Turning PCIe Hints into Cache Hits: Enabling Smart Data Cache Injection in Linux

Wei Huang
Manoj Panicker

2025 Linux Plumber Conference

AMD
together we advance_

Overview

About AMD SDCI

I/O performance enhancement via smarter data cache injection

Linux Kernel Integration

Available for device drivers to improve I/O efficiency and performance

Performance Benefits

Lower latency, higher throughput, and improved memory bandwidth

Community Collaboration

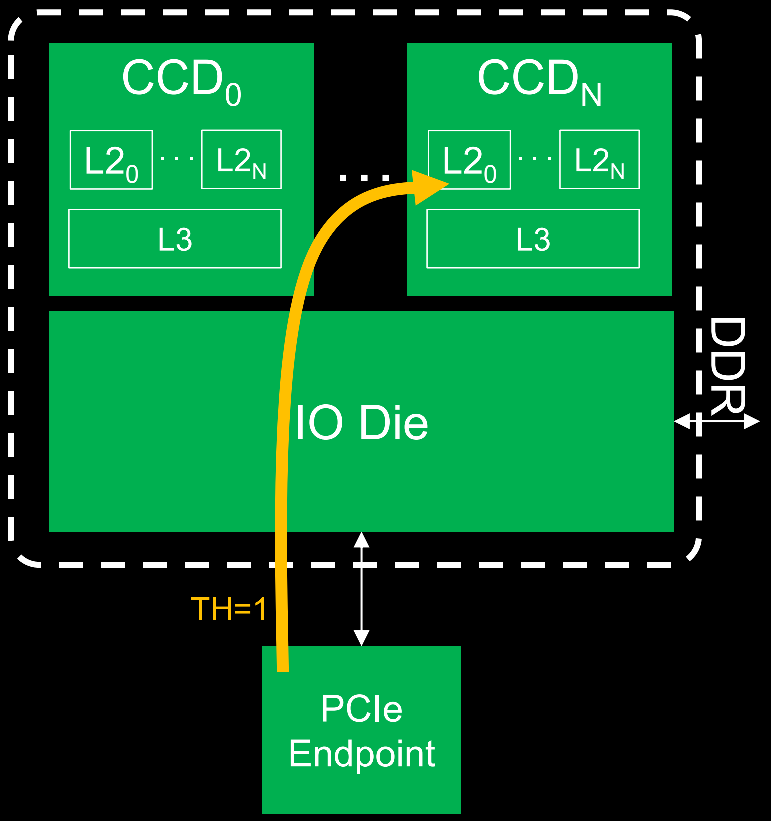
More vendor and device drivers support plus new features

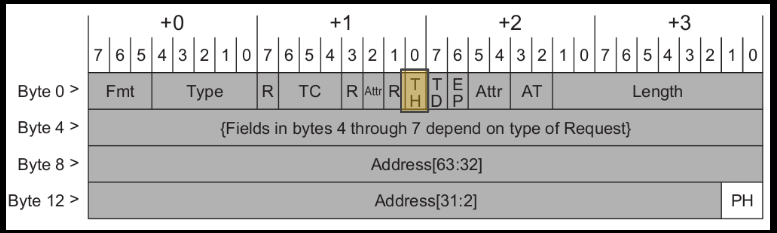
Background

- Fast delivery of I/O data into caches is key to improving processor performance.
- Existing cache injection technologies typically target at the last-level cache.
- Emerging chiplet-based, scale-out trends require a flexible solution.
  - Extensible with complex cache layouts
  - Allow endpoints to decide what traffic to inject
  - Standards-based approach
- Design goals
  - Open, vendor-neutral, flexible, and extensible

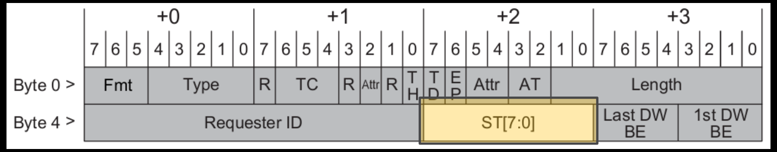
AMD's Smart Data Cache Injection (SDCI)

- Design
  - Based on industry standard: PCIe TLP Processing Hints (TPH) feature
  - Inbound I/O DMA write data is injected into the L2 cache directly
- Data flow
  - Host and endpoints must both support the TPH feature
  - Endpoint driver and FW chooses which DMA data for cache injection
  - The driver leverages OS ACPI _DSM interface to retrieve ST.
  - Vendor's IO Die decides the cache line placement policy accordingly.





