XDP - eXpress Data Path

XDP now with REDIRECT

Jesper Dangaard Brouer,

Principal Engineer, Red Hat Inc.

LLC - Lund Linux Conf

Sweden, Lund, May 2018

Intro: What is XDP?

Really, don't everybody know what XDP is by now?!

Basically: New layer in the kernel network stack

- Before allocating the SKB
  - Driver level hook at DMA level
- Means: Competing at the same “layer” as DPDK / netmap
- Super fast, due to
  - Take action/decision earlier (e.g. skip some network layers)
  - No-memory allocations
- Not kernel bypass, data-plane is kept inside kernel
  - via BPF: makes early network stack run-time programmable
  - Cooperates with kernel

Intro: XDP: data-plane and control-plane

Overall design

Data-plane: inside kernel, split into:

- Kernel-core: Fabric in charge of moving packets quickly
- In-kernel BPF program:
  - Policy logic decide action
  - Read/write access to packet

Control-plane: Userspace

- Userspace load BPF program
- Can control program via changing BPF maps
- Everything goes through bpf system call

Intro: XDP actions and cooperation

What are the basic building blocks I can use?

BPF program return an action or verdict

- XDP_DROP, XDP_PASS, XDP_TX, XDP_ABORTED, XDP_REDIRECT

How to cooperate with network stack

- Pop/push or modify headers: Change RX-handler kernel use
  - e.g. handle protocol unknown to running kernel
- Can propagate 32Bytes meta-data from XDP stage to network stack
  - TC (clsbpf) hook can use meta-data, e.g. set SKB mark

Intro: Why kernel developers should love BPF

How BPF avoids creating a new kernel ABI for every new user-invented policy decision?

BPF is sandboxed code running inside kernel (XDP only loaded by root)

- A given kernel BPF hook just define:
  - possible actions and limit helpers (that can lookup or change kernel state)

Users get programmable policies (within these limits)

- Userspace "control-plane" API tied to userspace app (not kernel API)
  - likely via modifying a BPF-map
- No longer need a kernel ABI
  - like sysctl/procfs/ioctls etc.

Next slides

Why XDP_REDIRECT is so interesting?!

New XDP action XDP_REDIRECT

First lets cover the basics...

XDP got new action code XDP_REDIRECT (that drivers must implement)

- In basic form: Redirecting RAW frames out another net_device/ifindex
- Egress driver: implement `ndo_xdp_xmit` (and `ndo_xdp_flush`)

Performance low without using a map for redirect (single CPU core numbers):

- Using helper: bpf_redirect = 7.5 Mpps
- Using helper: bpf_redirect_map = 13.0 Mpps

What is going on?

- Using redirect maps is a HUGE performance boost, why!?

Novel: redirect using BPF maps

Why is it so brilliant to use BPF maps for redirecting?

Basic design: Simplify changes needed in drivers

- "Redirect" is more generic, than "forwarding"

First trick: Hide RX bulking from driver code

- Driver still processes packets one at a time - calling `xdp_do_redirect`
- End of driver NAPI poll routine "flush" (max 64 packets) - call `xdp_do_flush_map`
- Thus, bulking via e.g. delaying expensive NIC tailptr/doorbell

Second trick: invent new types of redirects easy

- Without changing any driver code!
- Hopefully last XDP action code(?)

Redirect map types

What kind of redirects are people inventing?!

The “devmap”: BPF_MAP_TYPE_DEVMAP

- Contains net_devices, userspace adds them via ifindex to map-index

The “cpumap”: BPF_MAP_TYPE_CPUMAP

- Allow redirecting RAW xdp frames to remote CPU
  - SKB is created on remote CPU, and normal network stack invoked
- The map-index is the CPU number (the value is queue size)

Upcoming AF_XDP - “xskmap”: BPF_MAP_TYPE_XSKMAP

- Allow redirecting RAW xdp frames into userspace
  - via new Address Family socket type: AF_XDP
  - (More in Björn Töpel’s talk later....)

Next slides

What is this CPUMAP redirect?

XDP_REDIRECT + cpumap

What is cpumap redirect?

Basic cpumap properties

- Enables redirection of XDP frames to remote CPUs
- Moved SKB allocation outside driver (could help simplify drivers)

Scalability and isolation mechanism

- Allows isolating/decouple driver XDP layer from network stack
  - Don't delay XDP by deep call into network stack
- Enables DDoS protection on end-hosts (that run services)
  - XDP fast-enough to avoid packet drops happen in HW NICs

Another use-case: Fix NIC-HW RSS/RX-hash broken/uneven CPU distribution

- Proto unknown to HW: e.g. VXLAN and double-tagged VLANs

Cpumap redirect: CPU scaling

Tricky part getting cross CPU delivery fast-enough

Cpumap architecture: Every slot in array-map: dest-CPU

- MPSC (Multi Producer Single Consumer) model: per dest-CPU
  - Multiple RX-queue CPUs can enqueue to single dest-CPU
- Fast per CPU enqueue store (for now) 8 packets
  - Amortized enqueue cost to shared ptr_ring queue via bulk-enq
- Lockless dequeue, via pinning kthread CPU and disallow ptr_ring resize

