Towards Scalable SDN Switches: Enabling Faster Flow Table Entries Installation

Roberto Bifulco, Anton Matsiuk
NEC Laboratories Europe, Germany – E-mail: <firstname.lastname>@neclab.eu

CCS Concepts

•**Networks** → **Bridges and switches;** *Network performance analysis;*

Keywords

Software-defined Networking, Reactive Control Plane, OpenFlow

1. INTRODUCTION

SDN applications operate the switching hardware in new ways, uncovering some strict constraints of current switching devices. The large flow table entries installation time is among these constraints, since it limits the applicability of hardware SDN switches, in particular in those applications that require a reactive logic. The source of this performance limitation lays in the technological solutions adopted by current ASICs, which typically rely on Ternary Content Addressable Memory (TCAM) to implement the flow tables. While TCAM can be used to implement very fast and powerful lookups, for instance to search on a range of values, unfortunately their write operations are quite slow. Furthermore, the implementation of flexible lookups may require the writing of more than one entry in the memory, in a precise order, sensibly increasing the installation time.

Inspired by previous work that suggests the combination of software and hardware forwarders [2], and noticing the trend in adding powerful CPUs to commodity *white box* switches[1], we argue that a hybrid software-hardware switch may help in lowering the flow table entries installation time and present ShadowSwitch (sSw), an OpenFlow switch prototype that implements such design. Notice that our work complements the previous work that uses similar hybrid architectures to increase the switch's flow table size [2, 1].

sSw builds on two key observations. First, software tables are very fast to be updated, hence, forwarding tables updates always happen in software first. Eventually, entries are moved to the TCAM-based forwarding tables, to achieve higher overall throughput and offload the software forwarder. However, partially overlapping entries may break the forwarding decision semantic. In fact, an entry in the hardware tables will always have an actual priority which is higher than the priorities of the entries in the software tables, since the lookup in software is performed only in case there are no entries matching a packet in hardware. To re-establish the correct relative priority ordering between overlapping entries, while keeping a small installation time, ShadowSwitch builds on a second observation: deleting entries from TCAM is in most of the cases much faster than adding them. Hence, ShadowSwitch may translate an entry installation in a mix of installation in software tables and deletion from hardware tables.

[1]http://www.opencompute.org/projects/networking/

SIGCOMM '15 August 17-21, 2015, London, United Kingdom

ACM ISBN 978-1-4503-3542-3/15/08.

DOI: http://dx.doi.org/10.1145/2785956.2790008

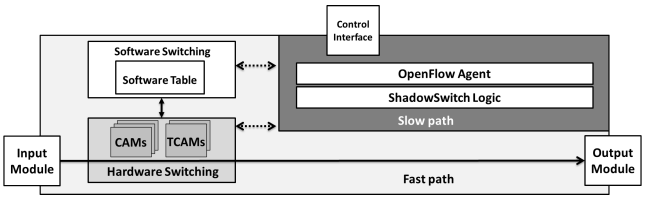

Figure 1: High-level ShadowSwitch's design

2. SHADOWSWITCH

The sSw architecture is shown in Figure 1: a high-performance software switching layer is introduced in the switch's fast path, that is, the part of the switch that performs packet forwarding operations. The software switching layer (SwSw) runs on a general purpose system, generally composed of a CPU with its caches hierarchy, implemented by SRAMs, a slower central memory based on DRAM and a high-speed system bus, which connects the CPU with the central memory and the switching ASIC. The *ShadowSwitch Logic* (sSwLogic), which is located in the sSw's slow path, manages the flow table entries (FTEs) installation.

flow_mod handling. An OpenFlow's *flow_mod* message is used to install, delete or modify a FTE. In sSw the installation of a new FTE is always performed in the SwSw, where the installation time is usually in the sub-millisecond range and constant, being not affected by other entries already installed in the forwarding tables, as it happens when the installation is done in TCAMs. Once the FTE is placed in the SwSw, the *flow_mod* processing terminates and packets start to be forwarded in the fast path. However, since FTEs are eventually moved to the hardware layer (HwSw), additional actions are required if the HwSw contains lower priority FTEs that (partially) overlap the newly installed FTE (which exists only in software). In these cases, together with the FTE installation in the software layer, the sSwLogic performs an additional action that is either the deletion from the HwSw of (all) the FTE(s) that have a dependency on the newly installed one[2] or the installation of the new FTE in the HwSw. The rationale behind this solution is that FTE deletion from a hardware table is, in principle, a faster and simpler operation, while a FTE installation may require a reorganization of the

[2]This may actually need the deletion of the entries which have dependencies also with the ones that have been just deleted.

previously installed FTEs. Nevertheless, deletion of the FTEs from the HwSw is not always possible. In fact, the sSwLogic checks the throughput (in terms of packets and bytes per second) contributed by the FTEs in the HwSw, deleting them only if their aggregated throughput could be afforded by the SwSw. In the other cases, sSw falls back to a direct installation of the new FTE in the HwSw.

When a *flow_mod* contains a FTE deletion request, the sSwLogic deletes the FTE either from the software switching layer or from both layers if the FTE was moved to hardware.

3. PROTOTYPE AND EVALUATION

We implemented a sSw prototype using a commercial hardware OpenFlow switch (HwSw) and an OpenvSwitch (OVS) instance running on a HP DL380G7 server equipped with an Intel Xeon L5640 (6 cores @ 2.26 GHz). The server is connected to the HwSw using both the switch's control port and two 1Gbit data plane ports. The sSwLogic, implemented as a user-level application, works as a proxy between the switches and an OpenFlow controller, which we implemented using POX on the same server. A second server, connected to two switch's ports, works as traffic generator and receiver. In all the tests we compare sSw performance with the performance of a high-end hardware OpenFlow switch.

