NVMe over CXL (NVMe-oC): An Ultimate Optimization of Host-Device Data Movement

Bernard Shung, San Chang, Terry Cheng

Wolley Inc.

Outline

- Problem Statement
- Prior Art on Host-Device Data Movement Optimization
- NVMe over CXL (NVMe-oC): Core Concept
- Application Examples Supporting NVMe-oC Core Concept
- NVMe-oC Performance Analysis
- NVMe-oC SSD versus Memory-Semantics SSD
- Local versus Network NVMe-oC
- Wolley's NVMe-oC Prototypes – Work in Progress
- Summary

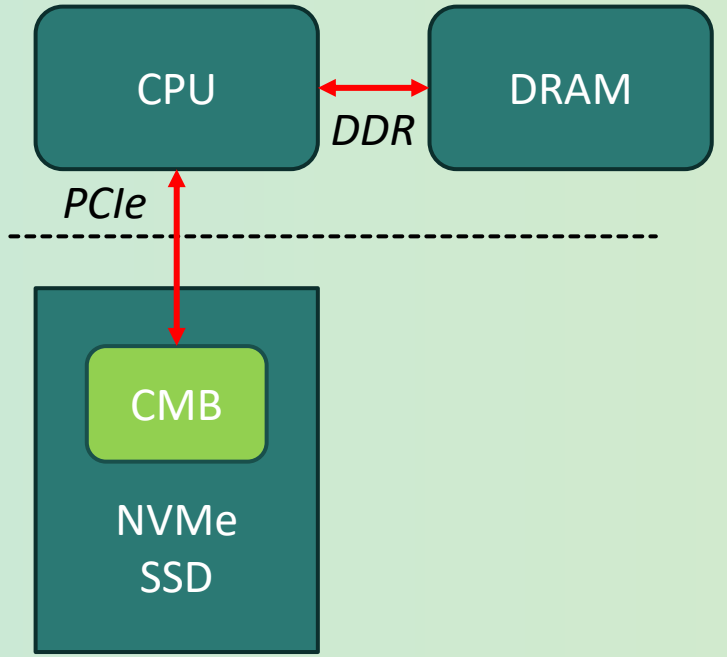
Problem Statement

- As an NVMe SSD solution provider, why should I bother with CXL (if my NVMe only uses PCIe or CXL.io)?
  - What is the benefit to upgrade my controller IP from PCIe to CXL?

Prior Art on Host-Device Data Movement Optimization – Controller Memory Buffer (CMB)

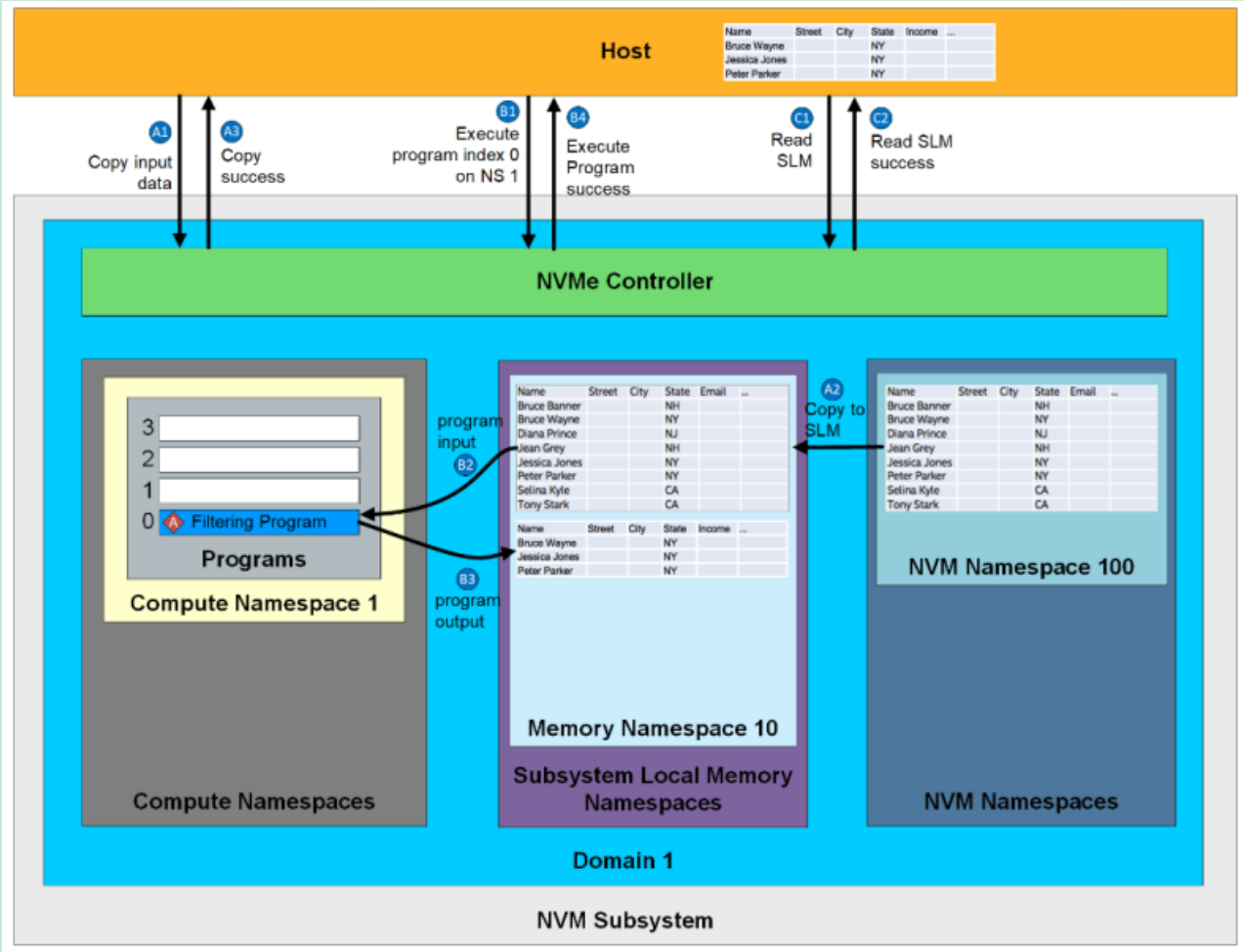
- **Controller Memory Buffer (CMB)**
  - Introduced to NVMe in version 1.2 since 2014
  - Hope to reduce host-device data movement
  - CPU accesses CMB via memory read/write TLP (MRd or MRw) and CMB is implemented as part of PCIe BAR
- **As host CPU can not access CMB as efficiently as DDR memory, CMB is generally used as *a DMA buffer for block data transfer*, which only optimize the data movement among PCIe devices**

