DPDK

DATA PLANE DEVELOPMENT KIT

Accelerating vHost Data Plane with DMA in the CPU

JIAYU HU, INTEL

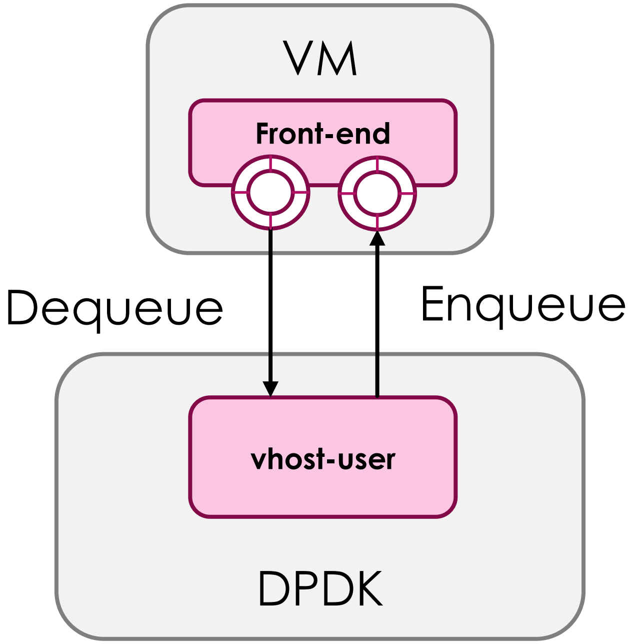
Para-Virtual I/O

- Para-virtual I/O is a virtualization technique to enhance VM I/O performance.

- VirtIO is a standard of para-virtual I/O, which consists of VirtIO front-end in VM and backend in hypervisor. User-space backend in DPDK is vhost-user.

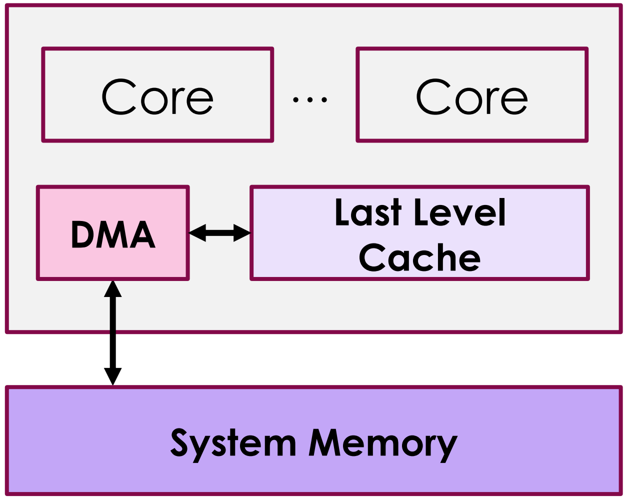
- vHost-user exchanges data with front-end via **copying packet buffers** between DPDK and VM memory.

***Copying large bulk of data** takes a major part of CPU cycles and becomes **hotspot** inside vhost-user.*



DMA Engine in the CPU

- DMA engine in Intel CPU is **extremely efficient** in performing **memory copy**.
  - No CPU intervention during data transfer.
- DMA engine in Intel CPU
  - Crystal Beach DMA (CBDMA) in Ice Lake and former CPUs.
  - Data Streaming Accelerator (DSA) in Sapphire Rapids CPUs.

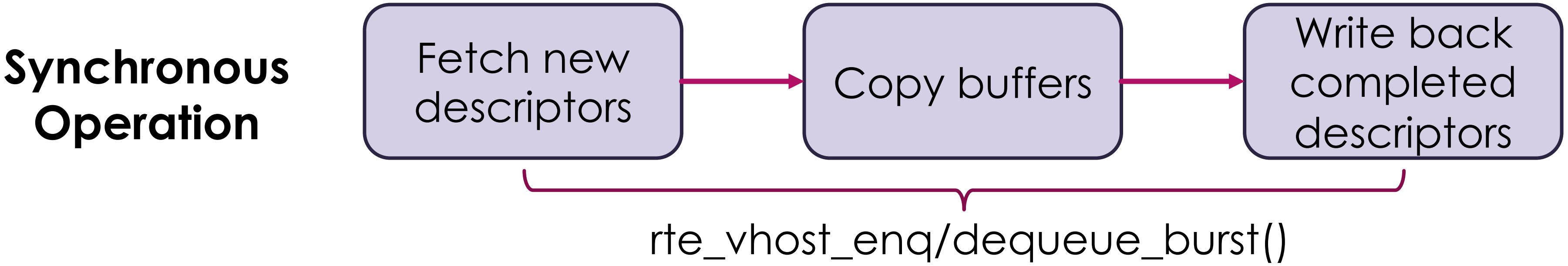


- DPDK provides IOAT driver and copy API for applications to leverage CBDMA and DSA.

