IN5050: PCI Express Non-Transparent Bridging

Lars Bjørlykke Kristiansen

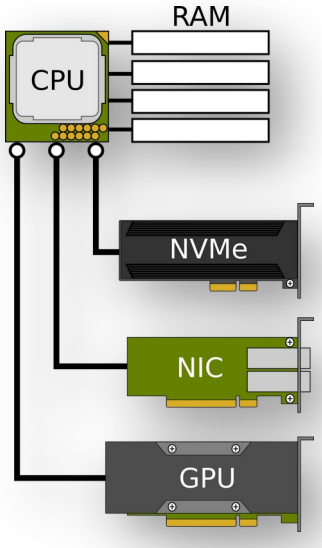
Jonas Markussen

Dolphin Interconnect Solutions

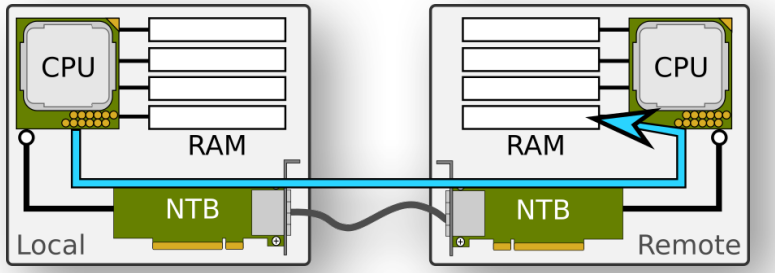
25th April 2024



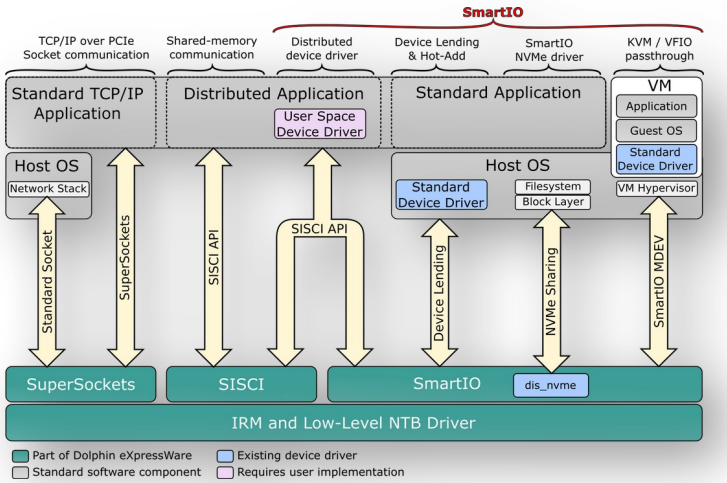
Outline

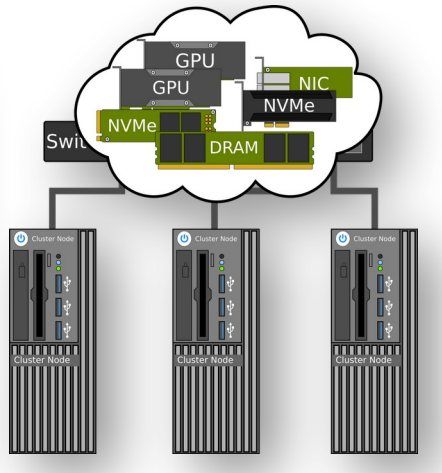


PCIe Fundamentals



PCIe networking & shared-memory with Non-Transparent Bridging





Intro to Dolphin's Solution Stack

PCI Express (PCIe) is a standard for connecting devices to a computer system



Graphics card (GPU)

- The standard defines both the physical hardware connection and how software should interact with a device
  - Software that is controlling a device is usually called a "driver"



Non-Volatile Memory SSD (NVMe)



InfiniBand Network Card (NIC)

PCI Express (PCIe) is a standard for connecting devices to a computer system



PCIe Gen4 x16
~= 32GBytes/sec
(Or 512Gbit/sec)

- Currently at PCIe Gen 5.0
  - Gen6 spec is out
- Many different form factors
  - Add-in cards, m.2, cables, …
- Each generation roughly doubles bandwidth
  - Gen5 is ~4GBytes/sec per lane
  - Gen6 will be ~8 GBytes/sec per lane
- Supports x1, x2, x4, x8 and x16 lanes

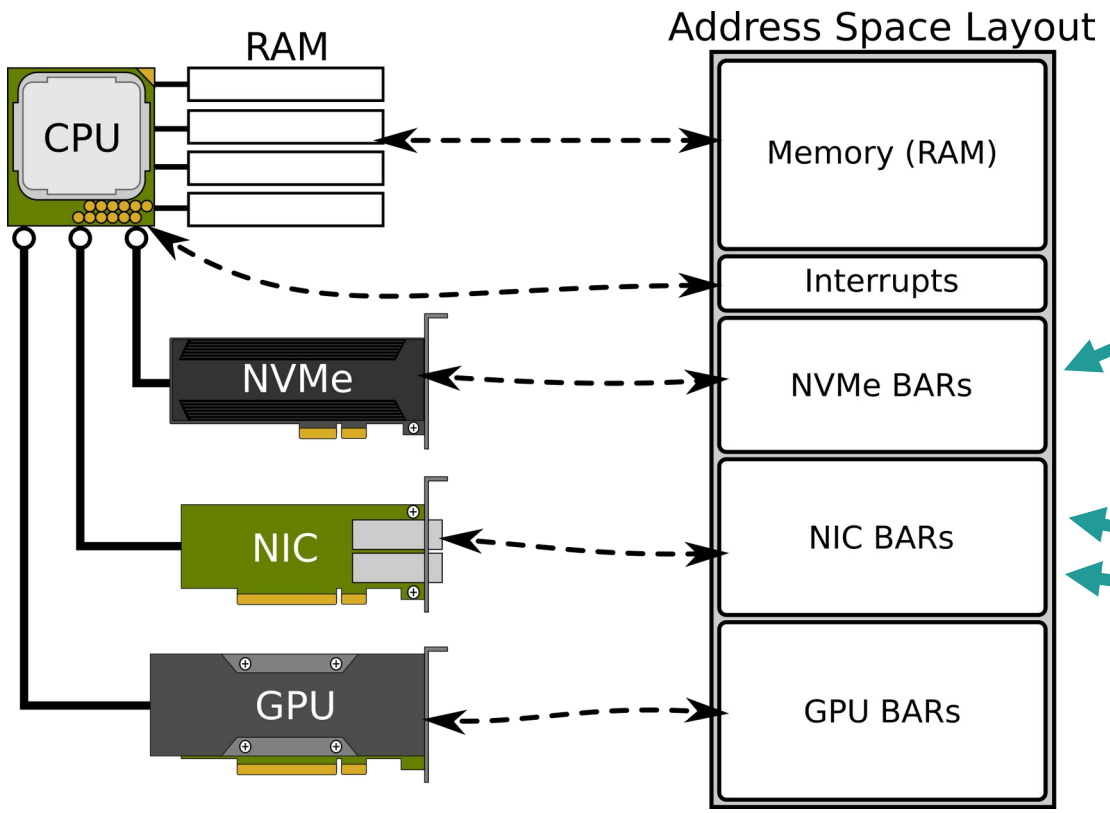


PCIe Gen4 x4 ~= 8GBytes/sec



PCI Gen5 x16
~= 64 GBytes/sec

PCIe devices, such as GPUs, NVMes, and network cards, are mapped to the same physical address space as system memory and the CPU



BIOS (and OS) "enumerates" the PCIe bus during boot, and maps device memory regions to physical addresses

```
# lspci -v

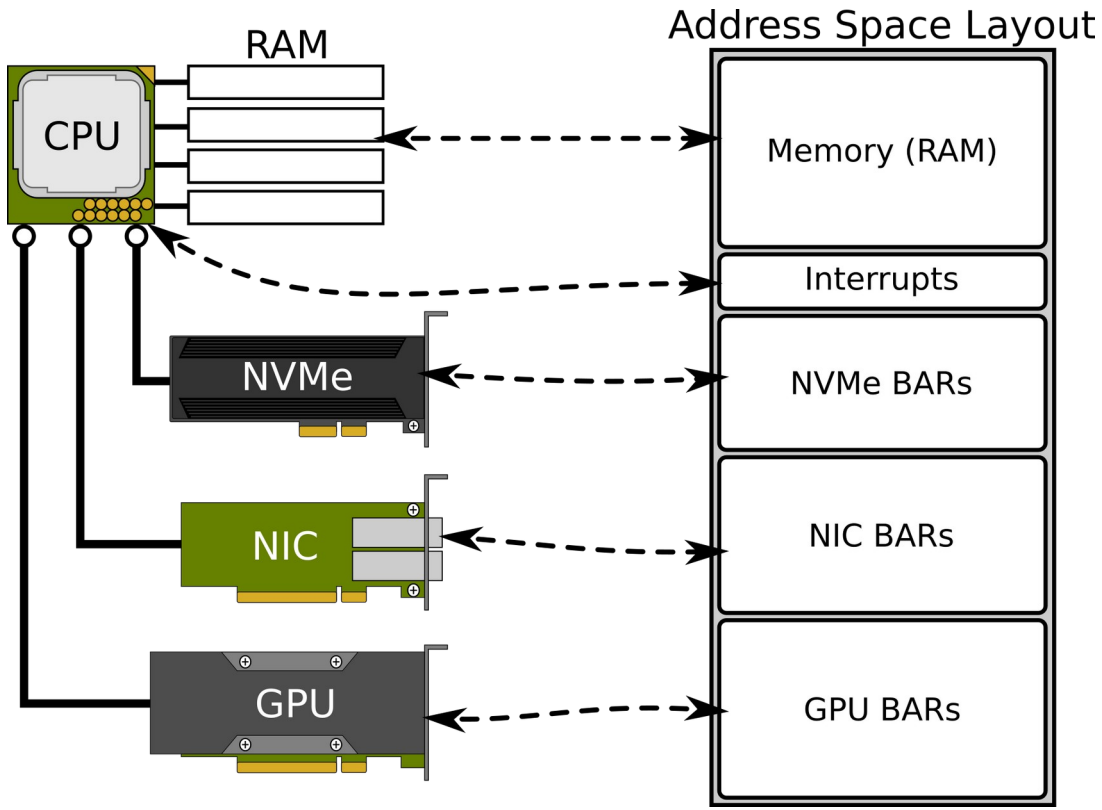
08:00.0 Non-Volatile memory controller: Kingston Technology Company,
Inc. Device 500f (rev 03) (prog-if 02 [NVM Express])
        Subsystem: Kingston Technology Company, Inc. Device 500f
        Flags: bus master, fast devsel, latency 0, IRQ 34, NUMA node 0,
IOMMU group 15
        Memory at fc600000 (64-bit, non-prefetchable) [size=16K]
        Capabilities: [40] Power Management version 3
        Capabilities: [50] MSI: Enable- Count=1/8 Maskable+ 64bit+
        Capabilities: [70] Express Endpoint, MSI 00
        Capabilities: [b0] MSI-X: Enable+ Count=16 Masked-
        Kernel driver in use: nvme
        Kernel modules: nvme

0a:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd.
RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 15)
        Subsystem: ASUSTeK Computer Inc. PRIME B450M-A Motherboard
        Flags: bus master, fast devsel, latency 0, IRQ 36, IOMMU group
15
        I/O ports at f000 [size=256]
        Memory at fc504000 (64-bit, non-prefetchable) [size=4K]
        Memory at fc500000 (64-bit, non-prefetchable) [size=16K]
        Capabilities: [40] Power Management version 3
        Capabilities: [50] MSI: Enable- Count=1/1 Maskable- 64bit+
        Capabilities: [70] Express Endpoint, MSI 01
        Capabilities: [b0] MSI-X: Enable+ Count=4 Masked-
        Kernel driver in use: r8169
        Kernel modules: r8169

(abbreviated)
```