<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0 ></td><td colspan="3">Fmt</td><td colspan="5">Type</td><td>R</td><td colspan="3">TC</td><td>R</td><td>Attr</td><td>R</td><td>TH</td><td>TD</td><td>EP</td><td colspan="2">Attr</td><td colspan="2">AT</td><td colspan="10">Length</td></tr>
<tr><td>Byte 4 ></td><td colspan="32">{Fields in bytes 4 through 7 depend on type of Request}</td></tr>
<tr><td>Byte 8 ></td><td colspan="32">Address[63:32]</td></tr>
<tr><td>Byte 12 ></td><td colspan="30">Address[31:2]</td><td colspan="2">PH</td></tr>
</table>

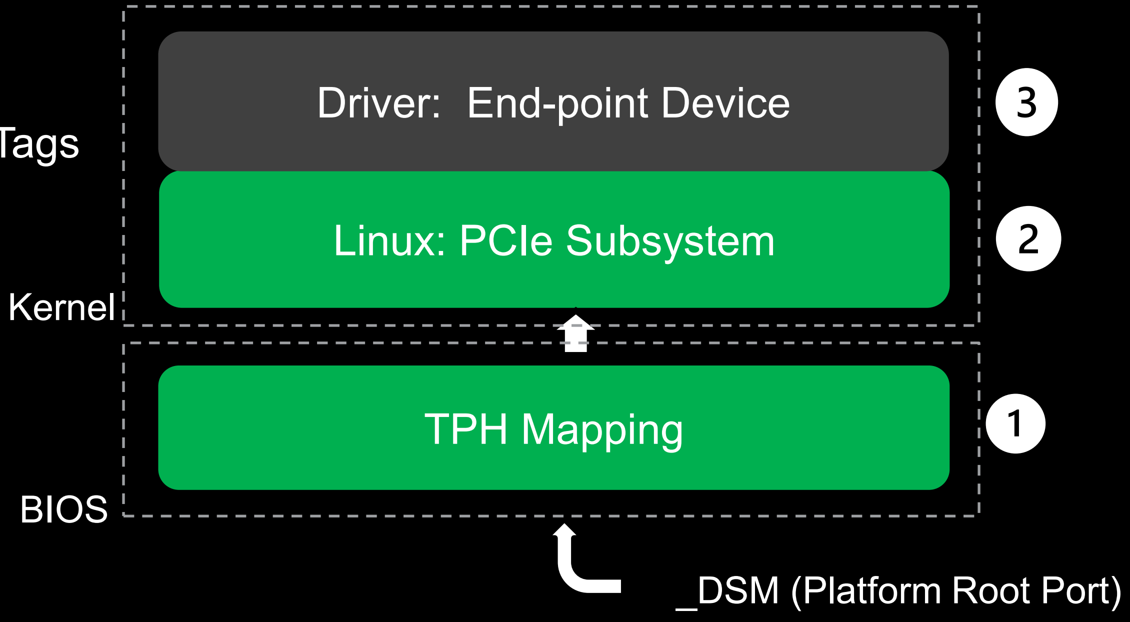


<table>
<tr><td></td><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td></td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>Byte 0 ></td><td colspan="3">Fmt</td><td colspan="5">Type</td><td>R</td><td colspan="3">TC</td><td>R</td><td>Attr</td><td>R</td><td>TH</td><td>TD</td><td>EP</td><td colspan="2">Attr</td><td colspan="2">AT</td><td colspan="10">Length</td></tr>
<tr><td>Byte 4 ></td><td colspan="16">Requester ID</td><td colspan="8">ST[7:0]</td><td colspan="4">Last DW BE</td><td colspan="4">1st DW BE</td></tr>
</table>

PCIe Express TLP Headers

Software Stack

1. BIOS: TPH mapping
   - Provide SoC specific mapping information: CPU UIDs ==> ST Tags
2. Linux: Generic PCIe TPH support
   - TPH capability detection and configuration
   - ST table configuration and update
3. Driver: What is injected?
   - Parsing TPH ST mapping for device specific setup



