# OVS-DPDK
# vHost async data path
# using DMA-dev

# Agenda

Ensure that all of

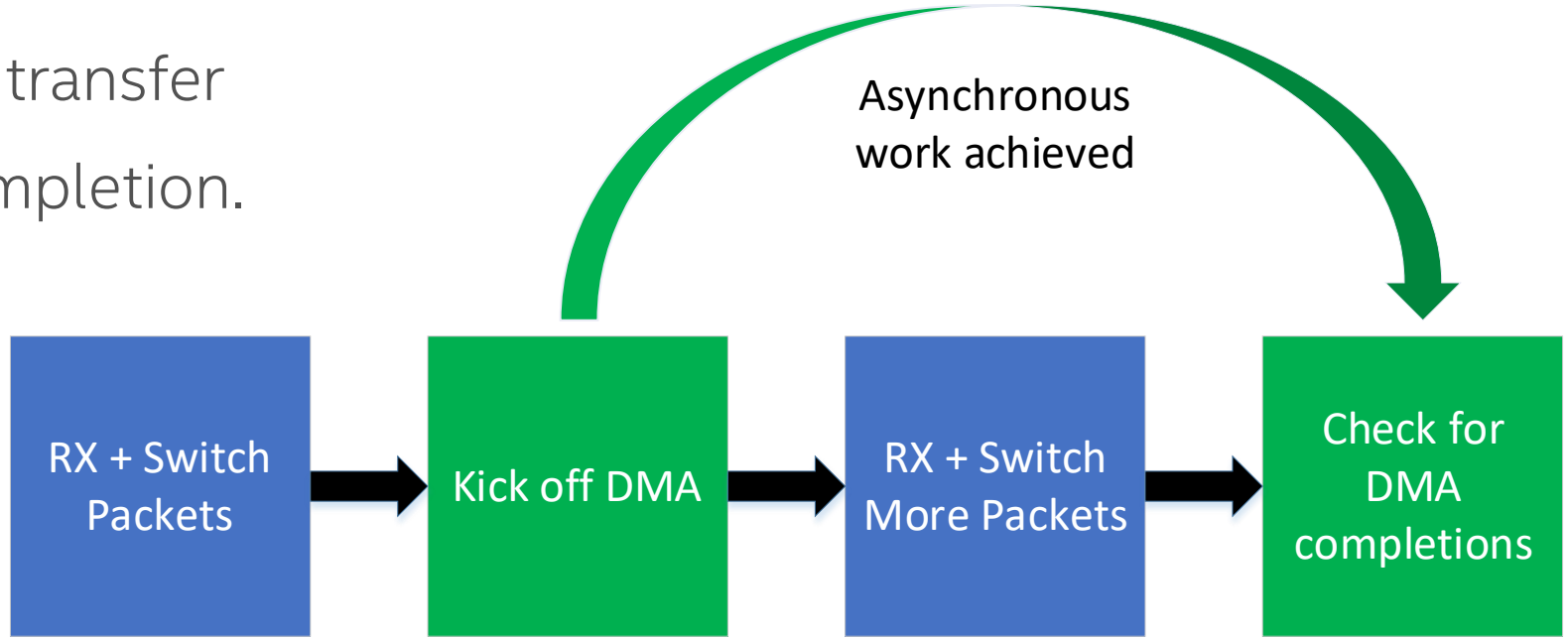
- DPDK DMA-dev library,
- DPDK vHost library (consuming DMA-dev for acceleration) and
- OVS (as an end user of the DPDK DMA-dev & vHost libraries)

