IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-0926-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOMWKSHPS54753.2022.9798228

# STATOEUVER: State-aware Load Balancing for Network Function Virtualization

Wendi Feng*, Ranzheng Cao*, and Zhi-Li Zhang†
*Beijing Information Science and Technology University
†University of Minnesota – Twin Cities

## I. INTRODUCTION

By running network functions as software on commodity servers instead of on expensive dedicated hardware middleboxes, network function virtualization (NFV) offers many benefits. It provides flexible control and management of network functions (NFs), and it lowers network management and deployment costs. In particular, NFV makes it easier and cheaper to *scale out* to meet growing traffic demands by deploying NF instances on more CPU cores, hence has been widely touted as the future of networking.

These benefits notwithstanding, recent studies show NF states play critical roles in retaining peak performance [1], [2]. Using a simple synthesized NF as an example, a network monitor (NM) maintains a state variable (a counter) to record the number of packets received. However, balancing virtualized NF (VNF) instance loads with the awareness of states (NFV-SLB) is challenging because correctness must be guaranteed without compromising performance. In particular, when traffic is dispatched to multiple NM instances, the counter is shared across instances and accessed exclusively. Thus, significant performance penalty encounters (see Sec. II-B).

State-of-the-art NFV load balancers are either stateless load balancing based purely on the packet fields (*e.g.*, Sprayer [3]) or load-state scheme that each instance can only access state locally (*e.g.*, RSS++[4]). However, state access patterns by various NFs and the flow granularity of different NFs are neglected but yet critical. In this paper, we propose STATOEUVER, a state-aware load balancer for NFV. Based on the findings of our previous NFV profile work [2], it takes state access patterns, state sizes, the number of live flows, and CPU loads into account and generates practical traffic dispatching rules for efficient traffic steering along with state-aware VNF instance load balancing. Our contribution is two-fold:

- We identify and describe the NFV-SLB problem, and
- we present the design of the STATOEUVER load balancer.

## II. MOTIVATION

This section motivates the NFV-SLB problem with examples. From state sizes, shared states, to state access patterns, we gradually demonstrate issues behind these factors, which consequently contributes a state-aware NFV load balancer that takes all these factors into consideration and makes a judicious decision to attain the ultimate performance.

### A. State Size Hurts

Fig. 1a depicts the throughput experiments on our testbed of the Layer-4 load balancer (L4LB)[1] as the size of state changes. When the state size increases[2] the throughput decreases after passing the point $p_1$ or $p_2$ in different packet size experiments. This is due to the cache misses when the state size exceeds the CPU cache capacity [2]. Finally, the performance stops decrease as all memory accesses become DRAM-bounded, and the NF packet processing is significantly impacted.

### B. Shared States Pitfalls

Another experiment shows performance penalty resulted from multiple NF instances accessing a shared state variable. The same testbed setup is employed, and NM[3] is used. As shown in Fig. 1b, when traffic is "sprayed" to the two instances, each instance updates the same state variable every time a packet arrives. Hence, the state variable bounces between the two instances in the modern hierarchical memory architecture [2], and cache misses are significant. Thus, we can see from Fig. 1c, throughput is less than that of a single instance. Whereas the overall performance of the non-shared state version observed a linear increase as the number of instances increase. Unfortunately, the non-shared state version cannot always guarantee the correctness of NM (as the state on each NM instance only counts a portion of the whole traffic).

### C. State Access Pattern Impacts

So the question is: do NFs always access states on a per-packet basis? The answer is no. Many NFs (*e.g.*, Firewall, intrusion detection system, deep packet inspection) access (especially update) states only a few times each NF flow. In these cases, completely mitigating shared states can lead to load imbalance on NF instances, as shown in Fig. 1d, which compromises the performance. Hence, tradeoffs (see Fig. 1e) between shared state accesses and the number of instances that can be employed for processing packets is the key to achieving the optimal performance (throughput).