https://doc.dpdk.org/guides/rawdevs/ioat.html

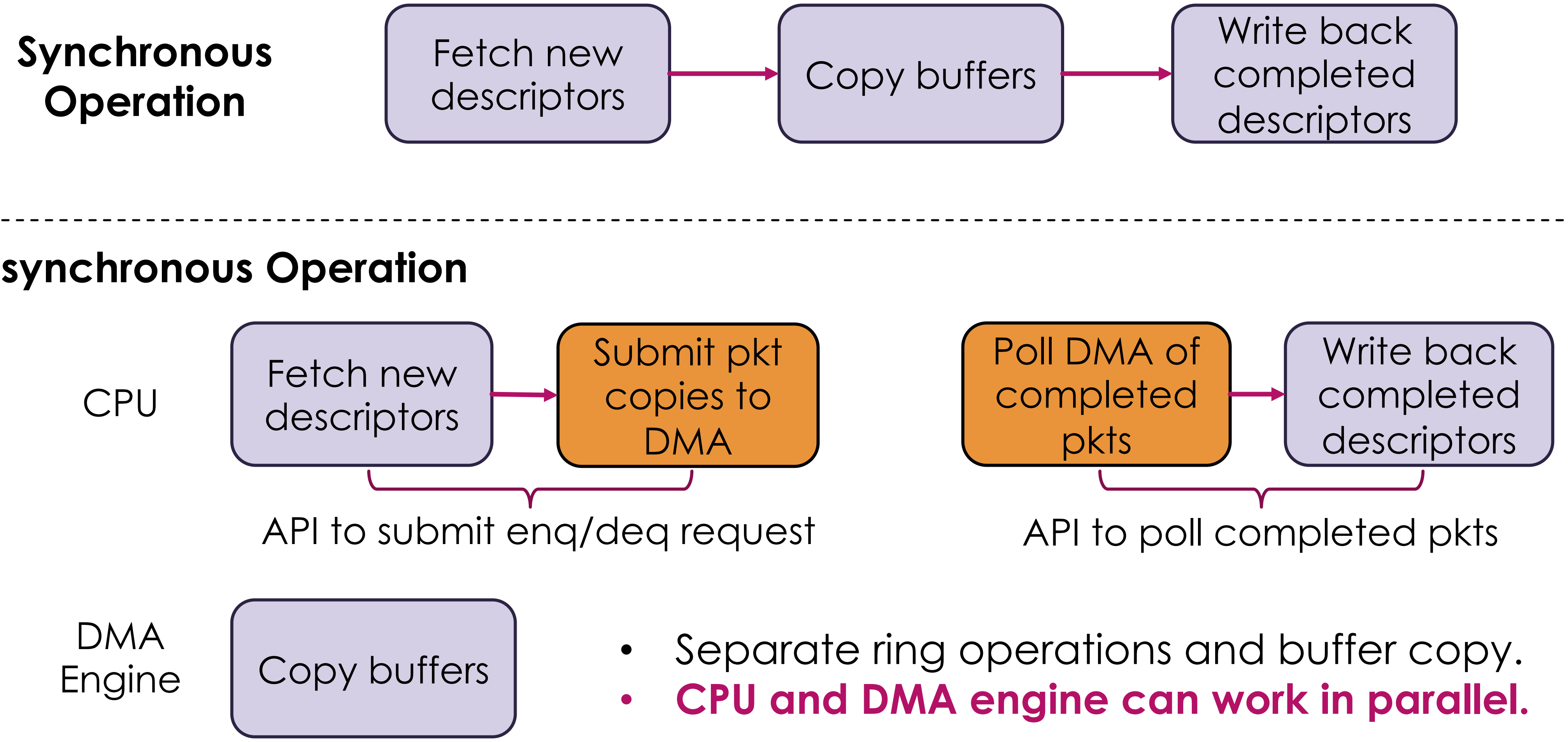
Challenge of Using DMA Engine in vHost-User (1)

- CPU and DMA engine **working in parallel** can significantly improve performance. **But** enqueue/dequeue API is **synchronous**.


Synchronous Operation
Fetch new descriptors
Copy buffers
Write back completed descriptors
rte_vhost_enq/dequeue_burst()