AMD

Endpoint Vendor

AMD
together we advance_

Linux TPH Core API

1. End-point device's TPH extended capability will be probed by Linux by default.

2. TPH enable and disable functions
   - `pcie_enable_tph()` activates TPH for devices in various modes
   - `pcie_disable_tph()` disables TPH and clears related bits

3. Steering tag management
   - `pcie_tph_get_cpu_st()` retrieves CPU-optimized ST values via ACPI
   - `pcie_tph_set_st_entry()` programs these tags into steering tables

Kernel support status

- Full SDCI Support in Linux since **Kernel version 6.13**
- Configuration and control support is enabled via CONFIG_PCIE_TPH
- TPH support can be disabled globally using the pci=notph in kernel boot parameter

Sample Driver - BNXT

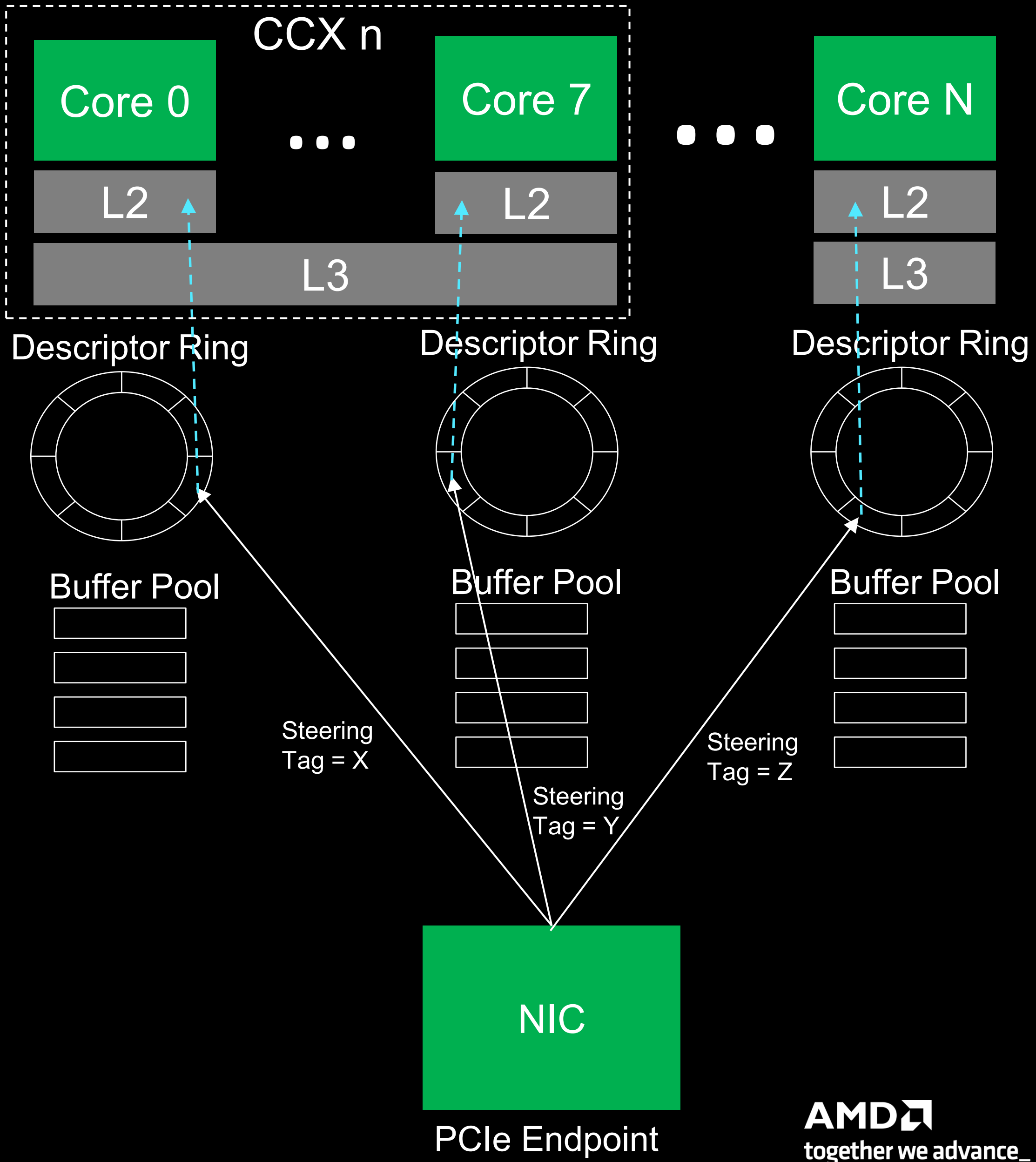
TPH support enablement flow in bnxt driver:

1. Enable per-device TPH during IRQ setup
   - Calls `pcie_enable_tph(PCI_TPH_ST_IV_MODE)`.
   - Each MSI-X vector associates a CPU mask that will drive ST selection.