04	RO	Impl Spec	**Write Data Support (WDS):** If this bit is set to '1', then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the host to the controller (e.g., Write). If this bit is cleared to '0', then data and metadata for commands that transfer data from the host to the controller shall not be transferred from the Controller Memory Buffer.
03	RO	Impl Spec	**Read Data Support (RDS):** If this bit is set to '1', then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the controller to the host (e.g., Read). If this bit is cleared to '0', then data and metadata for commands that transfer data from the controller to the host shall not be transferred to the Controller Memory Buffer.
02	RO	Impl Spec	**PRP SGL List Support (LISTS):** If this bit is set to '1', then the controller supports PRP Lists in the Controller Memory Buffer. If this bit is set to '1' and SGLs are supported by the controller, then the controller supports Scatter Gather Lists in the Controller Memory Buffer. If this bit is set to '1', then the Submission Queue Support bit shall be set to '1'. If this bit is cleared to '0', then PRP Lists and SGLs shall not be placed in the Controller Memory Buffer.
01	RO	Impl Spec	**Completion Queue Support (CQS):** If this bit is set to '1', then the controller supports Admin and I/O Completion Queues in the Controller Memory Buffer. If this bit is cleared to '0', then Completion Queues shall not be placed in the Controller Memory Buffer.
00	RO	Impl Spec	**Submission Queue Support (SQS):** If this bit is set to '1', then the controller supports Admin and I/O Submission Queues in the Controller Memory Buffer. If this bit is cleared to '0', then Submission Queues shall not be placed in the Controller Memory Buffer.



Prior Art on Host-Device Data Movement Optimization – NVMe Computational Storage (CS)

- **NVMe CS standardization**
  - TP4091: Computational Programs
  - TP4131: Subsystem Local Memory (SLM)
- **NVMe CS runs program inside SSD to operate data in Subsystem Local Memory**
- **NVMe CS makes computation closer to storage data to reduce data movement, but it requires tailored program that can run on device CPU**



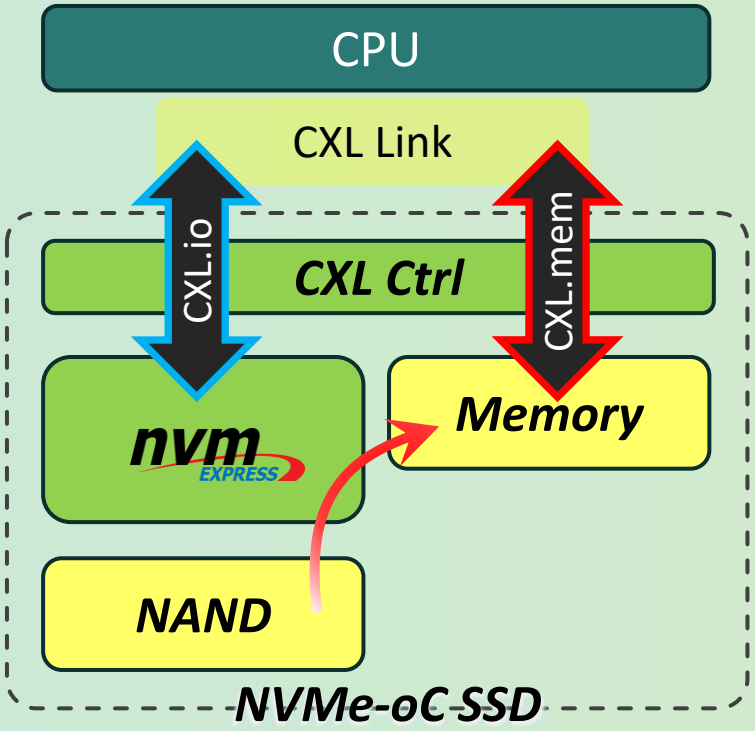
https://nvmexpress.org/wp-content/uploads/FMS-2023-Computational-Storage-Subsystem-Local-Memory.pdf

Animation

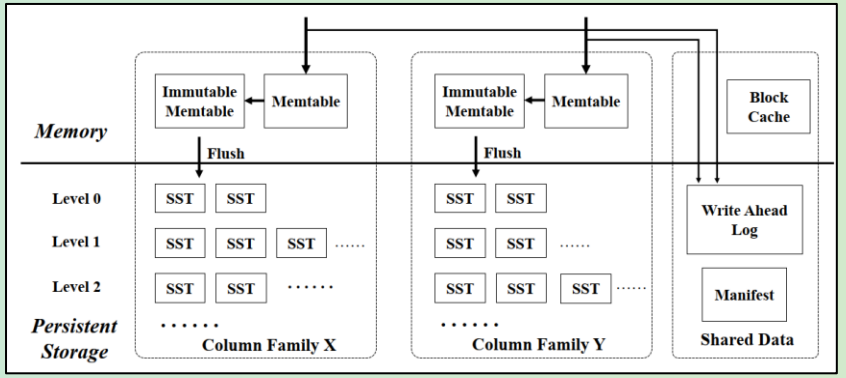
- **An animation video (about 40sec) will be inserted here**
  1. Use container to depict NVMe 4KB shipment from device to host
  2. Use “sending a worker to device side to unpack container” to depict the action of NVMe CS; the shipment from device to host is reduced
  3. Use Drone to depict an efficient and reduced shipment from device to host

