redhat.

TC SW datapath: a performance analysis

Paolo Abeni, Davide Caratti, Eelco Chaudron,
Marcelo Ricardo Leitner - Red Hat

LPC, Vancouver 2018

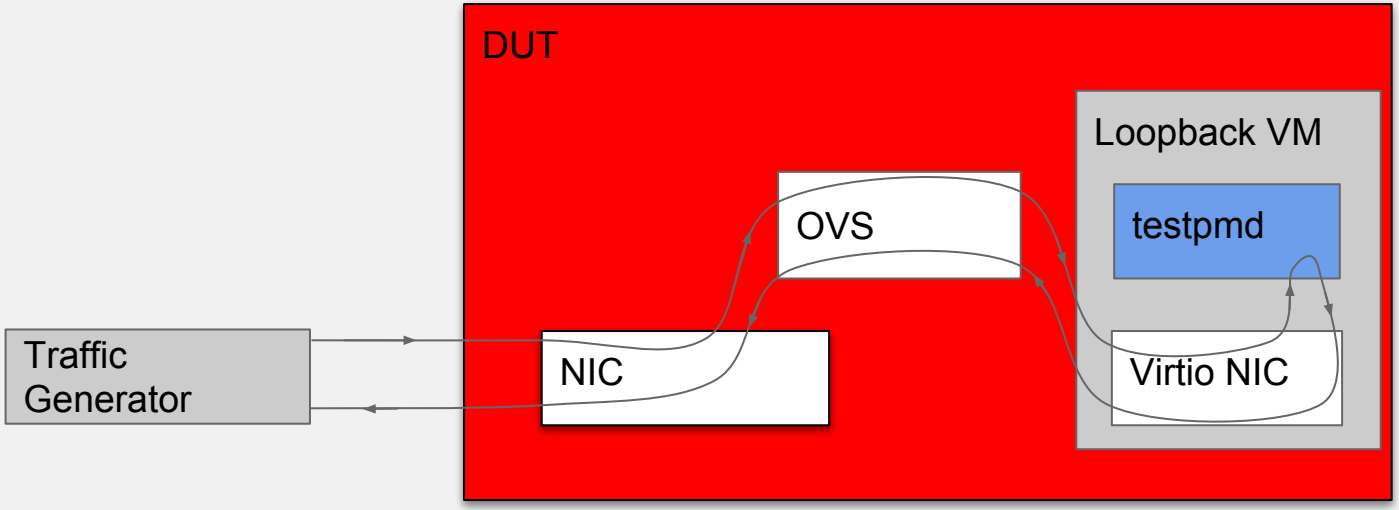
Outline

- An history of 2 datapaths
- The testing scenario
- Performance analysis, recent and current status
- Will eBPF save the world?