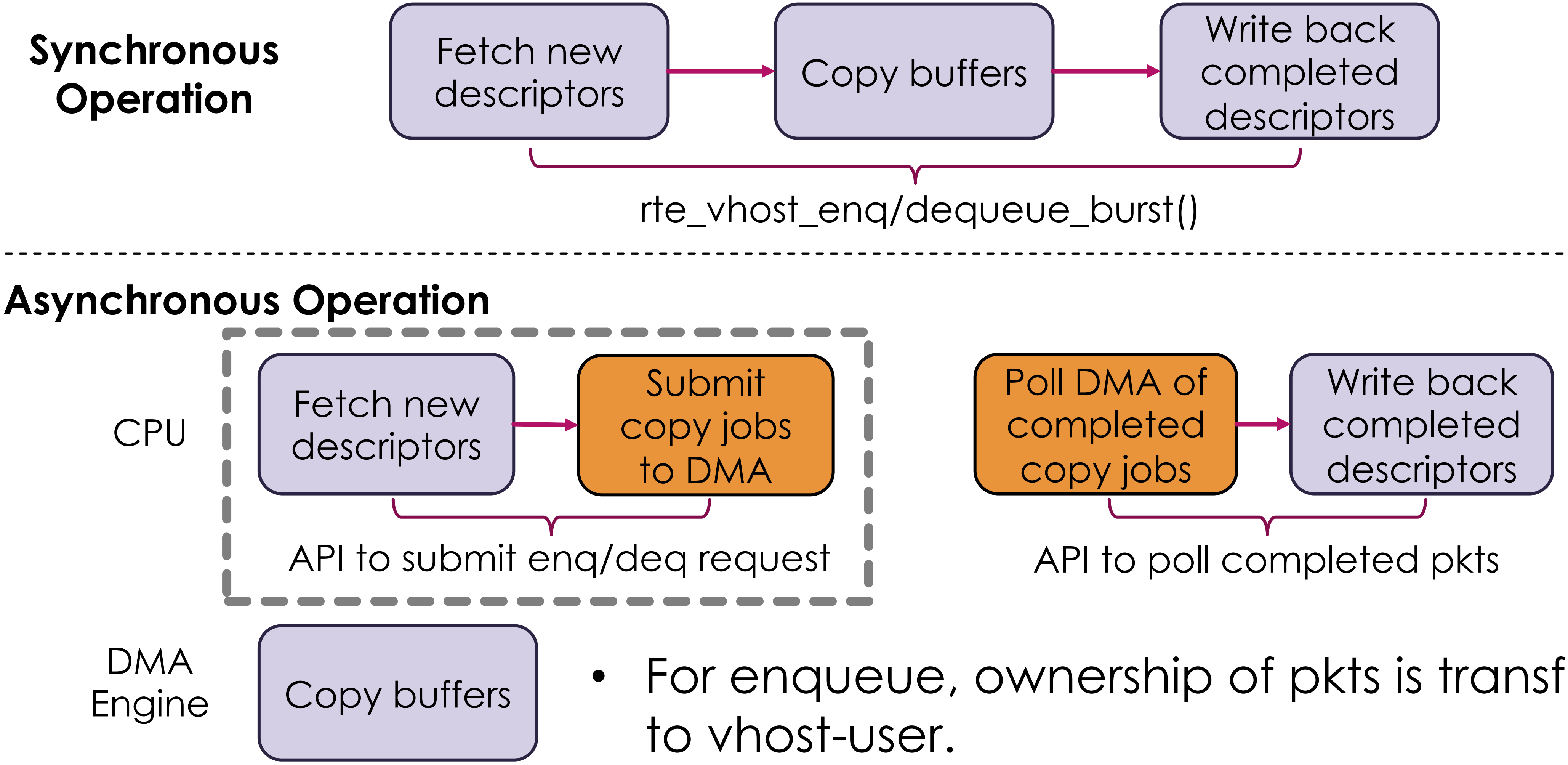
Ring operations and buffer copy cannot be parallelized.

Asynchronous Enqueue/Dequeue Operation



- Separate ring operations and buffer copy.
- **CPU and DMA engine can work in parallel.**

Asynchronous Enqueue/Dequeue Operation

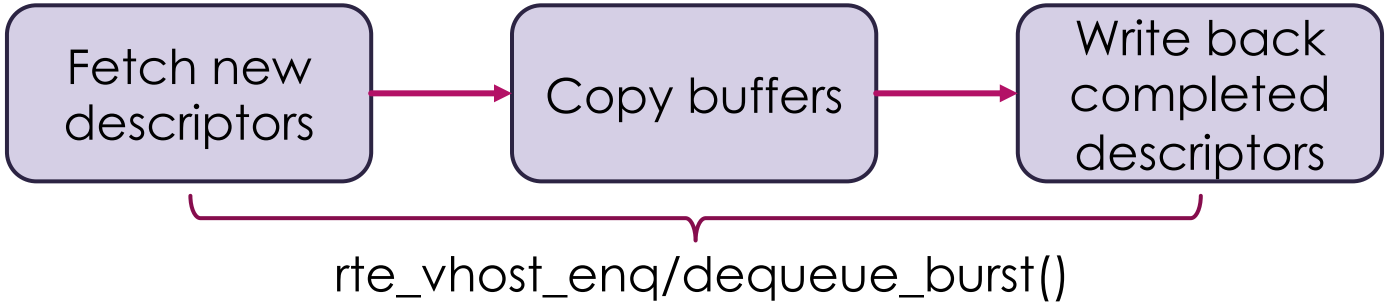


Synchronous Operation
Fetch new descriptors
Copy buffers
Write back completed descriptors
rte_vhost_enq/dequeue_burst()
Asynchronous Operation
CPU
Fetch new descriptors
Submit copy jobs to DMA
API to submit enq/deq request
Poll DMA of completed copy jobs
Write back completed descriptors
API to poll completed pkts
DMA Engine
Copy buffers

- For enqueue, ownership of pkts is transferred to vhost-user.

