Accelerating Packet Processing with FPGA NICs

DPDK Summit - San Jose – 2017
Boris Pismenny

#DPDKSummit
NIC packet processing offloads has been proven to significantly assist packet processing, e.g.,
- TCP/UDP checksum
- TCP segmentation offloads
- RSS

The recently introduced `rte_security` APIs allowed NICs to accelerate crypto operations *inline*
- Received packets are decrypted by the NIC before being scattered to memory
- Sent packets are encrypted by the NIC before being sent to the wire
- No need to enqueue the packets to another `cryptodev` PMD
The benefits of inline acceleration can be generalized to support *any* application-specific action by FPGA-capable NICs!

- A single NIC may support multiple Inline Acceleration Functional Units (I-AFUs) provided by multiple parties
- The I-AFU can be programmed in the field to do any packet processing task
- Any packet flow can be redirected to any I-AFU

We have a good toolbox for handling flows which is constantly evolving

- Count, Mark, Steer, modify...

Generic acceleration flow actions are a natural fit

- Steer any flow to any I-AFU
- Continue packet processing according to steering
- Application-specific byte-intensive packet transformation
- Application-specific flow-steering
  - Accelerator parses packet and modifies header fields accordingly
  - Flow processing resumes normally afterwards
Discovery
- What I-AFUs are currently installed on the NIC?

Control
- Discovering the capabilities of an I-AFU
- Configuring an I-AFU

Flow processing
- Packet flows are matched normally
- Opaque action specifies the I-AFU that should handle matching packets

Data path
- Report/deliver I-AFU specific information via opaque mbuf meta-data
Reports the following information
- Vendor ID – This is the ID of the accelerator provider
- Product ID – Uniquely identifies a product of the provider
- Version – Product version

Given this information, applications uniquely identify the I-AFU
- Semantics are known to the application a-priori
Control Path

- Opaque command
  ```c
  struct rte_accel_session
  *rte_accel_session(uint16_t id,
    struct rte_accel_sess_conf *conf,
    struct rte_mempool *mp,
  );
  ```

- Create/Destroy/Configure Session

```c
struct rte_accel_session_conf {
    unsigned short vendor_id;
    /**< AFU vendor ID */

    unsigned short product_id;
    /**< AFU product ID */

    unsigned int cmd_id;
    /**< AFU command ID */

    unsigned int length;
    /**< AFU command buffer length */

    unsigned char buf[0];
    /**< AFU command buffer */
};
```
New non-terminating action “call accelerator”

For example: Customer AFU replaces FOO with BAR in payload of matching packets

```c
/**
 * security session configuration parameters */
 struct rte_accel_session_conf accel_cmd = {
   .vendor_id = 0x1234,
   /**< Customer AFU vendor ID */
   .product_id = 0x5678,
   /**< Customer product ID*/
   .cmd_id = 1,
   .length = 8;
   buf = “FOO|BAR”
   /**< String to replace */
};

/** flow parameters */
attr->ingress = 1; /**< attr->egress = 1 */

pattern[0].type = RTE_FLOW_ITEM_TYPE_ETH;
pattern[1].type = RTE_FLOW_ITEM_TYPE_IPV4;
pattern[2].type = RTE_FLOW_ITEM_TYPE_UDP;
pattern[3].type = RTE_FLOW_ITEM_TYPE_END;

action[0].type = RTE_FLOW_ACTION_TYPE_ACCEL;
action[0].conf = accel_session;
action[1].type = RTE_FLOW_ACTION_TYPE_END;
```
Related Work

- **rte_prgdev** – focused on burning/loading images into programmable devices
  - Complementary to this proposal

- **rte_raw_dev** – abstracted the PMD device functionality for accelerators
  - Seems like a good direction for FPGAs that act as CPU-assists
  - Complements inline packet acceleration
Questions?