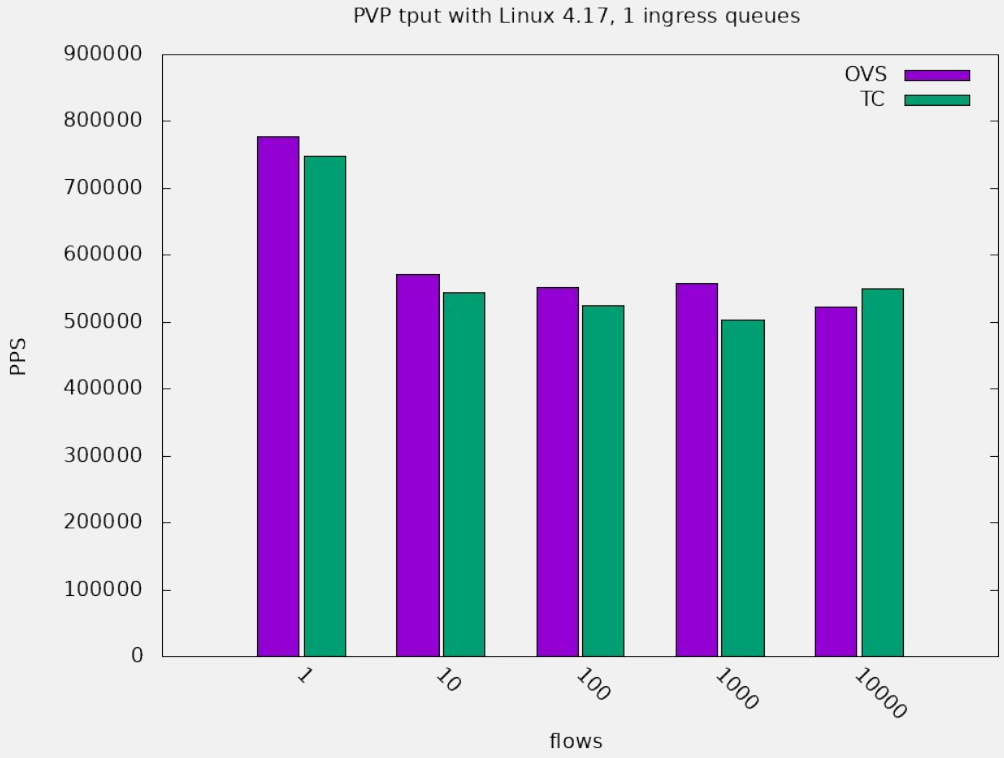
Why we need 2 in kernel OVS datapath?

- “Old” kernel OVS datapath
  - first “fast” OVS datapath implementation
  - Feature-complete
- TC S/W:
  - Created to allow for H/W offload
  - Considered slower
  - Lacks some features - conntrack

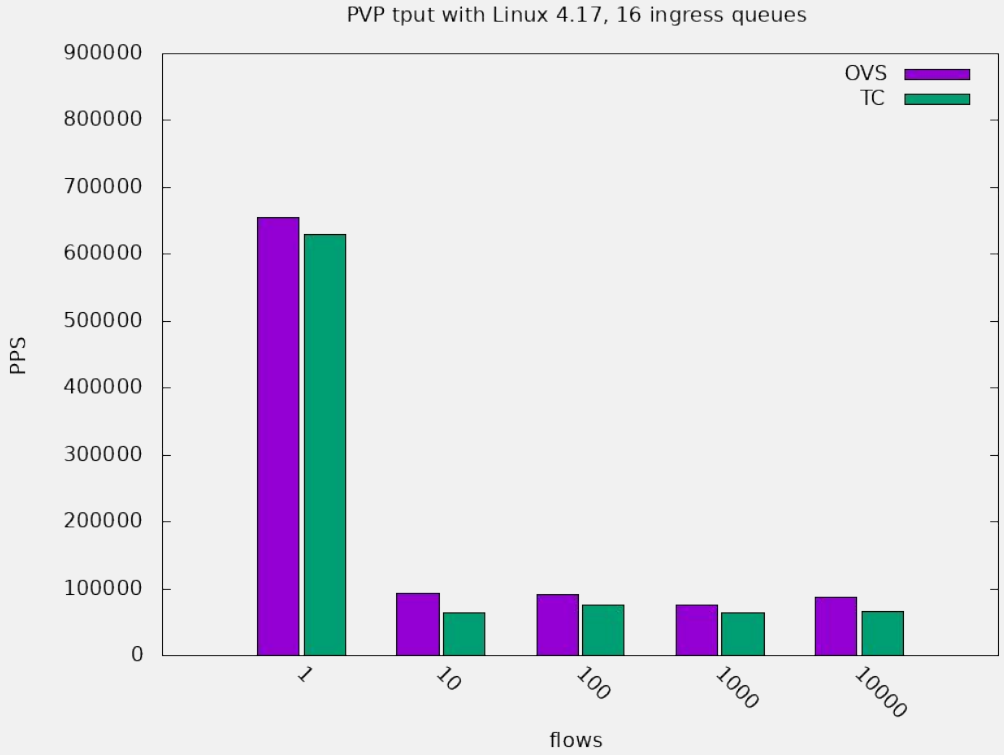
The PVP test scenario



Let’s see the numbers



How about scaling?