are working well together and

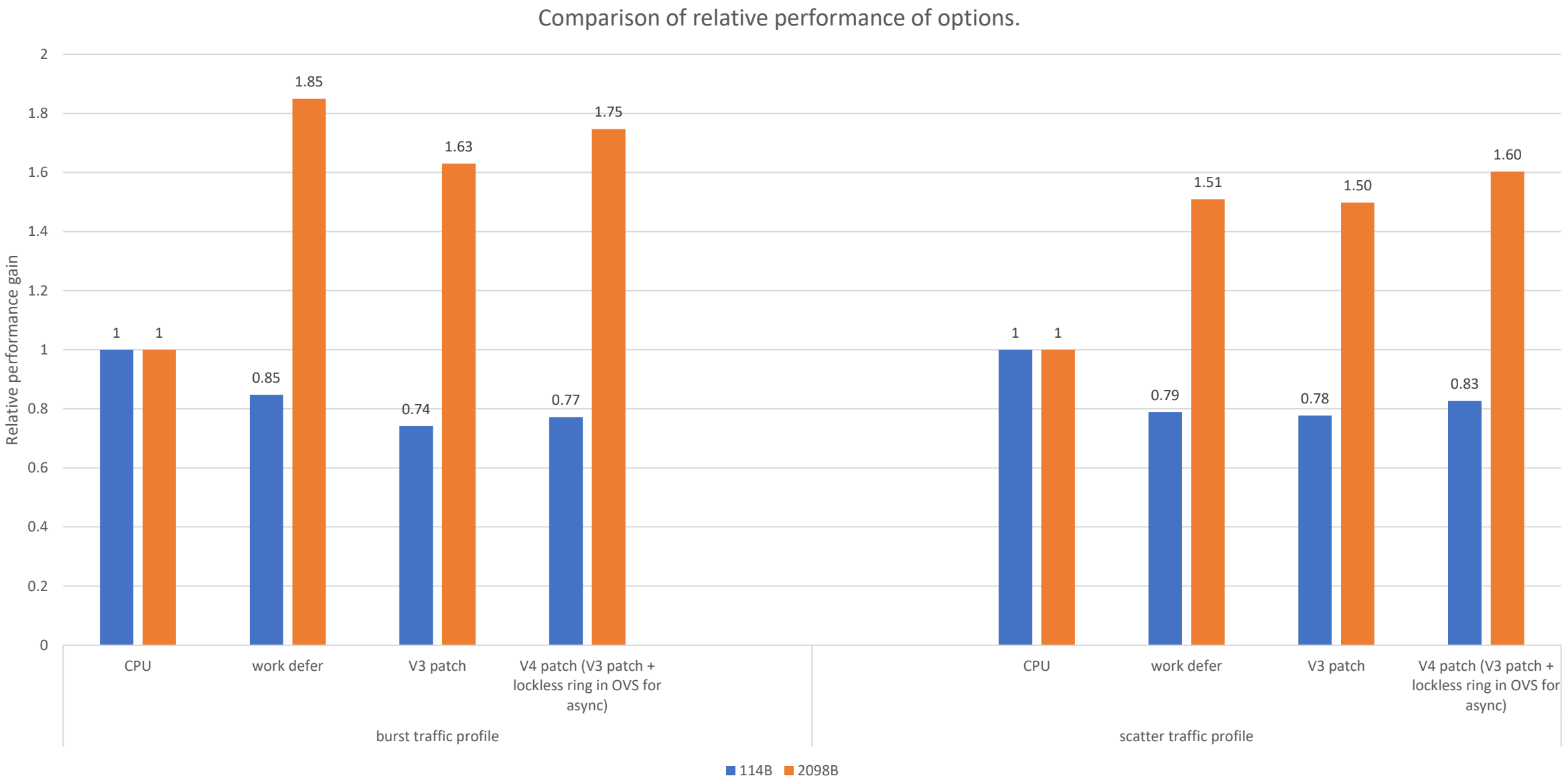
that the maintainers & contributors to those libraries are aware of the design & architecture in OVS consumption.

# Motivation

- Packet copy into guest can be slow for large packets.
  - Accelerate this using DMA engine
- Don't stall the CPU while DMA engine is active
  - i.e **asynchronous acceleration**.
  - Call DPDK API's to start transfer
  - Check back later for completion.