Asynchronous Enqueue/Dequeue Operation

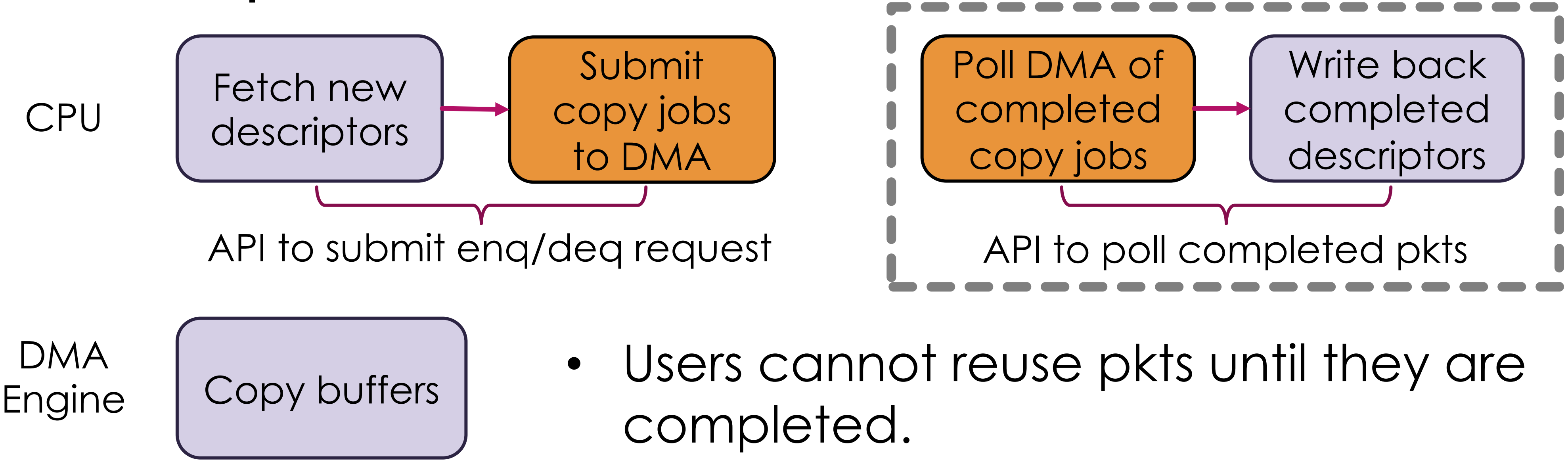
Synchronous Operation



Fetch new descriptors → Copy buffers → Write back completed descriptors

rte_vhost_enq/dequeue_burst()

Asynchronous Operation



CPU

Fetch new descriptors → Submit copy jobs to DMA

API to submit enq/deq request

Poll DMA of completed copy jobs → Write back completed descriptors

API to poll completed pkts

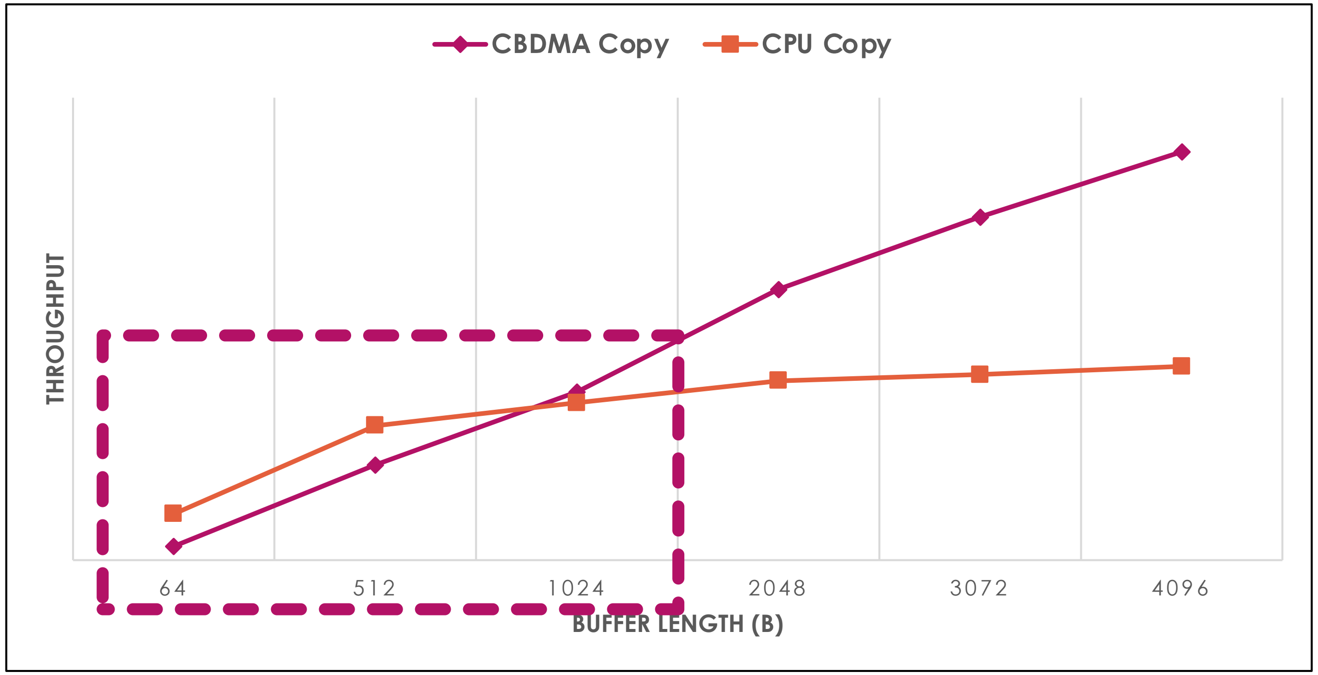
DMA Engine

Copy buffers

- Users cannot reuse pkts until they are completed.