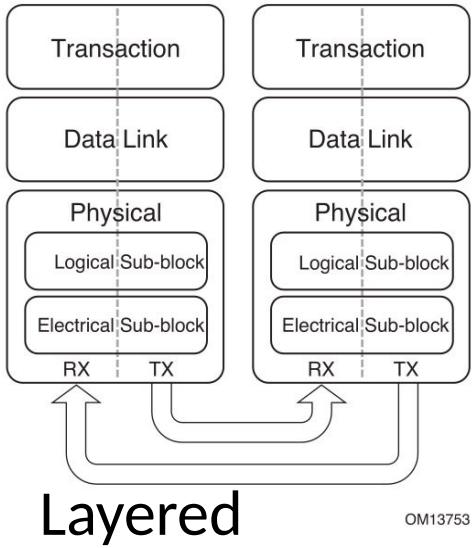
The "lspci" command shows memory address mappings (and more, such as capabilities, enabled registers etc)

PCIe devices, such as GPUs, NVMes, and network cards, are mapped to the same physical address space as system memory and the CPU

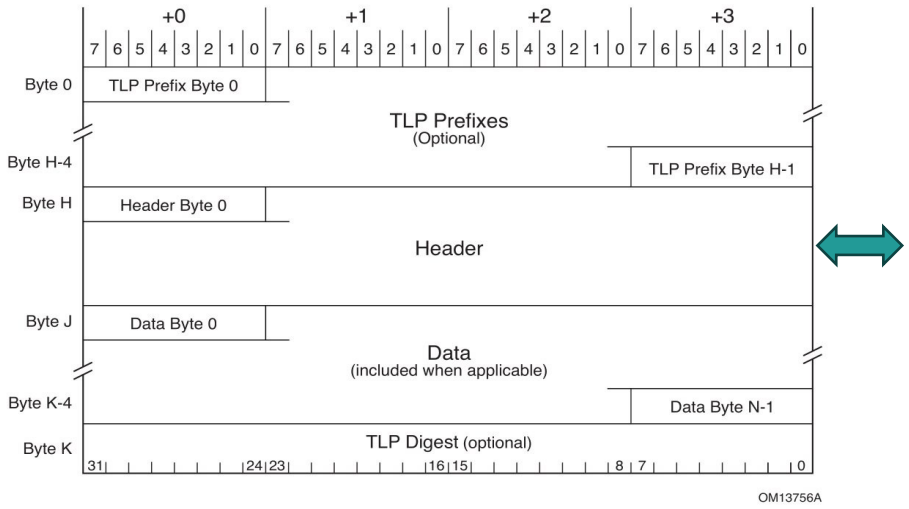


- The CPU can read and write to device memory using memory-mapped I/O (MMIO)
  - Example: Reading/writing to device registers
  - Slower than accessing RAM (~10x slower)
  - Note: MMIO is sometimes called "PIO"
- MMIO is same as accessing a memory pointer (to RAM) from the software's perspective
  - Some considerations: write-combining, alignment, etc.
- Devices can read and write from system memory using direct memory access (DMA)
  - DMA transfers can be used to copy data without involving the CPU (offloading)
  - Example: cudaMemcpy(D2H / H2D)
  - On some platforms I/O cache coherency can be difficult

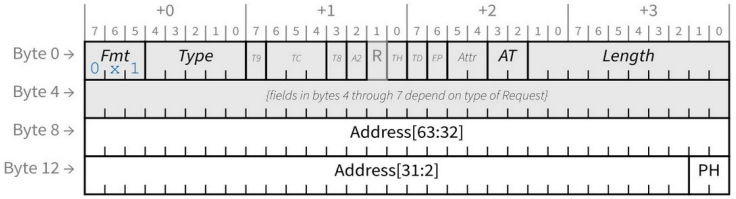
PCIe uses request/completion packets with headers, containing the memory address destination and ID of the requestor



Layered protocol



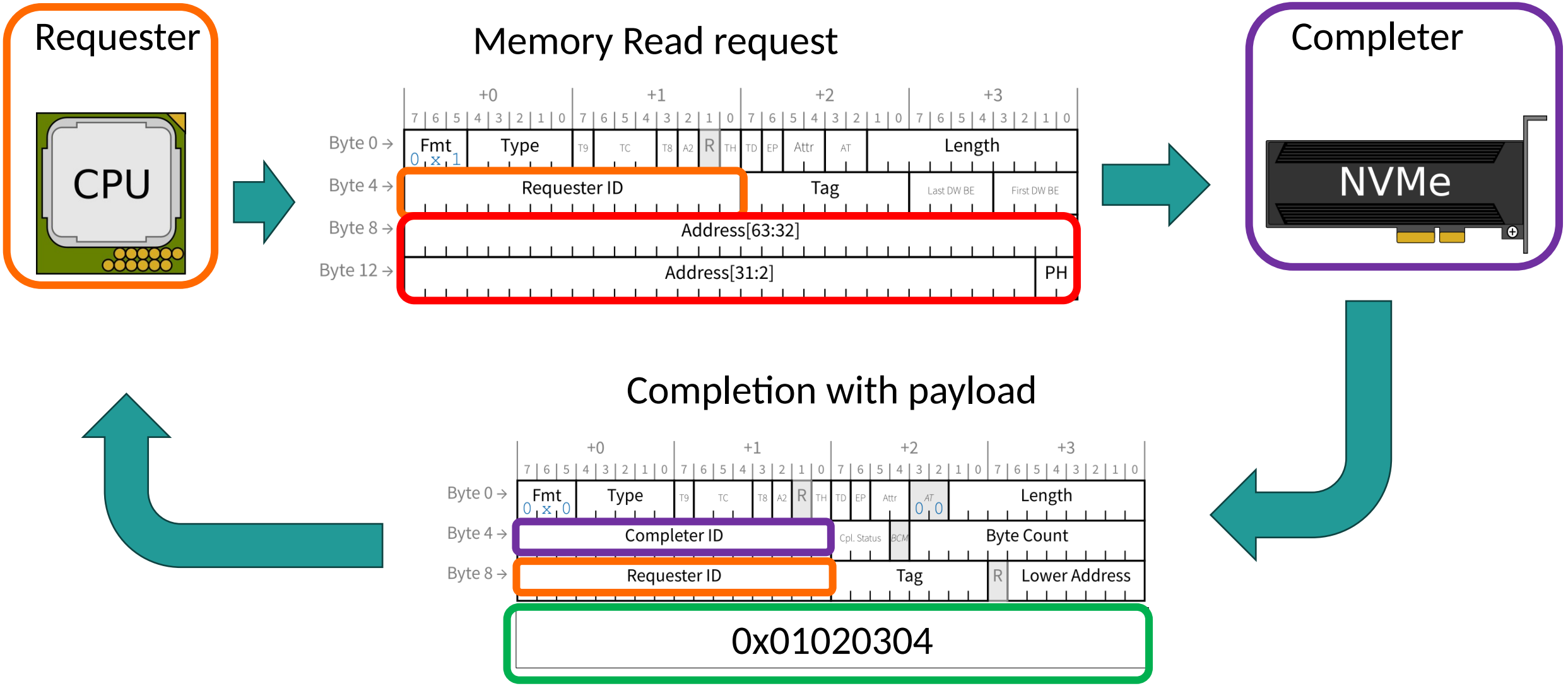
Single transaction layer packet (TLP)



