Network Intelligence: NIC, switch or xPU?

Bojie Li

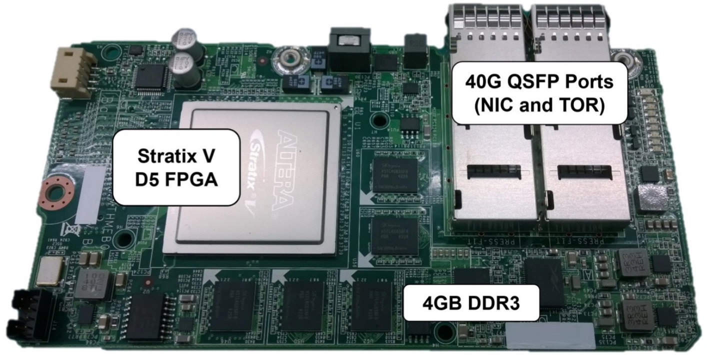
Co-Founder, Logenic AI

Sep 16, 2023

Contents

- Trend 1: Intelligent Network Devices
  - SmartNIC: FPGA, ASIC, NP and DPU
  - Programmable Switch
- Trend 2: Fast Interconnect
  - NVLink and CXL: Direct P2P with Memory Semantics
  - Convergence of AI and Cloud Networking

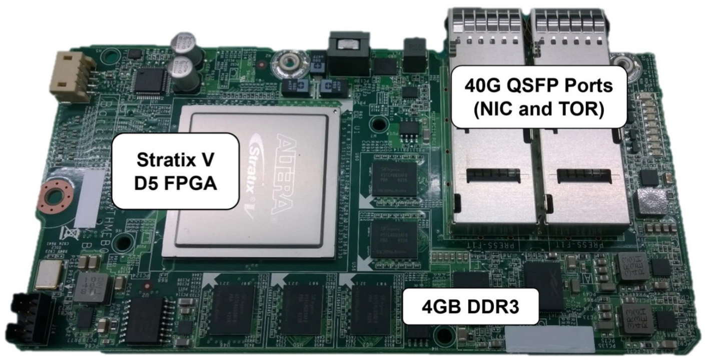
Trend 1: Intelligent Network Devices



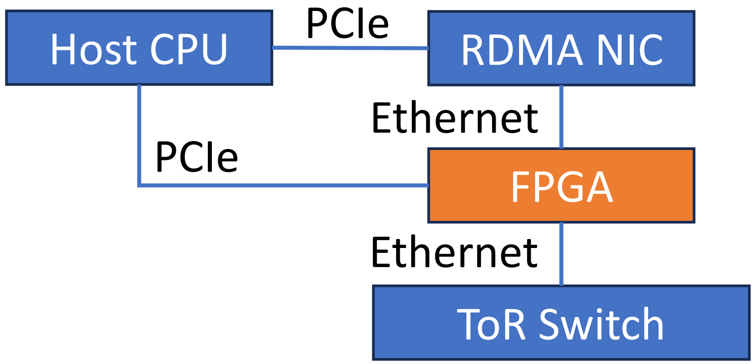
SmartNICs
(Microsoft)

Programmable Switches
(Barefoot Tofino)

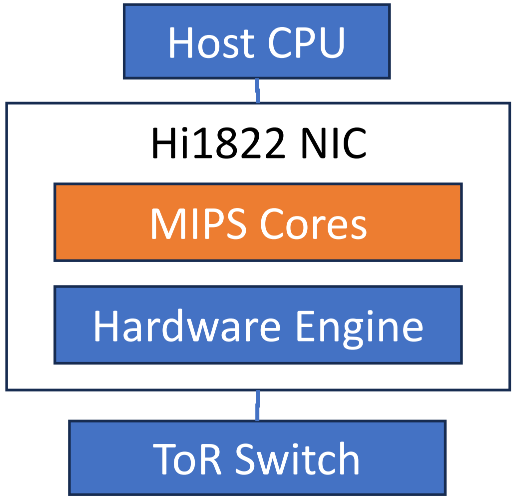
Types of SmartNICs

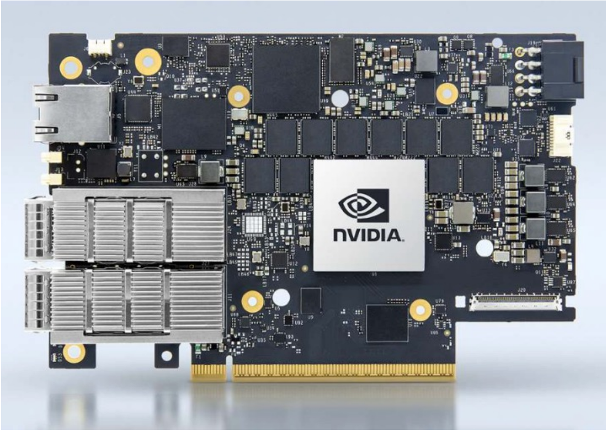


FPGA/ASIC On-Path SmartNICs
(Microsoft Catapult)

