Who fears the spectres?

Performance impacts of spectre/meltdown counter-measures and possible improvements

Paolo Abeni - Red Hat

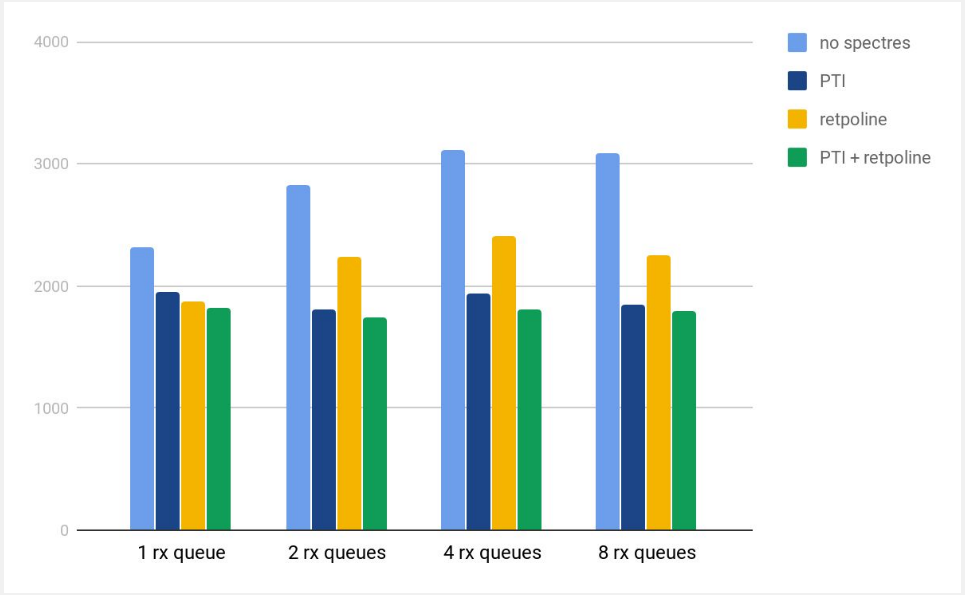
NetConf, Boston 2018

Outline

- Performance impact of PTI and retpoline on current net-next
- Possible improvements, PTI and retpoline-related
- Possible improvements, misc

UDP RX performances

64 bytes ipv4 packets, single sink, different RX queues



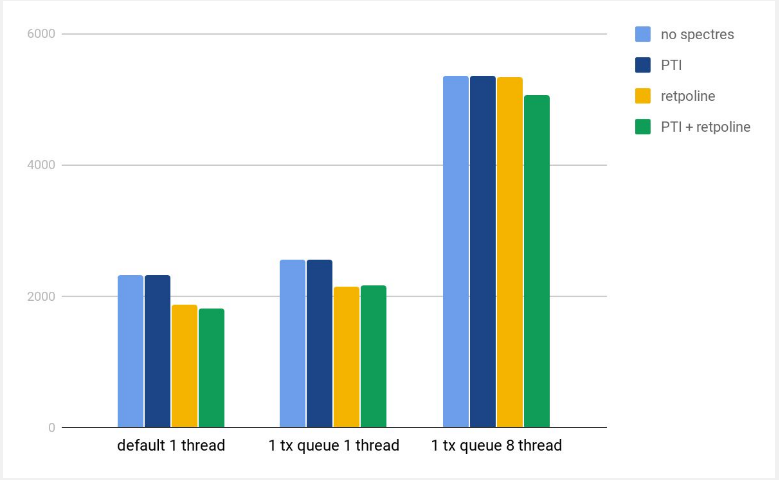
Digging with perf

Topmost perf offenders for UDP RX test - receiver process, compared

NO mitigations	Retpoline only	PTI only
11.41% copy_user_generic_unrolled	10.95% udp_recvmsg (delta 2.73%)	13.49% syscall_return_via_sysret
9.12% udp_recvmsg	7.72% copy_user_generic_unrolled	10.49% 0xfffffe000016601b
5.25% __slab_free	6.40% avc_has_perm (delta 2.36%)	7.11% copy_user_generic_unrolled
5.20% page_frag_free	5.51% page_frag_free	6.40% udp_recvmsg
4.55% __sys_recvfrom	5.12% __sys_recvfrom	4.15% page_frag_free
4.38% entry_SYSCALL_64_after_hwframe	4.35% __slab_free	3.72% __sys_recvfrom
4.08% do_syscall_64	4.07% __skb_recv_udp (delta ~1.54%)	3.61% do_syscall_64
4.04% avc_has_perm	3.93% entry_SYSCALL_64_after_hwframe	3.46% __slab_free
3.75% _copy_to_iter	3.40% do_syscall_64	3.18% entry_SYSCALL_64_after_hwframe