NVMe over CXL (NVMe-oC): Core Concept

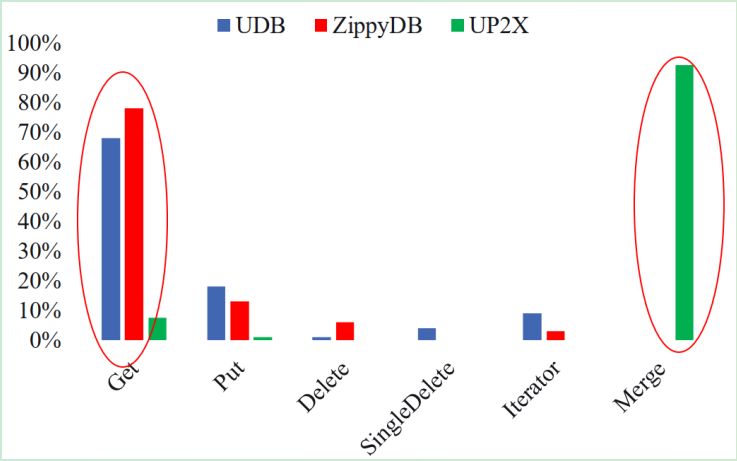
- **A NVMe-oC device uses CXL.io to access a NVMe SSD and CXL.mem to access an HDM**
  - CXL HDM enables cacheable high-performance data read/write not attainable in CMB
- **Key Advantages**
  - When the application needs only a fraction of the block data, the destination of the block data can be set to fall on the device HDM
  - Host CPU can selectively retrieve the necessary bytes from the block data via CXL.mem, reducing the amount of host-device data movement



Application Example (1): Key-value Store



- **Three typical RocksDB (KVS) production use cases at Facebook**
  - UDB (a MySQL storage layer for social graph data)
  - ZippyDB (a distributed key-value store)
  - UP2X (a distributed key-value store for AI/ML services)
- **GET operations dominate and mostly request small key-value access**
  - Unsorted and small-size key-value pairs are stored in different storage block



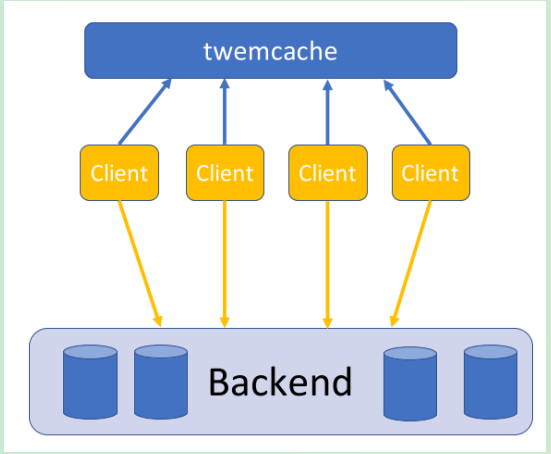
The average key size (AVG-K), the standard deviation of key size (SD-K), the average value size (AVG-V), and the standard deviation of value size (SD-V) of UDB, ZippyDB, and UP2X (in bytes)

	AVG-K	SD-K	AVG-V	SD-V
UDB	27.1	2.6	126.7	22.1
ZippyDB	47.9	3.7	42.9	26.1
UP2X	10.45	1.4	46.8	11.6

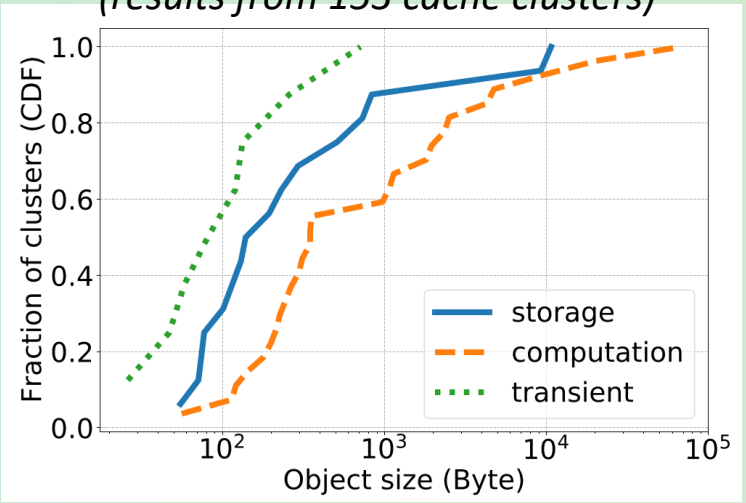
Cao, Zhichao, et al. "Characterizing, Modeling, and Benchmarking RocksDB Key-Value Workloads at Facebook" 18th USENIX Conference on File and Storage Technologies (FAST 20). 2020.

Application Example (2): Cache Systems

- **Twemcache is used by Twitter as a look-aside cache**
  - Upon a cache miss, cache clients read from the backends and then write to cache.
  - Upon cache writes, clients write to backends, then write into the cache
- **In Twitter production cache clusters**
  - The median of the object size curve for storage is slightly larger than 100 bytes
  - Top four production clusters' mean object sizes are 230, 55, 294 and 72 bytes, respectively



(results from 153 cache clusters)



Yang, Juncheng, Yao Yue, and K. V. Rashmi. "A Large-scale Analysis of Hundreds of In-memory Key-value Cache Clusters at Twitter" ACM Transactions on Storage (TOS) 17.3 (2021): 1-35.

Application Example (3): fsync()

- **Why fsync() is important?**
  - causes all modified data in the open file to be saved to permanent storage
  - commonly used by many applications to guarantee the durability of a file
  - blocks until the flushing data is completed
- **fsync() involves a bunch of small random read-modify-write operations, significantly impacting application performance**
  - Average performance of an individual TPC-C query experiences severe (4.9-8.7x) drop on NVMe SSDs

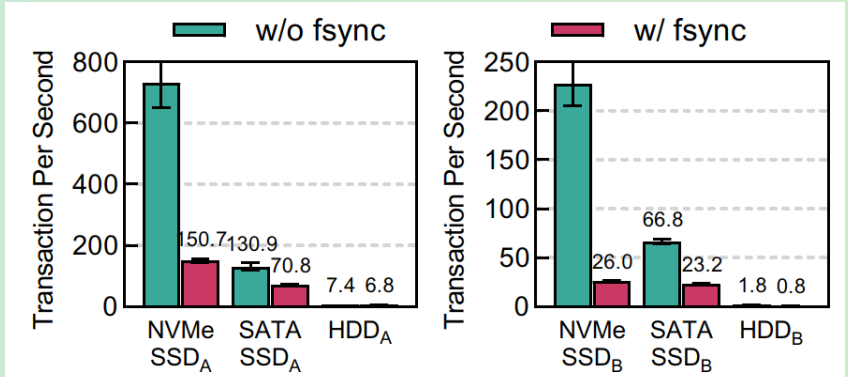


Figure 3: Performance drop of MySQL on NVMe SSD, SATA SSD and HDD after aggressively issue fsync.