## III. STATOEUVER DESIGN

This section presents the primary design of STATOEUVER state-aware network function load balancer. In a word, STATOEUVER determines the number of instances needed and how traffic should be dispatched to them. Note that a system profile is necessary in practice but is out of the scope of this paper. Please refer to our previous work [2] for more details.

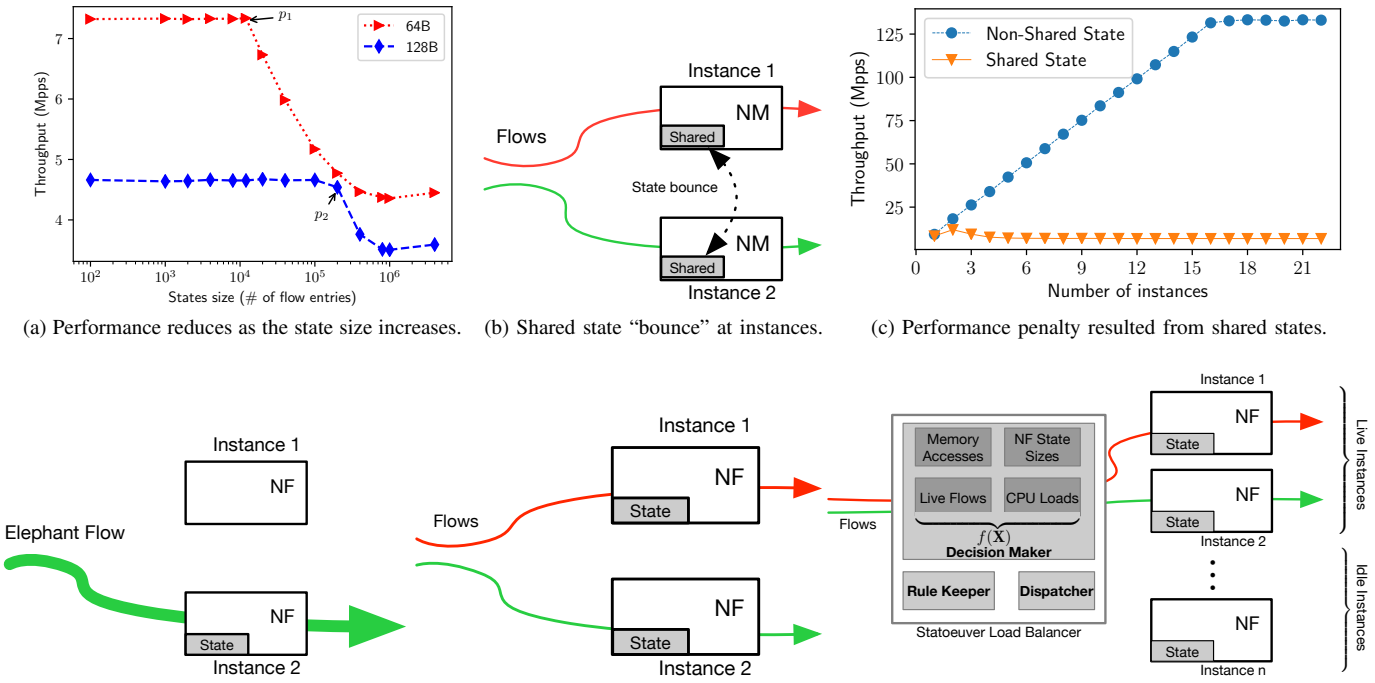
### A. Workflow Overview

As demonstrated in Fig. 1f, STATOEUVER consists of three key components, *Decision Maker*, *Rule Keeper*, and *Dispatcher*.

[1]L4LB balances flows based purely on the 5-tuple information.

[2]We use "# of flow entries" to indicate the state size as each state variable is associated with an L4LB flow (5-tuple flow in this scenario), and hence, the overall state size is proportion to the number of flows.

