software

Xing Li*§, Xiaochong Jiang*, Ye Yang§*, Lilong Chen*, Tianyu Xu*, Chao Xu§, Longbiao Xiao§, Fengmin Shi§, Yi Wang§, Taotao Wu§, Yilong Lv§, Hangfeng Gao§, Zikang Chen§, Yisong Qiao§, Hongwei Ding§, Yijian Dong§, Chengkun Wei*, Zihui Zhang*, Shunmin Zhu†§, Wenzhi Chen*

*Zhejiang University §Alibaba Cloud †Tsinghua University

KEYWORDS

Virtual Switch, SmartNICs, Programmability, Flexibility

1 INTRODUCTION

The vSwitch, as a critical component for Virtual Machine (VM) network connectivity in cloud environments, has prompted increasing attention towards its forwarding performance. While software optimization schemes have limitations in meeting the expanding network capacity demands [11, 12, 15, 17, 18], hardware offloading architectures leveraging SoC, FPGA, and ASIC have been proposed to transfer the match-action workload [1, 3, 6, 7, 13, 16], addressing the growing need for network capacity.

However, the existing hardware offloading solutions introduce a redundant datapath to the software vSwitch, creating new challenges. For instance, the VFP offloading solution [13] and OVS-DPDK offloading solution based on Mellanox ASAP2 [1] divide the packet forwarding process into two separate datapaths: the *software path*, which handles complete packet processing, and the *hardware path*, which accelerates packet matching as a cache for the software path. We use the term *off-path model* to refer to these solutions that distinguish software and hardware paths based on the hotness of the flow. However, the two datapaths in off-path model are not comparable in terms of performance and both require maintenance to support incoming services. This presents challenges for Cloud Service Providers (CSPs) in the following aspects:

Performance: Hardware offloading solutions enhance vSwitch forwarding capacity but introduce unpredictable VM network experience. The significant performance gap between the two datapaths (the software path and the hardware path) results in the varied treatment of network traffic, compromising VM network Service Level Agreement (SLA) guarantees. Specific scenarios like short connections or routing rule refreshes worsen the network experience. Moreover, resource consumption on SmartNICs or DPUs doubles with both datapaths, reducing concurrent connections and vNIC density originally supported by the vSwitch.

Flexibility: Flexibility plays a critical role in enabling the continuous delivery of cloud services. However, it faces challenges stemming from hardware development and resource constraints. Firstly, we observed that relying solely on the OpenFlow network programming model[14] is insufficient to support new services comprehensively. For example, advanced functionalities, like traffic mirroring[5, 9] and Flowlog[2, 8], necessitate expanding matching fields and actions, adding development complexity. Engineers must now navigate designing functions across two datapaths, leading to longer development cycles. Secondly, the existing architecture amplifies operation and maintenance costs due to system complexity and longer call chains.

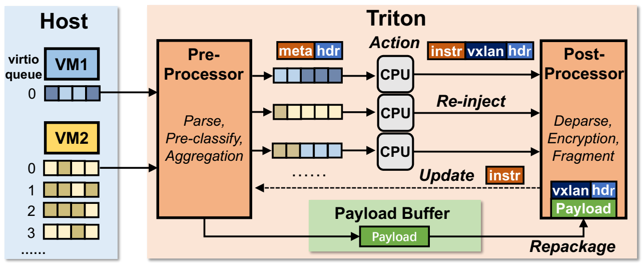

Figure 1: The architecture of Triton

To address these challenges, we present Triton, a novel hardware accelerated vSwitch system that leverages hardware assistance rather than bypassing software. Triton aims to offload I/O and memory-intensive tasks to FPGA units while preserving high flexibility tasks, such as action execution, within the software domain. By adopting this approach, Triton ensures guaranteed network Service Level Agreements (SLAs) and predictable performance through a unified datapath. Additionally, we introduce optimization methods like header-payload slicing and packet aggregation to enhance CPU-based software forwarding capacity through vectorization. The deployment of Triton in Alibaba Cloud validates its effectiveness. Evaluations demonstrate that compared to existing hardware offloading architectures, Triton achieves nearly a two-fold increase in Connections Per Second (CPS) with a minimal latency increase of only $10\mu s$. Furthermore, Triton maintains unified performance metrics and flexibility.

2 SYSTEM DESIGN

2.1 Design Overview

Triton's core principle is integrating hardware and software within a unified datapath to ensure consistent performance metrics. However, a crucial challenge lies in effectively splitting and distributing packet processing loads among different processing units.