Why are we so slow? Can perf tell us?

Topmost perf offenders for vhost (1 queue)

5.49%	vhost_get_vq_desc
4.99%	skb_release_data
4.83%	__qdisc_run
4.68%	tun_do_read
4.36%	__skb_flow_dissect
3.81%	_copy_to_iter
3.76%	translate_desc
3.36%	iov_iter_advance
3.19%	kmem_cache_free
3.18%	ixgbe_xmit_frame_ring
2.79%	tun_get_user
2.72%	handle_rx

Topmost perf offenders for vhost (16 queues)

10.15%	tun_do_read
8.77%	skb_release_data
7.72%	vhost_get_vq_desc
6.22%	_copy_to_iter
5.41%	__slab_free
4.86%	handle_rx
4.26%	vhost_net_buf_peek
4.26%	translate_desc
4.23%	kmem_cache_free
3.81%	__check_object_size
3.40%	iov_iter_advance
2.65%	skb_release_head_state

Not entirely obvious...

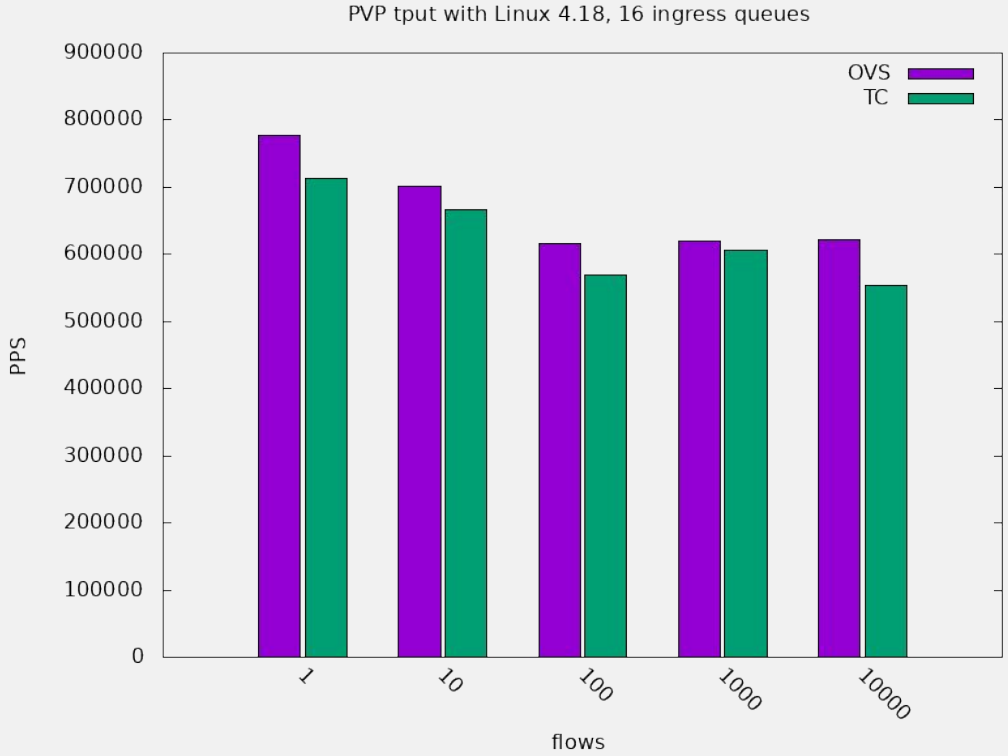
More help from perf

call-graph accounting for vhost (16 queues)

```
100.00%         vhost-<pid>
|
 --       |--87.85%--ret_from_fork
          |           kthread
          |           vhost_worker
          |           |
                      |--79.25%--handle_rx
                                 |--55.45%--tun_recvmsg
[...]
                     | --8.60%--handle_tx
                                 |
                                 |--7.42%--tun_sendmsg
 [...]
```

vhost forwarding is asymmetric! And fixing it is simple: apply the same limits to both handle_rx() and handle_tx(). Implemented in the 4.18 release cycle

Did we improve?



