Optimizing Dataplane OS through Intra-host Network Analysis

Abstract

As datacenter networks scale to multi-hundred gigabit speeds, understanding the *intra-host* network (*e.g.*, the data path between the NIC and CPU/memory) becomes increasingly critical. While significant attention has been given to optimizations of network hardware and system software (*e.g.* kernel bypassing OS), potential opportunities of OS optimizations according to complex interactions within the host's I/O subsystem remain poorly understood.

This paper presents a systematic analysis of intra-host network bottlenecks in high-speed packet processing using DPDK. We dissect the Integrated I/O (IIO) stack, including the PCIe Root Complex, IOMMU (Intel VT-d), IOTLB, and Data Direct I/O (DDIO), to understand how these components interact and where performance bottlenecks arise. Through detailed measurements combining Intel PCM counters, NIC firmware telemetry, and DPDK profiling, we identify critical bottlenecks for I/O intensive workloads.

Our analysis provides actionable insights for optimizing CPU-based packet processing and lays the groundwork for understanding similar bottlenecks in more heterogeneous systems for ML workloads. As modern ML workloads increasingly rely on high-bandwidth communication between GPUs, NICs, and storage devices within a single host, the intra-host network analysis methodology we develop here becomes essential. Our future work aims to extend this characterization to GPU hosts, where PCIe/NVLink interconnects and GPU memory hierarchies introduce additional complexity to the intra-host data path.

1 Introduction

Reference Documents.

- **Intel Virtualization Technology for Directed I/O** [7]: Defines the hardware architecture and system software interface for Intel VT-d technology, which supports I/O device virtualization, such as DMA or interrupt remapping (*i.e.*, hardware-level address/ID translation), etc., on Intel platforms.
- **Intel Data Direct I/O Technology (Intel DDIO): A Primer** [4]: Explains how Intel DDIO enables I/O devices to read from and write directly to the CPU's Last Level Cache (LLC), bypassing main memory to reduce latency and improve system performance and power efficiency.
- **Intel Xeon Processor Scalable Memory Family Uncore Performance Monitoring** [5]: Defines the architecture, events, and control registers for Performance Monitoring Units (PMUs) in the uncore domain (memory controller, IIO, interconnect, etc.) of Intel Xeon Scalable processors.
- **Utilizing the Intel Xeon Processor Scalable Family IIO Performance Monitoring Events** [6]: Explains IIO (Integrated I/O) traffic flow in Intel Xeon Scalable processors and describes how to use performance monitoring events (PerfMon) to analyze I/O bandwidth and performance.
- **Intel Xeon Processor Datasheet Volume Two: Registers**: Documents uncore Configuration Space Registers (CSRs), IIO MMIO registers, and core MSRs (Machine Specific Registers).
  - Intel Xeon Processor Scalable Family (1st Gen Skylake-SP, 2nd Gen Cascade Lake) [3]
  - 3rd Gen Intel Xeon Scalable Processor (Ice Lake) [2]

2 background

Throughout this section, we use Ⓛ to denote logical/abstract concepts and Ⓗ to denote physical hardware units.

2.1 IIO Stack

This section describes the Integrated I/O (IIO) stack based on [3,6,7]. Figure 1 illustrates the flow of a DMA request from