He, Haochen, et al. "When Database Meets New Storage Devices: Understanding and Exposing Performance Mismatches via Configurations." Proceedings of the VLDB Endowment 16.7 (2023): 1712-1725.

low fsync performance

SSD with hundreds of thousands random 4KB read/write IOPS

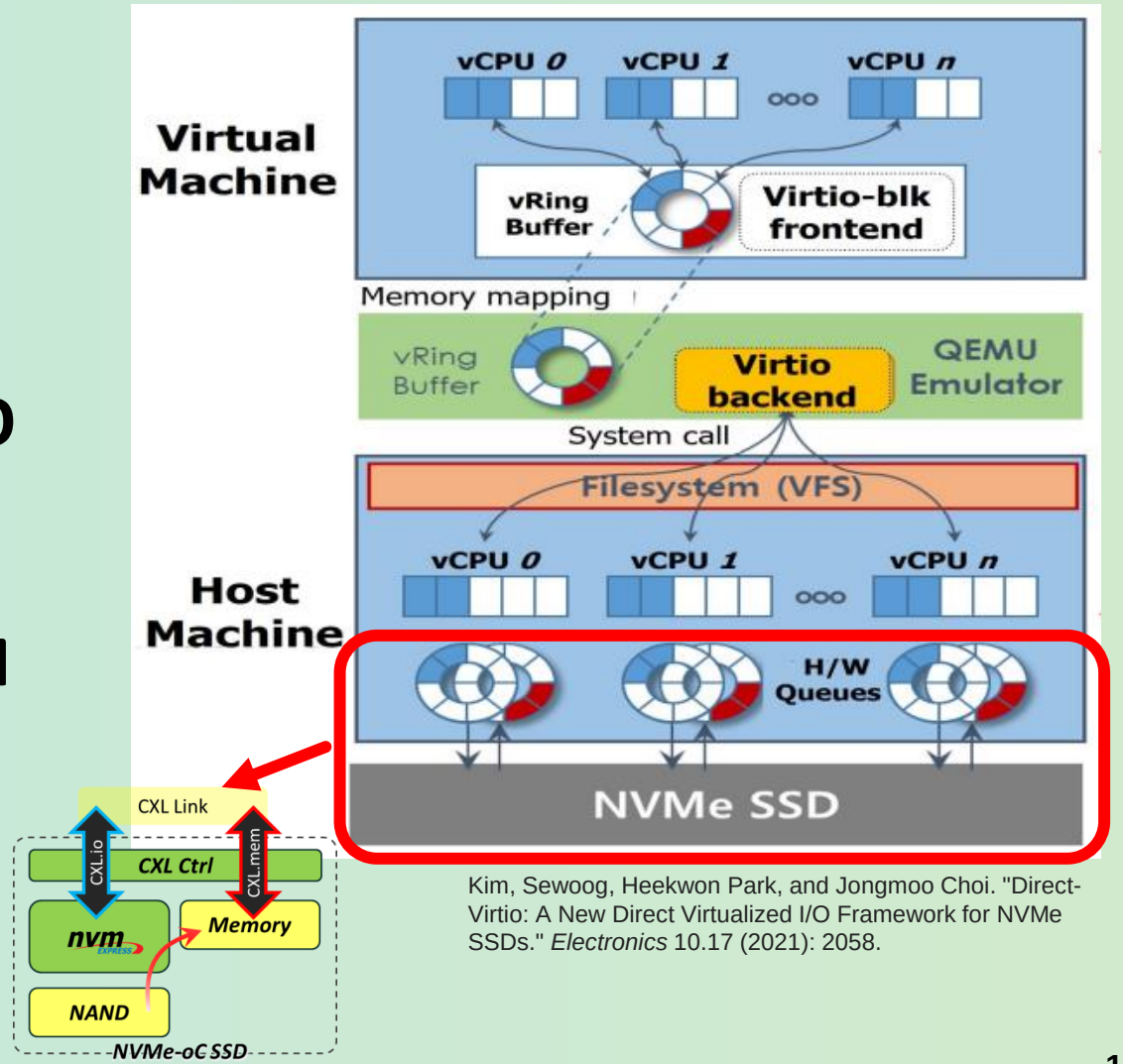
Drive	rate	latency
SAMSUNG MZ7LN512	160/s	6.3ms
Crucial_CT480M500SSD1	108/s	9.3ms
Intel 520	2031/s	0.49ms
SAMSUNG MZVPV512HDGL	104/s	9.6ms
Samsung SSD 960 PRO	267/s	3.8ms
Intel PC-3100	1274/s	0.79ms
Intel 750	2038/s	0.49ms
Intel PC-3700	7380/s	0.14ms

