

# CXL, SmartNICs and OVS: a paradigm change?

Alejandro Lucero

[alucero@os3sl.com](mailto:alucero@os3sl.com)

OVSCON

December 2023

# Disclaimer

I work for AMD but I do not represent AMD's view in this presentation nor any AMD's roadmap regarding CXL or SmartNICs. My goal is to discuss my vision of current SmartNICs start of the art regarding potential CXL uses.

# CXL, SmartNICs and OVS: a paradigm change?

## Agenda

1. Introduction
2. SmartNICs with MATs ... and DRAM
3. Slow-fast paths: the Linux/OVS/Openflow way
4. CXL into the picture
5. Counters and Multitenant Openflow
6. Conclusions

# 1. Introduction

CXL brings sharing memory efficiently between Host CPUs and high performance devices (initially targeting GPUs).

PCIe devices have “shared” memory with the Host through PCI BARs windows but lacking the performance and mainly the coherency required.

CXL can be used in the datapath ... and in the control path.

# 1. Introduction

Being part of SmartNIC projects in two different companies supporting OVS-TC offload.

And the control path through the kernel TC (and conntrack) not progressing too much. Does it need to?

(My point ) Massive virtualization can not rely on TC: the rate of change has been ignored.

Latencies due to a far from optimal software datapath plus latencies due to how the hardware offload is done imply more cpu cycles “stolen” from VMs.

# 1. Introduction

I could not help thinking all this just can not efficiently work in real massive virtualization systems.

Improvements can be done but when you need to go to the Moon, changing a bike by a car does not help too much.

Then CXL for the Control Path popped into my head: it just made sense.

## 2. SmartNICs, MATs and DRAM

The SmartNIC type I want to talk about is the one used for improving networking in virtualization.

VMs talk straight to the NIC/HW (SRIOV, VFs, IOMMU, VDPA, ...)

Packets sent by VMs and packets coming from the wire are first handled by the NIC using Match and Action Tables (MATs). Example: a packet matching a rule will end up being redirected to a specific VM and the tunnel header removed. This is the hardware datapath.

A packet not matching any rule has a default action: drop ... or send to the Host. This is the software datapath.