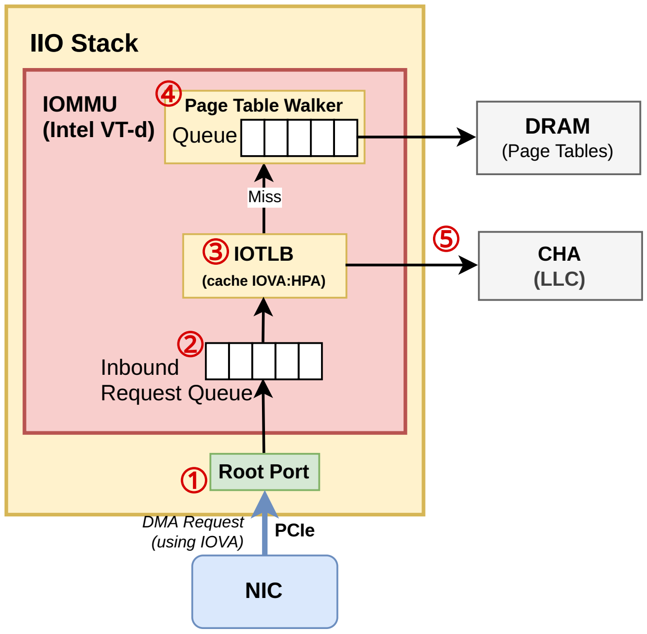

Figure 1: DMA request flow through the IIO stack. ① The NIC initiates a DMA request using an IOVA, which traverses the PCIe link to the Root Port inside the IIO. ② The request enters the Inbound Request Queue of IOMMU (VT-d engine). ③ The IOMMU checks the IOTLB for address translation (IOVA to HPA). ④ On a IOTLB miss, the request enters the Page Table Walker (PTW) Queue; the PTW then reads the page tables from DRAM (incurring latency overhead) and caches the result in the IOTLB. ⑤ On a hit (or after ④), the request reaches the target CHA determined by address hashing.

the NIC through the IIO stack. Figure 2 illustrates system architecture of our testbed nodes (node5 and node6).

2.1.1 PCIe Hierarchy.

The **Root Complex (RC)** is the logical root of the PCIe hierarchy (Ⓛ), connecting the processor and system memory to the PCIe bus, and is physically implemented within the IIO stack (Ⓗ). It translates system memory addresses for PCIe devices and vice versa, making PCIe devices appear local to the CPU. A **Root Port (RP)** Ⓗ is a specific port within the Root Complex that creates a PCIe link to downstream devices (*e.g.*, NICs, GPUs) or switches. The **Integrated I/O (IIO)** Ⓗ is a block of circuitry on the CPU that integrates the Root Complex, Root Ports, DMA engines, and other I/O controllers into a single unit, streamlining CPU-to-I/O communication.

2.1.2 IIO as Gateway.

The IIO stack serves as a bridge between PCIe lanes and the processor's internal mesh interconnect. For PCIe-based network devices such as NICs, the IIO stack acts as a gateway that manages all traffic between the device and the host system. When a NIC reads data from main memory for transmission (DMA Read) or writes received data to memory (DMA Write), the data flow must pass through the IIO stack.

2.1.3 IOMMUⒽ (Intel VT-dⓁ).

Intel VT-d (Virtualization Technology for Directed I/O) is Intel's architecture specification of I/O Memory Management Unit (IOMMU), integrated within the IIO stack alongside the PCIe root complex. The IOMMU intercepts DMA requests from I/O devices before they reach main memory, providing address translation and memory protection. When a device performs DMA, it uses I/O Virtual Addresses (IOVAs) rather than physical addresses. The IOMMU translates these IOVAs to Host Physical Addresses (HPAs) using multi-level page tables stored in DRAM, similar to CPU virtual memory. This DMA remapping enables devices to access non-contiguous physical memory as if it were contiguous, simplifying buffer management, while also providing memory isolation: each device (or group of devices) can have its own address space, preventing errant or malicious devices from accessing unauthorized memory regions.

2.1.4 IOTLBⒽ.

Since page tables reside in DRAM, each address translation would require multiple memory accesses to walk the page table hierarchy. To avoid this overhead, the IOMMU caches recent translations in the I/O Translation Lookaside Buffer (IOTLB), a SRAM structure inside the IOMMU. When a DMA request arrives, the IOMMU first checks the IOTLB. On a hit, the translation is returned immediately and the DMA proceeds.

2.1.5 Page Table WalkerⒽ.

On an IOTLB miss, the request is queued to the Page Table Walker (PTW) circuit block, which traverses the multi-level page table in DRAM to resolve the translation. Once complete, the result is cached in the IOTLB and the DMA request proceeds. High IOTLB miss rates can significantly degrade I/O performance, as each miss incurs multiple DRAM accesses and queueing delays in the Page Table Walker.

2.2 Intel DDIO (Data Direct I/O)

This section summarizes Intel's Data Direct I/O technology based on [4].

