FlowShadow: a Fast Path for Uninterrupted Packet Processing in SDN Switches

Yi Wang, Dongzhe Tai, Ting Zhang, Boyang Xu, Linxiao Jin, Huichen Dai, Bin Liu*
Dept. of Computer Science and Technology
Tsinghua University, Beijing, China

Xin Wu
Big Switch Networks, Inc.
Santa Clara, CA 95054

ABSTRACT

Updating rules in the flow tables of SDN switches are complex and time-consuming. Therefore, we propose a cache-based scheme (named *FlowShadow*) to improve the packet processing performance and keep continuous operating while updating rules in the flow tables. FlowShadow caches the microflows in the hash table to build a fast path for packet processing. By leveraging the Action Table, FlowShadow achieves update consistency and good update performance. In order to examine the reliability, validity, utility and scalability of FlowShadow, we implement FlowShadow on the Open VSwitch and conduct numerous experiments with different settings to measure the performance of FlowShadow. The experimental results demonstrate that FlowShadow achieves a lookup speed of 75 million packets per second on a commodity PC under the real backbone traces; the system with FlowShadow speeds up 3.4× times of the original Open VSwitch.

Categories and Subject Descriptors

C.2.6 [**Computer Communication Networks**]: Internetworking

General Terms

Algorithms, Design

Keywords

Software-defined Networks; Fast Path; Update

1. INTRODUCTION

Software-Defined Networking (SDN) allows various network applications to process packets by deploying fine-grained rules on the flow tables of switches [1]. Updating rules in physical flow tables are complex and time-consuming. Often, a single rule change leads to multiple rules switched into the flow tables because rules are interdependent. Earlier work [2] on measuring the performance of inserting rules into commodity switches shows that the per-rule update time is 3.3 ms when the rules have the same priority; and the per-rule update time reaches 18 ms when inserting 600 rules with different priorities.

The lookup process is suspended while the flow tables being updated. Frequent rule updates cannot be avoided in current commodity SDN switches, which dramatically degrades the lookup performance and exacerbates the processing latency. Therefore, *it is imperative to find a solution that can keep packets being processed while the flow tables are updated*.

Caching the microflows is a potential solution. Based on the observation that network communications exhibit strong locality, Congdon *et al.* [3] use a prediction circuit before the flow tables to speedup flow classification of incoming packets. However, the prediction circuit cannot preserve the counters of rules and does not address the issue of statistics. Open vSwitch [4] (OVS), an open-source software implementation of an OpenFlow switch, implements two datapaths, the *fast path* and the *slow path*, to process the incoming packets. The fast path is the cache of exact-match rules, and the slow path is the original flow table. The first packet of each microflow undergoes the slow path to identify the highest-priority matching wildcard rule [5], and subsequent packets of the microflows in the cache are processed by the fast path. OVS's fast path stores the reference of rules in the microflows' associated data to preserve the rule counters and keep update consistency, but it only supports the single flow table (e.g., OpenFlow 1.0). Meanwhile, the lookup performance of the OVS' fast path is poor since each packet undergoing the fast path still needs to match the rule to get the final action.

In this paper, we proposes FlowShadow, a general solution based on cache, to keep working continuously with frequent rule updates. More specifically, our central contributions include:

1. The design of a new scheme for keeping update consistency between the multiple flow tables and the cache. We organize the actions of the microflows in a hash table (named *Action Table*), and leverage the state of each action (valid, or invalid) to indicate the states of the microflows' corresponding rules. Many microflows can share the same entry in the Action Table that is small enough to fit into the on-chip memory.
2. The design of a new data structure for caching microflows and their corresponding counters, actions and other associated data. The data structure has fast lookup speed and high cache replacement performance.