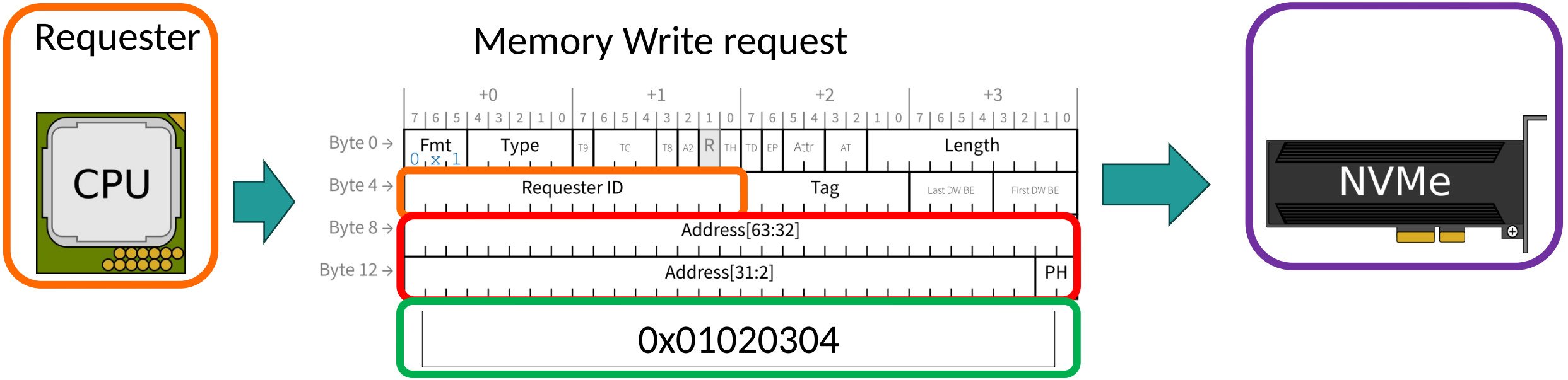
TLP Header for a 64-bit Memory Request

From software it is the same as reading/writing to a physical memory address

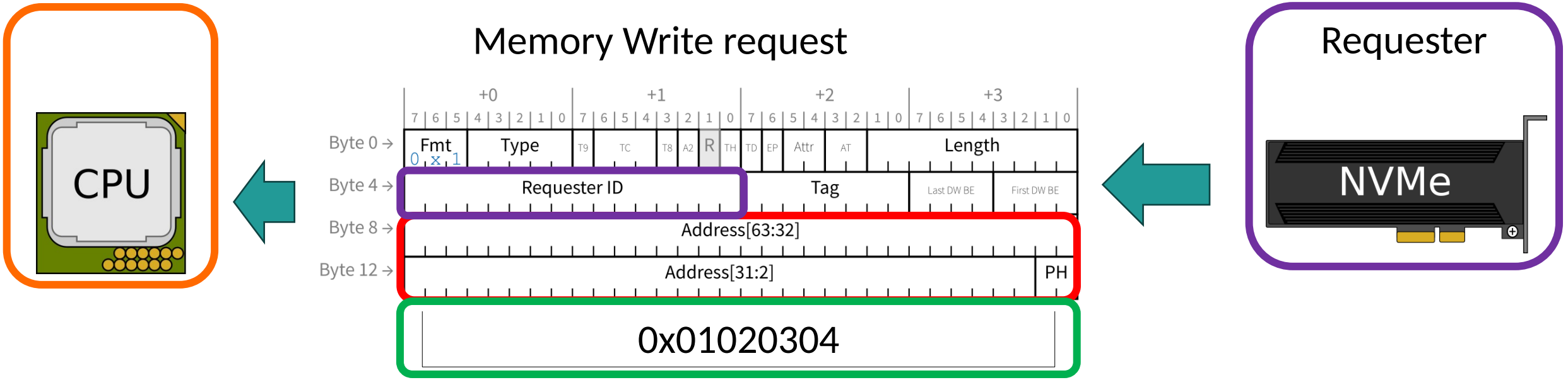
Example: CPU reading a device register



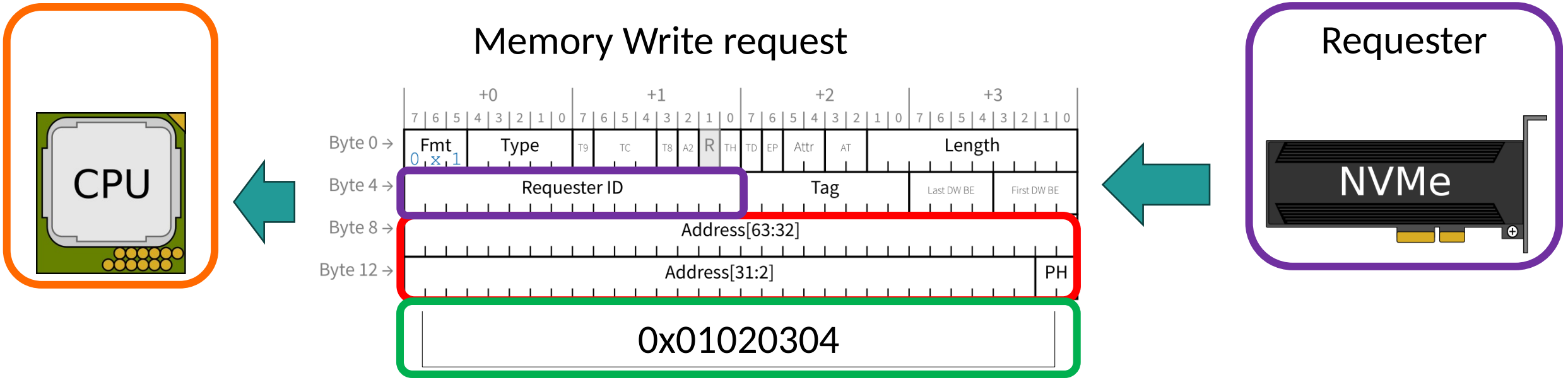
Example: CPU writing a device register



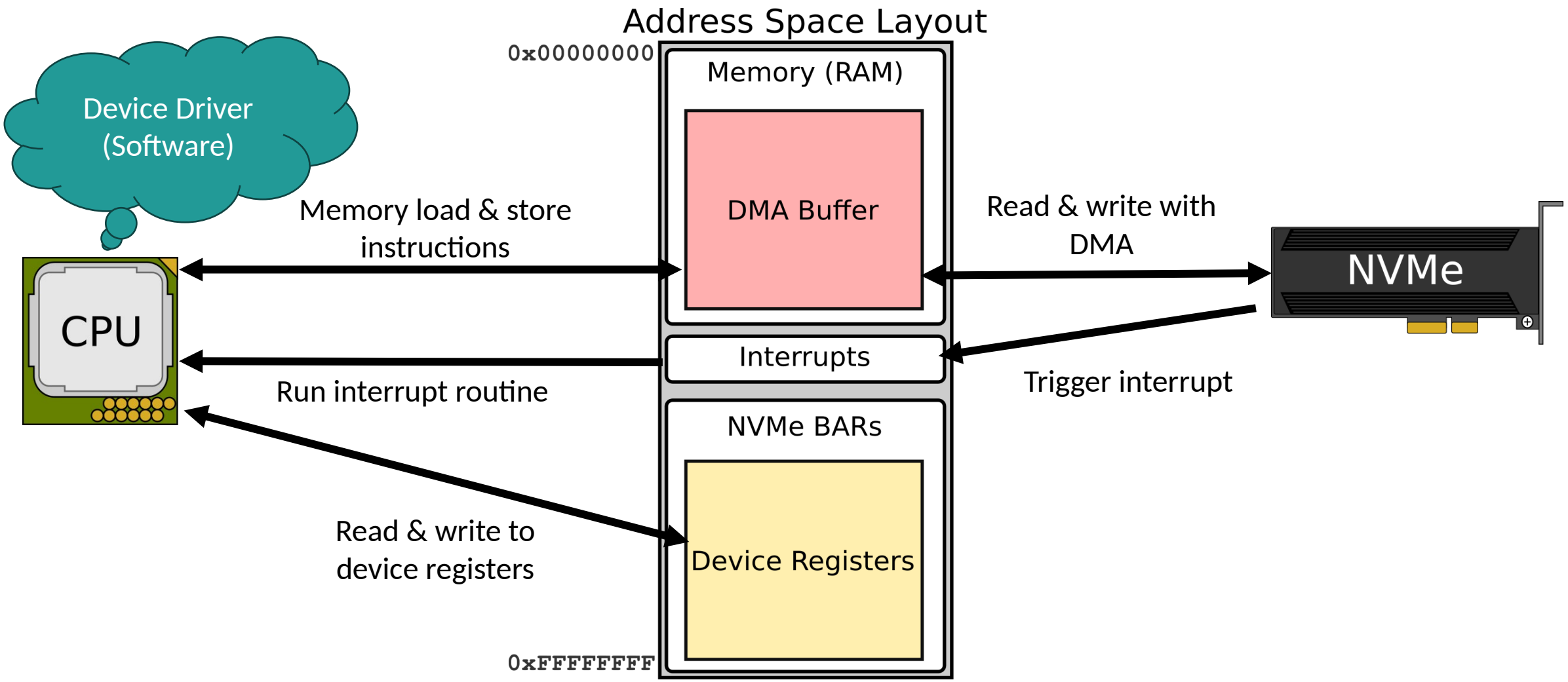
Example: Device writing to RAM (DMA)



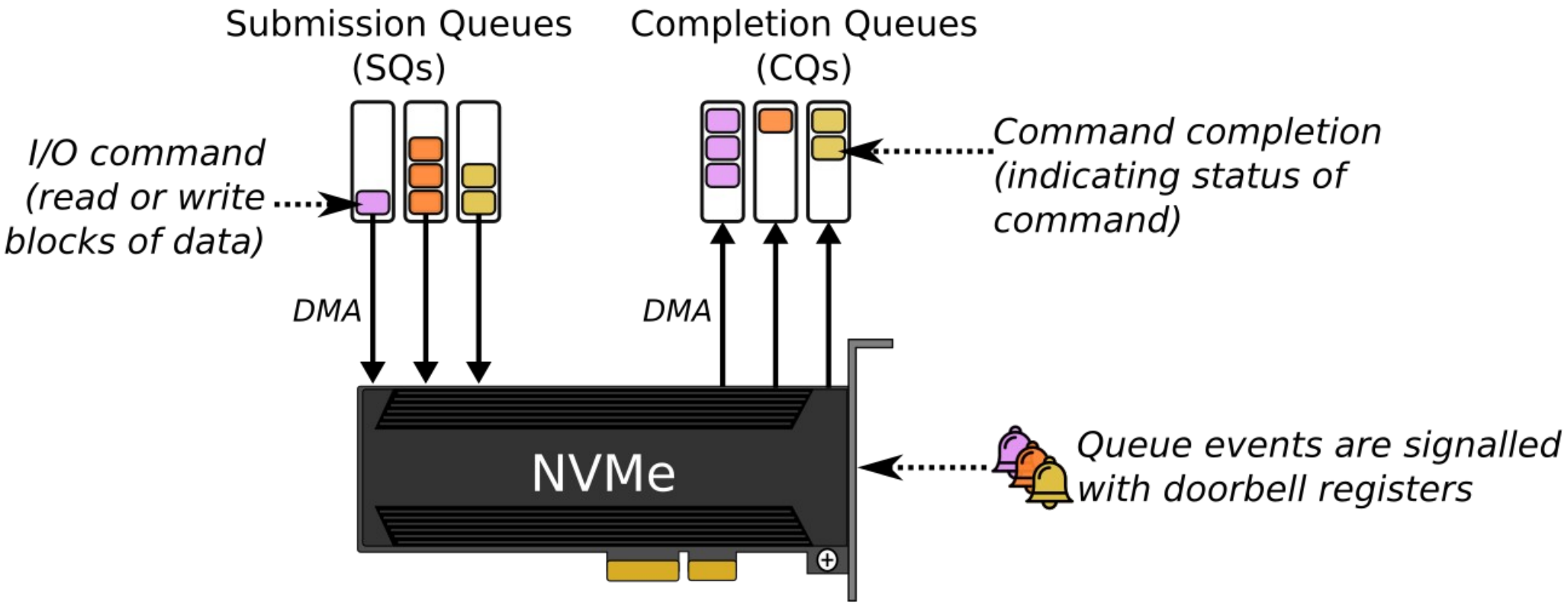
Example: Device sending interrupt



Most device drivers and PCIe devices follow a similar data and control pattern, using device registers, DMA transfers and interrupts



Example: Reading a block from NVMe disk



Example: Reading a block from NVMe disk

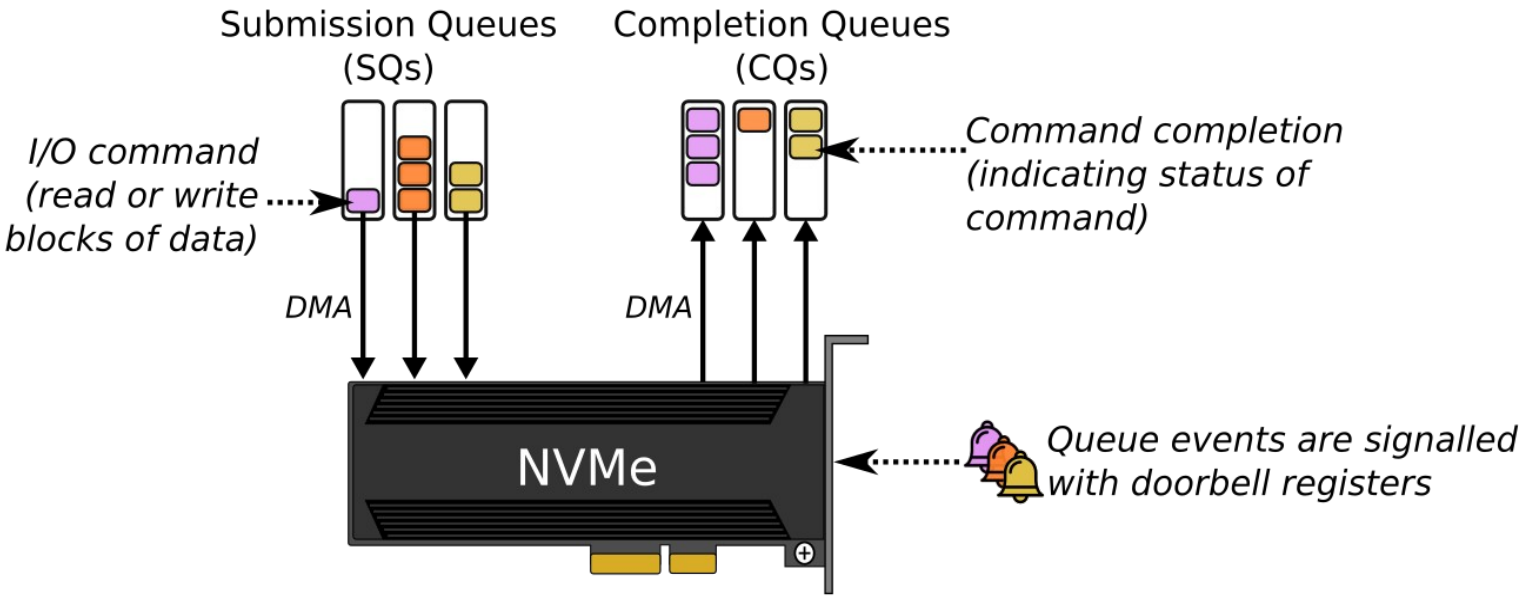
Submission Queue Entry ("Command")

	31	23	15	7
DW0	Opcode	Flags	Command Identifier	
DW1	Namespace Identifier			
DW2	*Reserved*			
DW3	*Reserved*			
DW4	Metadata pointer			
DW5	Address of memory page 1 (bits 63:32)			
DW6	Address of memory page 1 (bits 31: 0)			
DW7	Address of memory page 2 (bits 63:32)			
DW8	Address of memory page 2 (bits 31: 0)			
DW9	Start block (bits 63:32)			
DW10	Start block (bits 31: 0)			
DW11	Number of blocks		RW Flags	
DW12	Dataset management			
DW13	*Reserved*			
DW14	*Reserved*			
DW15	*Reserved*			