Traditional I/O architectures were designed assuming slow networking speeds and small caches, treating main memory as the primary destination and source for I/O data rather than the scarce cache. This forced frequent trips to memory for I/O data transfers, degrading system performance and increasing

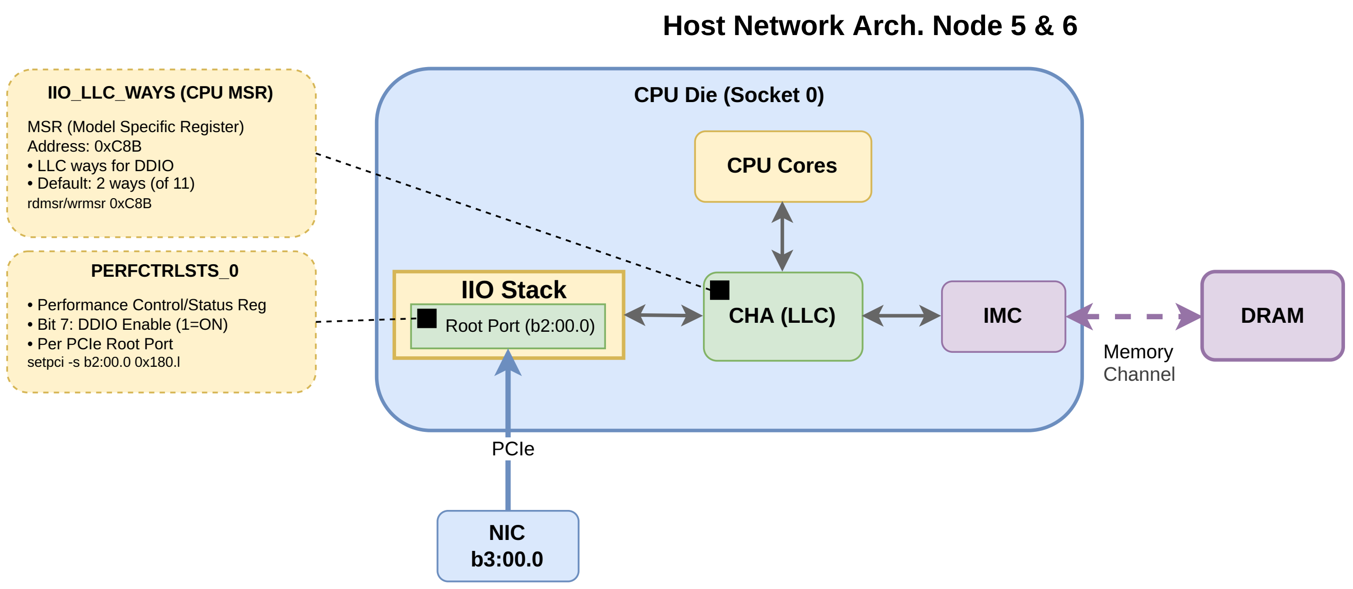

Figure 2: Node 5/6 architecture.

power consumption. Intel DDIO was introduced to leverage faster networking and larger caches more efficiently, enabling NICs to communicate directly with the host processor's LLC and drastically reducing memory access. As a result, DDIO increases system I/O bandwidth, reduces latency, and lowers power consumption, increasing overall system performance.

2.2.1 DMA Write (RX Path).

When a NIC receives a packet, it initiates DMA write operations to transfer the packet data and control structures (*e.g.*, descriptors) to host memory for protocol processing by the CPU. **Without DDIO**: Data is written directly to memory, invalidating any cached copy; when the CPU later reads the data for processing, it incurs a cache miss and must fetch from memory. **With DDIO**: Data is delivered directly to the LLC via *Write Update* (in-place update if address is already cached) or *Write Allocate* (new allocation if not cached), so subsequent CPU reads hit the cache. Write Allocate is limited to 10% of LLC capacity to prevent cache pollution from high-rate streams that exceed CPU processing speed; this is not reserved space—it remains fully available to CPU applications.

2.2.2 DMA Read (TX Path).

When a NIC transmits a packet, it initiates DMA read operations to fetch packet data and control structures (*e.g.*, descriptors) from host memory. The CPU first creates the packet data in its cache hierarchy, then notifies the NIC to begin transmission. **Without DDIO**: Each NIC read causes the data to be evicted from cache, so when the CPU later accesses the buffer (*e.g.*, to free it), it must fetch the data from memory again. **With DDIO**: The NIC read is directly served from cache *without eviction*, keeping the data available for software reuse and achieving zero memory trips in the ideal case. Note that if data is not found in cache, it is fetched from memory but *not allocated* into the cache hierarchy (unlike DMA writes).

