SDC20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Platform Performance Analysis for I/O-intensive Applications

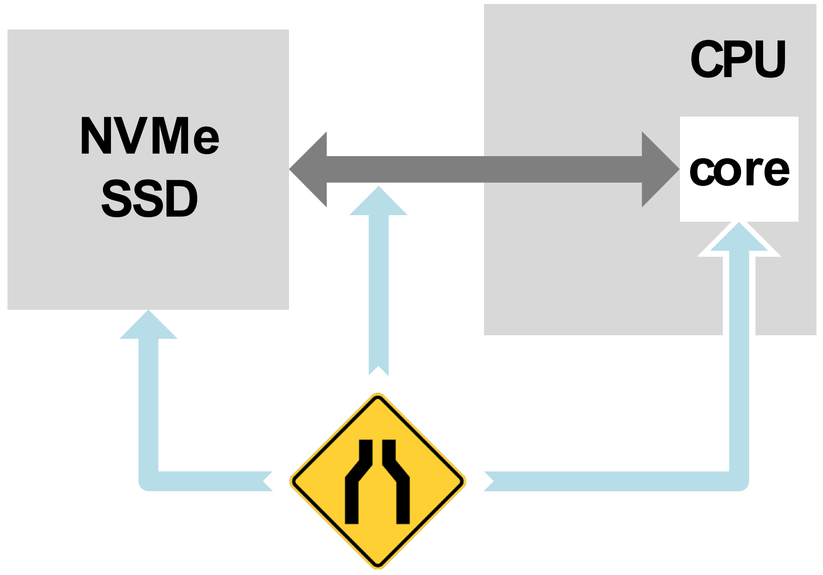
Ilia Kurakin | ilia.kurakin@intel.com
Perry Taylor | perry.taylor@intel.com
Intel Corporation

Agenda

- Introduction
- Architectural background
  - Intel® Xeon® Scalable Processor Overview
  - Intel® DDIO details
- Performance analysis
  - Platform-level observability
  - Intel® VTune™ Profiler: Input and Output analysis
  - Methodology – Directions

IO-intensive Apps Performance Bottlenecks

Domain	Performance is limited by…	How to detect and address
I/O device bound	… device capabilities	Refer to datasheet
Core bound	… algorithmic or microarchitectural code issues	Core-centric analyses (hotspots, uarch exploration, threading, Intel® Processor Trace - based, …)
Transfer bound	**… non-optimal interactions between devices and CPU**	**Growing "uncore"-centric analyses**



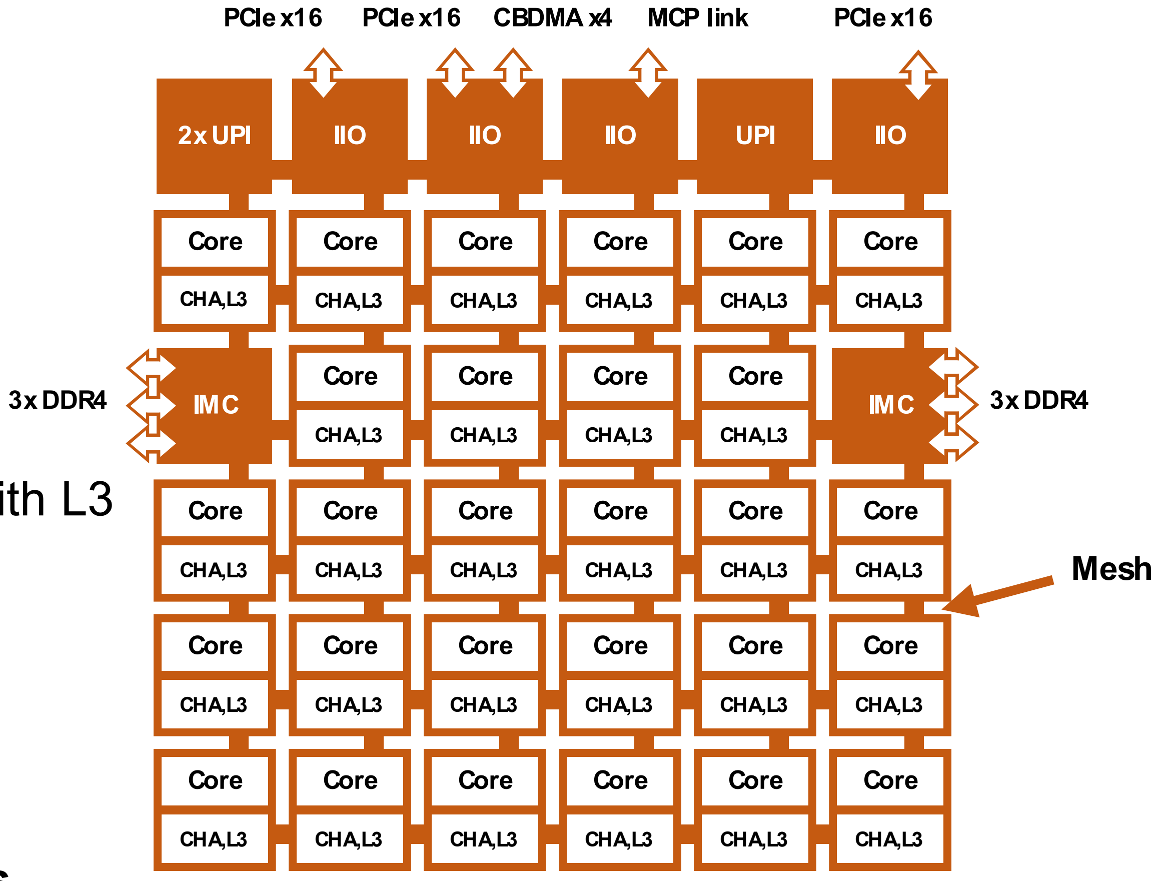
This presentation focuses on the latter domain, which introduces most challenging issues weakly covered with easy-to-follow methodologies