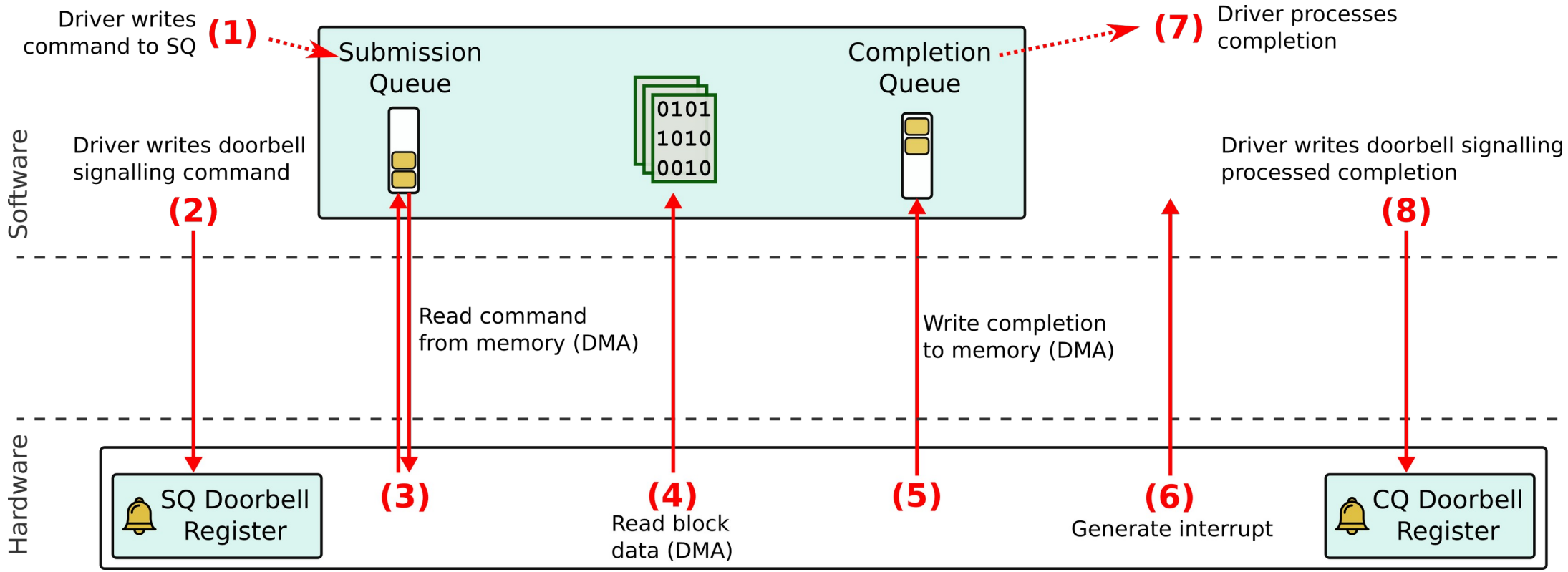
Data structures are placed in RAM

Completion Queue Entry ("Completion")

	31	23	15	7	0
DW0	Command Specific				
DW1	Reserved				
DW2	SQ Identifier		SQ Head Pointer		
DW3	Status Field		P	Command Identifier	



Example: Reading a block from NVMe disk



Industries we work with

- High Performance Computing
- Embedded Systems & Rugged Applications
- Instrumentation
- Automotive Industry



That's us :)

Solutions we provide

- Ultra-low latency and high-bandwidth networking over PCIe
- PCIe-based cluster computing
- Shared-memory communication
- Cluster-wide I/O expansion (SmartIO)
- TCP/IP over PCIe
- … and a lot more!

It is possible to connect machines and external devices in a PCIe cluster, using cables, rack switches, and plugin adapter cards



Small PCIe cluster with switch, three nodes, and two expansion chassis



MXH930 Host Adapter
Gen4 x16 copper



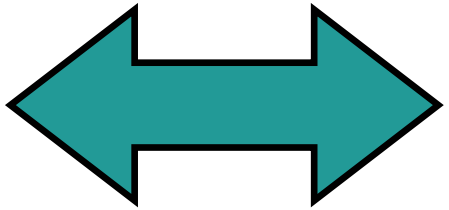
MXS924 Switch
24 x4 gen4 ports

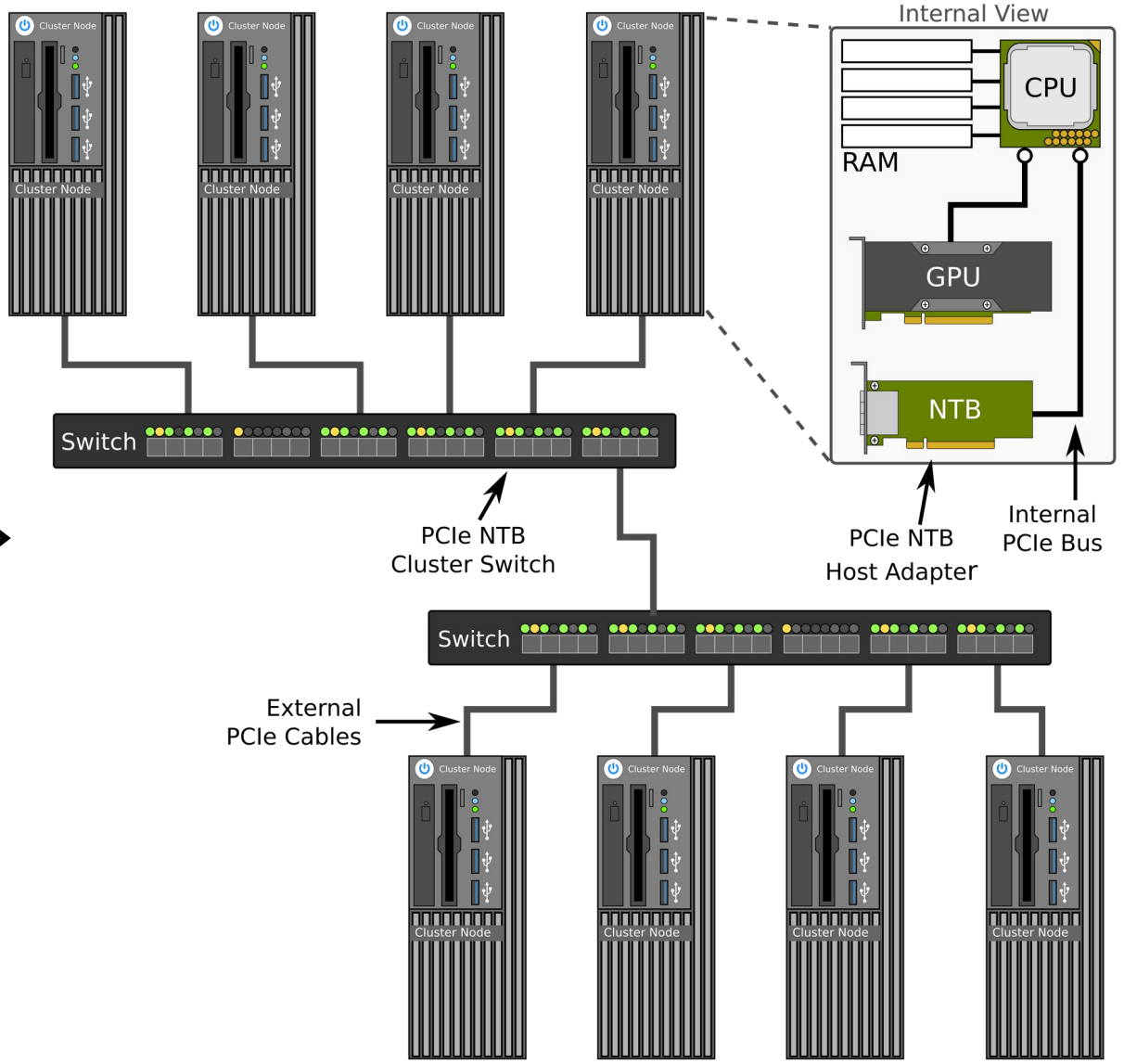


MXH940 Host Adapter
Gen4 x16 fibre-optic

In a PCIe cluster, CPUs and devices (such as GPUs and FPGAs) are connected to the same PCIe network fabric, enabling direct remote memory access