2.2.3 Enabling/Disabling DDIO Write.

DDIO Write can be controlled through the `Use_Allocating_Flow_Wr` bit (bit 7) of `PERFCTRLSTS_0` register (offset: `180h`) in the PCIe Root Port [3]. For example, the register value on node6 can be checked with `$ setpci -s b2:00.0 0x180.l`). This register bit controls whether DMA writes use allocating flow to LLC (with DDIO) or non-allocating flows to DRAM (legacy mode without DDIO). In the non-allocating mode, if the data being delivered happens to be in the CPU's caching hierarchy, it is invalidated [4]. Importantly, this bit only affects write operations: setting it to 0 disables LLC allocation for writes, but read operations continue to use the cache-efficient DDIO behavior (no eviction on LLC hit). The "legacy mode" described in [4] (*i.e.*, eviction after read) refers to older architectures without DDIO, not the state when this register bit is unset. In other words, the DDIO disable switch is a half-switch that only controls write allocation; read-side benefits remain active as long as the processor supports DDIO.

2.2.4 Tuning LLC Occupancy.

Althought [4] mentions that Write Allocate operations are restricted to 10% of the LLC and its fixed (cannot be changed),

[1] (in Section 4.4) mentions that it can be adjusted via the `IIO LLC WAYS` MSR with the address of 0xC8B. The behavior may vary across different CPU architectures [8]. This needs to be further investigated.

2.2.5 CHA Processing Flow.

After the DMA request arrives at the target CHA (Caching Home Agent), the CHA performs coherency checks and cache operations differently depending on whether it is a DMA Write (RX) or DMA Read (TX). Figure 3 illustrates both flows.

3 Performance Monitorning Tools

pcm-pcie The `pcm-pcie` tool reports coherence lookup results at the CHA (Caching and Home Agent) for PCIe transactions: a *hit* indicates the target address already exists in LLC and requires coherence action, while a *miss* indicates the address is not in LLC and requires no coherence action. Note that it has nothing to do with DDIO hit/miss. To verify the impact of DDIO ON/OFF, we use `pcm-memory` to measure DRAM bandwidth: DDIO ON results in low DRAM write, while DDIO OFF causes high DRAM write.

4 Evaluation

Setup. All experiments use 64-byte packets and default DPDK configuration parameters unless otherwise noted.

4.1 TX Scaling Bottleneck Analysis

We analyze why TX throughput does not scale linearly with the number of cores without inline mode. Measurements were collected using: **(1)** `pcm-iio` for IIO stack counters (outstanding reads, IOTLB hit/miss), **(2)** NIC firmware parameters (`LOG_MAX_OUTSTANDING_WQE`, currently 128), and **(3)** IOMMU disable tests to isolate IOTLB overhead.

1 Core vs 2 Cores Comparison. Table 1 summarizes the key metrics. With 2 cores, TX rate improves by only 1.26× (not 2×), while PCIe latency increases by 52%.

Little's Law Analysis. Using Little's Law (Throughput = Outstanding Requests / Latency):

- **1 Core:** (15.99 Gb/s × 1627 ns)/(64 B × 8) ≈ 51 outstanding reads
- **2 Cores:** (19.99 Gb/s × 2468 ns)/(64 B × 8) ≈ 96 outstanding reads

While 2 cores issue ∼2× outstanding requests, latency also increases proportionally, limiting throughput gains.

Table 1: TX scaling metrics: 1 core vs 2 cores.

Metric	**1 Core**	**2 Cores**	**Ratio**
TX Rate			
TX Rate (Mpps)	19.05	23.90	1.26×
pcm-iio (IIO Stack)			
IB Read (GB/s)	1.48	1.85	1.25×
IOTLB Hit (M/s)	20.30	25.32	1.25×
IOTLB Miss (M/s)	11.25	14.13	1.26×
IOTLB Miss Rate	35.65%	35.82%	≈same
NeoHost (NIC Internal)			
PCIe Inbound BW (Gb/s)	15.99	19.99	+25%
PCIe Avg Latency (ns)	1627	2468	+52%
PCIe Stall (No Compl.)	0	0	–
MTT L0 Miss Rate	4.56%	6.70%	+47%

Key Findings.