## 2. SmartNICs, MATs and DRAM

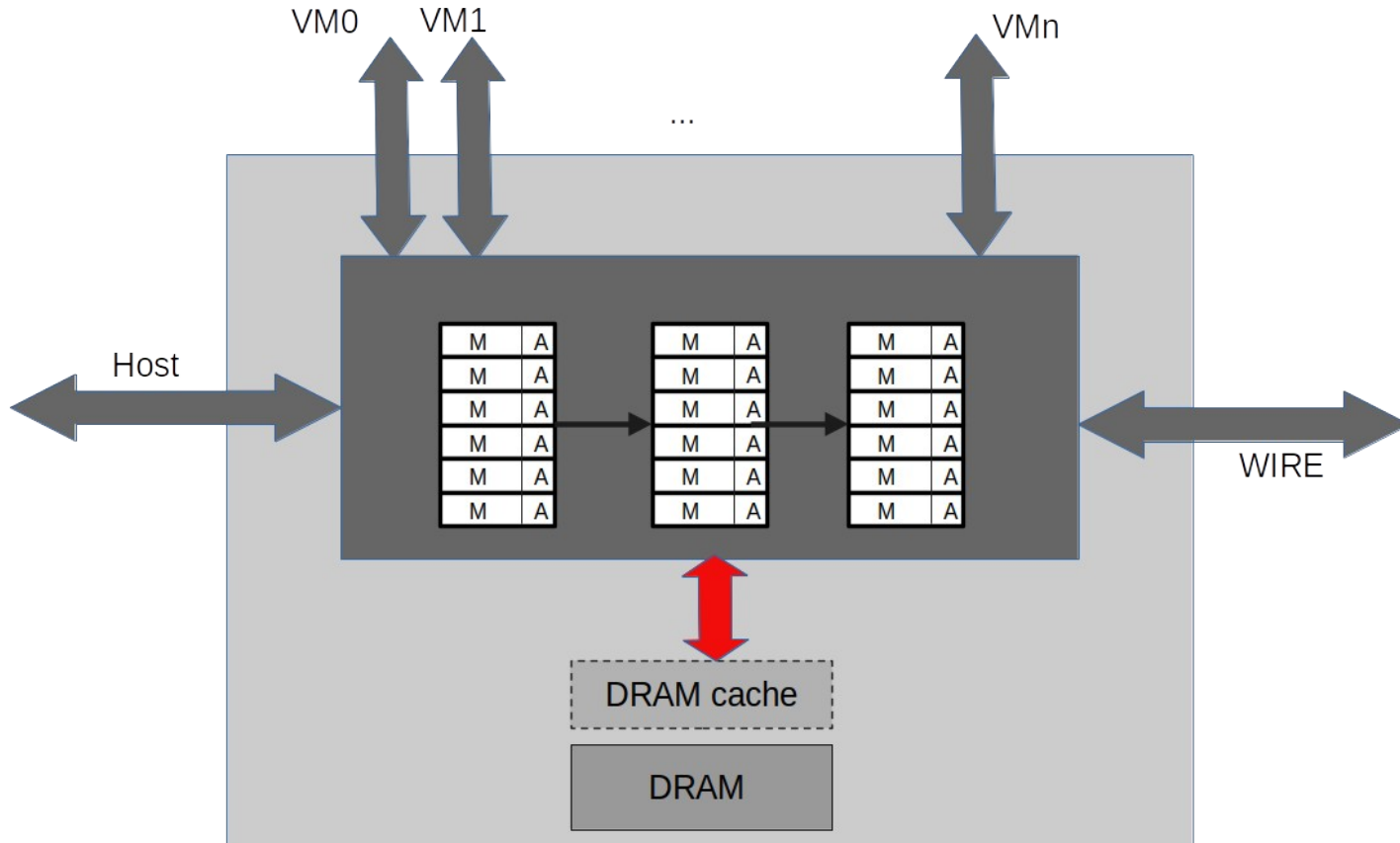
MATs functionality are more efficiently implemented with HW xCAMs where the matching can be performed against all the rules in a MAT table in parallel.

This parallelism has limitations due to power requirements so the MATs have usually maximum sizes far from what massive virtualization requires.

A DRAM is used where rules and flows can be used in the millions.



## 2. SmartNICs, MATs and DRAM





### 3. SW/HW datapaths: The OVS/Linux way

Does the SW datapath make sense? Let's assume so:

- ✓ Openflow rules: proactive vs reactive. Rate of change?
- ✓ Slow path (Host) having more resources than HW (more rules/flows handled).
- ✓ Supporting those rules/actions (and protocols) the HW can not deal with.
- ✓ Connection tracking based on software (Linux kernel conntrack).

### 3. Slow/Fast datapaths: The OVS/Linux way

For the sake of the discussion, the SW datapath is the kernel OVS-TC.

Packets are handled by the kernel Traffic Control (TC) which has matching and action capabilities (TC Flower).

If no match there, the OVS kernel switch sends the packet to the OVS userspace component: this is the slow path.

A new TC rule will be installed for processing this and related packets (flow) in the future. This can imply to offload that rule to the NIC.

### 3. SW/HW datapaths: The OVS/Linux way

With conntrack, the SW datapath is still necessary after the HW gets the rule.

The HW rule will need the connection state related to the packet: the packet will be sent through the SW datapath.

The kernel TC rule will create the connection state and it will be offloaded once established.

This is always reactive and the expected rate of change far higher than rules/actions updates.

### 3. SW/HW datapaths: The OVS/Linux way

Why using kernel TC for the SW datapath?

First OVS in HW implementations had private per-vendor solutions for offloading, but unlike other similar cases, there was not convergence nor a Linux kernel configuration option selecting one ...

The kernel Traffic Control was proposed because it had match/action and it could be extended (TC Flower).

Does it work?

### 3. SW/HW datapaths: The OVS/Linux way

TC was designed at a time where virtualization was mainly a mainframe solution (Dad, what's a mainframe?) ...

At a time where ATM networks seemed the future ...

