Queue API

Goals and direction

1. Reset / reload single queue for page-pool changes
2. Reset / reload for AF_XDP ?
3. Use the queue-by-queue reset model for all config changes
4. Store configuration in the core, to allow per-queue config
5. On-demand creation and deletion of queues

Challenges:

- no support for graceful shutdown in general (FW limitations etc.)
- people are too precious about their drivers :)

Object lifetime model

Already have:

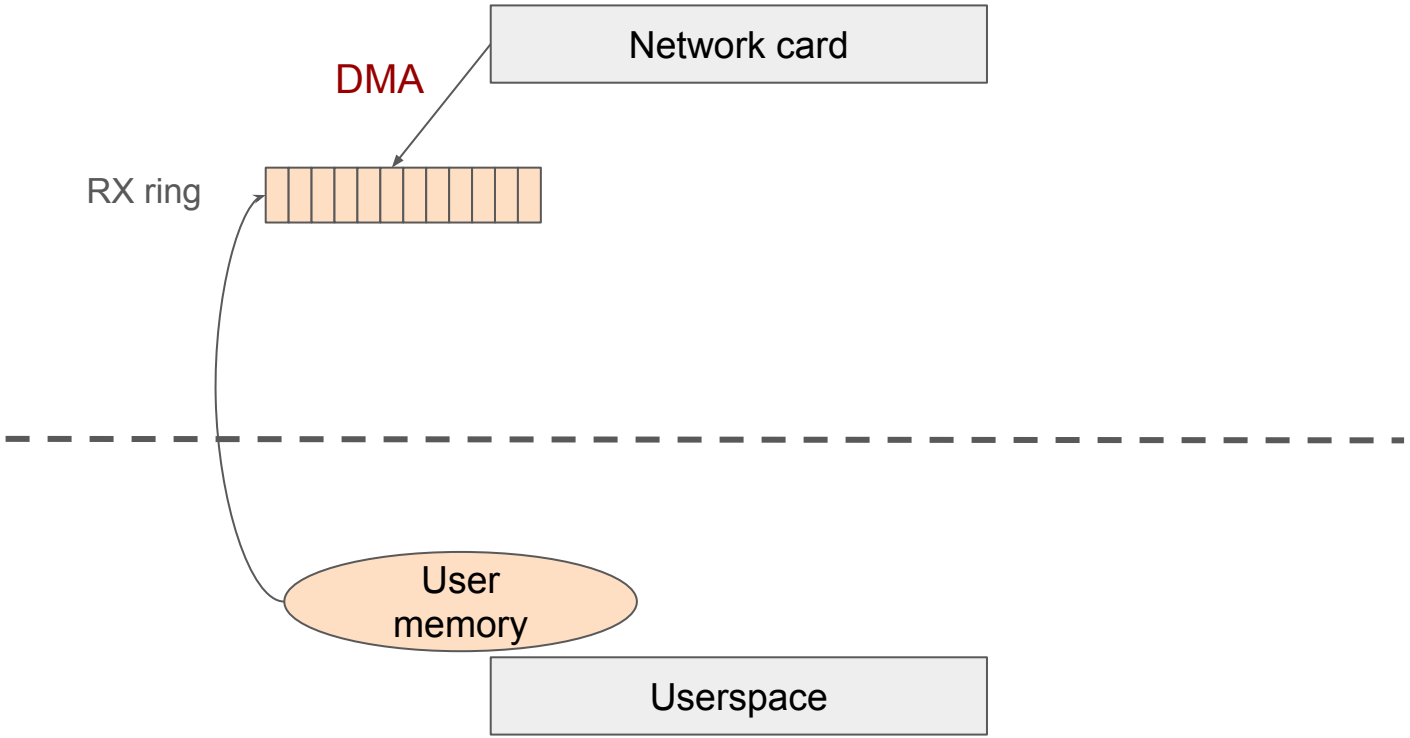
- objects can be attached to a netlink socket
- when socket gets closed objects are auto-destroyed