1. **Not PCIe tag saturation**: PCIe Stall (No Completion) = 0, well below 256-tag limit.
2. **Bottleneck is PCIe latency increase**: 52% latency growth from 1→2 cores.
3. **IOTLB miss rate unchanged** (∼35.6%): per-request miss rate is core-independent, but total misses increase with request volume, causing IIO stack queueing delays.
4. **NIC-side MTT cache contention**: MTT L0 miss rate increases 47% (4.56%→6.70%).

Q: Why does IOTLB miss rate stay the same (∼35%) with more cores? IOTLB caches *page-level* translations (IOVA→HPA), not per-packet addresses. With DPDK's 2MB huge pages, one IOTLB entry covers thousands of 64B packets. Multiple cores sharing the same mempool access the same huge pages, so the working set of unique pages remains similar regardless of core count. The miss *rate* stays constant, but **total misses increase** (11.25→14.13 M/s), and these additional page table walks cause IIO stack queueing delays.

Conclusion. The shared bottleneck is the **IIO Stack** (PCIe Root Complex), not PCIe tags or IOTLB miss rate per se. The 35% IOTLB miss rate causes each DMA read to incur additional memory accesses for page table walks, creating queueing delays in the IIO stack that worsen as core count increases. Figure 5 shows the key metrics for 1, 2, 4, and 8 TX cores, and Figure 6 shows the full set of metrics.

4.2 DDIO Impact Analysis

Figure 7 shows the impact of enabling/disabling `Use_Allocating_Flow_Wr` bit (as discussed in §2.2.3) on both PKTGEN (TX) and L3FWD (RX) nodes.

(a) DMA Write (RX Path)

```
CHA receives DMA Write request
  |
  +-> Snoop Filter (SF) Lookup
  |
  +- [SF Hit] Core has line (M/E/S)
  |    |
  |    +-> Snoop Invalidate to core
  |    +-> Core invalidates line
  |    +-> Ack to CHA
  |    +-> (continue to SF Miss path)
  |
  +- [SF Miss] (or after invalidation)
       |
       +-> L3 Tag Array Lookup
       |
       +- DDIO Disabled:
       |    Invalidate if L3 hit
       |    Write to DRAM via IMC
       |
       +- DDIO Enabled:
            L3 hit -> Write Update
            L3 miss -> Write Allocate
```

(b) DMA Read (TX Path)

```
CHA receives DMA Read request
  |
  +-> Snoop Filter (SF) Lookup
  |
  +- [SF HitM] Core has modified line
  |    |
  |    +-> Snoop request to core
  |    +-> Core flushes L1/L2 to mesh
  |    +-> Forward: Data -> IIO -> NIC
  |
  +- [SF Miss] Core not modified
       |
       +-> L3 Tag Array Lookup
       |
       +- [L3 Hit]:
       |    Data from LLC -> IIO -> NIC
       |    (no eviction with DDIO)
       |
       +- [L3 Miss]:
            Fetch from DRAM via IMC
            Data -> IIO -> NIC
```

Figure 3: CHA processing flow for DMA Write (RX path) and DMA Read (TX path). Both paths begin with a Snoop Filter lookup for cache coherency, followed by L3 tag array lookup. DDIO affects the write allocation policy (a) and preserves cache contents on reads (b).

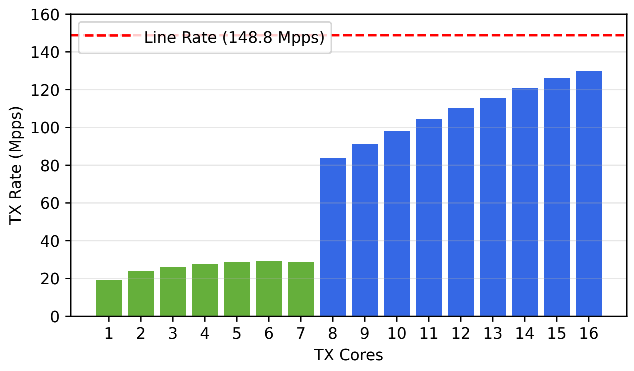

Figure 4: PKTGEN TX rate scaling with increasing TX cores on node5 using the default DPDK configuration.

Key observations:

- `pcm-pcie` results (PCIe Read/Write) are not affected.
- On PKTGEN (Pkt TX, DDIO Read), Both DRAM Read and Write are not affected.
- On L3FWD (Pkt RX, DDIO Write), Both DRAM Read and Write are significantly increased when DDIO Write is off, confirming that incoming packets bypass LLC and go directly to DRAM.