And when processing packet by cpus was not a big problem. The performance gap has significantly increased: less cpu cycles per packet for dealing with increasing network bandwidth.

So it can hardly (efficiently) work for massive virtualization where the SW datapath is not just the poor man but one important key behind performance.

### 3. SW/HW datapaths: The OVS/Linux way

TC problems:

- TC Qdiscs can not be updated in parallel (not needed because it is the control path ...).
- TC rules are matched against sequentially.
- Parsing and matching done per rule.
- TC syntax gap: Openflow → TC → HW specs
- TC ossification (was not a HW thing?) ... Here it comes P4.
- Conntrack: Hash Tables! ... but processing linked to TC processing (flow connection state as a matching field).



### 3. SW/HW datapaths: The OVS/Linux way

P4 as the frontend for the software datapath. Hardware and software datapaths from same “source”.

P4 software backend? eBPF? I really do not care as long as the implementation is overcoming current TC problems: optimization for properly supporting the massive virtualization case.

I do care about the control data representation, data access, data updates.

Could not the slow path and the fast path, coming from same source, share the same data? ...

### 3. SW/HW datapaths: The OVS/Linux way

TC/contrack offload problems:

- TC rule offload synchronously done when rule update through specific driver code. But TC Qdiscs can not be updated in parallel ...
- Driver needs to check if the rule/actions are supported.
- Driver needs to translate the TC rules to the HW intrinsics.
- The rule needs to be sent to the HW (datapath, control channel)
- HW Acking (sync/async). Identifier.
- MATs population: When is the rule really seen by the switching logic? How long is the latency? Is it deterministic?

### 3. Slow/Fast paths: The OVS/Linux way

TC/contrack offload problems:

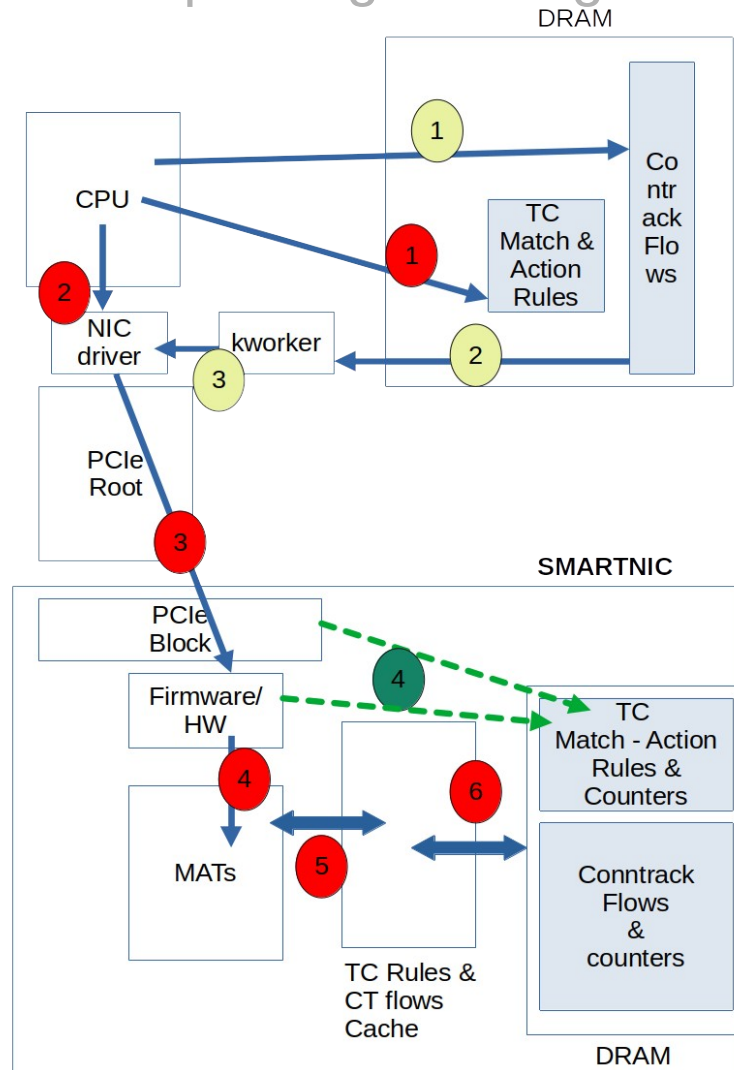
- Asynchronous contrack flow offload when established. Kernel kworkers used ... one work per flow direction!
- Kworker executing specific driver code. When?
- Driver checks if the contrack state is supported.
- Driver needs to translate the flow to the HW intrinsics.
- The flow state needs to be sent to the HW (datapath, control channel)
- HW Acking (sync/async)
- When is the flow state really seen by the switching logic? How long is the latency? Is it deterministic?