Challenge of Using DMA Engine in vHost-User (2)

- DMA engine is **inefficient** in performing **small copies**, as a result of overhead of launching DMA engine.



Offloading all copies to DMA engine will underutilize DMA resources.

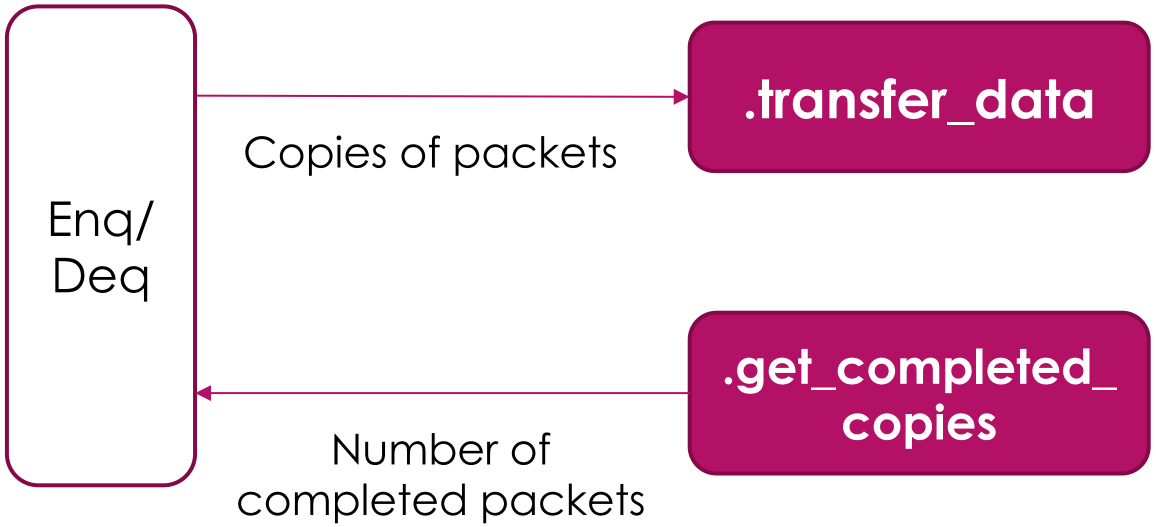
Dynamic Job Assignment

- In asynchronous operations, copies of packets are assigned to DMA engine or the CPU according to **copy lengths**.

- Copies whose lengths are greater than or equal to a ***threshold*** are assigned to **DMA engine**; others are assigned to the **CPU**.

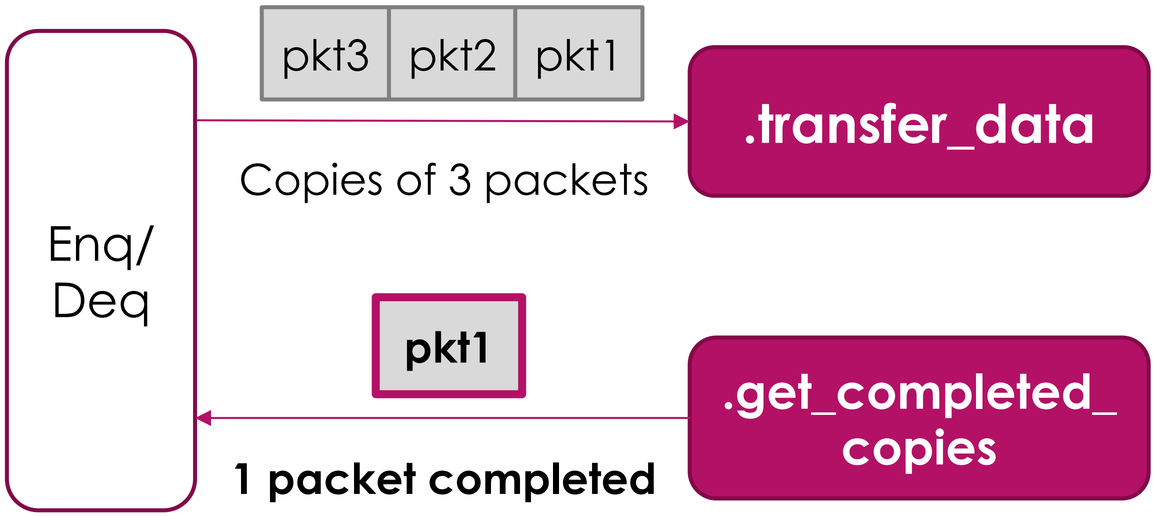
- The **value** of threshold is **decided** by **users** according to specific platforms and usage scenarios.