2. FLOWSHADOW

2.1 The Framework of FlowShadow

FlowShadow achieves fast packet processing and supports uninterrupted update by caching microflows. The framework of an SDN datapath applying FlowShadow is illustrated in Figure 1, here we take the packet processing pipeline based on TCAM as an example of the original flow tables. There are two paths for packet processing: the fast path (FlowShadow) and the slow path (the original datapath based on the flow tables). When the system receives a packet, it first parses this packet and partitions the packet into the fields and the payload. The fields are handled by the datapath to find the rules that the packet should follow; and the payload is stored in the main memory waiting to be forwarded.

In the beginning, the packet's fields are searched in the FlowShadow to find the packet's corresponding microflow. If the microflow is found, the actions of the microflow will be executed on the packet; Otherwise, the fields will be searched in the original flow tables to find the rules and actions. Since not all of the rules are stored in the flow tables, it is possible that the packet cannot find any matching rule. If that happens, the switch will send a *packet-in* message to the controller. In the case of successfully finding the

*Corresponding Author: Bin Liu, liub@tsinghua.edu.cn. This work is supported by 863 project (2013AA013502), NSFC (61373143, 61432009, 61402254), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20131019172), Tsinghua University Initiative Scientific Research Program (20121080068), CISCO Award Fund, China Postdoctoral Science Foundation (No. 2014M550734), and Jiangsu Future Networks Innovation Institute: Prospective Research Project on Future Networks (No. BY2013095-1-03).

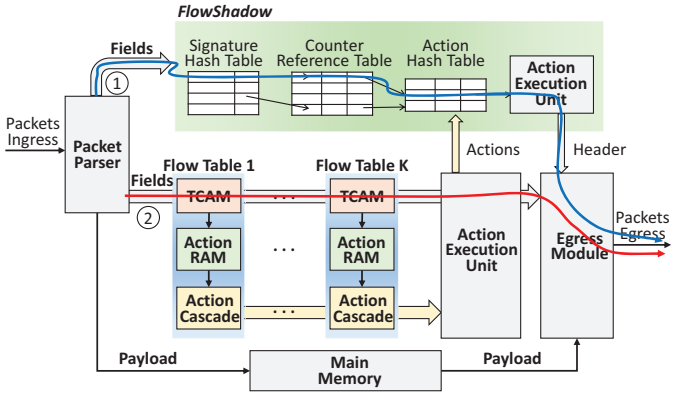

Figure 1: The framework of an SDN datapath applying FlowShadow.

matching rules, FlowShadow will cache the fields (the microflow) and its associated data that includes the references of counters and the actions.

Based on the observation that the amount of different actions is small, we store all the actions in a hash table (*Action Table*). When a new microflow is inserted into the hash table, FlowShadow first searches the Action Table to find the matching one. If there is a matching action in the Action Table, the reference of this action will be returned; otherwise, the new action is inserted into the Action Table, and the reference of this new action is returned. In the associated data of a microflow, the action list records the references of actions stored in the Action Table.

2.2 Update Consistency

In an SDN switch with a single flow table, the rule and the action is a one-to-one correspondence. The modification of a rule is directly reflected on its action. Meanwhile, the number of actions is very small. Based on these observations, we leverage the Action Table to implement update consistency. When a microflow finds its corresponding action is invalid in the Action Table, it will be removed from the cache.

However, in the multiple flow tables, a final action stored in the Action Table is cascaded by the multiple middle actions of the rules. In other words, one action of a rule cannot be found in the Action Table anymore. To leverage the Action Table for update consistency, FlowShadow records each rule's corresponding final actions. When a rule is modified, FlowShadow can set the actions to invalid by looking up the Action Table to obtain the references of the actions.

The Action Table is easy to maintain and is small enough to be fit into the on-chip memory. We can achieve weak update consistency without losing any performance by exploiting the Action Table in FlowShadow.