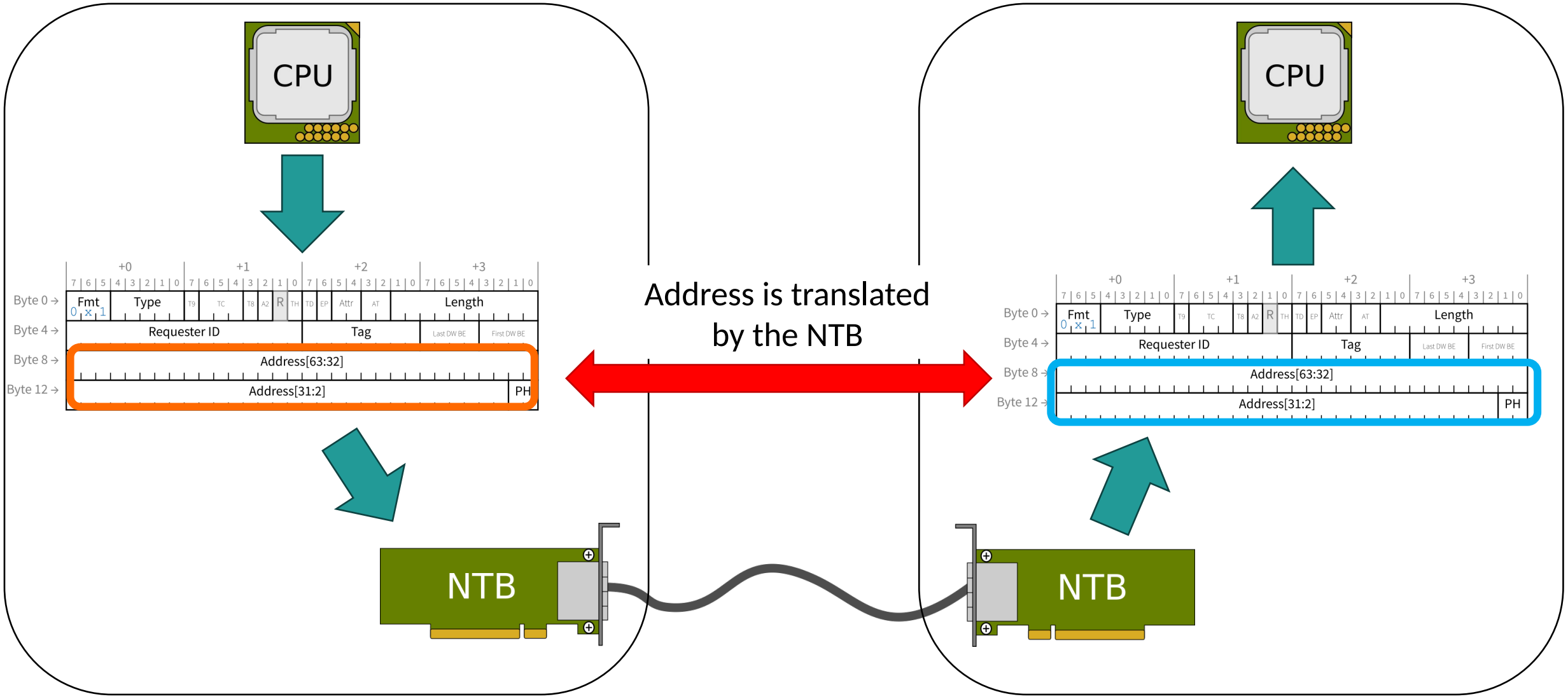




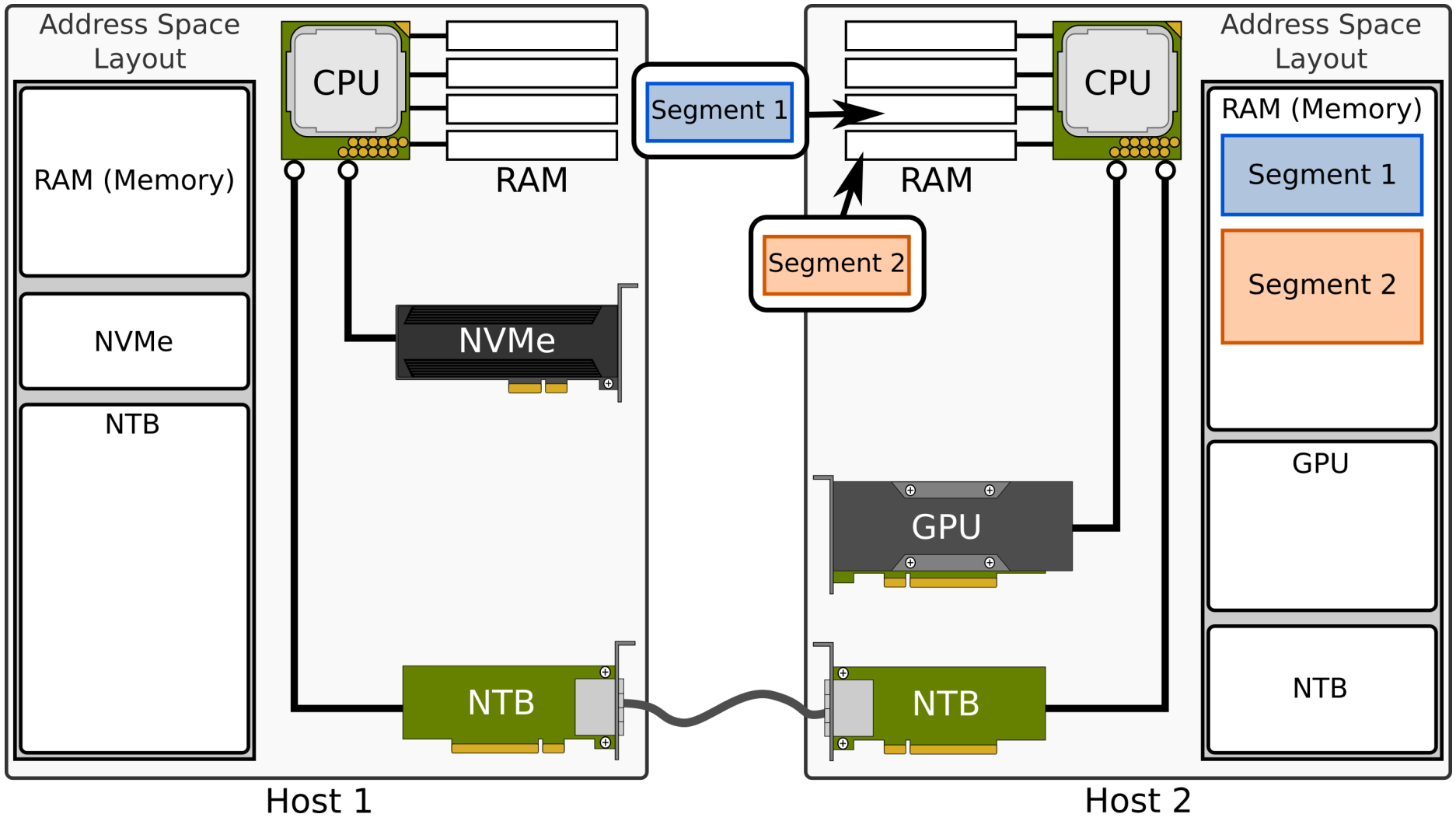


Non-Transparent Bridging

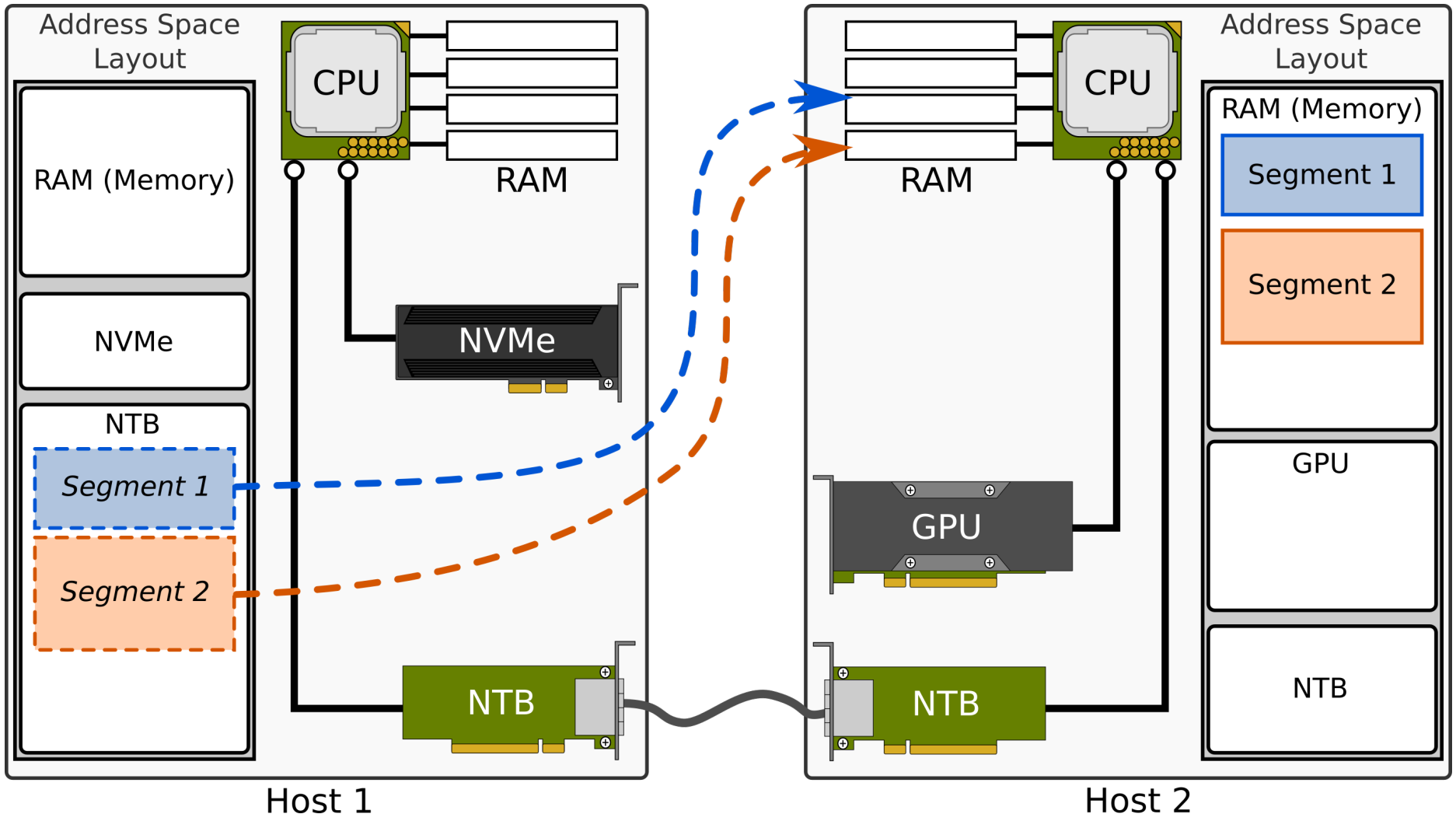
Non-Transparent Bridges (NTB) are a special kind of device that can forward memory transactions from the address space of one computer to another



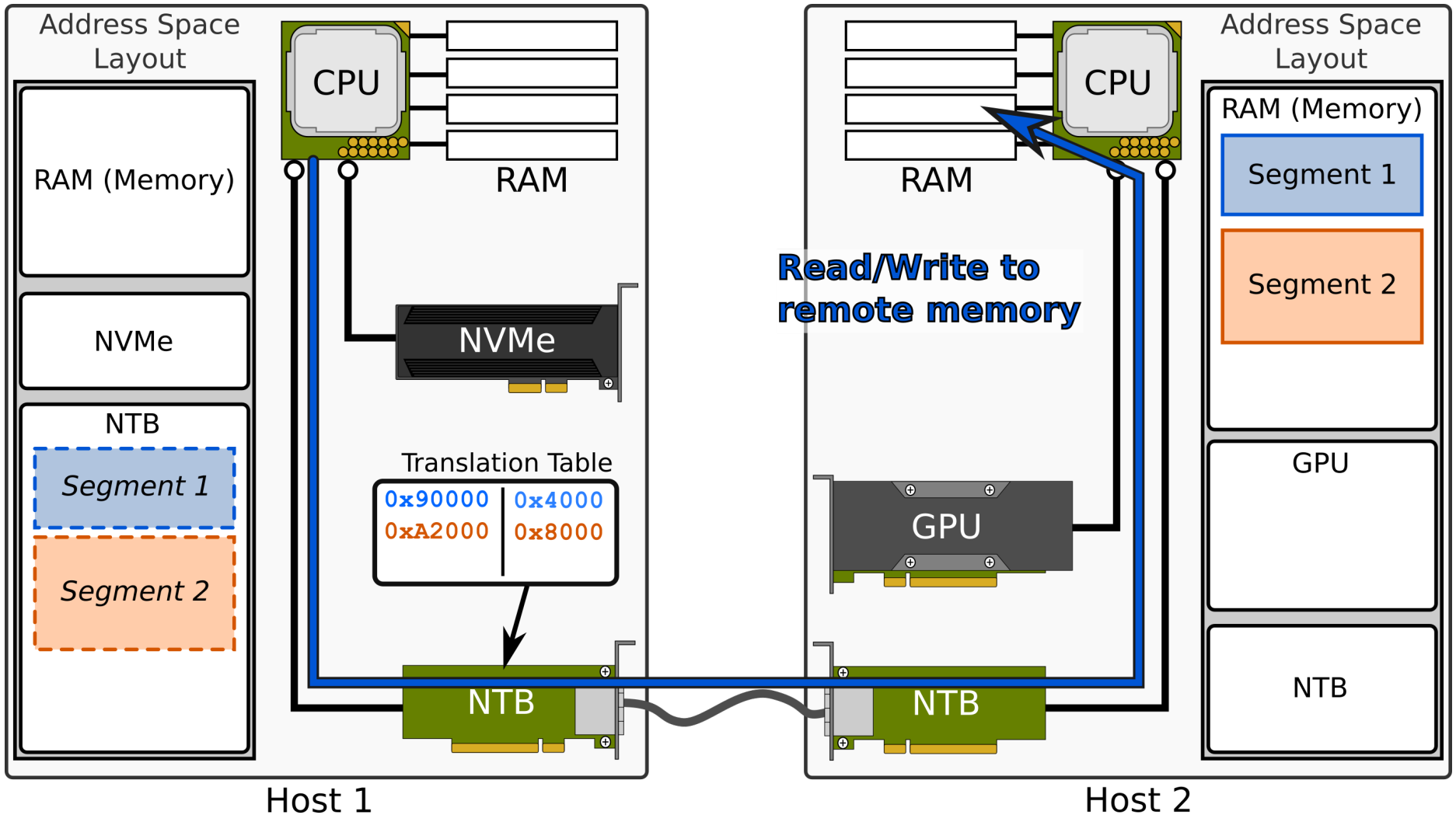
NTBs make it possible to memory-map different address offsets (segments) in a remote system, so a CPU can access remote memory using MMIO