References

[1] Alireza Farshin, Amir Roozbeh, Gerald Q Maguire Jr, and Dejan Kostić. Reexamining direct cache access to optimize {I/O} intensive applications for multi-hundred-gigabit networks. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pages 673–689, 2020.

[2] Intel Corporation. 3rd gen intel xeon scalable processor, codename ice lake, datasheet, volume two: Registers. Datasheet.

[3] Intel Corporation. Intel xeon processor scalable family, datasheet, volume two: Registers. Datasheet.

[4] Intel Corporation. Intel data direct i/o technology (intel ddio): A primer. Technical brief, February 2012. Revision 1.0.

[5] Intel Corporation. Intel xeon processor scalable memory family uncore performance monitoring. Reference manual, 2017.

[6] Intel Corporation. Utilizing the intel xeon processor scalable family iio performance monitoring events. https://www.intel.com/content/www/us/en/developer/articles/technical/

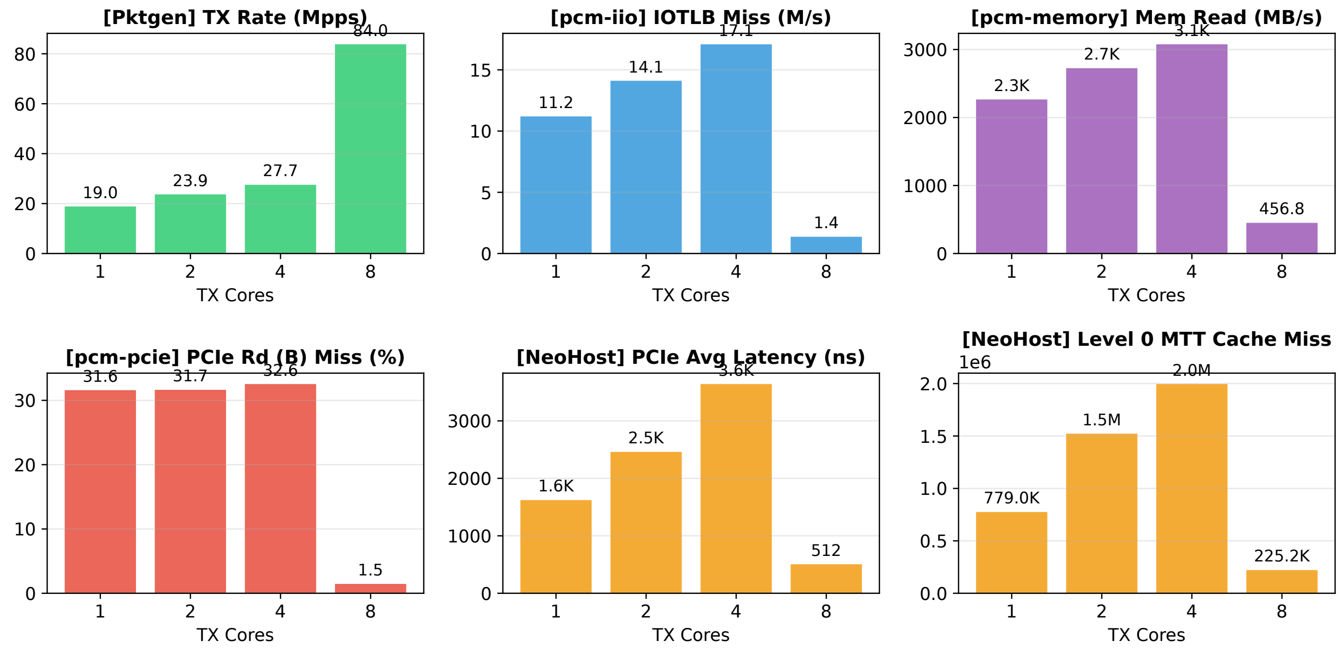

Figure 5: Key TX core scaling metrics (1, 2, 4, 8 cores): TX Rate, IOTLB Miss, Memory Read, PCIe Read Miss, PCIe Latency, and MTT Cache Miss.

utilizing-the-intel-xeon-processor-scalable-family-iio-performance-monitoring-events.html, 2019.

[7] Intel Corporation. Intel virtualization technology for directed i/o. Architecture specification, March 2023. Revision 4.1.