https://www.percona.com/blog/fsync-performance-storage-devices/

Application Example (4): VM IO Virtualization

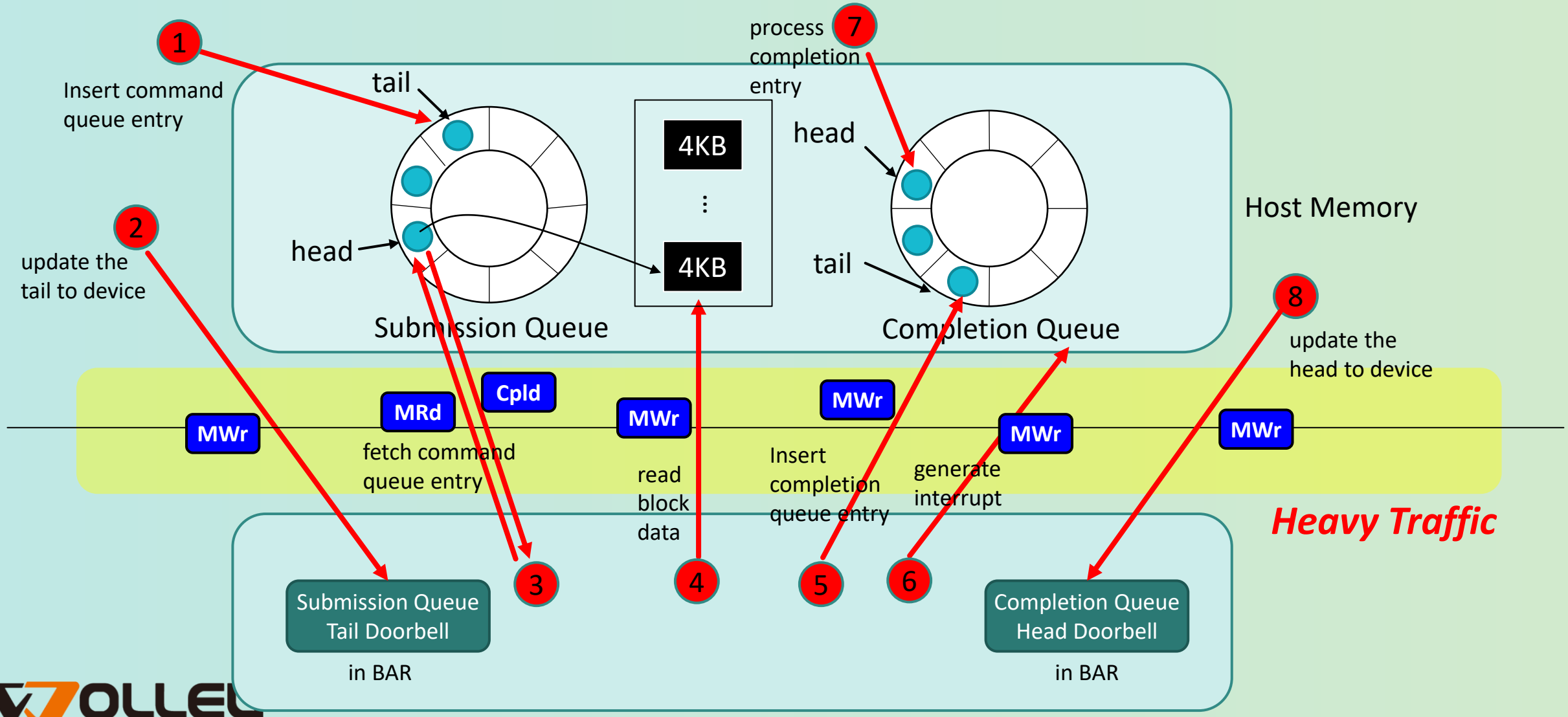
- **Virtio, a well-known software-based I/O virtualization framework**
  - Using frontend and backend drivers for the IO communication between VM and the host machine
- **As QEMU (or hypervisor) intercept IO requests from VM, the underlying software can fake data moving and return buffer address on device HDM using NVMe-oC**
  - *No modifications required to VM applications*