# CXL, SmartNICs and OVS: a paradigm change?

## TC/contrack offload problems:

- Driver involved: sync/async
- Data “duplicated”: updated twice\*.
- Extra Host cpu cycles
- Non deterministic
- Specific HW design
- ...
- Will not the data end up in device DRAM?

\* Drivers keep rules data, so it is really triplicated



### 3. SW/HW datapaths: The OVS/Linux way

TC/contrack offload problems:

- Any added latency implies subsequent flow packets going through the SW datapath. More SW datapath implies more latency ...
- Rate of change! It is not just about adding rules but also about removing them based on traffic (reactive solution).
- With contrack flows state this is just more critical ... and demanding.

## 4. CXL into the picture

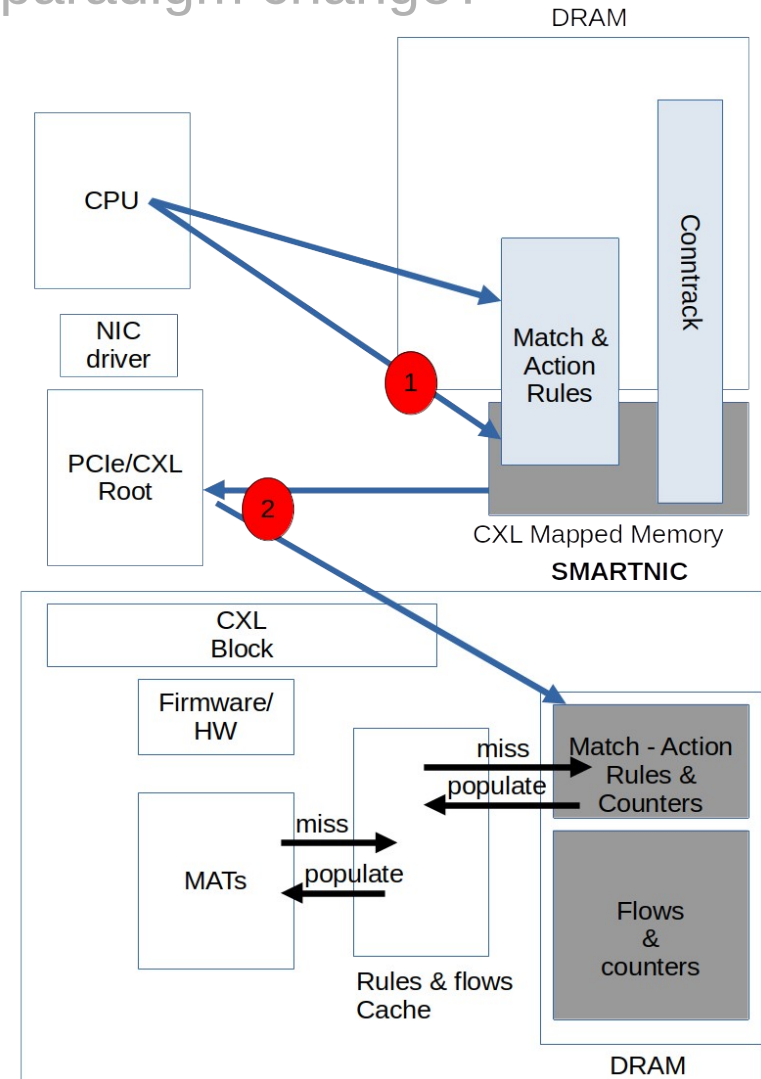
What about a Control Path where:

- 1) Updates would be a matter of CPU writes to memory addresses. No driver intervention for offloads.
- 2) A DRAM memory inside the SmartNIC backing those memory addresses. Standardization at the DRAM level: HW intrinsics per vendor but after accessing DRAM by the HW logic.
- 3) The CPU able to read such a memory as part of code execution: the SW datapath data.