DMA Engine in vHost-User



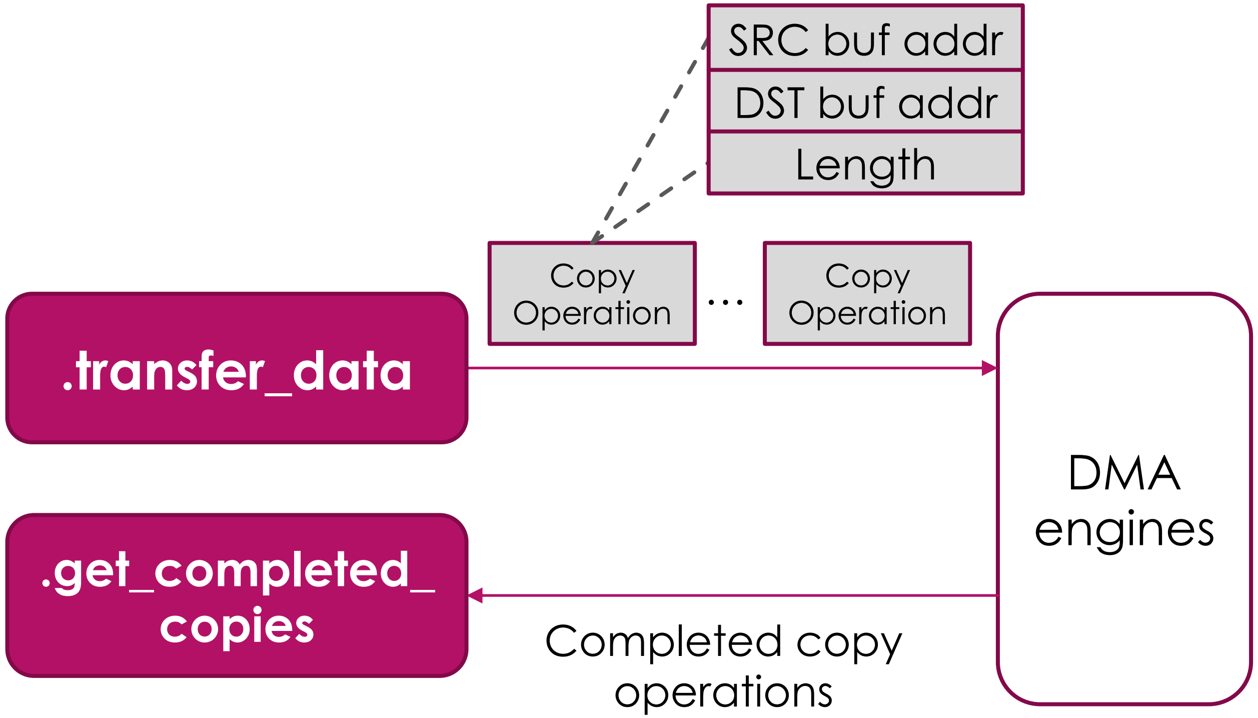
- DMA operations are abstracted as two callbacks: transfer_data, get_completed_copies.
  - Users provide callback implementations for specific DMA engines.
- **Order** of packets **submitted** to transfer_data must be the **same** as that of get_completed_copies **returned**.

DMA Engine in vHost-User



- DMA operations are abstracted as two callbacks: transfer_data, get_completed_copies.
    - Users provide callback implementations for specific DMA engines.
- **Order** of packets **submitted** to transfer_data must be the **same** as that of get_completed_copies **returned**.