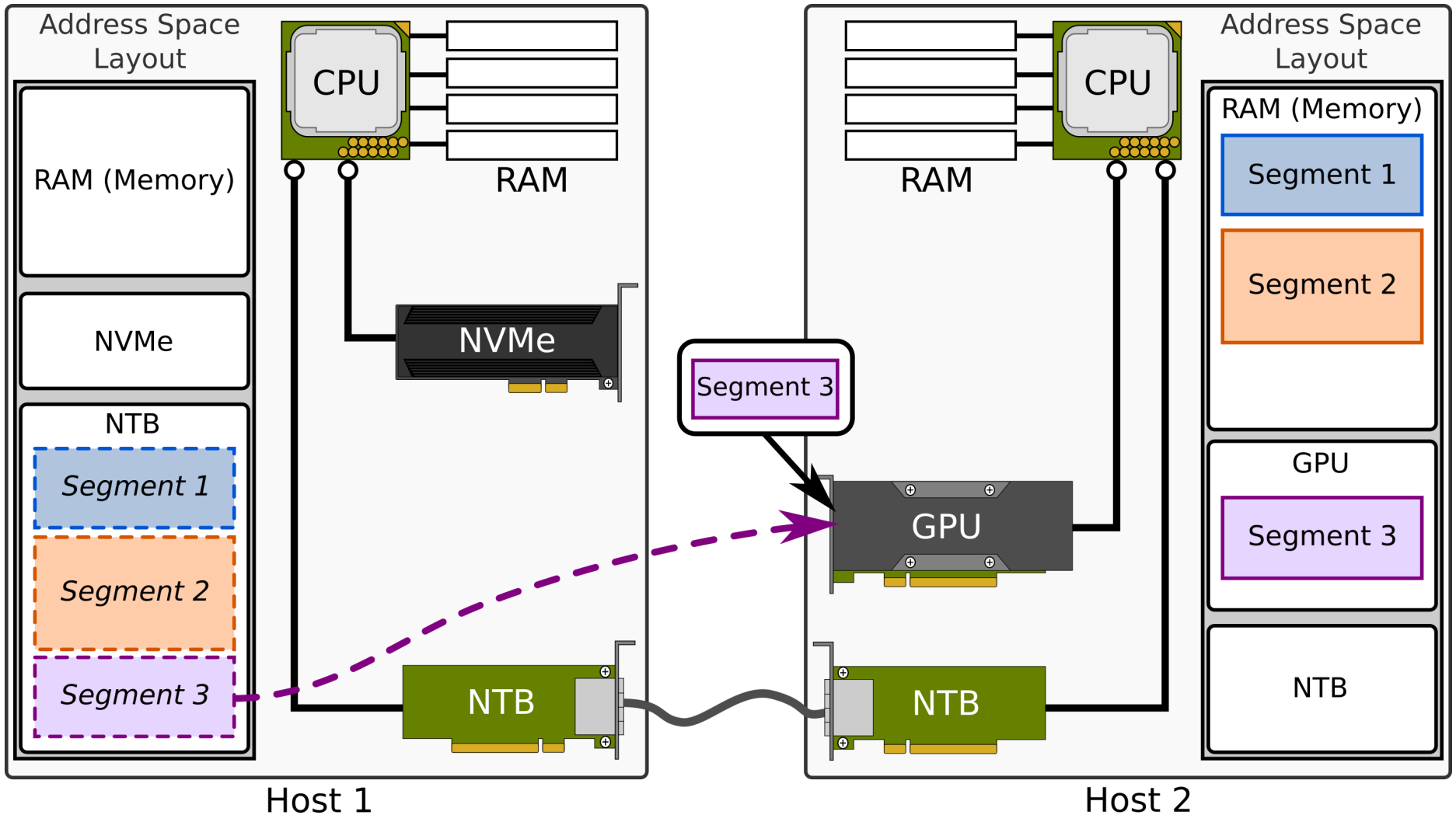
NTBs make it possible to memory-map different address offsets (segments) in a remote system, so a CPU can access remote memory using MMIO

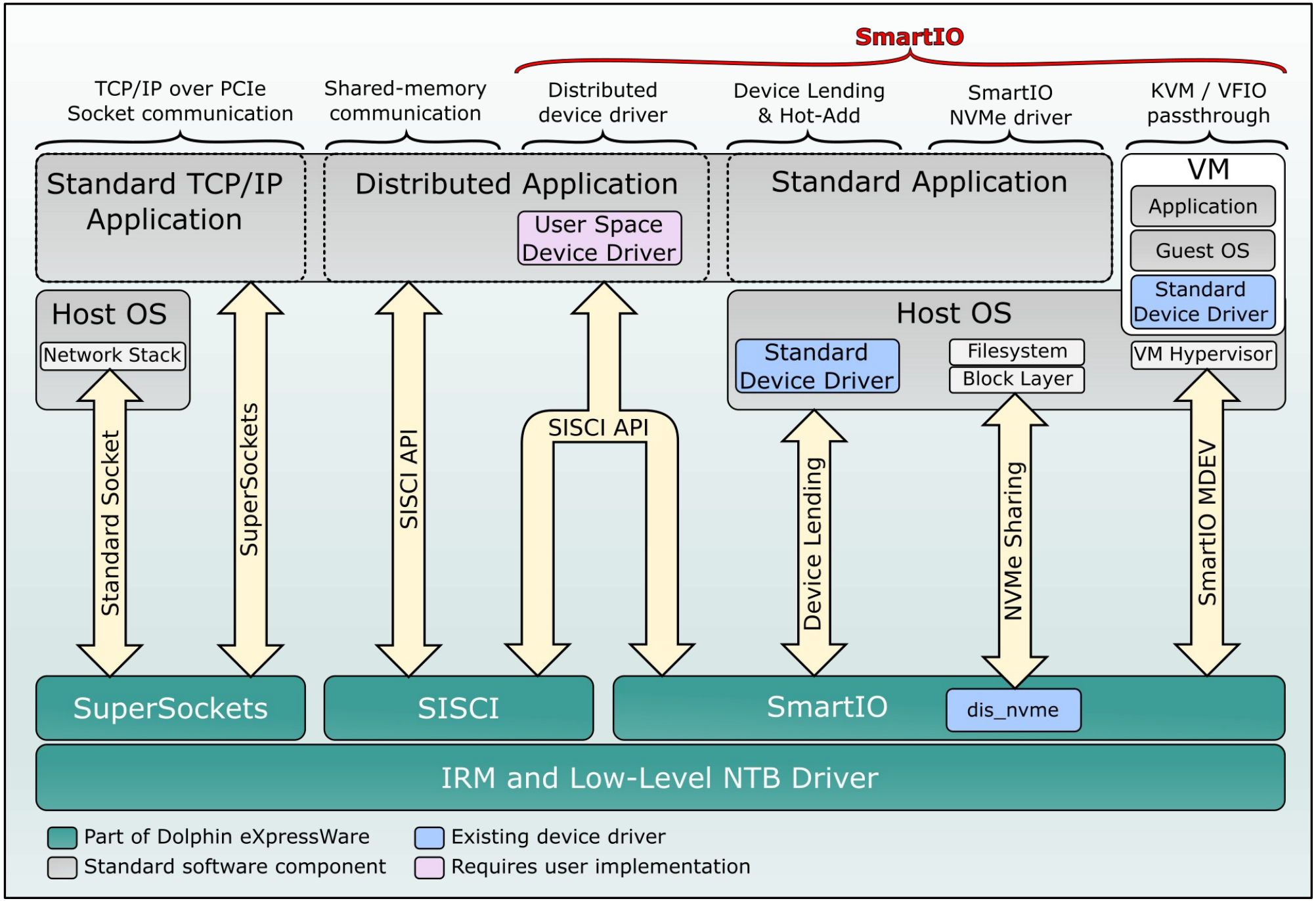


NTBs make it possible to memory-map different address offsets (segments) in a remote system, so a CPU can access remote memory using MMIO



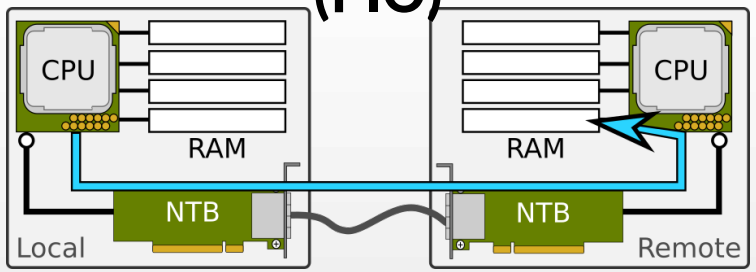
Since device memory is part of the same address space, memory regions of a remote device can also be mapped using an NTB




SmartIO
TCP/IP over PCIe Socket communication
Shared-memory communication
Distributed device driver
Device Lending & Hot-Add
SmartIO NVMe driver
KVM / VFIO passthrough
Standard TCP/IP Application
Distributed Application
User Space Device Driver
Standard Application
VM
Application
Guest OS
Standard Device Driver
Host OS
Network Stack
Host OS
Standard Device Driver
Filesystem
Block Layer
VM Hypervisor
Standard Socket
SuperSockets
SISCI API
SISCI API
Device Lending
NVMe Sharing
SmartIO MDEV
SuperSockets
SISCI
SmartIO
dis_nvme
IRM and Low-Level NTB Driver
Part of Dolphin eXpressWare
Standard software component
Existing device driver
Requires user implementation

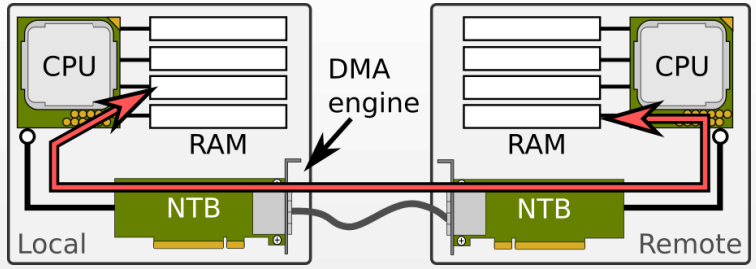
Dolphin provides a shared-memory API on top of PCIe NTBs called SISCI

Memory-mapped access (PIO)



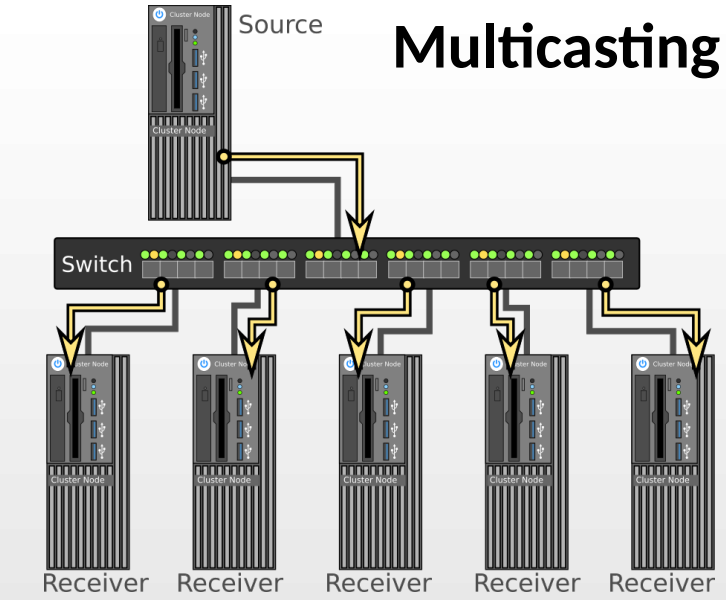
- Remote memory access based on shared memory segments
- CPU can natively read & write to remote memory
- Sub-microsecond latency (100s of nanoseconds)