Architectural Background

Intel® Xeon® Scalable Processor Overview

SoC compound

- **Mesh** that interconnects:
  - **Cores**
    = execution units + L1 cache + L2 cache
  - **Uncore units**
    - Slices of shared L3 cache (**LLC/SF**) with L3 cache controller (**CHA**)
    - Integrated memory controllers (**IMC**)
    - Intel® Ultra Path Interconnect (**UPI**) controllers
    - Integrated I/O (**IIO**) controllers
      – **interfaces to PCIe devices**



Any interaction between PCIe device and system
is handled by IIO and other uncore units standing on IO path

Integrated I/O Controllers (IIO)

Intel® Xeon® Scalable Processors (1st and 2nd gen) incorporate 5 IIO units:

- 3 units covering 48 PCIe Gen3 lanes (x16 each)
- 1 unit servicing DMI interface and CBDMA
- 1 unit servicing MCP (multichip package) link

IIO connects strictly ordered PCIe domain to the out-of-order mesh:

- Data is transferred in TLP payloads over PCIe and then translated by IIO into cache line (64B) requests.

IIO translates TLPs to cache line requests and vice versa.
These activities might be induced by both CPU and PCIe devices.

IIO Transactions for Core/Device Communication

Inbound transactions
initiated by I/O device, target system memory

- Inbound read = I/O device reads the system memory
- Inbound write = I/O device writes the system memory

driven by
Intel® Direct Data I/O
hardware technology

Outbound transactions
initiated by cores, target I/O device memory

- Outbound read = core reads the memory of I/O device
- Outbound write = core writes the memory of I/O device

typically done by
Memory-Mapped I/O
address space accesses

Though Intel® DDIO is transparent to SW,
there are pitfalls that may lead to suboptimal performance.

Intel® Data Direct I/O (Intel® DDIO) Details [1/2]

The inbound transactions are routed directly to the local L3 cache:

- **Inbound reads** are processed without L3 cache allocation
- **Inbound writes** require a related cache line to be allocated in the L3 and get processed in two phases:

Inbound Write Phase	**Details**
1. Get cache line ownership for IIO	Cache line location is tracked through L3 line, therefore L3 allocation is required.
2. IIO delivers modified data to the L3, releases ownership	This phase is done in different ways depending on chosen config: a. Allocating – data goes to the LLC b. Non-allocating – data goes to the DRAM

Inbound requests for data lead to L3 cache lookup resulting in L3 hit or miss scenarios.