***flow_mod* performance:** we measured the FTEs installation performance of the hardware switch when different FTEs priority patterns are used. Figure 2 shows the results. It is worth noticing that the FTE installation time increases almost exponentially with the number of already installed entries, when FTEs are installed in ascending priority order, demonstrating that TCAM entries reordering is happening. Instead, when FTEs have same priorities or they are installed in descending priority order, the installation time increases linearly with the number of entries in the forwarding table. Finally, FTE deletion time is much lower than the installation time and does not change in dependence of the FTEs' priorities (thus, we show a single curve for deletion in Fig. 2).

Our sSw prototype has two different performance behaviors. In the best case, it shows a constant FTE installation time, irrespective of the number of already installed FTEs and relative priorities, improving the hardware switch performance by 1 to 3 orders of magnitude. This is the behavior of sSw when the FTE that is going to be installed does not depend on any lower priority FTE in the HwSw. In case of dependencies, sSw deletes the dependent FTEs from HwSw in addition to installing the new FTE in SwSw. Figure 3 shows that the installation time grows linearly with the number of FTEs that have to be deleted. Also, Fig. 3 further shows that the deletion of several FTEs is (almost) always a faster operation than the addition of a single higher priority FTE to the HwSw. If the FTEs deletion is not possible, e.g., it would cause excessive load on the SwSw, sSw falls back to the hardware switch performance.

Reactive flow installation: in a second test we measure the installation delay of 10K network flows handled by a switch's port, using a reactive logic, i.e., a new flow generates an OpenFlow's *packet_in* message that is handled by the POX controller, which in turn installs a new FTE for such flow. We configure our traffic generator to timestamp the moment in which a flow's packet is sent and received. The flow installation delay is the time difference between these two timestamps for the first flow's packet. It is worth noticing that with this definition we include also the delay introduced by the POX controller, i.e., we look at the system as a black box. All the FTEs have same priority and define non-overlapping flows, which are generated at a rate of 1K flows per sec., over a 10 sec. period.

The measured flow installation delays are shown in Figure 4, where each flow is assigned with an id that reflects the generation order, i.e., first generated flow is assigned with id #1, the second

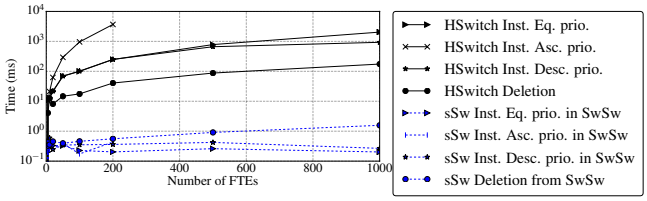

Figure 2: Cumulative execution time for forwarding table updates when installing/deleting a variable number of entries.

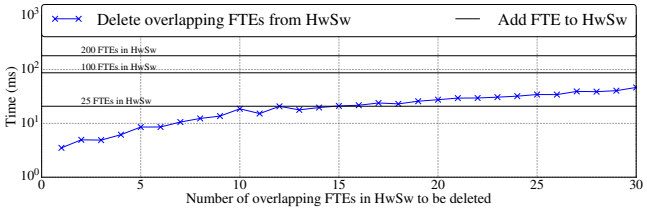

Figure 3: Time to delete a variable number of FTEs compared to the time to install a single FTE.

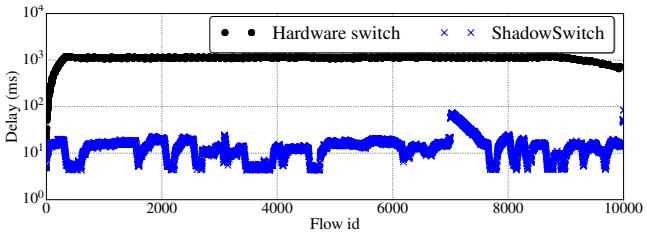

Figure 4: Flow installation delay (in log scale)

has id #2, and so on. The hardware switch drops about the 75% of the 10000 generated flows. The installation delay grows from the 5 ms experienced by flow #1 to about 1.1 s for flows with id bigger than 318. When using sSw no flows are dropped during the test, with an installation delay that is within 22 ms for the 95% of the flows. Figure 4 also shows that, when using sSw (lower line in figure), the flow installation delay experiences variations in the range 5 ms-20 ms. We believe these variations (and the higher delay peak for flows #7000 to #7500) are mainly introduced by our testbed, since we are running the POX controller, OVS and our user-space sSwLogic implementation on the same server.

4. FUTURE WORK

ShadowSwitch demonstrates that a fast software switching layer may help in reducing the flow table's entries installation time, however, the viability of the approach is dependent on the ratio between the software forwarding and hardware forwarding speeds, on the dependencies between flow entries, on the properties of the network flows in terms of packets and bytes per second. Our future work is to evaluate the system behavior when varying these parameters and to design smart algorithms that selects which entries should be moved first to the hardware flow table.

Acknowledgment

This work has been partly funded by the EU in the context of the "BEBA" project (Grant Agreement: 644122).

5. REFERENCES

[1] N. Katta, J. Rexford, and D. Walker. Infinite CacheFlow in software-defined networks. Proceedings of ACM SIGCOMM HotSDN, 2014.

[2] N. Sarrar, S. Uhlig, A. Feldmann, R. Sherwood, and X. Huang. Leveraging zipf's law for traffic offloading. *SIGCOMM Comput. Commun. Rev.*, 2012.