DMA Transfers



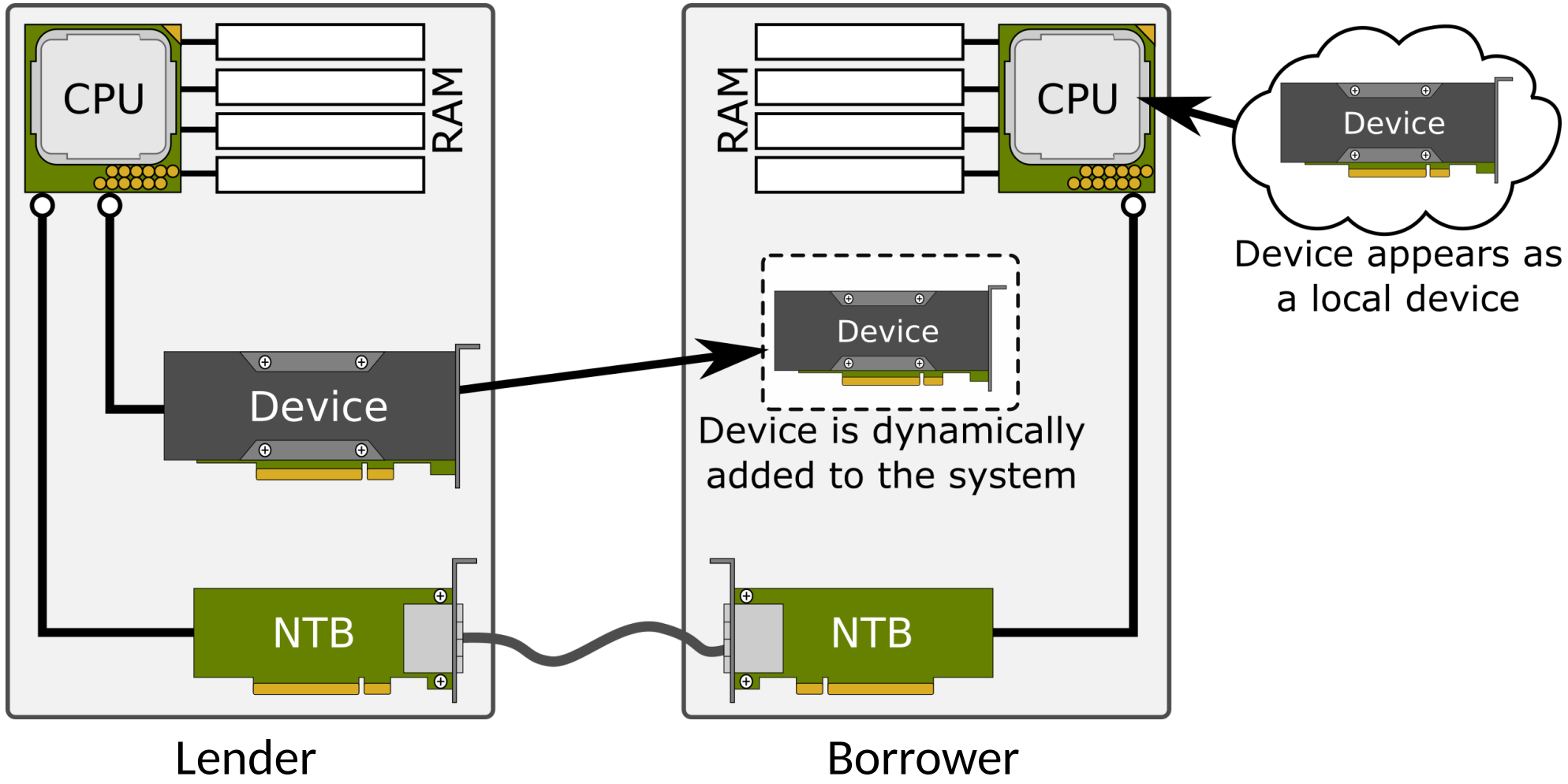
- High data throughput
- Offload data transfer to adapter -> Low CPU load
- Can be used for pipelining data streaming

Multicasting



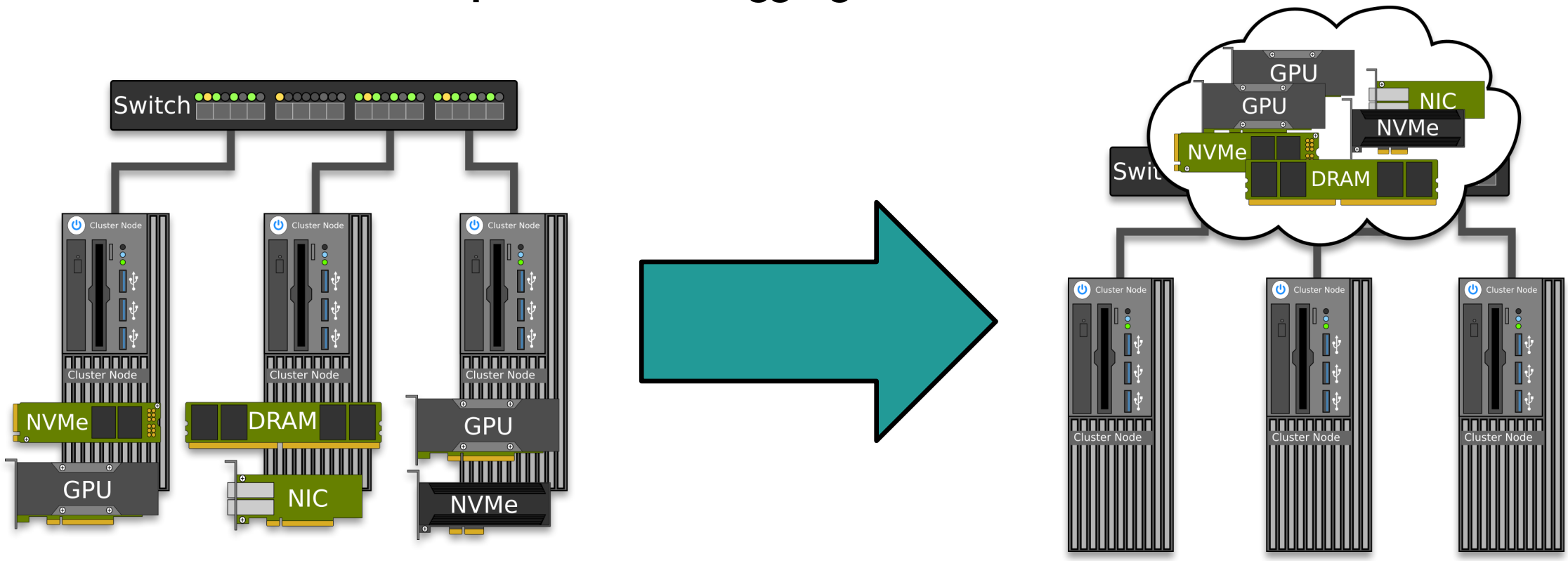
- Efficiently replicate data across several nodes
- May also DMA to multicast addresses

Dolphin's SmartIO solutions make it possible to dynamically share the internal devices of a host with remote hosts in the PCIe cluster



Using Dolphin SmartIO, resource utilization is increased as resources are decoupled from servers and can instead be pooled and shared

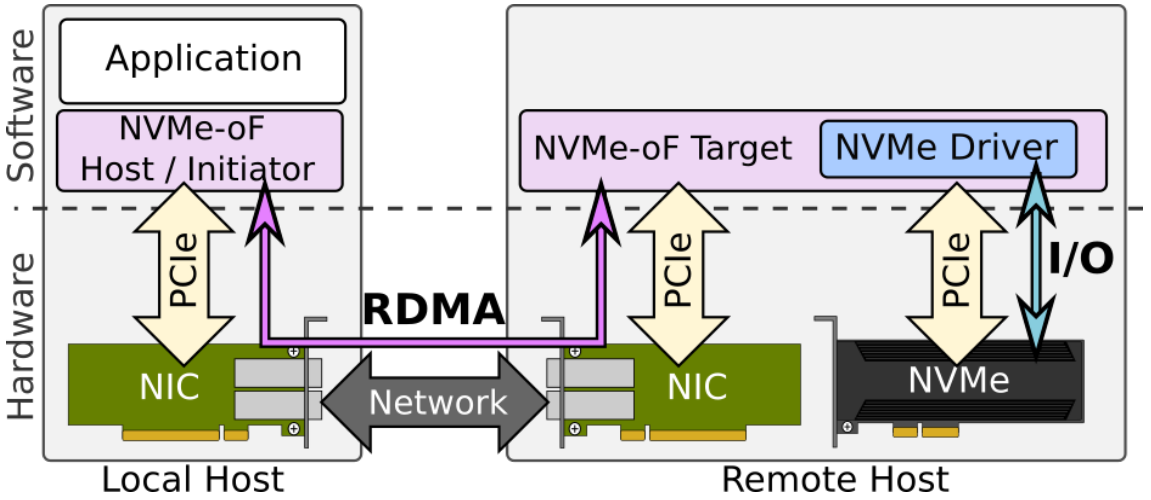
"Composable & Disaggregated Infrastructure"



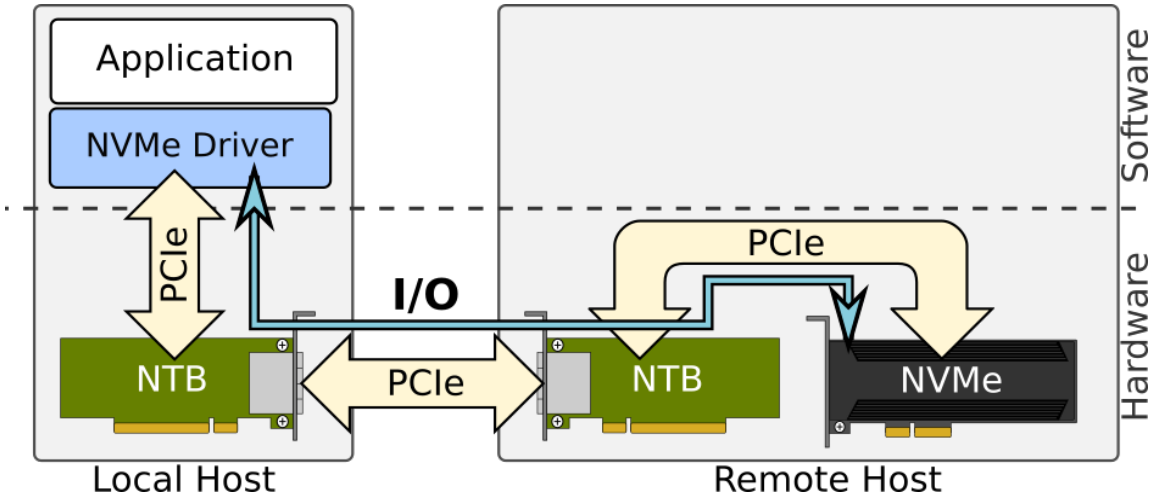
Traditional Server Nodes

Resource Pooling & Sharing

Device sharing is facilitated at the hardware level by using NTBs, in contrast to traditional distributed I/O over a network

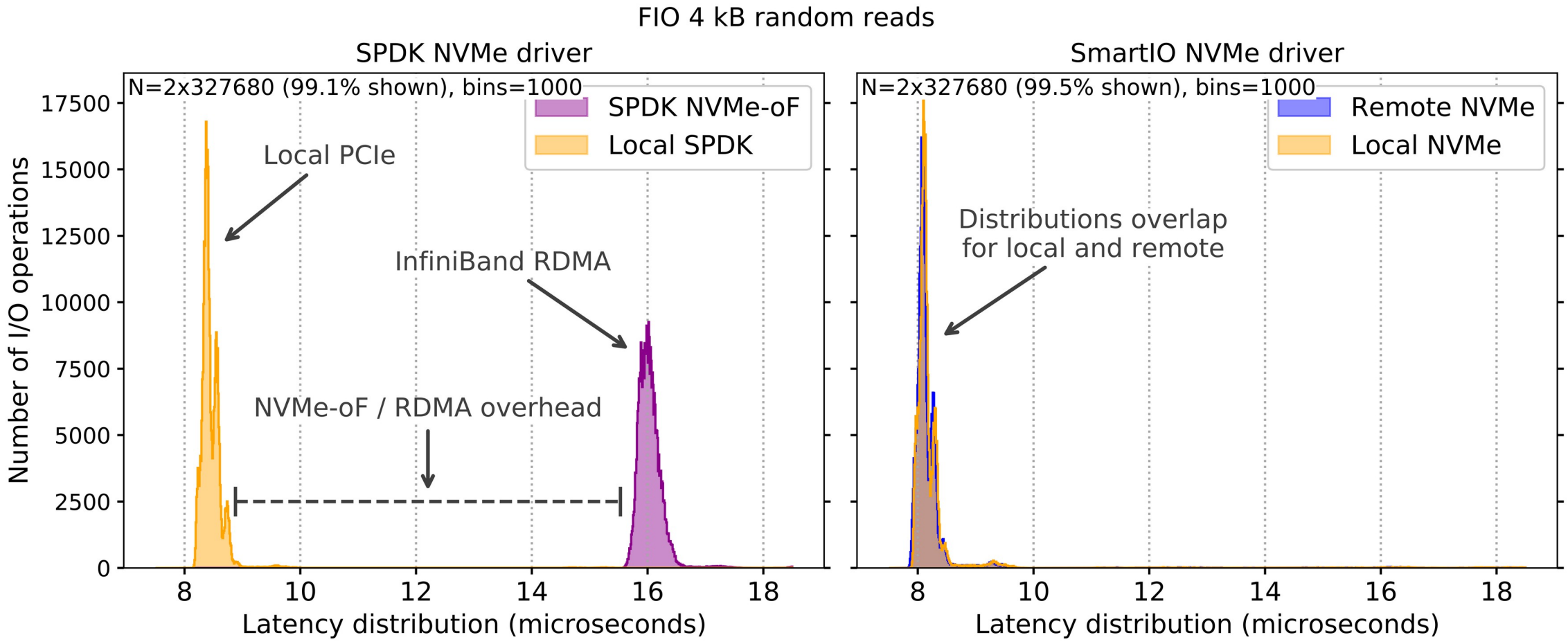


NVMe over Fabrics (NVMe-oF)