Intel® Data Direct I/O (Intel® DDIO) Details [2/2]

Following rules apply when platform processes inbound PCIe read and write:

Request	L3 Lookup	Implication
Inbound Read	Hit (good)	The data is read from L3 and sent to the PCIe device
	Miss (bad)	The data is read from the local DRAM or from the remote socket's memory subsystem and sent to the PCIe device
Inbound Write	Hit (good)	The cache line is overwritten with the new data
	Miss (bad)	Some cache line is first evicted. Then, in place of the evicted line, a new cache line is allocated. If the targeted cache line is used remotely, cross-socket accesses are required. Finally, the cache line is updated with the new data.

"DDIO misses" should be avoided for best latency/throughput and not wasting DRAM/UPI traffic and platform power

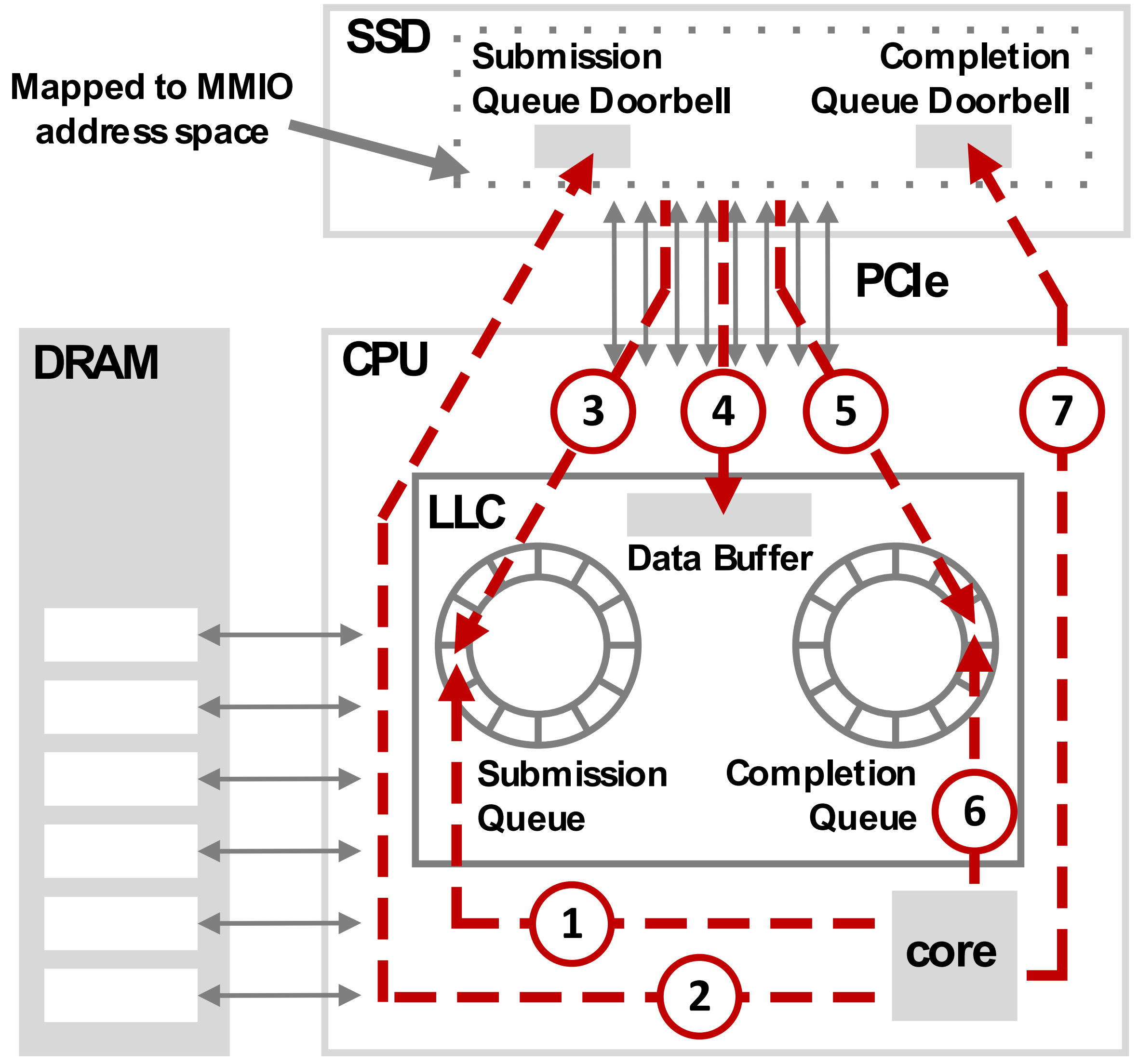
Memory-Mapped I/O (MMIO) Accesses

- **MMIO** access is a primary mechanism for performing **outbound PCIe** transactions
- MMIO access are quite expensive and should be limited:

Core Operation →	IIO Transaction	Cost
MMIO Read	Outbound PCIe Read	Most expensive I/O-related transaction from core perspective, since completion requires round trip to device.
MMIO Write	Outbound PCIe Write	Less costly transaction, but core still needs to get an acknowledge.