In Triton, workload distribution is determined through instruction-level analysis of the three stages involved in packet processing: parsing, classification, and action. The vSwitch datapath architecture of Triton is illustrated in Figure 1. The parsing stage, characterized by intensive memory access and jump instructions, is implemented as a pre-processing module in hardware units to alleviate the burden on the CPU. Hardware acceleration is used for best-effort acceleration in the classification stage, with intermediate data passed to

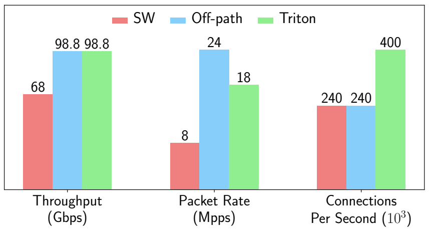

Figure 2: Performance comparison

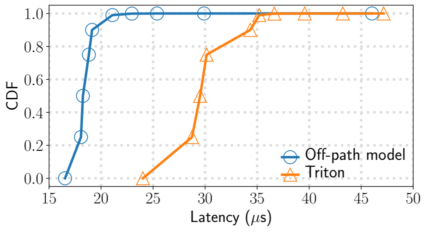

Figure 3: Latency comparison

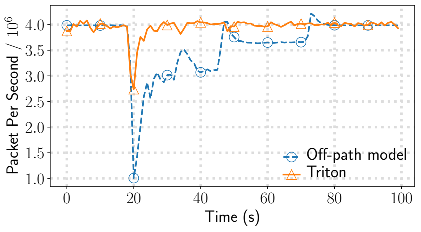

Figure 4: Predictable performance

software as metadata for faster packet matching. Action execution, requiring high flexibility, is fully implemented in software on the CPU to allow for quick modifications.

Triton offers the advantage of maintaining consistently high performance while preserving flexibility comparable to pure software solutions. Furthermore, Triton enhances fine-grained monitoring and debugging capabilities, addressing limitations observed in existing hardware offloading solutions, including features like comprehensive traffic statistics and real-time hooks.

2.2 Optimization Techniques

To overcome the shortcomings of software processing and improve the upper limit of performance in Triton, we design the following optimization techniques:

Overcome software forwarding capacity bottleneck: (1) *Header-payload slicing*: To alleviate the software forwarding capacity bottleneck, Triton implements header-payload slicing. The *pre-processor* slices packets into headers and payloads, as most vSwitch workloads primarily operate on headers. By efficiently storing payloads in a dedicated *payload buffer*, Triton minimizes PCIe traffic and enables higher bandwidth capacities. For actions that require access to the payload (e.g., encryption and fragmentation), the software will instruct the hardware's *post-processor* to perform corresponding operations on payloads, according to the directives in the *instr* field of re-injected packets. (2) *Packet aggregation*: Triton employs packet aggregation to enhance software processing efficiency. Packets belonging to the same flow are aggregated in hardware and processed in batches by CPUs. CPUs can use SIMD instructions for vectorized processing like [10], resulting in improved packet rate, reduced cache miss rate, and lower software processing latency.

Minimize the interaction between software and hardware: (1) *Implicit hardware updates*: To mitigate CPU overhead in hardware synchronization, Triton employs implicit hardware updates. The CPU can insert pre/post-processing rules into the *instr* field of re-injected packets, allowing the hardware to update functions after resolving the issued rules; (2) *Lightweight rules recycling*: Triton delegates the management of the processing pipeline to the hardware to minimize the overhead of software-driven recycling of pre/post-processing rules. The hardware initiates the recycling process if a specified threshold is exceeded. This approach reduces software overhead and enhances efficiency.

3 PRELIMINARY EVALUATION

Triton is developed and deployed in Alibaba Cloud Infrastructure Processing Unit (CIPU) [4], with only 4 CPU cores and a small number of LUT units are used.

Triton Performance: Figure 2 presents a performance comparison among the software vSwitch (SW), vSwitch accelerated with off-path model, and Triton. By implementing the techniques discussed in Section 2.2, Triton achieves throughput comparable to that of off-path model. Regarding packet rate, Triton demonstrates a more than twofold improvement compared to SW solutions. Although Triton introduces a nominal latency difference of approximately $10\mu s$, as shown in Figure 3, compared to the off-path model, its impact is negligible for most typical cloud workloads, like MySQL and Redis, that typically have latencies around 10 *ms*. Furthermore, Triton offers predictable high performance in multiple dimensions and increased flexibility, providing significant advantages.

Predictable Performance: Figure 4 illustrates the packet rate variation over a duration of 100 seconds for different offloading models, all initially supporting 2 million connections. At 20s, we update the routing rules in the vSwitch. The off-path model exhibits substantial performance deterioration, with a decline of approximately 75% in scenarios involving frequent rule changes, lasting up to 50 seconds. In contrast, Triton utilizes the mechanisms outlined in Section 2.2 to alleviate the strain on the CPU and install new rules through the data plane. As a result, Triton experiences only a 25% performance decrease for approximately 5 seconds. This demonstrates Triton's capability to deliver tenants a more predictable performance.

