OVS-DPDK vHost async data path using DMA-dev

(session2 - 29/03/22)

Agenda

- Opens
- **DMA Mapping:** Static or Dynamic allocation of resources?
  - Informs below discussion on implementations
- **Implementation Options:** Continued discussion OVS Dataplane
  - Work Defer
  - V3 patch
  - V4 patch (V3 patch + lockless ring)
  - Design Feedback on 3 approaches above
- **Code Availability:** Easy-to-use code from Github
- **Closing:** Next Steps

Dynamic Alloc : Pro/Con Overview

	PMD-thread	VRing
Static Assignment	✓ **Simple access** to DMA HW Direct static mapping: thread to DMA id. ✓ **Lockfree** access to device Direct static mapping: thread to DMA id. ✓ **Scaling** as 1:1 mapping has no contention Adding PMD threads adds more DMA capability too! ✓ **Simple Config:** DMA queue available for thread? Yes; claim it and use it. No; Use CPU memcpy for that PMD thread (like today).	✓ **Simple** access to DMA HW Direct static mapping from VRing to DMA id. ✗ **Locking Required** Multiple vrings map to same DMA id, thread contention. ✗ **Scaling suboptimal** Contention based on traffic, vrings = DMA id, oversubscription! ✓ **Simple Config:** DMA queue available for thread? Yes? Use it. No? CPU copy for this vring.
Dynamic Assignment	? **Simple/Complex** access to device, is it shared? Sharing requires runtime check for DMA id to use? ✗ **Locking required** Sharing == multi-threaded accesses. Drain DMA-completions from 2+ threads? ✗ **Bad Scaling** Active PMD threads map to same DMA id, contention. ✗ **Complex Config** Adding PMD requires other PMD threads to change access from "simple" to "complex" at runtime?	? **Simple/Complex** access to device, is it shared? Sharing requires runtime check for DMA id to use? ✗ **Locking required** Sharing == multi-threaded accesses. Drain completions from 2+ DMA-engines for single virtq? ✗ **Bad Scaling** Active VRings map to same DMA id, contention. ✗ **Complex Config** Adding VRings requires other VRings threads to change access from "simple" to "complex" at runtime?

Agree to merge ***static*** **PMD-thread** mapping now?
Optionally enable ***dynamic*** **PMD-thread** mapping in future?

DMA Implementation Overview

	Defer Work	**V3**	**V4** (v3 + Lockless Ring)
Link to Patchset (Date initially posted)	Defer Work RFC Patch (2021-09-07, ~6 months ago; V1 RFC: 2021-04-12, ~11 months ago)	V3 RFC patch (2022-01-04, ~3 months ago)	V4 RFC patch (2022-03-21, ~1 week ago)
Performance (Scaling and tx-thread contention)	**Good** **(baseline)**	**Lowest** **(-22% on Defer Work)**	**Good** **(~same as Defer Work)**
OVS Code Complexity (DPIF thread-level code)	**High**	**Lowest**	**Medium**
RX/TX "Clean Split" (Independent Rx and Tx DMA-completions)	**Yes**	**No**	**No**
Memory Footprint Increase (Lockless rings for packets)	**No** (work-item tracking)	**No**	**Yes** (per-packet tracking, pre-alloc per vhost txq)
Packet Drop Potential (*more* packet "ring full" scenarios)	**No**	**No**	**Yes** (wait or drop)

Ask to community : **What implementation do we merge?**

Next Steps, open thread on OVS Mailing List to finalize decision?

Getting the Code : Availability & Ease-of-Use

- Code will be available on Github (after choice of impl)
  - DPDK: https://github.com/istokes/dpdk/tree/dma-tracking
  - OVS: https://github.com/istokes/ovs/tree/dpdk-dma-tracking
  - Consolidated set of patches for VHost enabled DMA-dev acceleration in OVS
  - Simplifies getting/testing the code, as its all available in a single git branch (per project)
- Instructions for compile/use
  - Clone DPDK, compile & install as normal
  - Clone OVS, compile & install as normal
  - Enable DMA acceleration using command :

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:vhost-async-support=true
```

Next Steps

- Progress on
  - Decision for DMA Static PMD-thread allocation
  - Decision for OVS DMA datapath implementation
- Collaborate on OVS Mailing List
  - Code availalable on Github for ease of use
- Future calls, discussions/opens?
- Closing


intel