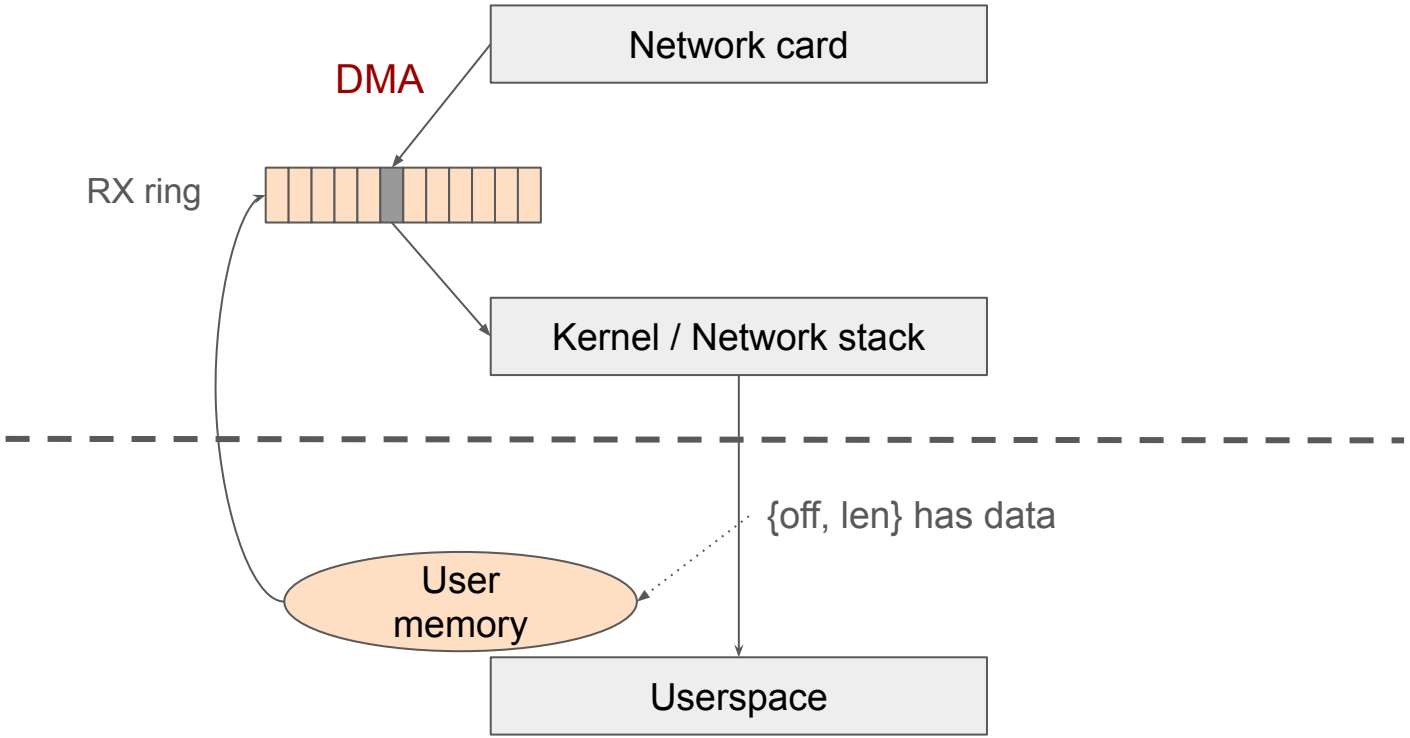
Potential improvements:

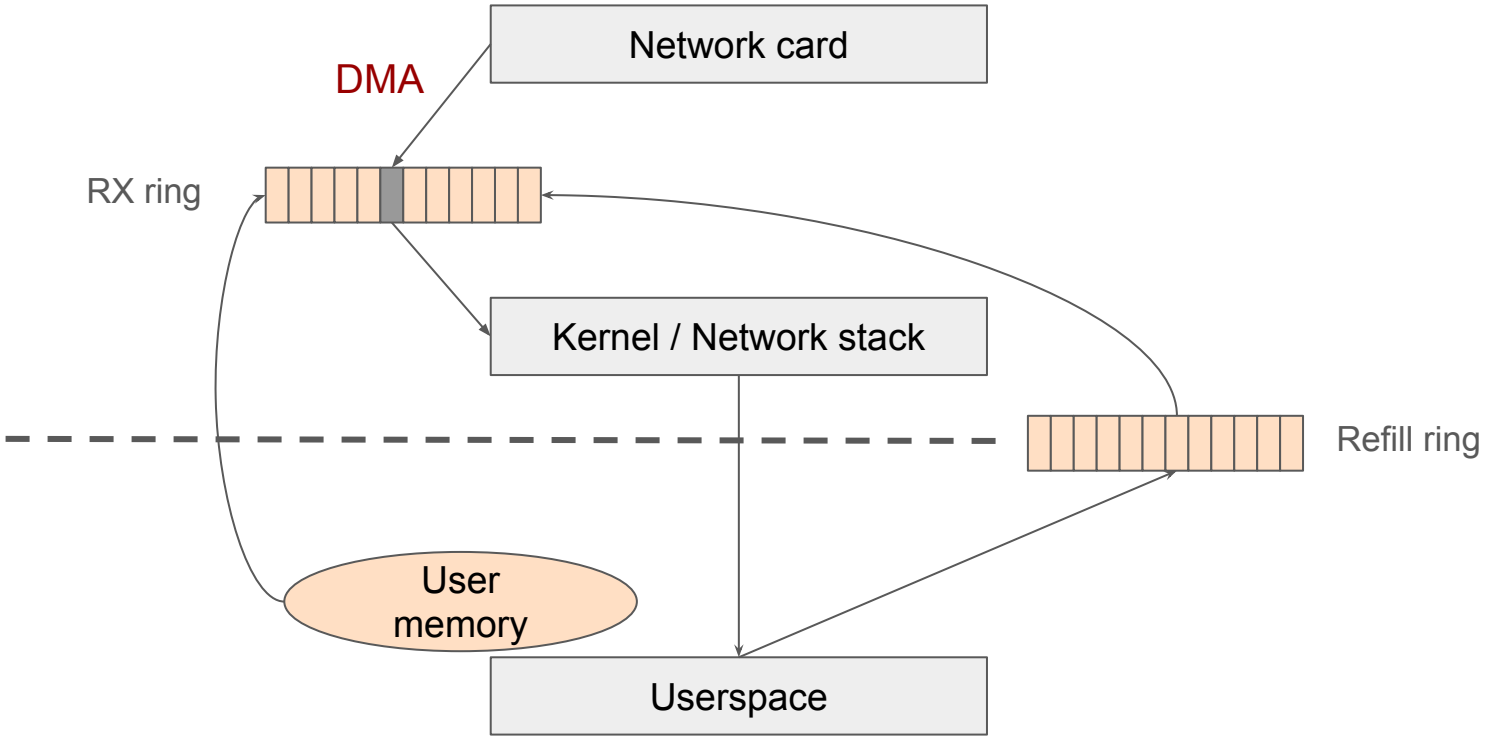
- better API for generic object store rather than per-family priv store
- tracking dependencies (n-tuple rule -> RSS context -> queues)
  - for ordered removal and error checking in the core
- introspection of who is holding which object
- object passing (w/ priv delegation)?