DMA Engine in vHost-User



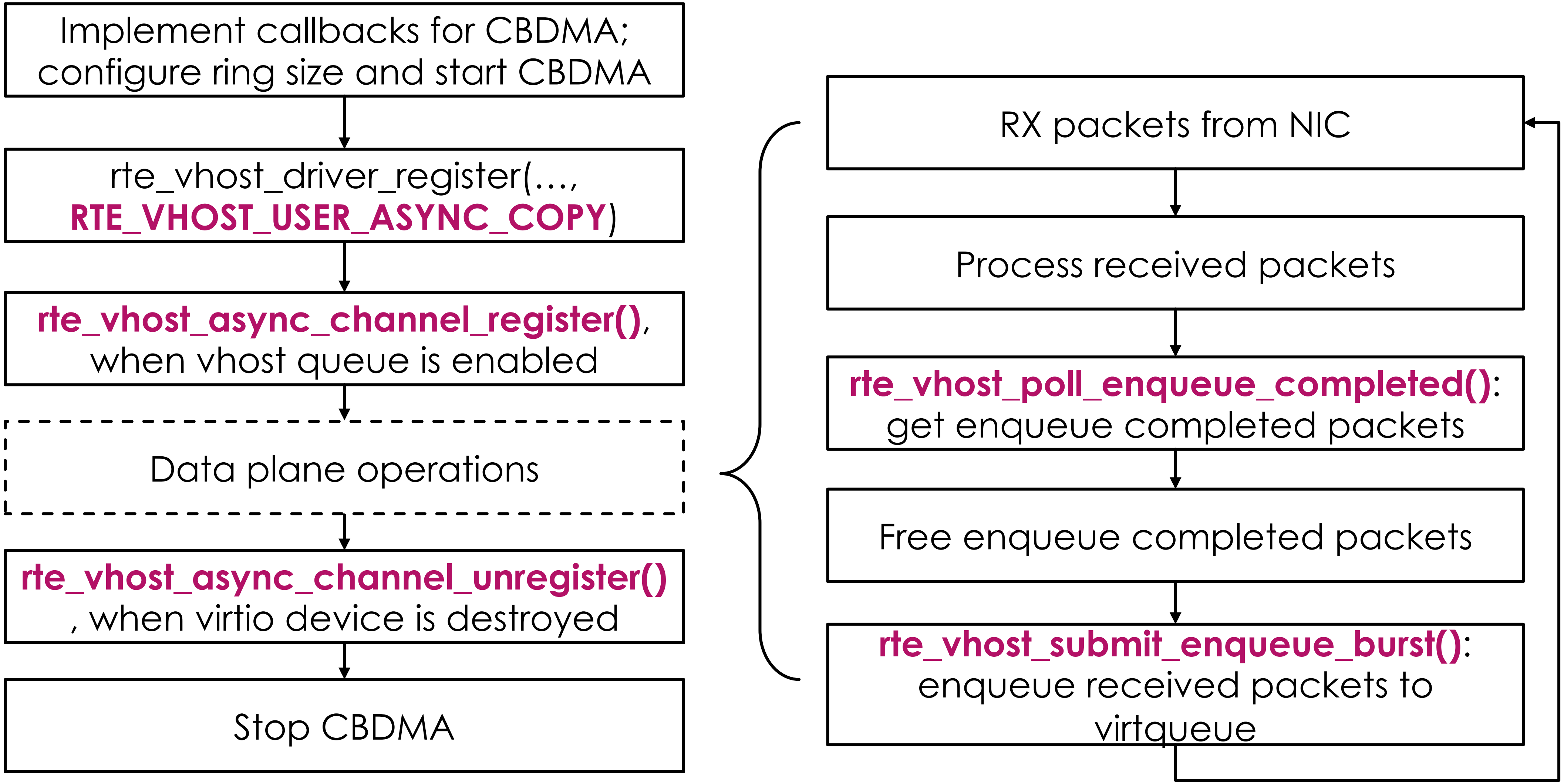
- DMA engines are **managed** by **users**.
  - Users configure/start/stop DMA engines.
- **Users assign** DMA engines to vhost queues.

Asynchronous APIs in vHost-User

- Control plane API
  - *rte_vhost_async_channel_register(vid, queue_id, …, ops)*
  - *rte_vhost_async_channel_unregister(vid, queue_id)*

- Data Plane API
  - *rte_vhost_submit_enqueue_burst(vid, queue_id, pkts, count, …)*
  - *rte_vhost_poll_enqueue_completed(vid, queue_id, pkts, count)*

Example of Using Asynchronous API



Implement callbacks for CBDMA; configure ring size and start CBDMA

rte_vhost_driver_register(…, **RTE_VHOST_USER_ASYNC_COPY**)