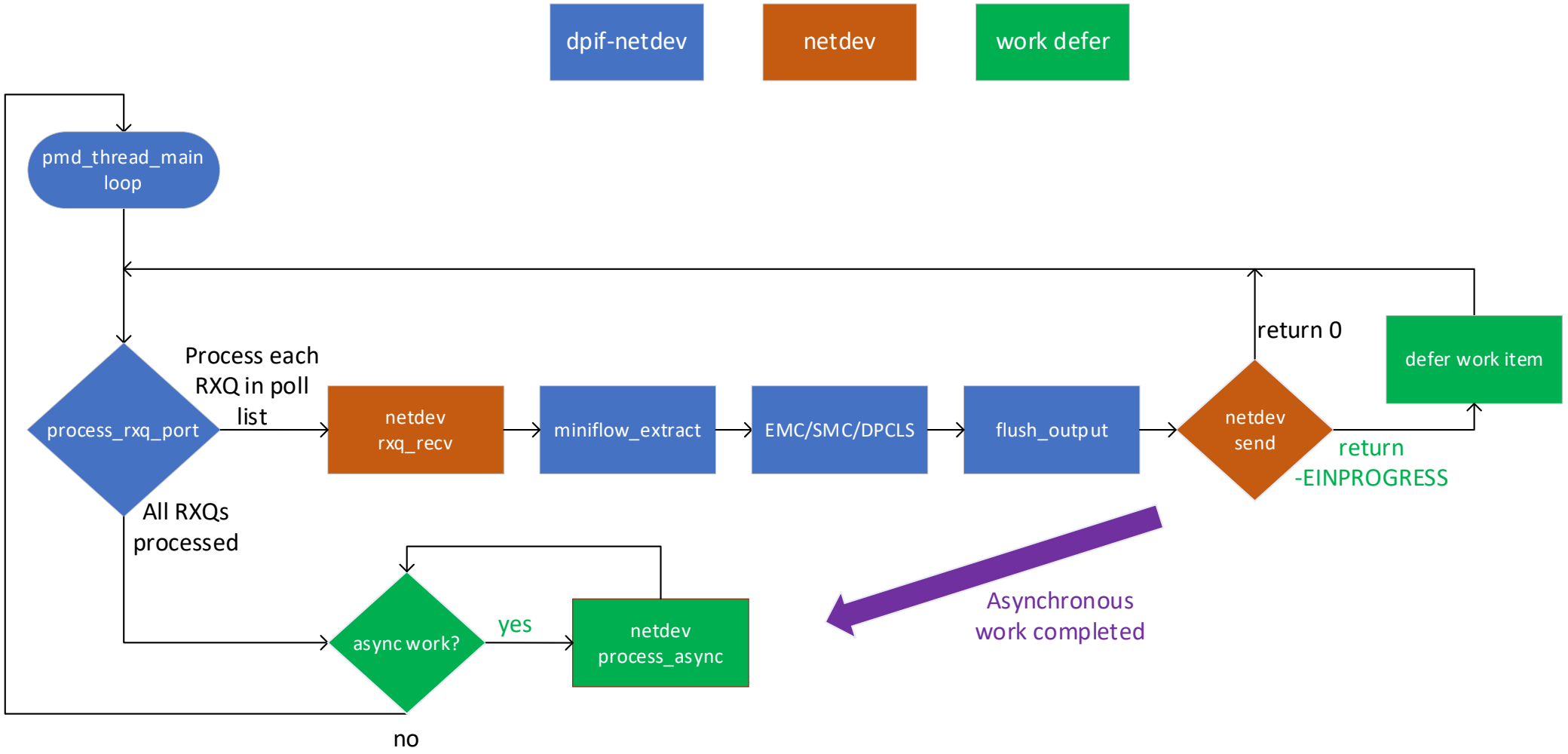
# Performance



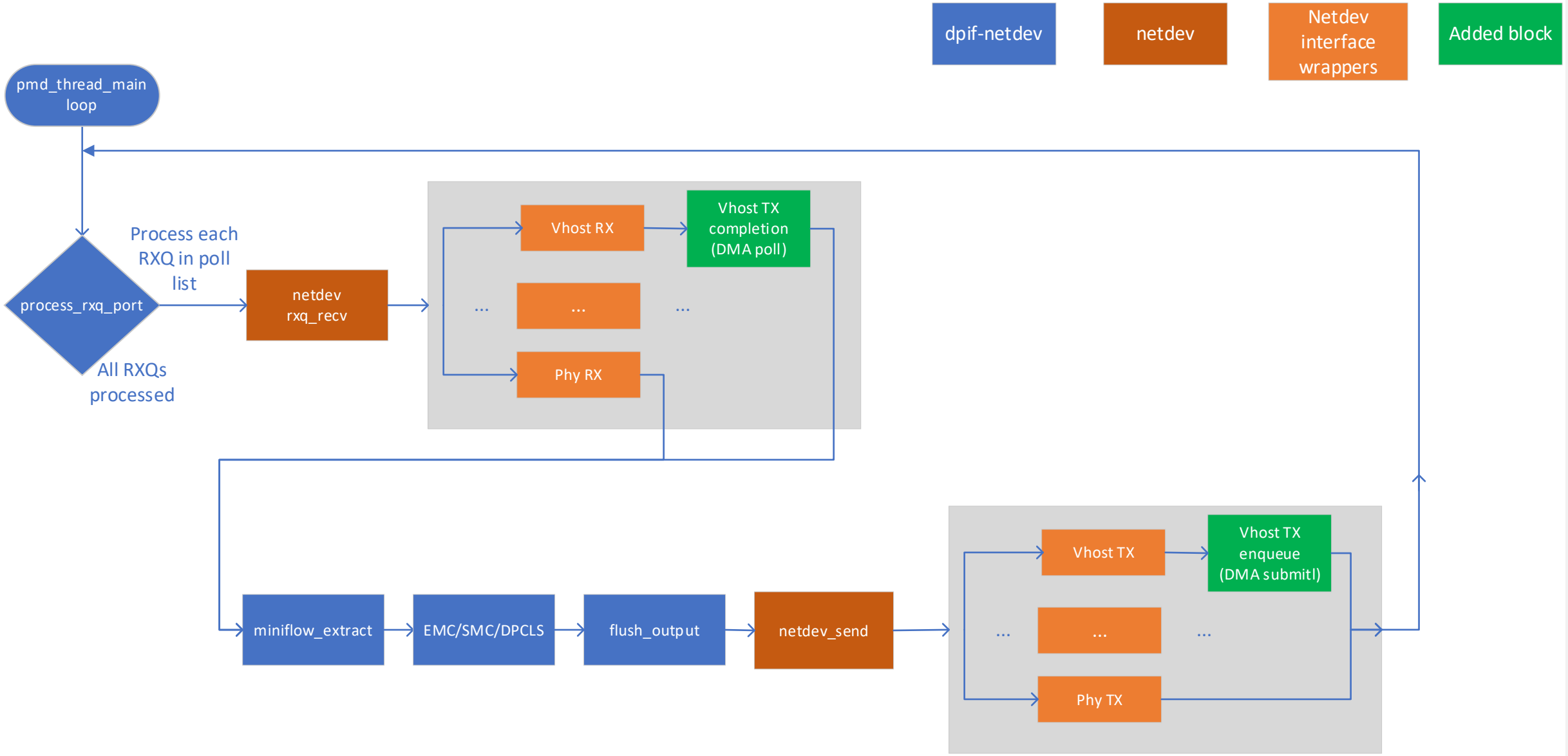

This data is based on the POC's implemented and tested on new Intel hardware that supports DSA , CPU@1.8Ghz ,