Zero Copy Rx with io_uring

David Wei, Pavel Begunkov


Network card
DMA
RX ring
User
memory
Userspace


Network card
DMA
RX ring
Kernel / Network stack
{off, len} has data
User memory
Userspace


Network card
DMA
RX ring
Kernel / Network stack
Refill ring
User
memory
Userspace

- Page pool memory providers
  - Page pool API, seamless for drivers
  - io_uring callback pulls buffers from the refill ring …
  - … or from a slow path stash
- Pages / buffers are wrapped into `struct net_iovs`
  - Turns into `netmem_ref` for the net stack / drivers
  - Refcounts buffers / helps controlling lifetime
  - Doesn’t need backing pages
- Copy fallback
  - skb linear part, mixing with kernel pages, etc.

- Simplifying user API
- Completions returned in main Completion Queue
  - No double layered completions
  - No separate ring to inspect
  - Requires extended 32 byte CQEs, **`IORING_SETUP_CQE32`**
- No extra steps for intermediate buffer registration
  - Done in a single syscall while registering an ifq object.
- Reworked completion and refill entry layouts
  - No duplication, e.g. socket and queue id
  - More flexible token scheme, will help with future extensions

Zero copy performance

- Thrift RPC benchmark
- Echo request / response, 64 KB payload
- Single server worker thread
  - Pinned to same CPU as net rx softirq
- CPU bound, 24 client connections over 6 client threads
- Broadcom Thor NIC, 100/4 = 25 Gbps server bandwidth

epoll	zerocopy receive
20.6 Kqps	22.8 Kqps
10.08 Gbps	11.14 Gbps

~10% difference, ~14% of copy overhead

High user space overhead

bnxt queue API implementation

- Patchset merged upstream
  - …but buggy in testing
- A delicate dance between driver + FW
  - Must be careful with FW synced data structures
- Need to be able to quiesce a queue
  - FW vnic_update()
- Requires HW ntuple filtering + flow steering

Multiple threads per queue

- Broadcom Thor has 128 queues
- 1:1:1 HW queue:io_uring:thread association
- So CPUs with > 128 cores = 😢
- Need to have multiple threads share *something*

Zero copy and TLS

- kTLS negates the benefit of zero copy
- Plaintext + ZC:
  - NIC → User
  - 1 DMA trip over memory bus
- TLS + ZC:
  - NIC → User → Decrypt + Copy → User
  - Now 2 trips
- kTLS:
  - NIC → Kernel → Decrypt + Copy → User
  - Also 2 trips over memory bus, 1 DMA + 1 copy
- PSP?
  - NIC → HW PSP Offload → User
  - Back to 1 trip

Userspace RPC alignment

- Use case: block storage service, 4 KB page writes
- Want O_DIRECT to work
- Everything sent as RPC
- Opaque 4 KB data somewhere inside RPC frame
- RPC frame segmented into MTU sized packets and sent
- Server zero copy puts each packet into a separate page
- For O_DIRECT each iovec must be properly aligned
- Hard to guarantee this with HW HDS

netdevsim as test device

- Added packet forwarding
- Added NAPI and multi-queue
- Want to be a test device
  - For io_uring zero copy / devmem TCP
  - Increased selftest coverage of netdev core features
  - What else…?
- TODO:
  - Flow steering?
  - RSS?
  - Page pool?

Latest branch, waiting for net_iov
https://github.com/isilence/linux.git zcrx/v5-conf

Outdated RFC
https://lore.kernel.org/io-uring/20240312214430.2923019-1-dw@davidwei.uk/

Benchmarking
https://github.com/spikeh/netbench/tree/zcrx/next

io_uring mailing
io-uring at vger.kernel.org

Or email us directly for any questions:
Pavel Begunkov <asml.silence@gmail.com>
David Wei <dw@davidwei.uk>