Important properties from main shared queue ptr_ring (cyclic array based)

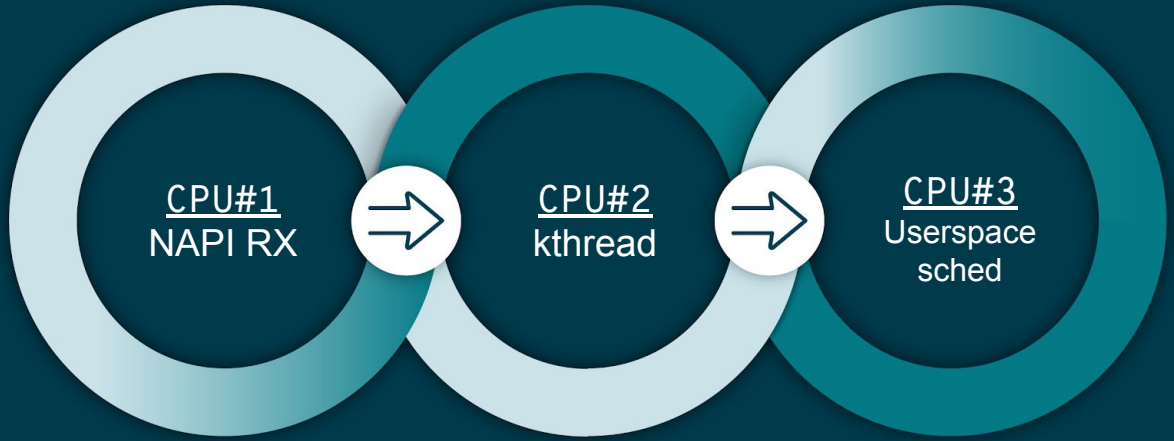
- Enqueue+dequeue don't share cache-line for synchronization
  - Synchronization happens based on elements
  - In queue almost full case, avoid cache-line bouncing
  - In queue almost empty case, reduce cache-line bouncing via bulk-enq

CPU scheduling via cpumap

Queuing and scheduling in cpumap

Hint: Same CPU sched possible

- But adjust /proc/sys/kernel/sched_wakeup_granularity_ns



Next slides

Recent changes to XDP core

Recent change: Information per RX-queue

Recent change in: kernel v4.16

Long standing request: separate BPF programs per RX queue

- This is not likely to happen… because

Solution instead: provide info per RX queue (`xdp_rxq_info`)

- Info: ingress net_device (Exposed as: `ctx->ingress_ifindex`)
- Info: ingress RX-queue number (Exposed as: `ctx->rx_queue_index`)

Thus, NIC level XDP/bpf program can instead filter on rx_queue_index

Recent change: queuing via xdp_frame

Very recent changes: only accepted in net-next (to appear in v4.18)

XDP_REDIRECT needs to queue XDP frames e.g. for bulking

- Queuing open-coded for both cpumap and tun-driver
- Generalize/standardize into struct xdp_frame
- Store info in top of XDP frame headroom (reserved)
  - Avoids allocating memory

Recent change: Memory return API

Very recent changes: only accepted in net-next (to appear in v4.18)

API for how redirected frames are freed or "returned"

- XDP frames are returned to originating RX driver
- Furthermore: this happens per RX-queue level (extended `xdp_rxq_info`)

This allows driver to implement different memory models per RX-queue

- E.g. needed for AF_XDP zero-copy mode

Next slides

Spectre V2 killed XDP performance

Performance issue: Spectre (variant 2)

CONFIG_RETPOLINE and newer GCC compiler
- for stopping Spectre (variant 2) CPU side-channel attacks

Hey, you killed my XDP performance! (Retpoline tricks for indirect calls)

- Still processing 6 Mpps per CPU core
- But could do approx 13 Mpps before!

Initial through it was `net_device->ndo_xdp_xmit` call

- Implemented redirect bulking, but only helped a little

Real pitfall: DMA API use indirect function call pointers

- Christoph Hellwig PoC patch show perf return to approx 10 Mpps

Thus, solutions in the pipeline...

End slide

... Questions?


plus.google.com/+JesperDangaardBrouer


facebook.com/brouer


linkedin.com/in/brouer


twitter.com/JesperBrouer


youtube.com/channel/UCSyplUCgtI42z63soRMONng

Thanks to all contributors

XDP + BPF combined effort of many people

- Alexei Starovoitov
- Daniel Borkmann
- Brenden Blanco
- Tom Herbert
- John Fastabend
- Martin KaFai Lau
- Jakub Kicinski
- Jason Wang
- Andy Gospodarek
- Thomas Graf
- Michael Chan (bnxt_en)
- Saeed Mahameed (mlx5)
- Tariq Toukan (mlx4)
- Björn Töpel (i40e + AF_XDP)
- Magnus Karlsson (AF_XDP)
- Yuval Mintz (qede)
- Sunil Goutham (thunderx)
- Jason Wang (VM)
- Michael S. Tsirkin (ptr_ring)
- Edward Cree