Virtio: IO Virtualization Framework

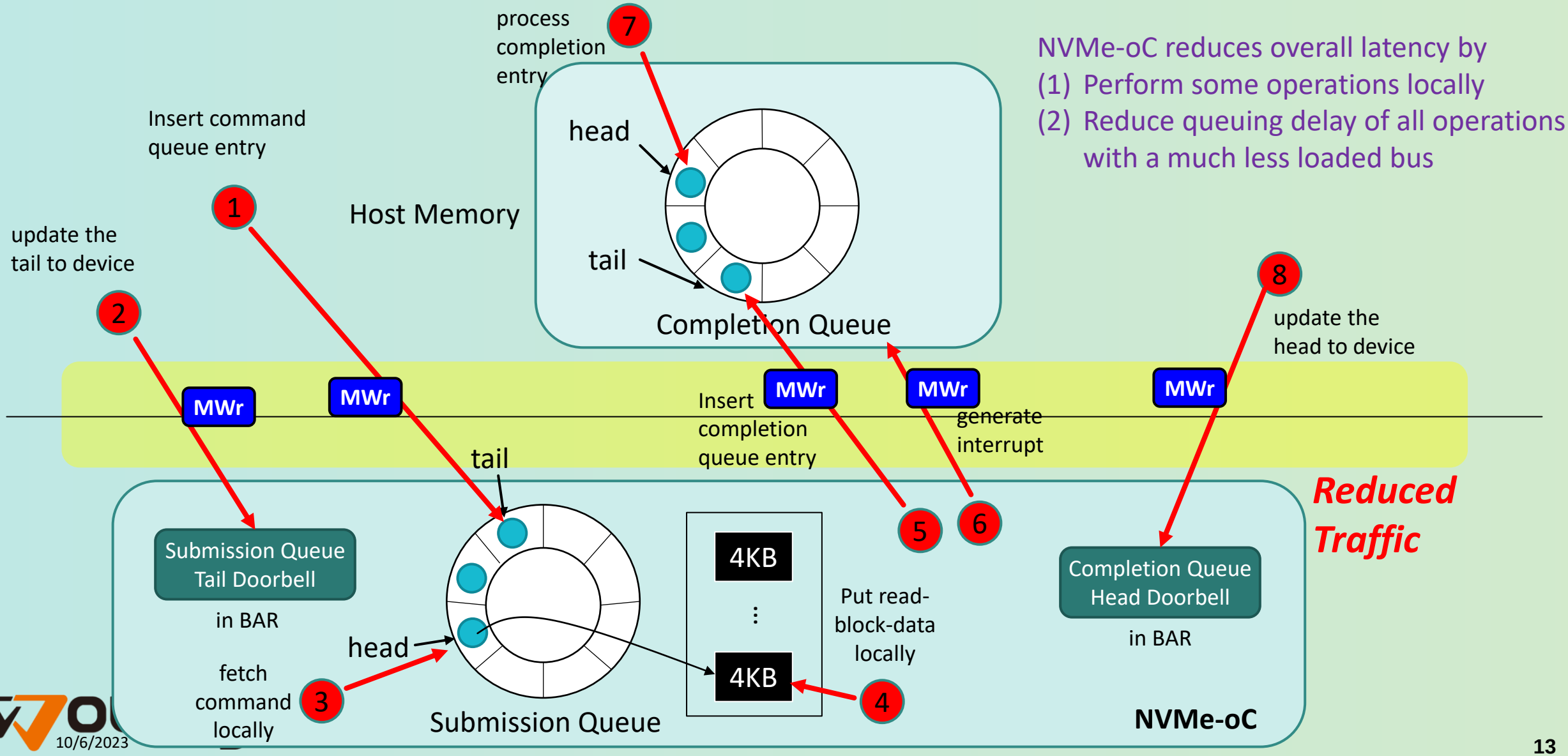


Kim, Sewoog, Heekwon Park, and Jongmoo Choi. "Direct-Virtio: A New Direct Virtualized I/O Framework for NVMe SSDs." *Electronics* 10.17 (2021): 2058.

Performance Analysis for Reading Data with *NVMe*



Performance Analysis for Reading Data with *NVMe-oC*



NVMe-oC reduces overall latency by
(1) Perform some operations locally
(2) Reduce queuing delay of all operations with a much less loaded bus

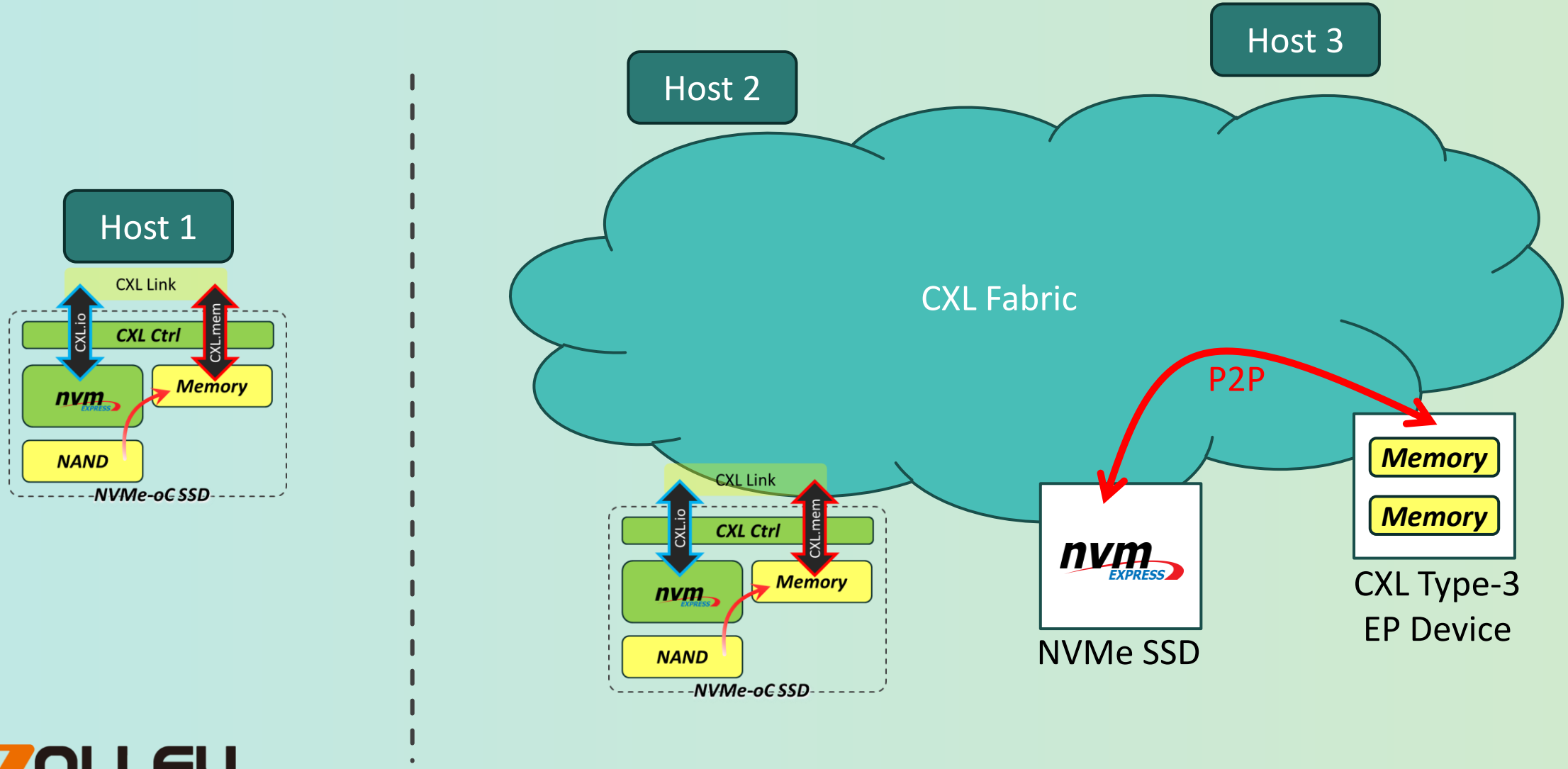
Reduced Traffic

NVMe-oC SSD versus Memory-Semantics SSD

- **Samsung was first to consider using CXL interface on a NVMe SSD, with their introduction of Memory-Semantics SSD in Flash Memory Summit 2022**
- **But fundamentally NVMe-oC SSD and Memory-Semantics SSD are different, as highlighted below**