QDisc performances

pktgen tput with queue_xmit mode, 64 bytes packets



Perf, again

Topmost perf offenders for Qdisc test, compared

NO mitigations

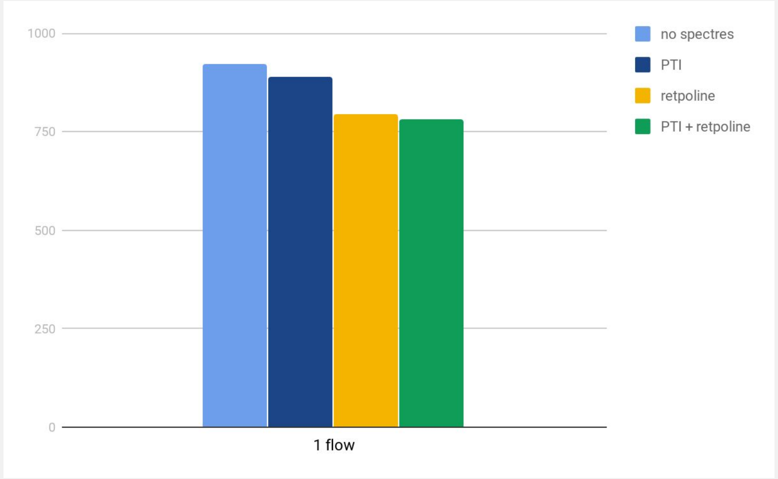
11.86%	pktgen_xmit
9.27%	ixgbe_xmit_frame_ring
8.71%	skb_unref.part.39
6.76%	pfifo_fast_dequeue
5.81%	ip_send_check
3.82%	__dev_queue_xmit
3.58%	mod_cur_headers
3.29%	__qdisc_run
3.15%	skb_put

Retpoline only

11.41%	ixgbe_xmit_frame_ring (delta 2.14%)
10.42%	pktgen_xmit
10.34%	pfifo_fast_dequeue (delta 3.62%)
4.98%	ip_send_check
4.74%	skb_unref.part.39
3.50%	__qdisc_run
3.33%	__dev_queue_xmit
2.97%	mod_cur_headers
2.60%	__build_skb

PVP performances

OVS kernel datapath, default flow configuration



One last perf comparison

Topmost perf offenders for PVP test - vhost process - compared

NO mitigations	Retpoline only	PTI only
5.78% vhost_get_vq_desc	5.29% tun_get_user	5.64% vhost_get_vq_desc
5.47% tun_get_user	5.06% vhost_get_vq_desc	5.18% tun_get_user
5.37% masked_flow_lookup	4.77% masked_flow_lookup	5.08% masked_flow_lookup
5.05% copy_user_generic_unrolled	4.65% ixgbe_xmit_frame_ring (delta 1.11%)	4.63% copy_user_generic_unrolled
4.62% translate_desc	4.20% pfifo_fast_dequeue (delta 1.47%)	4.57% translate_desc
4.01% iov_iter_advance	4.20% copy_user_generic_unrolled	3.92% iov_iter_advance
3.54% ixgbe_xmit_frame_ring	4.04% translate_desc	3.41% ixgbe_xmit_frame_ring
2.73% pfifo_fast_dequeue	3.76% iov_iter_advance	3.24% pfifo_fast_dequeue

Fighting spectres

- Bulking:
  - Potentially reduces the impact of both retpolines and PTI
    - But really affecting retpolines is usually less straight-forward - e.g. bulk_dequeue
  - Already there in several places (GSO, GRO, qdisc dequeue)
    - but routing and forwarding have no support
  - UDP is in a mixed state: GSO (and eventually GRO) for connected sockets, recvmmsg/sendmmsg for unconnected (?!?)
  - other options?

Still fighting spectres

might as well [not indirect] jump - indirect calls we can avoids

skb->destruct()

- Proposed by Hannes Sowa, originally to reduce skbuff size
- Use integer to demux the destruction action between the known ones
    - Some driver - chelsio - may still need indirect call
    - The expected gain is currently unknown

Indirect calls we want to avoid [II]

sch->enqueue and sch->dequeue

- We can check for build-in qdiscs and call the related ops directly,
  - We can avoid 2 indirect calls per packet
  - Still need them in some (most ?!?) cases
- With jump labels we can avoid all the indirect calls with the default configuration
  - And fall back to the above after any changes

More indirect calls we want to avoid

GRO and offloads

- A Lot of indirect calls per packet there
- At least for GRO removing all of them looks possible
  - But some code uglification looks unavoidable

Other targets?

Side-track: too many [virtual-]switches

- 2 in kernel datapaths for OVS (net/openvswitch and TC/flower)
- neither is near to the requested performances (for SDN)
  - But even bypass solutions do not meet pkt rate requirements
- TC/flower is needed for H/W offload
  - But it still misses some features
- Do we need both of them? Can we move towards TC/flower only?
- Crazy idea: can we attach TC ingress to the XDP hook?

And now for something completely different

- Leverage UMH to implement COMPAT_ code for xfrm, and remove compat kernel support from xtables (idea from Florian Westphal)
- Remote skb free (idea from Eric)
    - Any more details here?
- edmux for unconnected sockets (is that a dead cow?)

the hardships of an orphan[ed skb]

And now for something completely different - part II

- SKBs are orphaned the xmit path, when potentially crossing net-ns
- In presence of XPS this hurts TCP performance badly (due to OoO and lack of feedback towards the sender socket)
- Naive partial solution: disable XPS for orphaned sockets
  - Hurts UDP performances, don’t solve lack of feedback
- Alternative solution: access skb->sk via an helper in netfilter, do not really clear skb->sk while scrubbing the skb, just mark is as not accessible (via the helper)

redhat.

THANK YOU