rte_vhost_async_channel_register(), when vhost queue is enabled

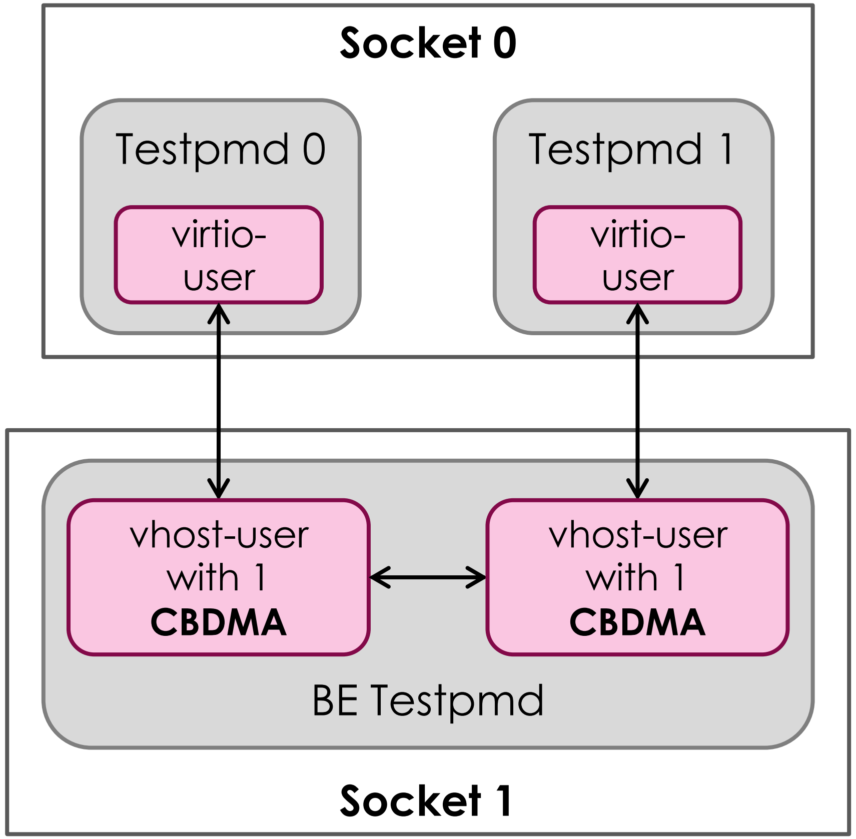
Data plane operations

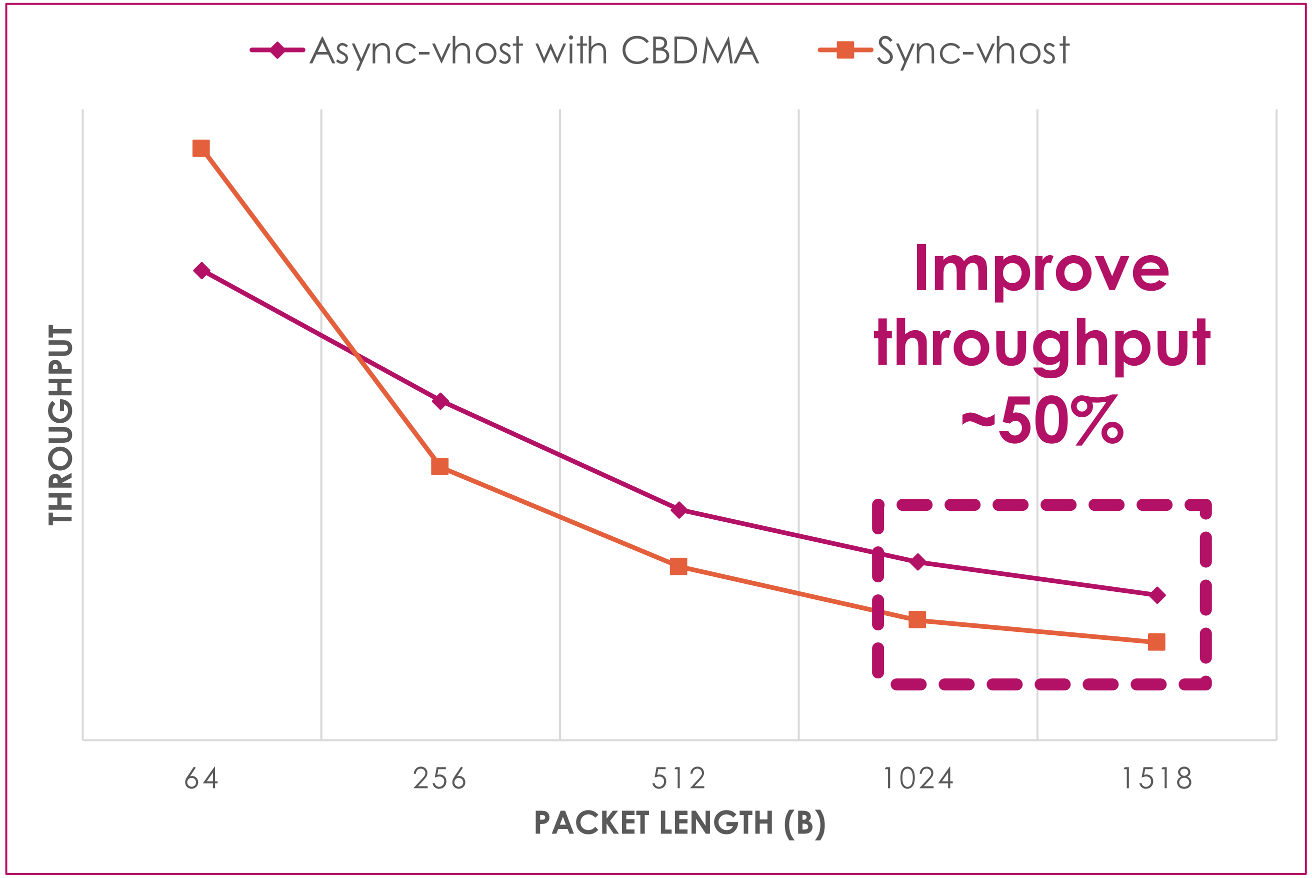
- RX packets from NIC
- Process received packets
- **rte_vhost_poll_enqueue_completed()**: get enqueue completed packets
- Free enqueue completed packets
- **rte_vhost_submit_enqueue_burst()**: enqueue received packets to virtqueue

rte_vhost_async_channel_unregister(), when virtio device is destroyed

Stop CBDMA

Asynchronous vHost-User Performance





FE: virtio-user with 4q4c and running csumfwd, BE: 4q1c and running csumfwd, Skylake 2.5GHz

Status and Plan

- DPDK 20.08
  - Supported asynchronous enqueue for split ring.
  - Enabled asynchronous enqueue in vhost example.
- Support asynchronous enqueue for packed ring in DPDK 21.05.

- Related references:
  - https://www.dpdk.org/wp-content/uploads/sites/35/2018/12/JiayuHu_Accelerating_paravirtio_with_CBDMA.pdf
  - https://www.dpdk.org/wp-content/uploads/sites/35/2019/10/Asynchronous.pdf
  - https://doc.dpdk.org/guides/prog_guide/vhost_lib.html
  - https://01.org/blogs/2019/introducing-intel-data-streaming-accelerator

Thanks

jiayu.hu@intel.com