REFERENCES

[1] 2020. Mellanox ASAP2 Accelerated Switching and Packet Processing. https://network.nvidia.com/files/doc-2020/sb-asap2.pdf. (2020).
[2] 2021. Logging IP traffic using VPC Flow Logs. https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html. (2021).
[3] 2021. Virtual Switching Offload on FPGA-Based Alveo® SmartNICs. https://www.youtube.com/watch?v=Gh1zSYrFymM. (2021).
[4] 2022. A Detailed Explanation about Alibaba Cloud CIPU. https://www.alibabacloud.com/blog/599183?spm=a3c0i.23458820.2359477120.3.76806e9bESi3SD. (2022).
[5] 2022. Amazon Virtual Private Cloud Traffic Mirroring. https://docs.aws.amazon.com/vpc/latest/mirroring/what-is-traffic-mirroring.html. (2022).
[6] 2023. AWS Nitro System. https://aws.amazon.com/cn/ec2/nitro/. (2023).
[7] 2023. Broadcom TRUFLOW™. https://www.broadcom.com/solutions/data-center/cloud-scale-networking. (2023).
[8] 2023. Overview of Flowlog. https://www.alibabacloud.com/help/en/virtual-private-cloud/latest/flow-logs-overview. (2023).
[9] 2023. Traffic mirroring overview. https://www.alibabacloud.com/help/en/virtual-private-cloud/latest/traffic-mirroring-overview. (2023).
[10] 2023. Vector Packet Processing (VPP). https://fd.io/. (2023).
[11] Alireza Farshin, Tom Barbette, Amir Roozbeh, Gerald Q. Maguire Jr, and Dejan Kostic. 2021. PacketMill: toward per-Core 100-Gbps networking. In *ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021*, Tim Sherwood, Emery D. Berger, and Christos Kozyrakis (Eds.). ACM, 1–17. https://doi.org/10.1145/3445814.3446724
[12] Daniel Firestone. 2017. VFP: A Virtual Switch Platform for Host SDN in the Public Cloud. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, Aditya Akella and Jon Howell (Eds.). USENIX Association, 315–328. https://www.usenix.org/

conference/nsdi17/technical-sessions/presentation/firestone

[13] Daniel Firestone, Andrew Putnam, Sambhrama Mundkur, Derek Chiou, Alireza Dabagh, Mike Andrewartha, Hari Angepat, Vivek Bhanu, Adrian Caulfield, Eric Chung, Harish Kumar Chandrappa, Somesh Chaturmohta, Matt Humphrey, Jack Lavier, Norman Lam, Fengfen Liu, Kalin Ovtcharov, Jitu Padhye, Gautham Popuri, Shachar Raindel, Tejas Sapre, Mark Shaw, Gabriel Silva, Madhan Sivakumar, Nisheeth Srivastava, Anshuman Verma, Qasim Zuhair, Deepak Bansal, Doug Burger, Kushagra Vaid, David A. Maltz, and Albert Greenberg. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 51–66. https://www.usenix.org/conference/nsdi18/presentation/firestone

[14] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (March 2008), 69–74. https://doi.org/10.1145/1355734.1355746 Place: New York, NY, USA Publisher: Association for Computing Machinery.

[15] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan J. Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Jonathan Stringer, Pravin Shelar, Keith Amidon, and Martín Casado. 2015. The Design and Implementation of Open VSwitch. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15)*. USENIX Association, USA, 117–130. event-place: Oakland, CA.

[16] Salvatore Pontarelli, Roberto Bifulco, Marco Bonola, Carmelo Cascone, Marco Spaziani Brunella, Valerio Bruschi, Davide Sanvito, Giuseppe Siracusano, Antonio Capone, Michio Honda, and Felipe Huici. 2019. FlowBlaze: Stateful Packet Processing in Hardware. In *16th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2019, Boston, MA, February 26-28, 2019*, Jay R. Lorch and Minlan Yu (Eds.). USENIX Association, 531–548. https://www.usenix.org/conference/nsdi19/presentation/pontarelli

[17] Alireza Sanaee, Farbod Shahinfar, Gianni Antichi, and Brent E. Stephens. 2022. Backdraft: a Lossless Virtual Switch that Prevents the Slow Receiver Problem. In *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*, Amar Phanishayee and Vyas Sekar (Eds.). USENIX Association, 1375–1392. https://www.usenix.org/conference/nsdi22/presentation/sanaee

[18] William Tu, Yi-Hung Wei, Gianni Antichi, and Ben Pfaff. 2021. Revisiting the Open VSwitch Dataplane Ten Years Later. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference (SIGCOMM '21)*. Association for Computing Machinery, New York, NY, USA, 245–257. https://doi.org/10.1145/3452296.3472914 event-place: Virtual Event, USA.