NFOS: Automatic Scaling of Single-threaded Network Functions


Lei Yan


Yueyang Pan


Diyu Zhou


Sanidhya Kashyap


George Candea

DSLab + RS3Lab

Appropriate abstractions can hide concurrency from developers while allowing them to convey enough information for NFOS to scale an NF.

Developing scalable NFs is hard

? Write parallel NF code → ? Understand scalability bottleneck → ? Remove scalability bottleneck

Need complex synchronization techniques for scalability, e.g., fine-grained locking.

Existing tools only show contention in code blocks. Unclear what are the real scalability issues: False sharing? Inefficient locks? Limitation of NF semantics?

Need to rewrite parallel code. May introduce new concurrency bugs or scalability issues.

NFOS: Develop scalable NFs *in a single-threaded manner*

Snippet of an NAT written in NFOS

```
fw-nat.c:
  ...
      // Allocate public <IP,port> with NFOS
      // index allocator interfaces.
      if (!nfos_alloc_index(pub_ip_port_pool, &
          index)) { drop_pkt(); }
      endp_pub = {.ip = to_ip(index),
                  .port = to_port(index)};
  ...
   /* Address translation */
    pkt->src_ip = endp_pub.ip;
    pkt->src_port = endp_pub.port;
  ...
```

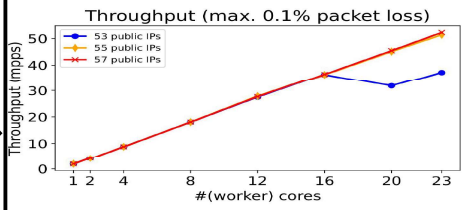
Write *single-threaded* NF code

NFOS profiler/recipe below shows bottleneck is the insufficient number of indexes in index allocator. For this NAT NF it means not enough public IPs.

```
Total txs: 94,713,483
Total tx aborts: 905,918
Total tx abort ratio: 1.0%
fw-nat.c:L42 if (!nfos_alloc_index(
    pub_ip_port_pool,
tx aborts: 890,919 98.3%
Recipe: Overprovision indexes.
```

Find & Understand scalability bottleneck *at the level of NF logic*

Increasing the number of public IPs from 53 to 55 makes the NAT scalable.



Remove scalability bottleneck *by changing the single-threaded code.*

NFOS abstractions

Programming model:
- Specify "packet set" and NF state local to it.
- Access "global state" shared by packet sets through NFOS state interfaces.

Transaction-abort proxy metric for scalability & Profiler for it

Scalability-enhancing recipes

NFOS runtime

Scale the NF leveraging the prog. model:
- Process a packet set on a single core to avoid synchronization on its local state;
- Process a packet in a transaction to exploit fine-grained parallelism on global state access.

The number of transaction aborts serves as an abstract proxy metric for determining the extent to which each line of single-threaded NF code hurts scalability due to conflict access to global state.

NF scalability depends on access to global state which are done via NFOS interfaces. Recipes show how to reduce the scalability impact of an interface operation, and hence improve NF scalability.

NFOS-based NFs (NAT, bridge, load balancer, and firewall) achieve competitive perf. as hand-parallelized Cisco VPP NFs.

Guided by NFOS profiler/recipes, developers can productively improve NF throughput by up to 91x through semantic relaxations.