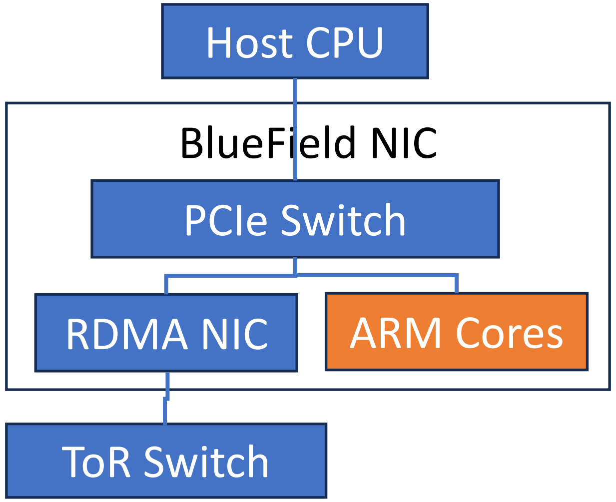


NP-based On-Path SmartNICs
(Huawei Hi1822)





Off-Path SmartNICs (DPU)
(Mellanox BlueField)



FPGA-based On-Path SmartNICs

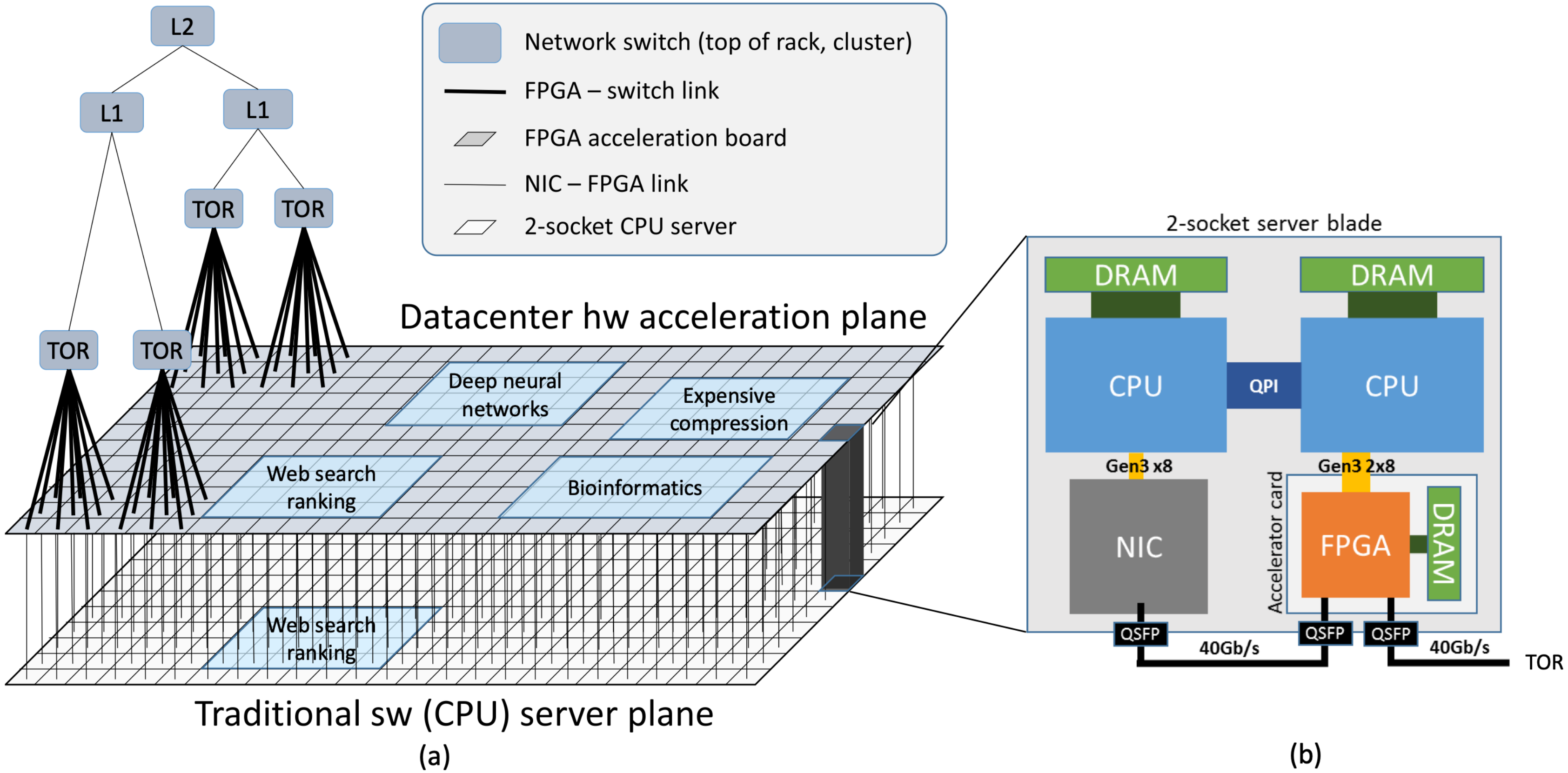


Fig. 1. (a) Decoupled Programmable Hardware Plane, (b) Server + FPGA schematic.

A Cloud-Scale Acceleration Architecture, ISCA '16 (Microsoft)

Since 2016, **every new server in Azure has deployed an FPGA-based SmartNIC**

- Network virtualization
- Storage virtualization
- Bing ranking acceleration
- Compression acceleration
- Encryption acceleration

SmartNICs in the Public Cloud

- Network Virtualization consumes CPUs, e.g., 5 physical cores per host.
- Each physical core sells for $0.1 per hour.
  - Max potential value $900 per year.
  - $4500 over the lifetime of a server.
- How much does an FPGA-based SmartNIC cost?