Avoid MMIO reads and use tricks to minimize MMIO writes on the data path.

IIO Flows Utilization in Storage Apps



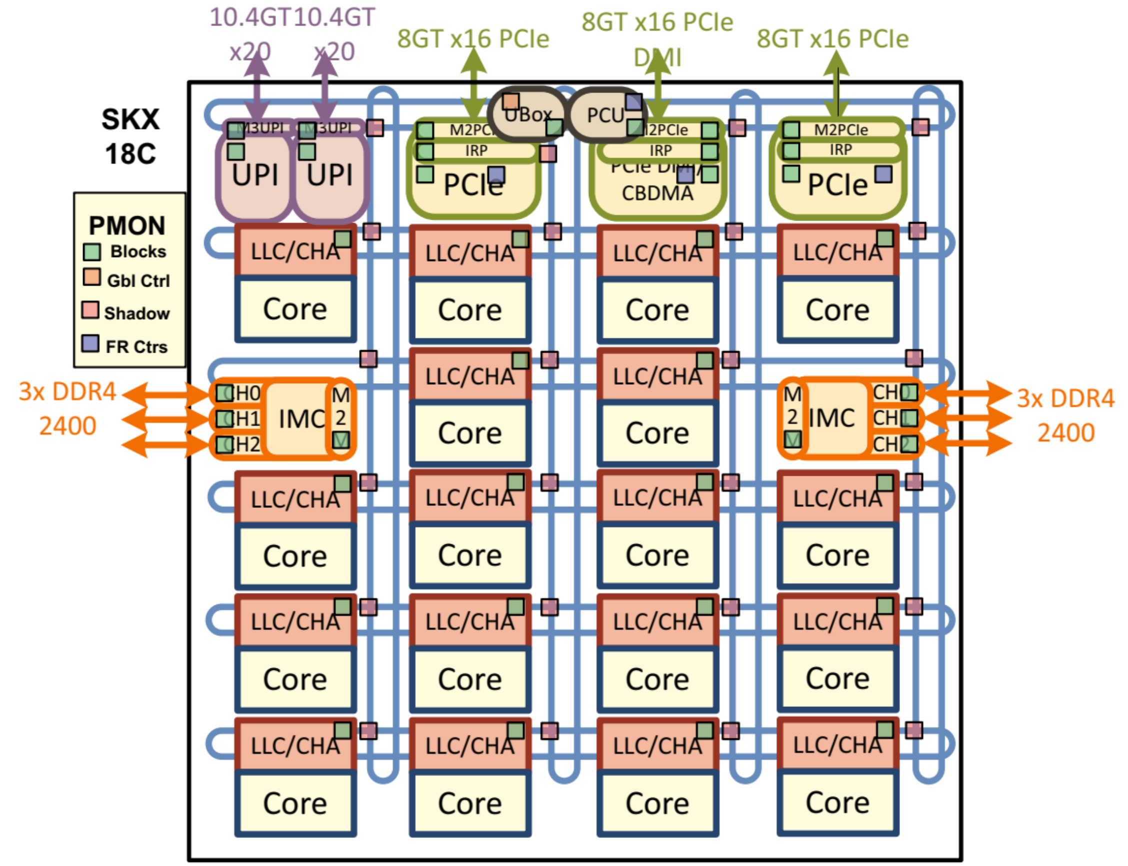
Mapped to MMIO address space
SSD
Submission Queue Doorbell
Completion Queue Doorbell
PCIe
DRAM
CPU
LLC
Data Buffer
Submission Queue
Completion Queue
core
1
2
3
4
5
6
7