NVMe over PCIe

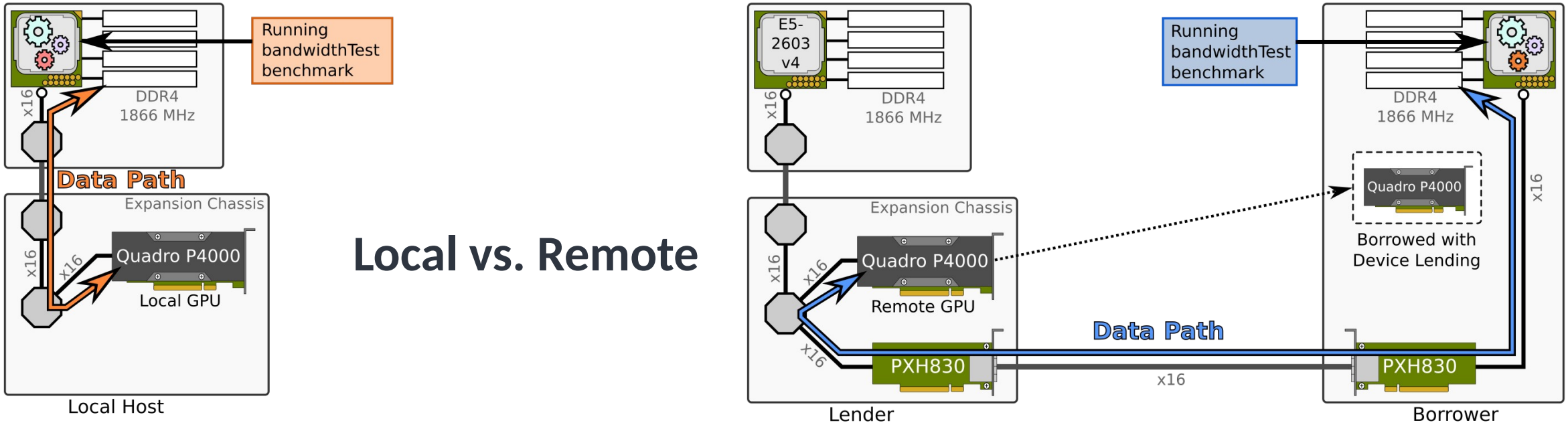
Device sharing is facilitated at the hardware level by using NTBs, in contrast to traditional distributed I/O over a network

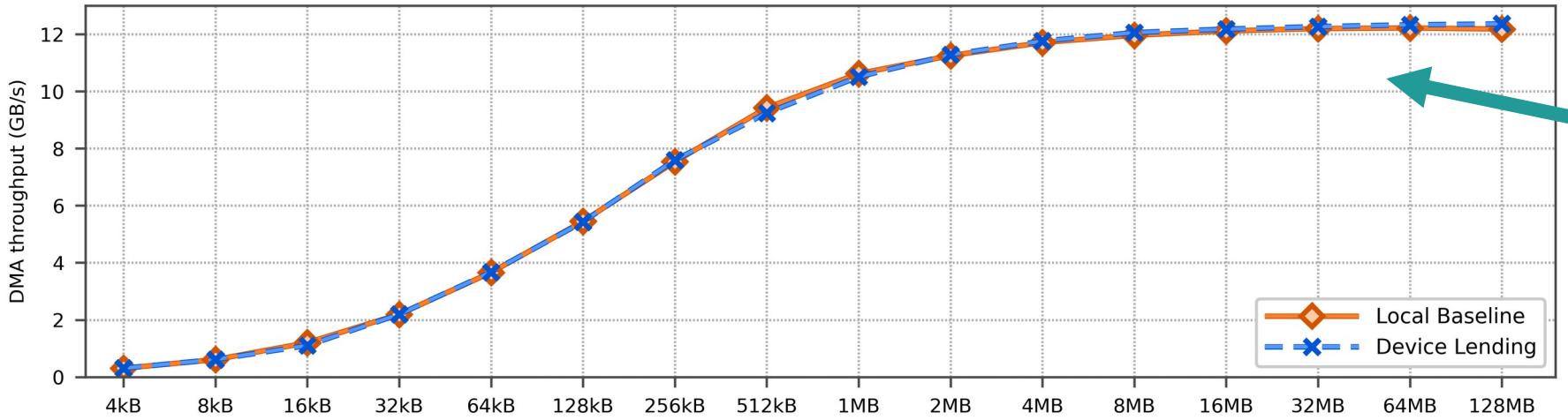


NVMe over Fabrics (NVMe-oF)

NVMe over PCIe

SmartIO allows remote devices to be used with native PCIe performance

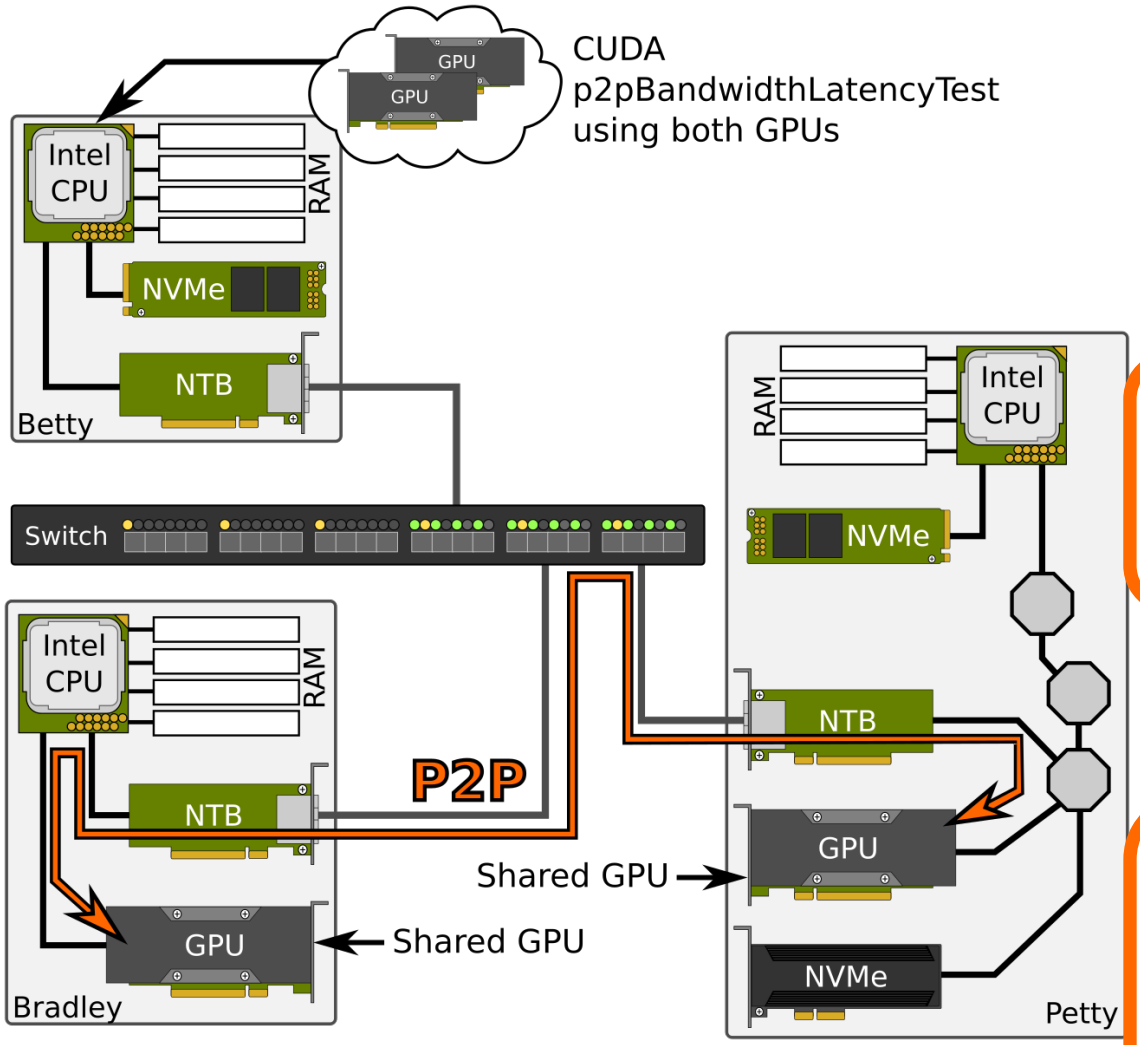




Same perf. for local and remote

bandwidthTest DtoH

SmartIO allows remote devices to be used with native performance




```
root@betty:~# /usr/local/cuda/samples/1_Utilities/p2pBandwidthLatencyTest/p2pBandwidthLatencyTest
[P2P (Peer-to-Peer) GPU Bandwidth Latency Test]
Device: 0, Quadro P4000, pciBusID: 4, pciDeviceID: 2, pciDomainID:0
Device: 1, Quadro P4000, pciBusID: 4, pciDeviceID: 3, pciDomainID:0
Device=0 CAN Access Peer Device=1
Device=1 CAN Access Peer Device=0

***NOTE: In case a device doesn't have P2P access to other one, it falls back to normal memcopy pro
So you can see lesser Bandwidth (GB/s) and unstable Latency (us) in those cases.

P2P Connectivity Matrix
     D\D     0     1
     0       1     1
     1       1     1
Unidirectional P2P=Disabled Bandwidth Matrix (GB/s)
   D\D     0      1
     0 193.76   2.65
     1   4.95 193.09
Unidirectional P2P=Enabled Bandwidth (P2P Writes) Matrix (GB/s)
   D\D     0      1
     0 193.57   9.07
     1   7.99 193.38
Bidirectional P2P=Disabled Bandwidth Matrix (GB/s)
   D\D     0      1
     0 194.44   4.26
     1   4.26 190.36
Bidirectional P2P=Enabled Bandwidth Matrix (GB/s)
   D\D     0      1
     0 193.38  14.71
     1  14.71 193.38
P2P=Disabled Latency Matrix (us)
   GPU     0      1
     0   2.09  22.69
     1  22.57   2.23

   CPU     0      1
     0   4.88  11.90
     1  12.05   4.91
P2P=Enabled Latency (P2P Writes) Matrix (us)
   GPU     0      1
     0   2.07   2.33
     1   2.38   2.21
```

2.6 GB/s
vs
9 GB/s

22.6 µs
vs
2.3µs


Dolphin
INTERCONNECT
SOLUTIONS