[8] Intel Support Community. Adjusting IIO_LLC_WAYS for increased ddio portion on intel xeon gold 6354. https://community.intel.com/t5/Software-Tuning-Performance/Adjusting-IIO-LLC-WAYS-for-Increased-DDIO-Portion-on-Intel-Xeon/m-p/1640393, 2024.

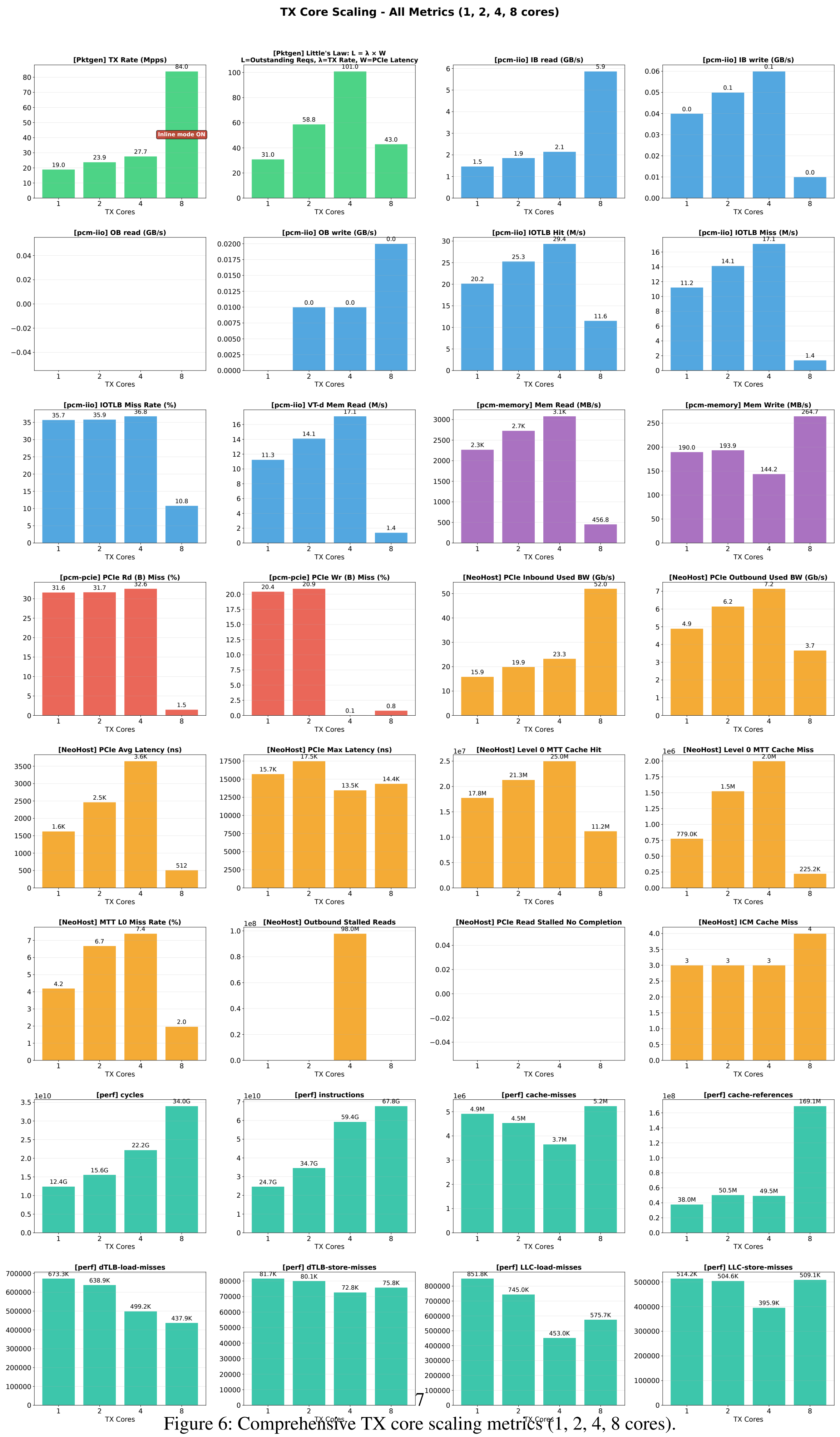

Figure 6: Comprehensive TX core scaling metrics (1, 2, 4, 8 cores).