Example: app reads from SSD

1. Core writes I/O command descriptor and starts polling completion queue element
2. Core notifies SSD that new descriptor is available (**Outbound PCIe Write**)
3. Device reads descriptor to get buffer address (**Inbound PCIe Read**)
4. Device writes I/O data (**Inbound PCIe Write**)
5. Device writes to the completion queue (**Inbound PCIe Write**)
6. Core detects that completion is updated
7. Core moves completion queue tail pointer (**Outbound PCIe Write**)

Performance Analysis

Platform-Level Observability

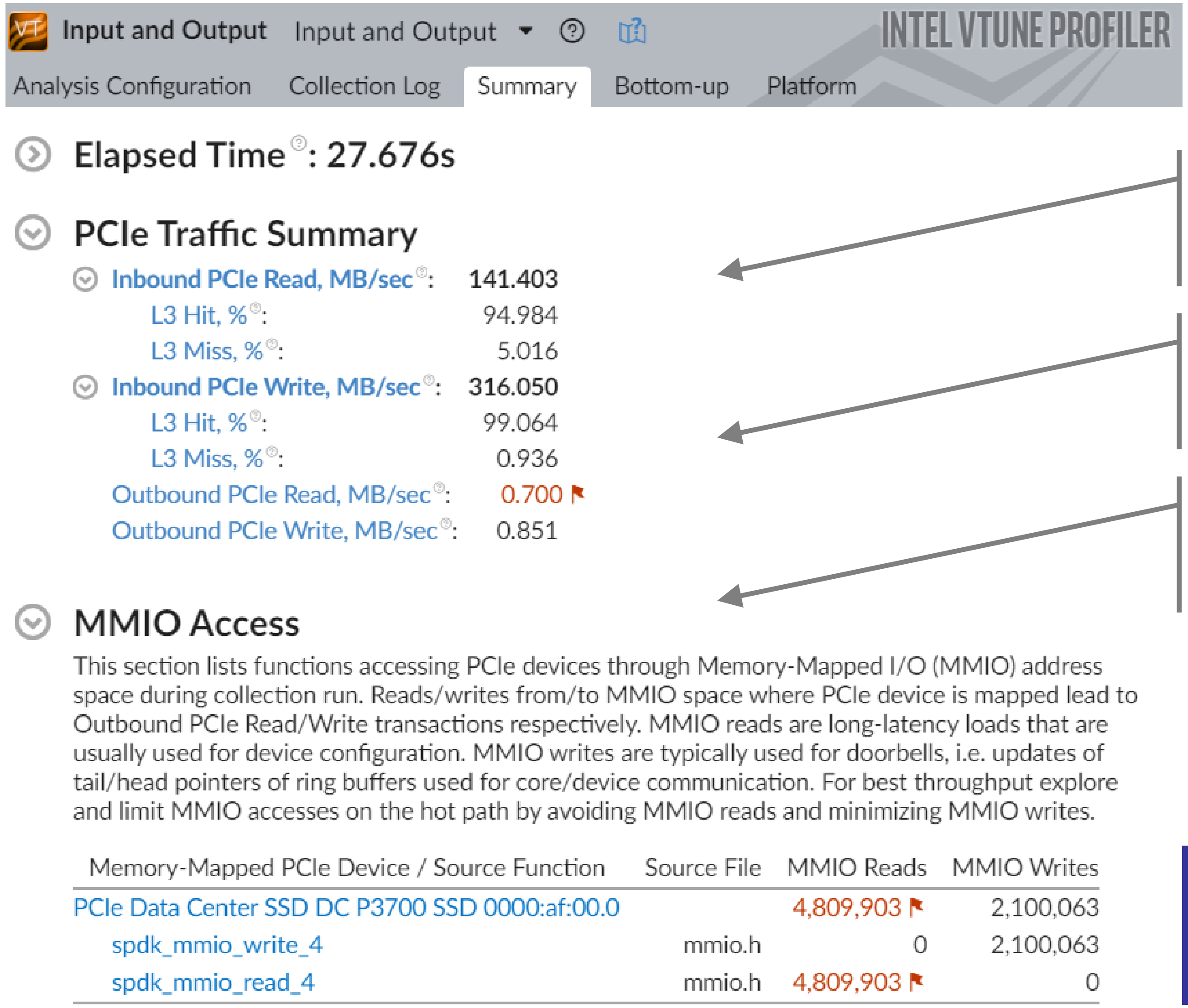


Thousands of uncore performance monitoring events incorporated in uncore Performance Monitoring Units (PMUs)