Can we do any better?

Topmost perf offender for vhost on Linux 4.18

4.18 with OVS backend

7.50% masked_flow_lookup
5.02% ixgbe_xmit_frame_ring
4.20% vhost_get_vq_desc
3.89% iov_iter_advance
3.18% translate_desc
2.92% pfifo_fast_dequeue
2.83% tun_build_skb.isra.57
2.81% tun_get_user
2.09% __dev_queue_xmit
1.99% handle_tx

With TC backend

5.08% ixgbe_xmit_frame_ring
4.63% vhost_get_vq_desc
4.37% skb_release_data
3.86% translate_desc
3.00% iov_iter_advance
2.94% tun_get_user
2.89% __skb_flow_dissect
2.76% memcmp
2.54% pfifo_fast_dequeue
2.17% rhashtable_jhash2

Again, not entirely obvious… let's look towards the bottom…

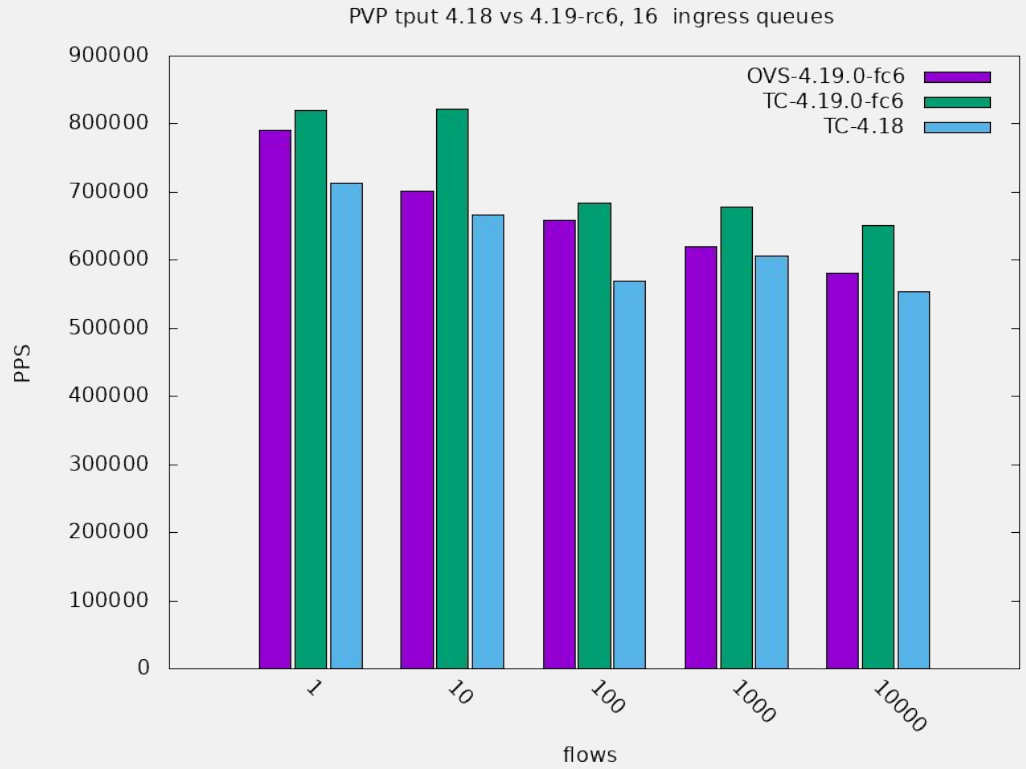
[...]

