Contribution ID: **148** Type: **not specified**

The Path to DPDK Speeds for AF_XDP

Tuesday, November 13, 2018 3:10 PM (35 minutes)

AF_XDP is a new socket type for raw frames to be introduced in 4.18 (in linux-next at the time of writing). The current code base offers throughput numbers north of 20 Mpps per application core for 64-byte packets on our system, however there are a lot of optimizations that could be performed in order to increase this even further. The focus of this paper is the performance optimizations we need to make in AF_XDP to get it to perform as fast as DPDK.

We present optimization that fall into two broad categories: ones that are seamless to the application and ones that requires additions to the uapi. In the first category we examine the following:

- Loosen the requirement for having an XDP program. If the user does not need an XDP program and there is only one AF_XDP socket bound to a particular queue, we do not need an XDP program. This should cut out quite a number of cycles from the RX path.
- Wire up busy poll from user space. If the application writer is using epoll() and friends, this has the potential benefit of removing the coherency communication between the RX (NAPI) core and the application core as everything is now done on a single core. Should improve performance for a number of use cases. Maybe it is worth revisiting the old idea of threaded NAPI in this context too.
- Optimize for high instruction cache usage through batching as has been explored in for example Cisco's VPP stack and Edward Cree in his net-next RFC "Handle multiple received packets at each stage".

In the uapi extensions category we examine the following optimizations:

- Support a new mode for NICs with in-order TX completions. In this mode, the completion queue would not be used. Instead the application would simply look at the pointer in the TX queue to see if a packet has been completed. In this mode, we do not need any backpreassure between the completion queue and the TX queue and we do not need to populate or publish anything in the completion queue as it is not used. Should improve the performance of TX for in-order NICs significantly.
- Introduce the "type-writer" model where each chunk can contain multiple packets. This is the model that e.g., Chelsio has in its NICs. But experiments show that this mode also can provide better performance for regular NICs as there are fewer transactions on the queues. Requires a new flag to be introduced in the options field of the descriptor.

With these optimization, we believe we can reach our goal of close to 40 Mpps of throughput for 64-byte packets in zero-copy mode. Full analysis with performance numbers will be presented in the final paper.

Presenters: TÖPEL, Björn (Intel); KARLSSON, Magnus (Intel)

Session Classification: Networking Track