2. Initialize STs for receive queues
   - Queries `pcie_tph_get_cpu_st()` with the chosen CPU ID
   - Writes ST into MSI-X entry via `pcie_tph_set_st_entry()`

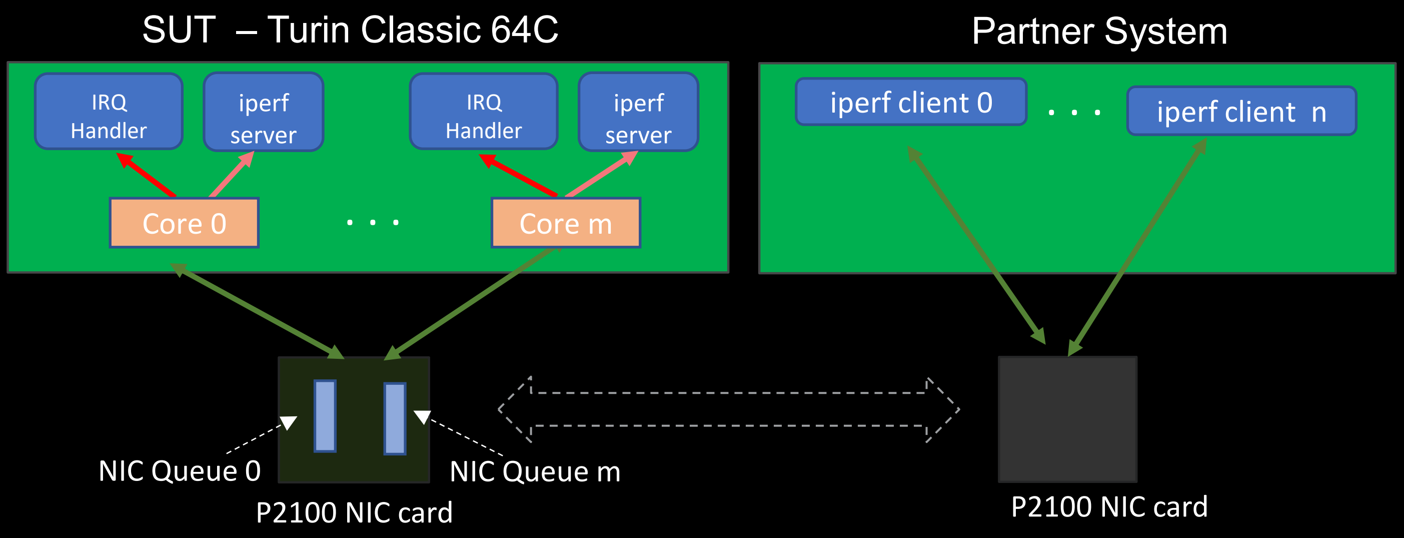
3. Track affinity changes via `bnxt_irq_affinity_notify()`
   - Reacts to user or kernel affinity updates
   - Reprograms MSI-X table and restarts the affected RX queue

4. Tear down
   - `bnxt_free_irq()` disables TPH for the device.



Perf Study – Memory BW Saving

- Configuration
  - AMD Turin 64-core, DDR 5600 * 12 channels
  - Broadcom P2100 100G NICs
  - Linux kernel 6.15.2



DRAM BW Utilization	Unidirectional 1 Queue		
	SDCI Off	SDCI On	% Change
UMC est read BW (GB/s)	11.54	1.68	-85%
UMC est write BW (GB/s)	10.37	3.6	-65%
Total R/W (GB/s)	21.91	5.28	-76%