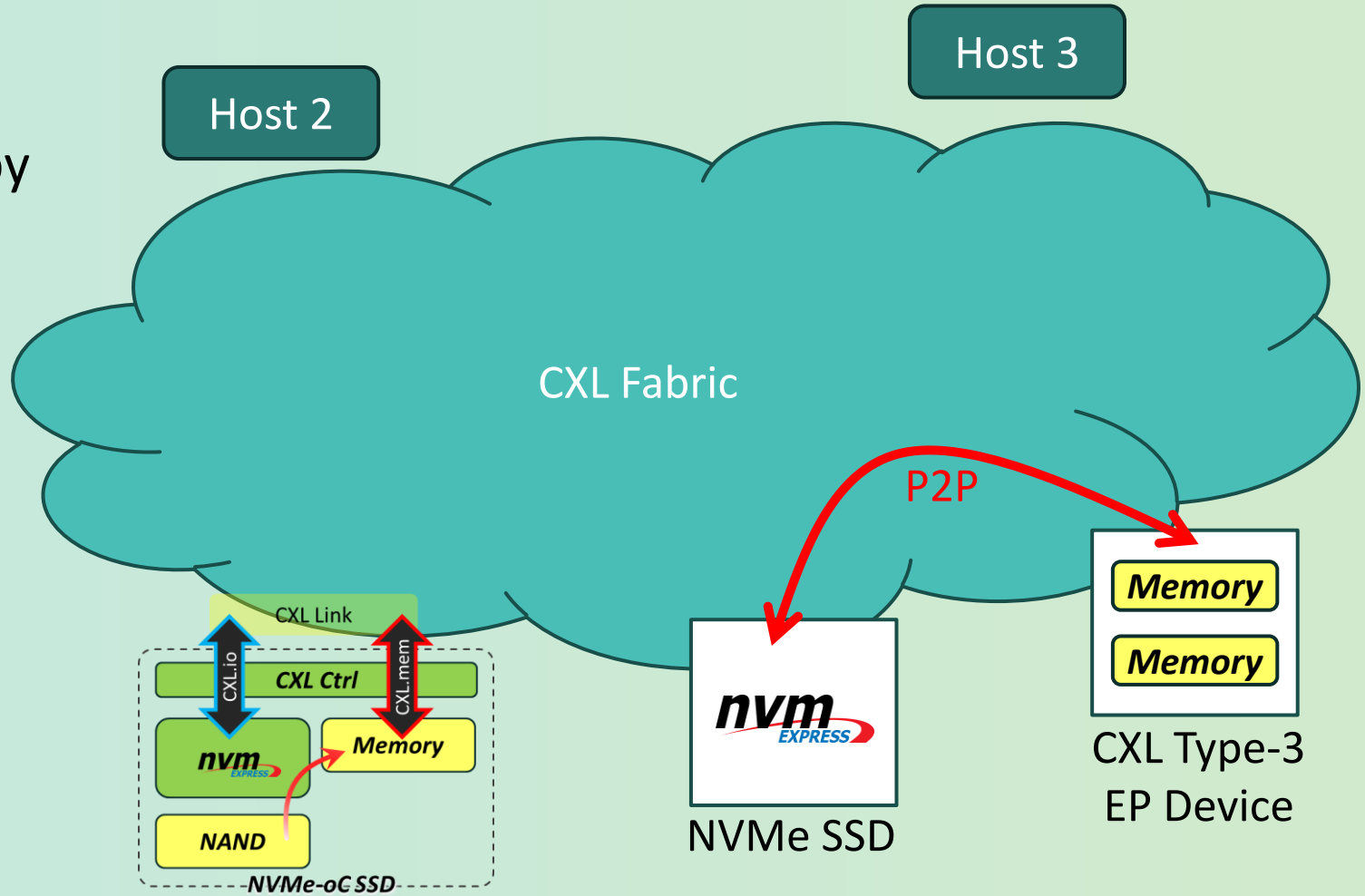
	NVMe-oC SSD	Memory-Semantics SSD
Host Interface	CXL	CXL
Media Components	NAND + DRAM	NAND + DRAM
Legacy NVMe SSD Support	Yes	Yes
Functional Device(s)	NVMe SSD and HDM memory	NVMe SSD or HDM memory
Roles of NAND and DRAM	DRAM as staging memory of NAND	DRAM as cache of NAND

Local versus Network NVMe-oC



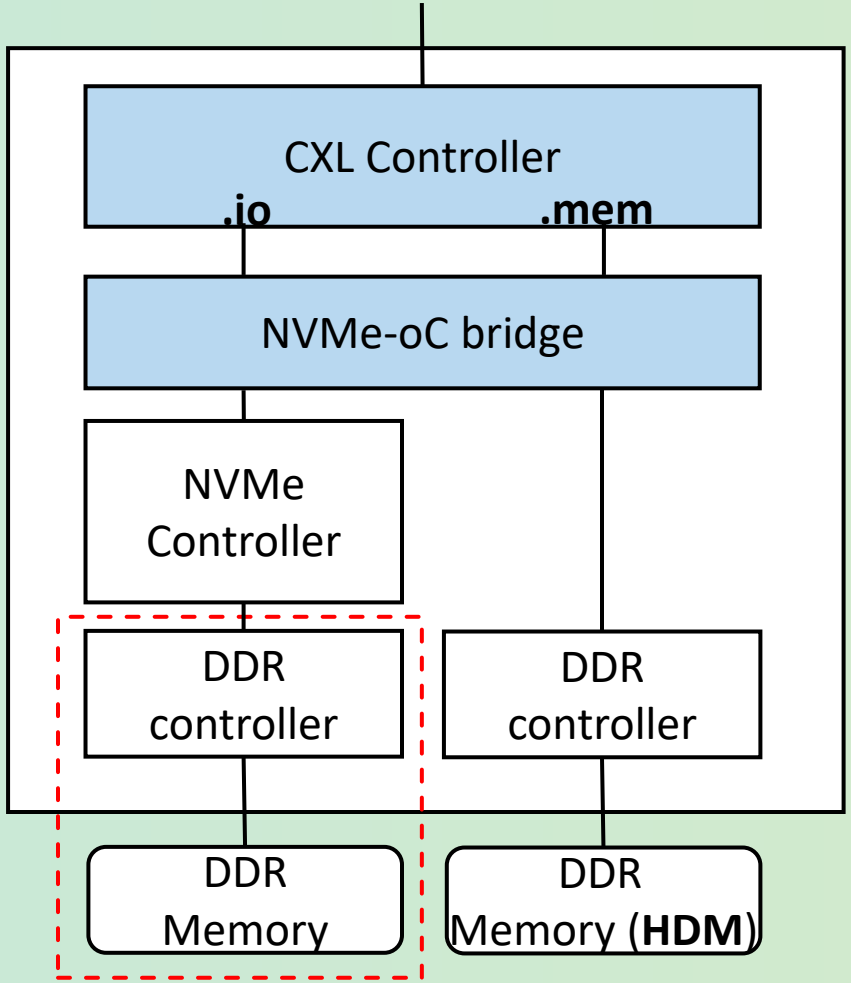
NVMe-oC Revolution: NVMe SSD as a Memory Device!