[3]NM counts the number of packets of incoming traffic. In the shared state version, only one state variable is used (a counter) by all instances. While in the non-shared state version, each instance has a dedicated state variable.




(a) Performance reduces as the state size increases. (b) Shared state "bounce" at instances. (c) Performance penalty resulted from shared states.

(d) Flow-level steering. Instances can be idle. (e) Low frequency access makes shared states acceptable. (f) STATOEUVER component structure.

Fig. 1: Motivation examples. (The testbed has two commodity servers, and each equipped with a 100Gbps NIC, 384GB DRAM, and 24-core CPUs @2.7GHz. Each instance uses a dedicated core. DPDK is employed, workload is generated by TRex.)

Decision Maker first makes load balancing strategies based on the status of the NFV system. The output of the load balancing strategy is traffic dispatching rules. These rules are stored by Rule Keeper, and it then generates leaner rules using traffic classifications to reduce the total amount. These rules are then installed to the hardware NIC by Dispatcher to attain line-rate speed packet dispatching.

### B. Considered Factors

The goal of STATOEUVER is to balance the traffic workload to VNF instances and achieve the best throughput performance. Following our analysis in Sec. II, Decision Maker leverages four main factors and makes tradeoffs between them to generate traffic dispatching strategies. *Memory Access* records the frequency of the state being accessed by the NF for each flow. *NF State Sizes* aggregates the total state size on each NF instance, which gives indications for the number of instances that should be employed. *Live Flows* records the number of flows received by the load balancer. *CPU Loads* logs a load of each instance utilizing the CPU core. After getting these information, the STATOEUVER load balancer leverages the STATOEUVER algorithm to generate *primary* dispatching rules.

### C. The STATOEUVER Algorithm

The NFV-SLB problem is an integer linear programming optimization problem, but traffic dispatching should be fast as multi-hundreds Gbps (or above) throughput requires nano-seconds-level latencies. Hence, the algorithm should be as simple as possible. We employ a heuristic approach briefly described as follows. **i)** STATOEUVER first reads state access pattern from Memory Access to determine an NF flow-level dispatching or a subflow-level dispatching[4]. When the number of state accesses is "little" in each flow processing, the subflow-level dispatching is utilized. **ii)** Next, STATOEUVER retrieves state variable sizes from NF State Sizes along with Live Flows to calculate the total state size. The size is used to determine whether the cache capacity is exceeded. If yes, more instances are needed. **iii)** Finally, STATOEUVER gets the CPU core loads. When a core is (almost) overloaded, new flows are dispatched to other cores (instances). If all live cores are (almost) overloaded, a new core is used (auto-scaling).

## IV. CONCLUSION

This paper has analyzed the NFV-SLB problem, in which NFV performance hinges critically on states. To address it, we have presented a novel state-aware load balancer – STATOEUVER that judiciously considers state access patterns and intelligently aligns state sizes, the number of live flows, and CPU loads into consideration for the near-optimal load balancing. We present the paper here to call for insightful and valuable comments for our future work to finalize STATOEUVER.

## REFERENCES

[1] R. Iyer, L. Pedrosa, A. Zaostrovnykh, S. Pirelli, K. Argyraki, and G. Candea, "Performance contracts for software network functions," in *Proc. of NSDI'19*.

[2] P. Zheng, W. Feng, A. Narayanan, and Z.-L. Zhang, "Nfv performance profiling on multi-core servers," in *Proc. of IFIP Networking'20*.

[3] H. Sadok, M. E. M. Campista, and L. H. M. Costa, "A case for spraying packets in software middleboxes," in *Proc. of ACM HotNets'18*.

[4] T. Barbette, G. P. Katsikas, G. Q. Maguire Jr, and D. Kostić, "Rss++ load and state-aware receive side scaling," in *Proc. of ACM CoNEXT'19*.

[4]Packets in one flow can be dispatched to different instances.