# How do we enable asynchronous acceleration ?

# Option1 : Defer work



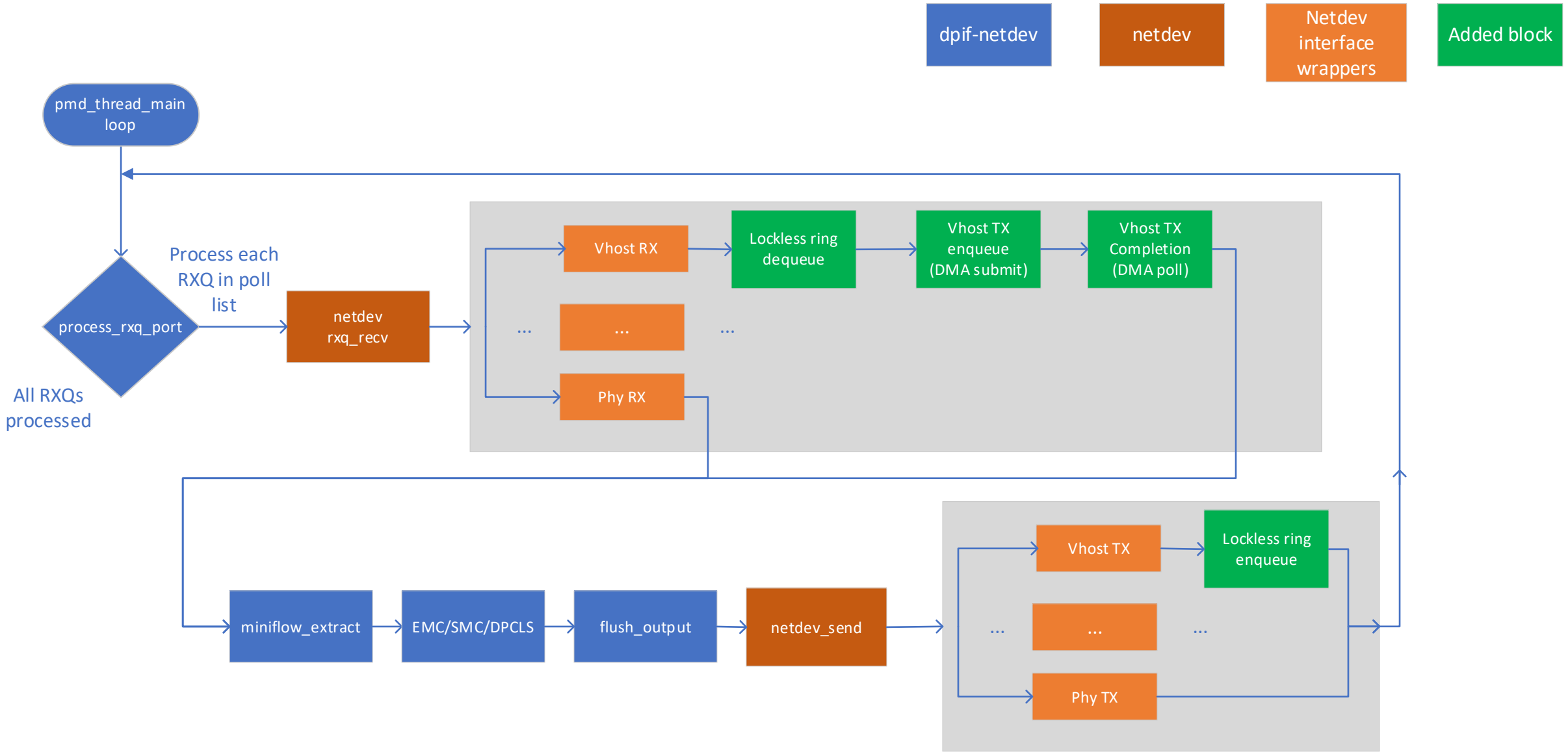
# Option2 : Tx completions from Rx context.