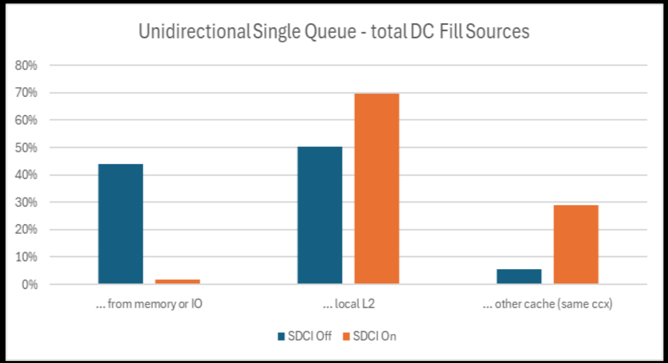
1 Queue: Only Queue 0 of NIC enabled

DRAM BW Utilization1	Unidirectional 4 Queues		
	SDCI Off	SDCI On	% Change
UMC est read BW (GB/s)	13.87	4.69	-66%
UMC est write BW (GB/s)	12.50	6.52	-48%
Total R/W (GB/s)	26.37	11.21	-57%

4 Queues: Queue 0-3 of NIC enabled

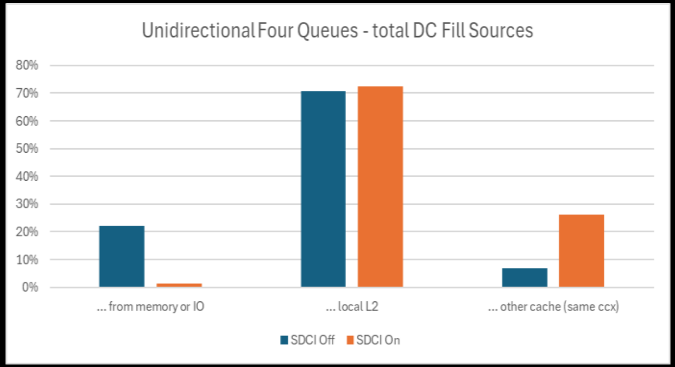
Perf Study – Memory BW Saving

- The source of memory bandwidth saving
  - The data cache fill requests are satisfied from local caches when SDCI is turned ON.
  - A steep drop in access to local memory is seen with both the single queue and multi-queue cases.



	Unidirectional 1 Queue					
For Cores running ->	irq	iperf	Combined	irq	iperf	Combined
DC Fill %	SDCI Off			SDCI On		
... from memory or IO	2%	51%	44%	1%	2%	2%
... local L2	87%	44%	50%	89%	66%	70%
... other cache (same ccx)	11%	5%	6%	11%	32%	29%

