DPDK
SUMMIT
APAC • 2021

Warp Speed Crypto

A NEW DPDK CRYPTODEV RAW DATA PATH API AND ITS USE IN FD.IO VPP

Agenda

DPDK SUMMIT APAC•2021

- Problem Statement
- New cryptodev symmetric raw datapath API
  - New Crypto data structure
  - Raw data path session context
  - Enqueue
  - Dequeue
- Cryptodev symmetric raw datapath API adoption to FD.io VPP
- Summary

Problem Statement

- DPDK Cryptodev is great, but there are limitations.
  - Based on the assumption that all data is described by *rte_mbuf*
  - Extra data structure (*rte_crypto_op*) to describe crypto workload
  - Pre-allocated, non-expandable mempools.
  - Best effort based enqueue and dequeue operations.
- All limitations are there for a sole purpose: extreme performance.
- Based on the assumption that your implementation is written on top of DPDK.
- What if you want to use DPDK Cryptodev API in your crypto driver/engine?

A Sample: DPDK Virtio-crypto

DPDK SUMMIT APAC•2021

- Virtio-crypto has its own data structure (e.g. `virtio_crypto_cipher_para`), and the data is described by virtio ring descriptor.
- To process virtio-crypto workload with DPDK Cryptodev, one must:
  - Get DPDK rte_crypto_op buffers from crypto op mempool.
  - Get DPDK `rte_mbuf` buffers from mbuf mempool.
  - Translate virtio ring descriptor into `rte_mbuf`.
  - Translate `virtio_crypto_cipher_para` into DPDK `rte_crypto_op`.
  - Enqueue to DPDK Cryptodev device queue and then dequeue.
  - Translate DPDK crypto operation status to virtio-crypto operation status.
  - Put DPDK `rte_mbuf` and `rte_crypto_op` back to mempools.

Next Example: VPP Crypto Async Infra

DPDK SUMMIT APAC•2021

- Same as virtio-crypto, VPP crypto async infra also has its own data structures to describe data and crypto operations.
- In addition, VPP wraps up to 64 crypto operations into a single crypto frame and requires complete enqueue and dequeue the whole frame or nothing.
- To overcome this, we had to
  - Keep a run-time counter to see if there is room to enqueue the frame.
  - Cache half dequeued frames and use complex logic to track the dequeue status for every cached frame.
- Even so the translation is done again by the cryptodev device driver.
- Furthermore, the “double looping” and “double translation” hurts the performance even more.

Problem Statement (Cont.)

DPDK SUMMIT APAC•2021

- To sum up, to enable DPDK cryptodev on top of an application with existing crypto infra, the following aspect may degrade performance:
  - Extra mempool get and put operations for rte_crypto_op and rte_mbuf.
  - Extra translation cost on converting implementation data structures to DPDK specific ones.
  - Extra loop and another translation for cryptodev driver.
  - Limited control over enqueue and dequeue burst operations forced enqueue check and dequeue cache.
- To make DPDK Cryptodev more friendly to external library/application and to widen Cryptodev usage, we propose DPDK Cryptodev symmetric Raw Data Path APIs.

Crypto Data Struct Redesign

DPDK SUMMIT APAC•2021

- Goal: Support any data, not mbuf centric.
  - The only data Cryptodev interested in mbuf:
    - Data virtual and physical addresses
    - Data lengths
    - SGL by mbuf chain.
  - Replace by the rte_crypto_sgl structure:
    - Reducing memory reads.
    - More compact crypto operation data structure.
    - Free from dependency to rte_mbuf.

```
struct rte_crypto_sym_op {
        struct rte_mbuf *m_src;
        struct rte_mbuf *m_dst;
        …
```

↓

```
struct rte_crypto_vec {
        void *base;
        rte_iova_t iova;
        uint32_t len;
};

struct rte_crypto_va_iova_ptr {
        void *va;
        rte_iova_t iova;
};

struct rte_crypto_sgl {
        struct rte_crypto_vec
*vec;
        uint32_t num;
};
```

New Raw Crypto Operation Data Structure

- Refactored `rte_crypto_sym_vec` structure to replace `rte_crypto_sym_op`.
  - For burst enqueue of packets with same session.
  - Contains only the required data field for all symmetric crypto operations.
  - Allocate from stack, not heap.
  - Simple status in return for enqueue: either success (0) or error (-errno).

```
struct rte_crypto_sym_vec {
        uint32_t num;
        struct rte_crypto_sgl *sgl;
        struct rte_crypto_va_iova_ptr *iv;
        struct rte_crypto_va_iova_ptr *digest;

        __extension__
        union {
                struct rte_crypto_va_iova_ptr
*auth_iv;
                struct rte_crypto_va_iova_ptr
*aad;
        };
        int32_t *status;
};
```