- IIO: inbound / outbound read / write bandwidth
- IRP: coherency-related IIO operations
- CHA: mesh and L3 cache controller
- IMC and M2M: memory bandwidth, memory directory access
- UPI: cross-socket bandwidth

Using raw events for performance analysis requires deep knowledge of hardware and appears a challenging task

Intel® VTune™ Profiler: Input and Output Analysis



Elapsed Time: 27.676s

PCIe Traffic Summary

Inbound PCIe Read, MB/sec:	141.403
L3 Hit, %:	94.984
L3 Miss, %:	5.016
Inbound PCIe Write, MB/sec:	316.050
L3 Hit, %:	99.064
L3 Miss, %:	0.936
Outbound PCIe Read, MB/sec:	0.700 ⚑
Outbound PCIe Write, MB/sec:	0.851

MMIO Access

This section lists functions accessing PCIe devices through Memory-Mapped I/O (MMIO) address space during collection run. Reads/writes from/to MMIO space where PCIe device is mapped lead to Outbound PCIe Read/Write transactions respectively. MMIO reads are long-latency loads that are usually used for device configuration. MMIO writes are typically used for doorbells, i.e. updates of tail/head pointers of ring buffers used for core/device communication. For best throughput explore and limit MMIO accesses on the hot path by avoiding MMIO reads and minimizing MMIO writes.

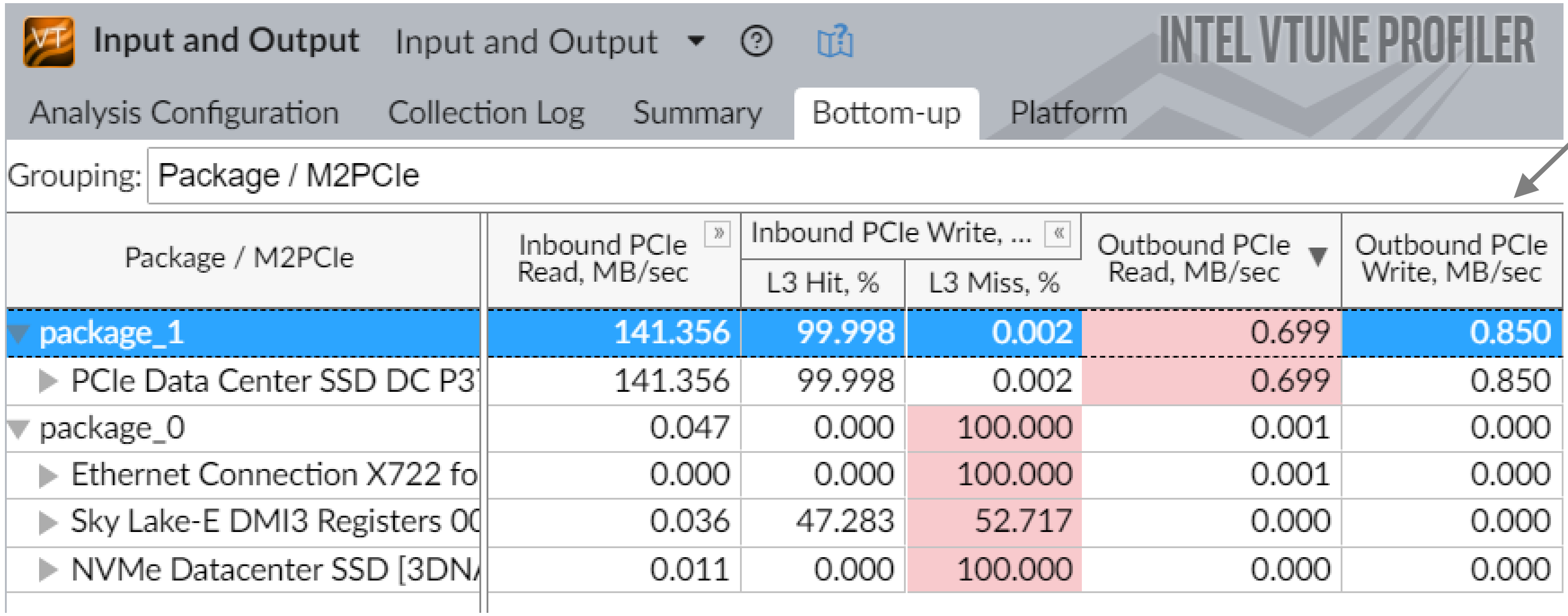
Memory-Mapped PCIe Device / Source Function	Source File	MMIO Reads	MMIO Writes
PCIe Data Center SSD DC P3700 SSD 0000:af:00.0		4,809,903 ⚑	2,100,063
spdk_mmio_write_4	mmio.h	0	2,100,063
spdk_mmio_read_4	mmio.h	4,809,903 ⚑	0