# Option2 : Tx completions from Rx context.

- Problems – more tx contention among threads
  - between poll and enqueue
- Mitigations – with lockless ring.
  - rte_ring - MP/SC in between to avoid the contention in pipeline.
  - Thread enqueues to rte_ring , NOT vhost/DMA.
  - Polling thread dequeues from rte_ring , enqueues to vhost/DMA, polls packets.

# Option3 : Tx completions from Rx context + lockless ring



# Concerns on option 2 and 3

- Dependent on vHost Rx to perform vHost Tx operations.
  - Need a vHost Rxq on **every** data plane thread for Txq completions to work.
    - Since DMA assignment is per data plane thread
  - PMD auto load balancing can potentially be incompatible in cases where it can cause few rxq's to be not polled on the same data plane thread.
    - TODO: find solution for this case -> PMD thread *must* handle completions regularly.
  - No such constraints for deferral of work(option1)
    - Per thread work ring ensures deferred work (Txq completions) is completed on the same thread

# Complexity

3 implementations: from most to least complexity in OVS codebase:

- Work Defer:
  - Complexity is added to dpif-netdev as well as netdev-dpdk, with async-free logic in both.
- V4 patch (V3 patch + lockless ring):
  - Complexity is added just to netdev-dpdk, with async-free logic in OVS under the RX API wrapper AND with lockless ring complexity added in netdev-dpdk.
- V3 patch:
  - Complexity is added just to netdev-dpdk, with async-free logic in OVS under RX API wrapper.

In all above implementations, **OVS is responsible for configuration of DMA devices**.
All **dataplane DMA-dev usage is abstracted inside DPDK VHost library**.

# Next steps

- Keeping OVS merge timelines in mind:
  - POC by 2.18(Aug 2022) and upstream by 2.19 (Feb 2023)
  - Need DPDK vHost API's to be finalized – non-experimental by 22.11 (Nov 2022)
- Open Questions:
  - **Defer work** has best performance, and elegant solution to DMA-completions
    - at cost of complexity at OVS pmd-thread level (handles corner cases e.g. no vhost-rxq)
    - Does the defer-work DMA-completion handling merit the OVS complexity?
  - **V3+Lockless Ring** has middle-performance
    - But needs a vhost-rxq on every PMD-thread…
    - or other solution to always handling DMA completions?

**DPDK and OVS Communities need to align to a Solution**

# Follow the discussion here on ML:

- http://patchwork.ozlabs.org/project/openvswitch/cover/20220104125242.1064162-1-sunil.pai.g@intel.com/

- Latest patch on ML V4: http://patchwork.ozlabs.org/project/openvswitch/list/?series=291382


intel

# Test Topology