All-in-one Device/Queue/Session Abstraction

- Original crypto cevice ID, queue pair ID, and session are abstracted into one `rte_crypto_raw_dp_ctx` structure, as:
  - Every thread owns a cryptodev queue pair and tends to not change frequently. Queue pair pointer can be directly mapped.
  - In most cases, multiple consecutive packets will share the same session, specific enqueue/dequeue handler can be mapped for the session to avoid huge enqueue/dequeue functions with minimum branches in them.
  - In case of session change, only the enqueue and dequeue handlers need to be updated.

```
struct rte_crypto_raw_dp_ctx {
        void *qp_data;

        cryptodev_sym_raw_enqueue_t enqueue;
        cryptodev_sym_raw_enqueue_burst_t
enqueue_burst;
        cryptodev_sym_raw_operation_done_t
enqueue_done;
        cryptodev_sym_raw_dequeue_t dequeue;
        cryptodev_sym_raw_dequeue_burst_t
dequeue_burst;
        cryptodev_sym_raw_operation_done_t
dequeue_done;

        /* Driver specific context data */
        __extension__ uint8_t drv_ctx_data[];
};
```

Data Enqueue

DPDK SUMMIT APAC•2021

- Two types of enqueue:
  - Enqueue burst: enqueue a burst of data with same session and safe cipher/auth offsets.
  - Enqueue single: only enqueue a single data, expect the data is cached and the crypto device is not “kicked” after enqueued.
- Enqueue single helps avoid “double loop” and saves one layer of translation.
- After expected pieces of data are enqueued, use `rte_cryptodev_raw_enqueue_done()` to “kick” the device queue to start processing cached data.
- To abandon cached data, simply reset the ctx.

```
uint32_t
rte_cryptodev_raw_enqueue_burst(
        struct rte_crypto_raw_dp_ctx *ctx,
        struct rte_crypto_sym_vec *vec,
        union rte_crypto_sym_ofs ofs,
        void **user_data, int *enqueue_status)

static __rte_always_inline int
rte_cryptodev_raw_enqueue(
        struct rte_crypto_raw_dp_ctx *ctx,
        struct rte_crypto_vec *data_vec,
        uint16_t n_data_vecs,
        union rte_crypto_sym_ofs ofs,
        struct rte_crypto_va_iova_ptr *iv,
        struct rte_crypto_va_iova_ptr *digest,
        struct rte_crypto_va_iova_ptr
*aad_or_auth_iv,
        void *user_data)

int
rte_cryptodev_raw_enqueue_done(
        struct rte_crypto_raw_dp_ctx *ctx, uint32_t
n);
```

Data Dequeue

- User can use callback function (preferably inline) to control expected dequeue count, either by parsing the first dequeued user data, or return a fixed number.
- User can use callback function to parse and set the dequeued status inflight to save status translation.
- Instead of loop all dequeued user data to find if there are errors, *n_success shows dequeue results immediately.
- Again, to abandon dequeue, simply reset ctx.

```
uint32_t
rte_cryptodev_raw_dequeue_burst(
        struct rte_crypto_raw_dp_ctx *ctx,
        rte_cryptodev_raw_get_dequeue_count_t

        get_dequeue_count,
        rte_cryptodev_raw_post_dequeue_t
post_dequeue,
        void **out_user_data, uint8_t
is_user_data_array,
        uint32_t *n_success, int *dequeue_status);

static __rte_always_inline void *
rte_cryptodev_raw_dequeue(struct
        rte_crypto_raw_dp_ctx *ctx, int
*dequeue_status,   enum rte_crypto_op_status
*op_status);

int
rte_cryptodev_raw_dequeue_done(struct
        rte_crypto_raw_dp_ctx *ctx, uint32_t n);
```

QAT PMD to Support RAW DP API

DPDK SUMMIT APAC•2021

- Each symmetric crypto algorithm type (Cipher/Auth/AEAD/Chain) has its own handler and is updated when session changed.
- In QAT, the shadow copy of Queue Pair head and tail are stored in `rte_crypto_raw_dp_ctx` during ctx initialization.
- Enqueue only write the jobs to the queue buffer and updates the shadow queue tail.
- On `rte_cryptodev_raw_enqueue_done()` call, kick the QAT queue to process written jobs and merge the shadow queue tail.
- Same way done to dequeue.

VPP Cryptodev Engine With RAW DP API

DPDK SUMMIT APAC•2021

- Cryptodev raw datapath API and the QAT support was merged in DPDK 20.11.
- VPP 21.08 unofficially, VPP 21.01 officially enabled raw datapath API in its dpdk_cryptodev engine.
  - Smaller memory footprint: saved one crypto op pool per socket.
  - More efficient:
    - Saved one loop of translation cost from VPP crypto frame into DPDK crypto op.
    - Saved crypto op status translation into VPP crypto op status.
    - Less cost on cached half-dequeued frame.
  - The end IPsec performance was improved by ~15%.

Summary

DPDK SUMMIT APAC•2021

- In general, DPDK Cryptodev Symmetric Raw Datapath API has the following advantages:
  - Compact and flexible crypto data structure, not dependant on mbuf, from stack.
  - All-in-one operation context.
  - Flexible and fine-grained enqueue and dequeue operations.
  - Avails close-to-native symmetric crypto performance to libraries/applications those are not mbuf centric and/or with its own crypto data structure.
- The API has been successfully integrated in FD.io/VPP, proven easier to adopt, more flexible, and more efficient.
- We hope this would encourage more changes made to DPDK, to make DPDK an even friendlier framework.


DPDK
SUMMIT
APAC•2021