1 Queue

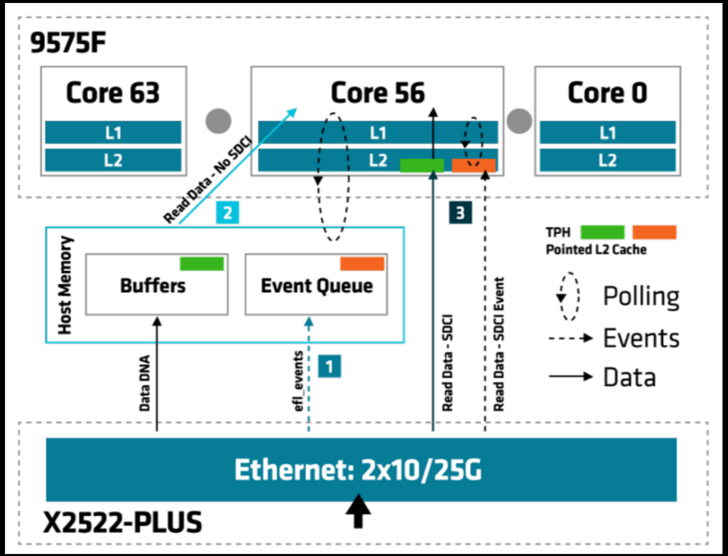


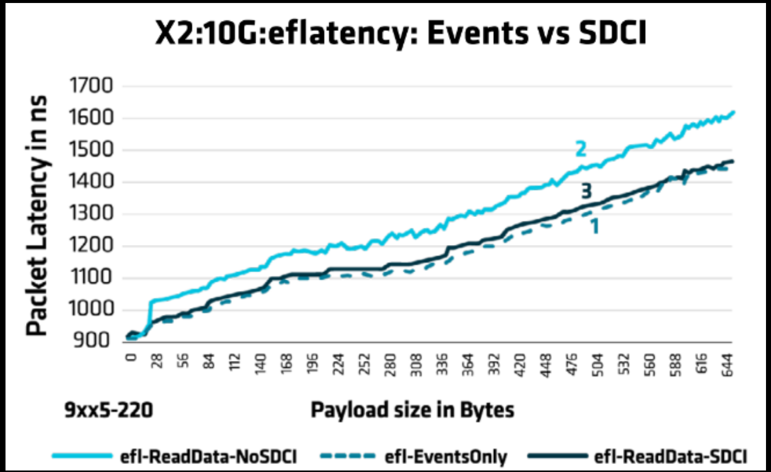
	Unidirectional 4 Queues					
For Cores running ->	irq	iperf	Combined	irq	iperf	Combined
DC Fill %	SDCI Off			SDCI On		
... from memory or IO	2%	28%	22%	1%	1%	1%
... local L2	83%	67%	71%	81%	70%	72%
... other cache (same ccx)	15%	5%	7%	18%	28%	26%

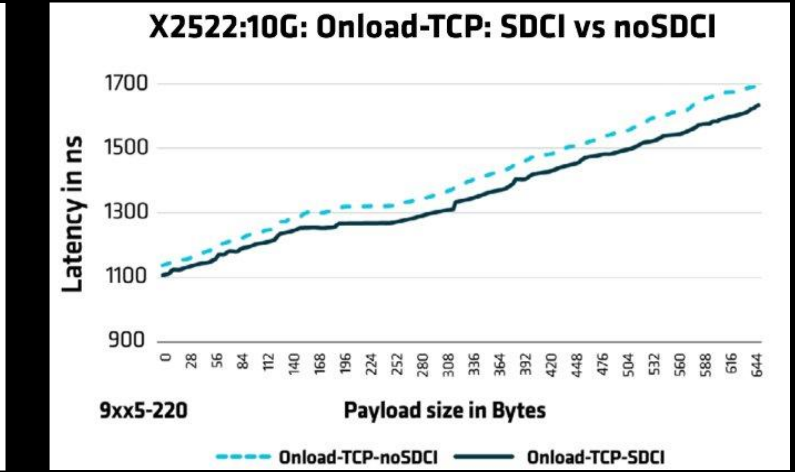
4 Queues

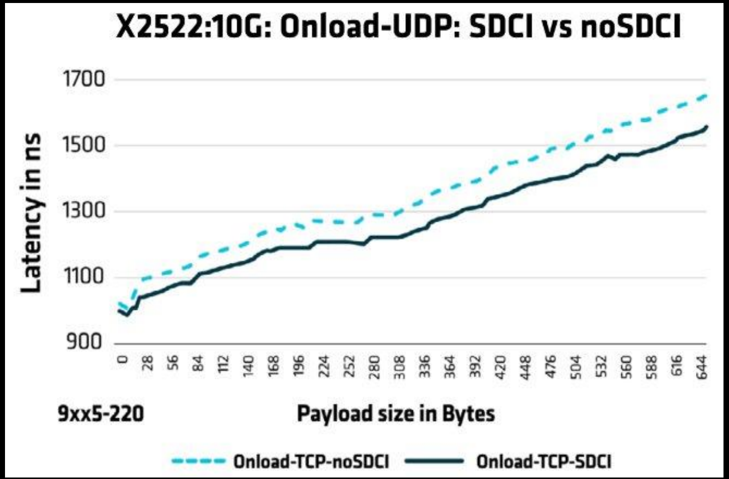
Perf Study – Latency Improvement

- Configuration
  - AMD EPYC™ Turin 9575F (64C), Ubuntu 25.04, kernel 6.14
  - AMD Solarflare X2522 NIC, SMT disabled, BIOS tuned for low latency
  - Benchmarks
    - eflatency benchmark
    - OpenOnload™ with TCP/UDP




X2:10G:eflatency: Events vs SDCI


X2522:10G: Onload-TCP: SDCI vs noSDCI


X2522:10G: Onload-UDP: SDCI vs noSDCI

Call for Collaboration

- Current Status
  - AMD SDCI was introduced in Zen5.
  - TPH support was in kernel 6.13 & Broadcom driver’s TPH support was introduced in 6.15.
  - AMD SDCI QoS (resctrl) was enabled in 6.18.

- Work in progress
  - TPH virtualization support

- Collaboration ideas
  - Enable TPH support in other vendor platforms
  - Identify and add TPH support to more device drivers
  - Enhance driver-level heuristics for dynamic cache injection
  - Explore integration with other frameworks