0.71% skb_clone

Killing bad clones

- In the TC S/W datapath packets are forwarded via the TC act_mirred action
  - It clones the skb and return a control action. The caller acts on the original skb accordingly
  - The TC S/W datapath uses DROP as the control action. We can avoid the clone and forward directly the original skb
  - Implemented in the 4.19 release cycle

The current status



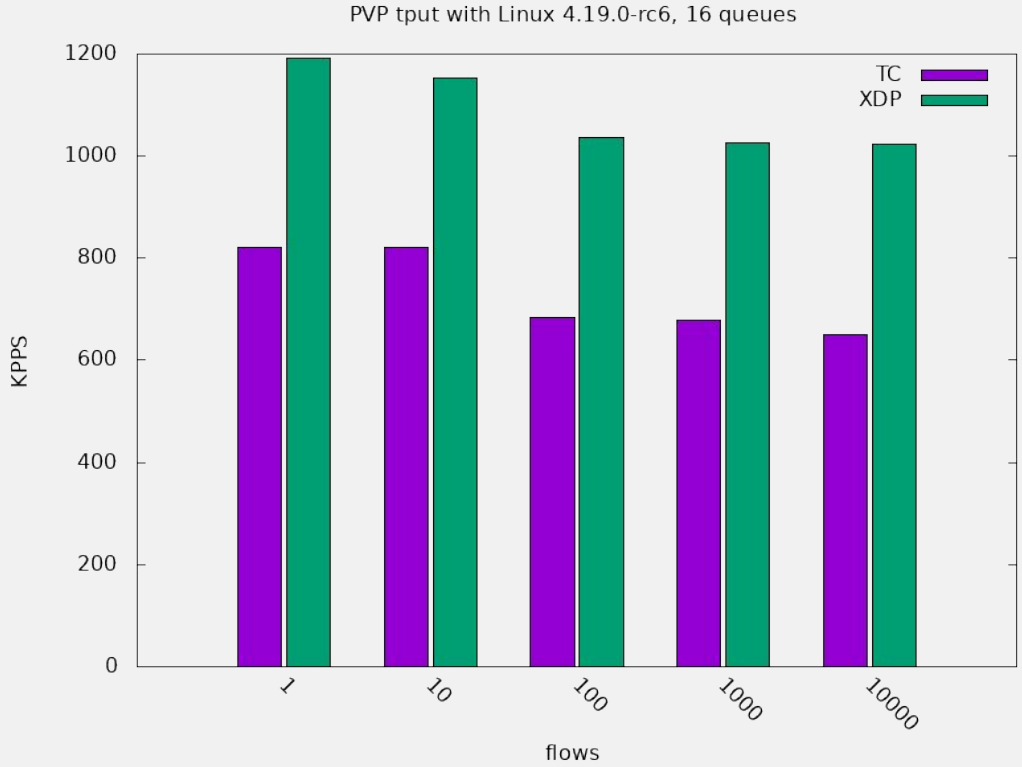
Things intentionally omitted - so far

- With more complex ruleset TC will hit a greater retpoline overhead
  - “listification” is hard to apply here, will not help with many flows
- Some specific TC actions do not scale well (per action spin_lock)
  - removal is a WIP - thanks to Davide Caratti
- We could almost double the tput using 2 vhost threads per virtio net queue (rx and tx)
  - That is almost alike using multiple virtio_net queues
- Still far away from line rate and carrier grade reqs (15x), less far from bypass solutions (3x)

Will eBPF save us?

- OVS support for XDP is under development. Is that a game changer? Let's perf it
  - Use a simple XDP program parsing ingress packet up to L3 and forwarding it using an user configured map
  - Nowhere near a complete solution, hopefully an upper bound of what we should expect with ovs-XDP

Will eBPF save us? [II]



A glance at the future

- XDP eBPF backend for OVS is not there yet
  - And next-to-come AF_XDP is possibly more interesting from performance PoV
- UDP GRO for forwarded packet can someday land into the kernel datapath.
  - Will help only with scenarios using a limited number of flows.

redhat.

THANK YOU