# CXL, SmartNICs and OVS: a paradigm change?

## TC/contrack offload with CXL:

- Driver not involved.
- Shared data: updated once.
- No Extra Host cpu cycles
- CXL protocol
- ...
- Data ends up in device DRAM, and Host CPUs using it (CXL coherency).
- Host DRAM extending functionality: supporting more rules/flows.



## 4. CXL into the picture

Standardizing DRAM contents

- 1) Translating to HW intrinsics up to the HW no the Host (cpu cycles).
- 2) In a ideal P4 scriptable datapath, this should not be needed: how the data is represented defined by the P4 source.
- 3) The first CXL-based solutions would likely require translation at the Host level as part of the transition to the full solution (or just as an option supported or not by the SmartNIC).



## 4. CXL into the picture

SW datapath using same CXL-backed data.

- 1) CXL coherency makes this possible. Type 2: cpu reads implying cpu cache misses reading from CXL memory. Slower slow datapath?
- 2) Saving memory (Type 2): just updates to one memory and based on CPU writes.
- 3) A scriptable slow datapath promises same functionality in both, the software and hardware. Can the access to the CXL memory be “possible”? Cuckoo hashing? Multiple hashes? Dynamic size hashes?

## 5. Counters and openflow multitenancy

Orthogonality based on just Host CPU reads/writes brings new possibilities.

- 1) Counters, which are hard to cope with when thousands/millions of entries, can be read on demand instead of sending updates periodically. Coherency helps here ... where it should be enforce at the time of reads. Granularity?
- 2) Users could read counters related to their specific rules/flows.
- 3) Users could have their own CXL-memory range ...

## 5. Counters and openflow multitenancy

Openflow and Multitenancy:

- Nowadays there is just one Openflow controller.
- It is the Host/cloud provider accessing the controller and updating rules/flows on behalf of the client.
- Multitenancy: each client owns his control path. Security, confidentiality. I do access my Openflow controller, and I do update my control path (CXL!).