Full Inbound (DDIO) and Outbound (MMIO) traffic

DDIO hit/miss ratios

Sources of MMIO accesses

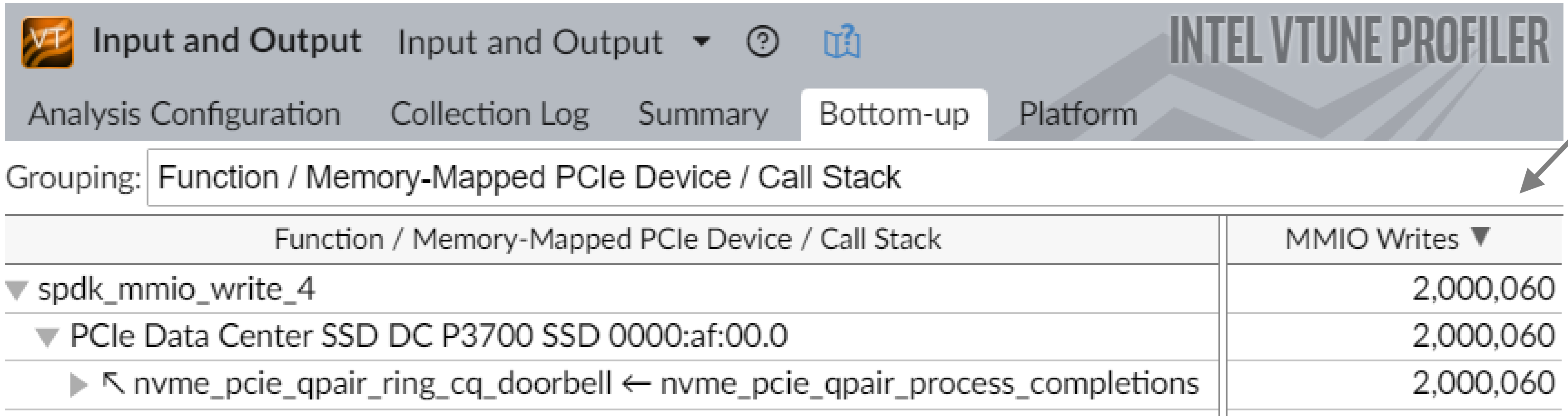
Need per device view?



Grouping: Package / M2PCIe

Package / M2PCIe	Inbound PCIe Read, MB/sec	Inbound PCIe Write, ... L3 Hit, %	Inbound PCIe Write, ... L3 Miss, %	Outbound PCIe Read, MB/sec	Outbound PCIe Write, MB/sec
package_1	141.356	99.998	0.002	0.699	0.850
PCIe Data Center SSD DC P3	141.356	99.998	0.002	0.699	0.850
package_0	0.047	0.000	100.000	0.001	0.000
Ethernet Connection X722 fo	0.000	0.000	100.000	0.001	0.000
Sky Lake-E DMI3 Registers 00	0.036	47.283	52.717	0.000	0.000
NVMe Datacenter SSD [3DN	0.011	0.000	100.000	0.000	0.000