3. EXPERIMENTS

3.1 System Implementation

FlowShadow is implemented based on the open source project Open vSwitch (release version 2.1.2 [6]). Specifically, we have done the following works: 1) We modify the *fast path* of OVS' datapath to implement the cache mechanism and the update scheme of FlowShadow; 2) We implement a module for supporting multiple flow tables by reusing the existing modules of the single flow table in OVS. This multiple hash table module still has 16-bit port numbers (Strating from OpenFlow 1.1, the later OpenFlow version uses 32-bit port numbers), and it can support the most of actions defined in OpenFlow 1.4; 3) For measuring the performance, the FlowShadow runs in the user space instead of the kernel space.

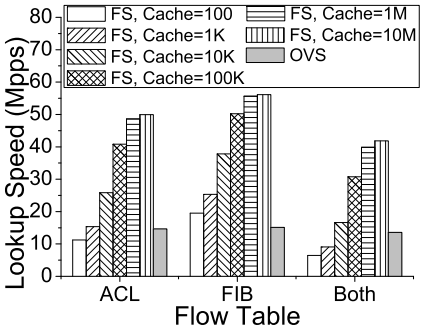

Figure 2: The lookup performances of the methods with different flow tables under the traffic of Backbone-1 (24 threads).

3.2 Experiments Setup and Results

Computational platform: The OVS with FlowShadow runs on a commodity PC with two CPUs (6-core, 1.6 GHz per core). The PC runs Linux Operating System in the version 2.6.41.9-1.fc15.x86_64. The part of multi-core parallel processing is developed using OpenMP API [7] in version 2.5.

Flow tables: The multiple flow table in our experiments consists of two real flow tables: 1) an IP prefix table downloaded from RIPE [8] (Equinix, 2012-01-01) contains 388,344 IP prefixes; 2) and an ACL set from the website [9] contains 752 rules that consider 5-tuple flows and subject to the format of ClassBench [10].

Traces: The traces used in our experiments are captured from the routers of Chicago and San Jose in Mar. 20, 2014. [11], and each trace sustains 5 minutes.

The system performance: The system applies FlowShadow as the fast path runs on the commodity PC that contains 2 CPUs (total 24 physical threads). The lookup performances of the system with FlowShadow and without FlowShadow on different flow tables are illustrated in Figure 2. Under the Backbone-1 trace, the system with FlowShadow (10M entries in the cache) achieves 49.94 Mpps, 56.11 Mpps and 41.86 Mpps on the flow tables of ACL, FIB, and both ACL and FIB, respectively. It is about 3.4× of the original OVS which achieves 14.67 Mpps, 15.11 Mpps and 13.58 Mpps on the flow tables of ACL, FIB, and both.

4. REFERENCES

[1] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[2] Xin Jin, Hongqiang Harry Liu, Rohan Gandhi, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Jennifer Rexford, and Roger Wattenhofer. Dynamic Scheduling of Network Updates. In *ACM SIGCOMM*, 2014.

[3] P.T. Congdon, P. Mohapatra, M. Farrens, and V. Akella. Simultaneously reducing latency and power consumption in openflow switches. *IEEE/ACM Transactions on Networking*, 22(3):1007–1020, June 2014.

[4] Ben Pfaff, Justin Pettit, Keith Amidon, Martin Casado, Teemu Koponen, and Scott Shenker. Extending Networking into the Virtualization Layer. In *HotNETs'09*.

[5] Gaetano Catalli. Open vSwitch: performance improvement and porting to FreeBSD.

[6] Open vSwitch 2.1.2. `http://openvswitch.org/releases/openvswitch-2.1.2.tar.gz`, 2014. [Online].

[7] The OpenMP API specification for parallel programming. http://openmp.org/wp/. `http://openmp.org/wp/`, 2014. [Online].

[8] RIPE NCC: RIPE Network Coordination Centre. `http://www.ripe.net/`, 2014. [Online].

[9] An Access Control List. `http://www.arl.wustl.edu/~hs1/project/filterset/acl1`, 2014. [Online].

[10] David E Taylor and Jonathan S Turner. Classbench: A packet classification benchmark. In *INFOCOM'05*, pages 2068–2079. IEEE, 2005.

[11] CAIDA Data. `http://www.caida.org/data/`, 2014. [Online].