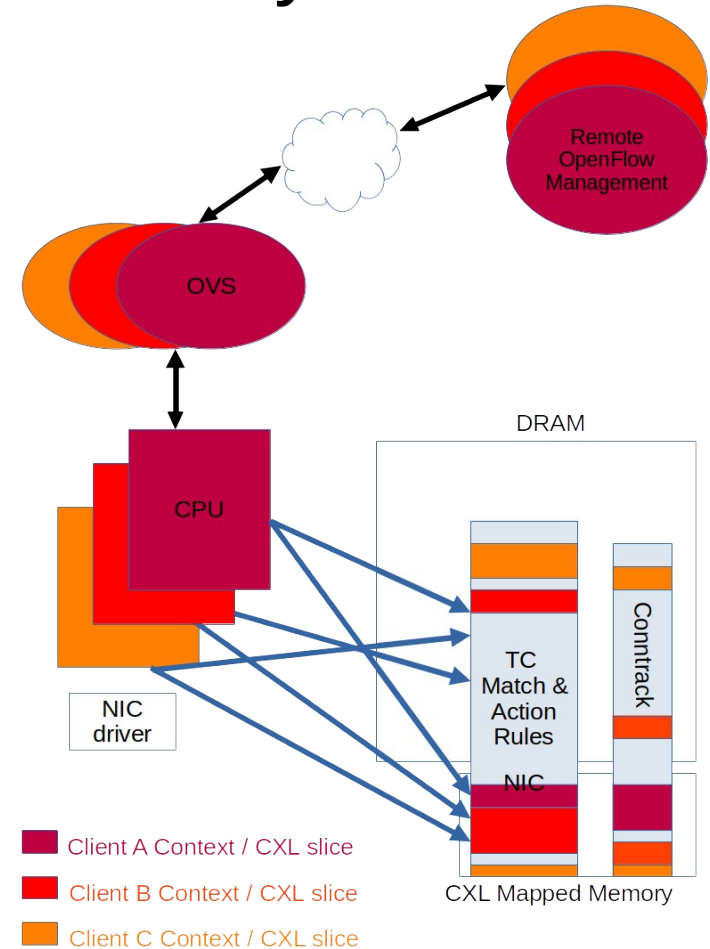
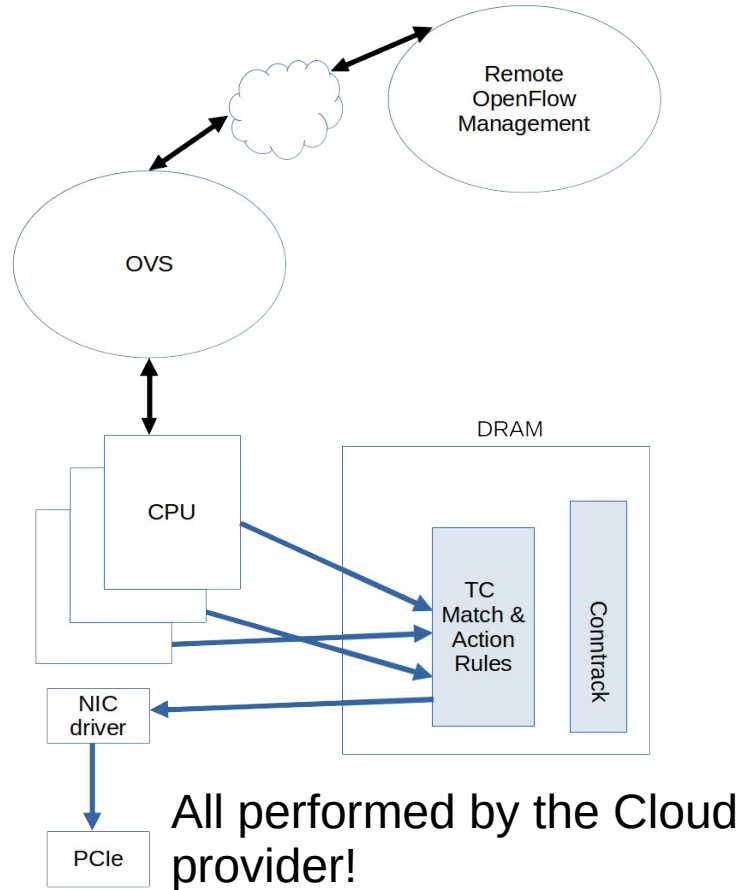
## 5. Counters and openflow multitenancy

### Openflow and Multitenancy: the Holistic Approach

- Switches and endpoints(SmartNICs) offering this control.
- Two levels: cloud provider and tenants.
- Cloud provider: here is your (virtual) network. This is the tunneling. This is your control path (CXL memory range).
- Hardware: CXL and SVE\*-like (Host and inside the SmartNIC).
- Tenant: None sees what I do. None can change my control path.
- QoS and counters could benefit from same orthogonality.

\* SVE (AMD), SGX (Intel), CCA (ARM)

# 5. Counters and openflow multitenancy



## 5. Counters and openflow multitenancy

Openflow and Multitenancy: the Linux way. How?

- SW datapath executed on behalf of a client (specific context). When?
- Cpu cycles and contention in those cpus allocated to the same client.
- CXL memory: only ranges owned by the client.
- Can we do that? Per client context inside the kernel? SVE applied to those kernel contexts? Userspace?
- Same contexts are needed inside the NIC
- ...
- No trivial, but possible. If it is really needed, CXL can help.

## 6. Conclusions

CXL is going to be disruptive.

Things easily done through CPU writes. Offloads!

It can help with the Control Path in massive virtualization and reinforce the scriptable software datapath approach.

It can help achieving the dream of full virtualization: multitenancy with security and confidentiality.

CXL, SmartNICs and OVS: a paradigm change?

Q & A