- Block data from NVMe SSD need to be turned into memory form to be useable by Host/Application
- Host/Application is to determine where to place staging data optimally
  - Large % of data needed -> place on host side
  - Small % of data needed -> place on device side



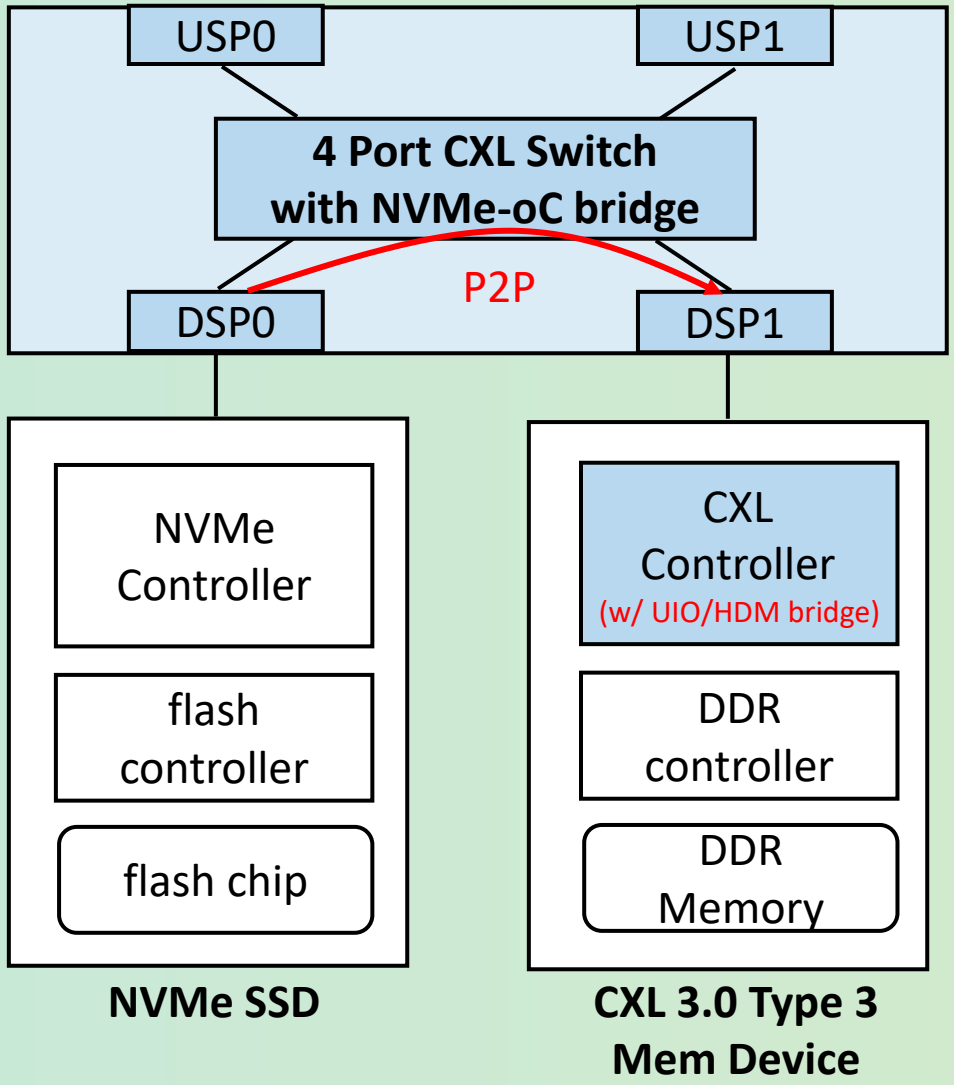
Wolley's NVMe-oC Prototype – Local NVMe-oC

- **FPGA Implementation Platform**
  - Virtex® UltraScale+™ XCVU13P
  - PCIe Gen3 x8 speed
  - 128GB DDR4 DIMM for NVMe RAM-disk
  - 4GB DDR4 DIMM for CXL HDM
- **CXL Controller + NVMe-oC bridge**
  - CXL 1.1/2.0/3.0 Support
  - Enable NVMe SSD and HDM coexist
  - Optimize data movement between NAND and HDM
- **Software**
  - Modified NVMe driver to utilize CXL HDM as data buffer



Wolley's NVMe-oC Prototype – Network NVMe-oC

- **4-Port CXL Switch Prototype (current):**
  - Virtex® UltraScale+™ VU19P
  - 2 Upstream Ports
  - 2 Downstream Ports
  - PCIe Gen3 x8 speed
  - CXL 1.1/2.0
- **NVMe-oC Switch Prototype (2024):**
  - Adding P2P Support
  - Converting TLP to UIO inside switch for P2P
- **CXL 3.0 Type 3 EP Prototype (2024):**
  - Adding UIO to HDM bridge



Summary

- **Many applications only use a fraction of the block data retrieved from storage**
- **NVMe-oC achieves “best of both worlds”**
  - Backward compatibility with NVMe storage infrastructure
  - Efficient CXL.mem access of device side data
- **NVMe-oC optimizes host-device data movement not attainable before**
  - Less data movement reduces power consumption and improves performance
- **“Local NVMe-oC” integrates NVMe SSD and HDM, while “Network NVMe-oC” takes advantage of P2P feature in CXL 3.0 fabric**
- **Host intelligence to optimally place staging data is an interesting/important topic**
- **Wolley hopes to contribute to hardware/software components of NVMe-oC to make the vision a reality**