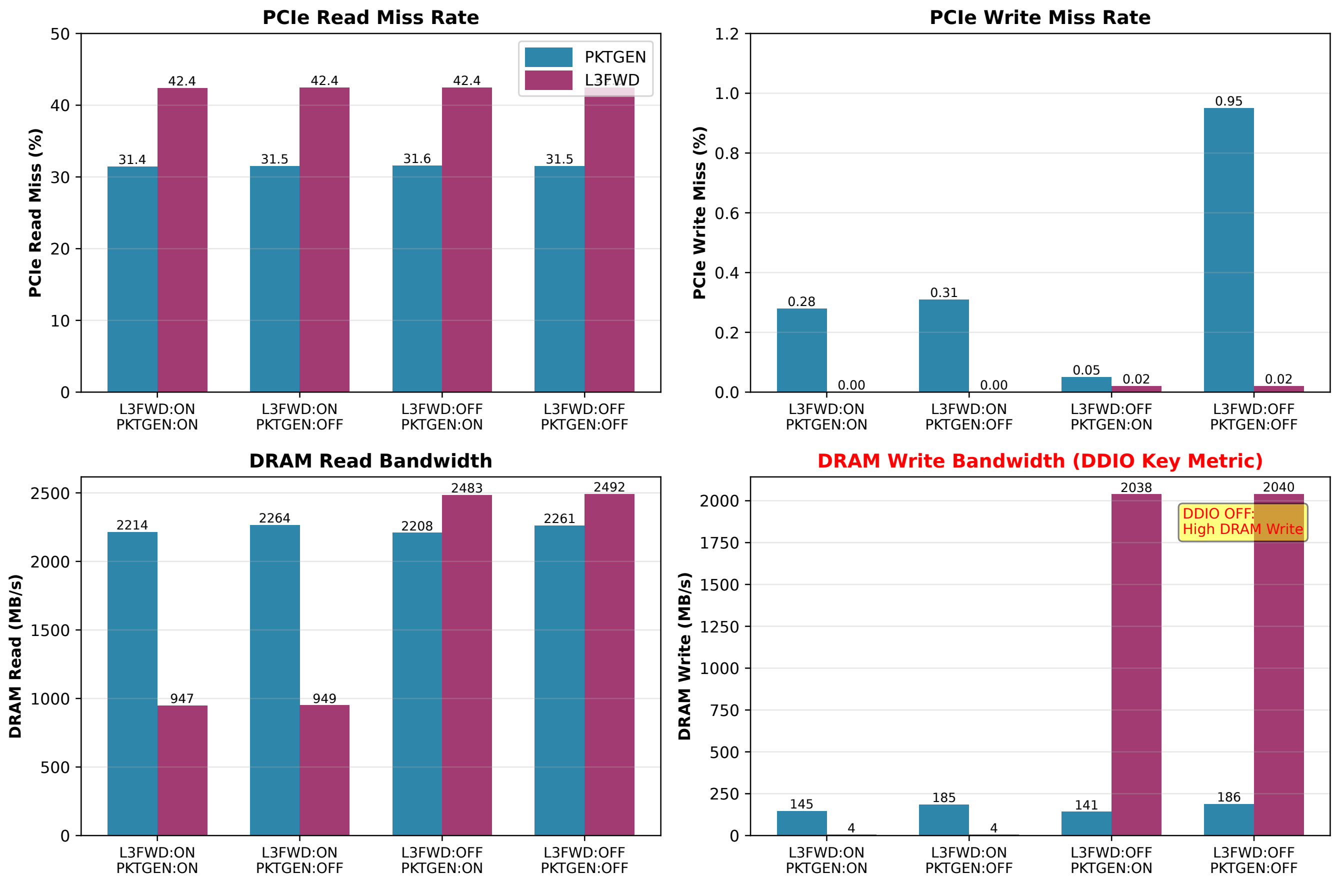

Figure 7: DDIO ON/OFF impact on pcm-pcie and pcm-memory metrics. DRAM Write bandwidth is the key indicator: when DDIO Write is OFF on the RX side (L3FWD), incoming packets bypass LLC and write directly to DRAM. PKTGEN: 1 TX core, 19 Mpps. L3FWD: 1 RX/TX core, 16 Mpps.