DDIO and MMIO metrics per end devices

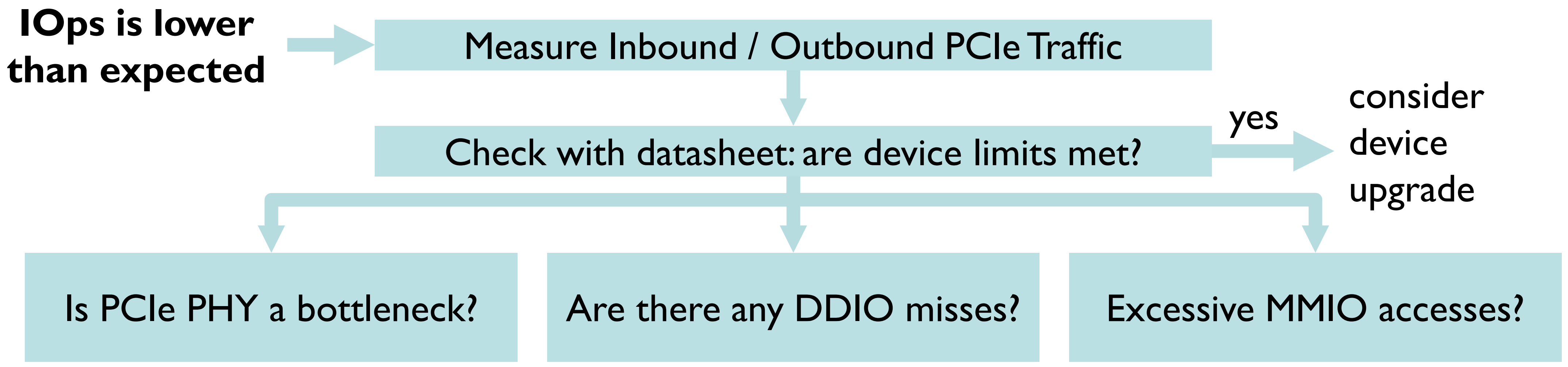


Grouping: Function / Memory-Mapped PCIe Device / Call Stack

Function / Memory-Mapped PCIe Device / Call Stack	MMIO Writes
spdk_mmio_write_4	2,000,060
PCIe Data Center SSD DC P3700 SSD 0000:af:00.0	2,000,060
↖ nvme_pcie_qpair_ring_cq_doorbell ← nvme_pcie_qpair_process_completions	2,000,060

Execution path leading to MMIO accesses

Methodology – Directions


IOps is lower than expected
Measure Inbound / Outbound PCIe Traffic
Check with datasheet: are device limits met?
yes
consider device upgrade
Is PCIe PHY a bottleneck?
Are there any DDIO misses?
Excessive MMIO accesses?

Possible solution

Estimate PCIe overhead, check if full BW + overhead fit into interface capabilities

1. Consider Max Payload Size and Max Read Request adjustments
2. Check if SSD coalesces requests

Multi-socket system? Check topology for localization

Consider L3 cache management techniques: prefetching, Intel CAT

Estimate Outbound PCIe traffic consumption per IO

Limit MMIO writes

Make sure there are no MMIO reads on the hot path

Estimate PCIe Bandwidth Consumption

Example: app reads from SSD

1. Core writes I/O command descriptor and starts polling completion queue element
2. Core notifies SSD that new descriptor is available (**Outbound PCIe Write**)
3. Device reads descriptor to get buffer address (Inbound PCIe Read)
4. Device writes I/O data (Inbound PCIe Write)
5. Device writes to the completion queue (Inbound PCIe Write)
6. Core detects that completion is updated
7. Core moves completion queue tail pointer (**Outbound PCIe Write**)

Outbound PCIe Write MB/sec =
(SQ_doorbell_sz [B] +
CQ_doorbell_sz [B]) * Read_Rate [M/sec]

Outbound Bytes per IO =
Outbound PCIe Write [MB/sec] / Read_Rate [M/sec]

Where to take **Read Rate** from?

SPDK-based app?
VTune profiling is integrated to SPDK

Other cases –
enhance the analysis with your own collector

Advanced Analysis

Customize Input and Output analysis by adding more uncore performance monitoring events that indicate:

- Snoop requests to IO and core/IO contentions
- Coherent operations issued by IIO to track full/partial write requests and allocating/non-allocating writes
- Intel® VT-d utilization

Follow this link to find a detailed view on analyzing raw events.

References

- What Every Programmer Should Know About Memory by Ulrich Drepper of Red Hat, Inc.
- Intel® Xeon® Processor Scalable Family Technical Overview
- Intel® Xeon® Processor Scalable Family Uncore Reference Manual
- Utilizing the Intel® Xeon® Processor Scalable Family IIO Performance Monitoring Events
- Benchmarking and Analysis of Software Data Planes
- Effective Utilization of Intel® Data Direct I/O Technology
- Intel® VTune™ Profiler Performance Analysis Cookbook