  - Less than $1000 when purchased in large bulk.

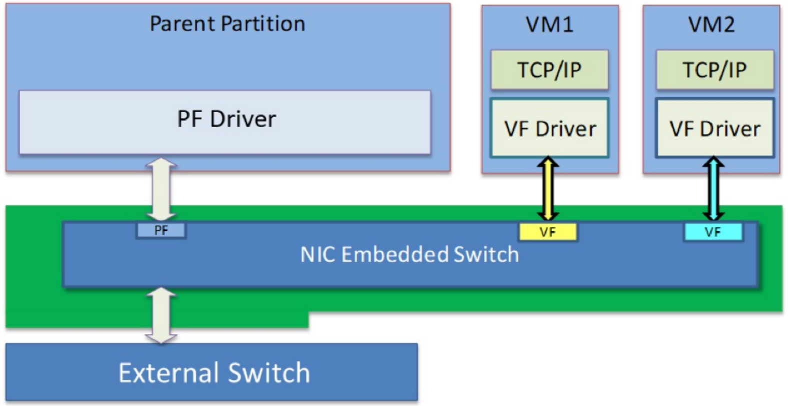


Figure 1: An SR-IOV NIC with a PF and VFs.

2. Aren't FPGAs very expensive?

While we cannot disclose vendor pricing publicly, the FPGA market is competitive (with 2 strong vendors), and we're able to purchase at significant volumes at our scale. In our experience, our scale allows non-recoverable engineering costs to be amortized, and the cost of the silicon becomes dominated by the silicon area and yield. Total silicon area in a server tends to be dominated by CPUs, flash, and DRAM, and yields are typically good for FPGAs due to their regular structure.

Azure Accelerated Networking: SmartNICs in the Public Cloud, NSDI '18 (Microsoft)

SmartNICs in the Public Cloud

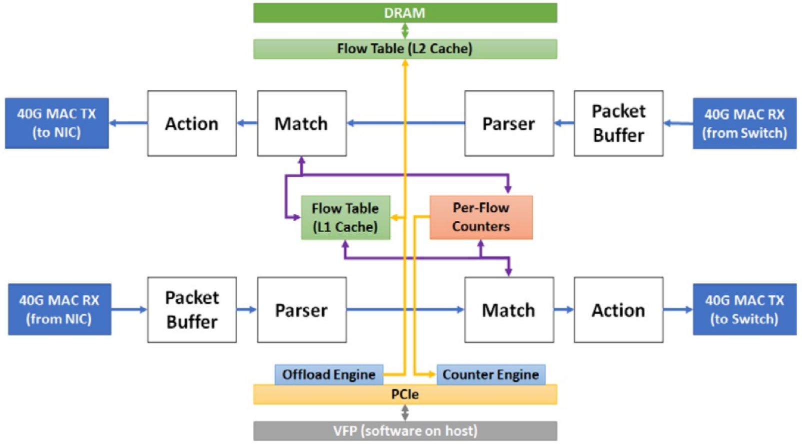


Figure 4: Block diagram of the GFT processing pipeline

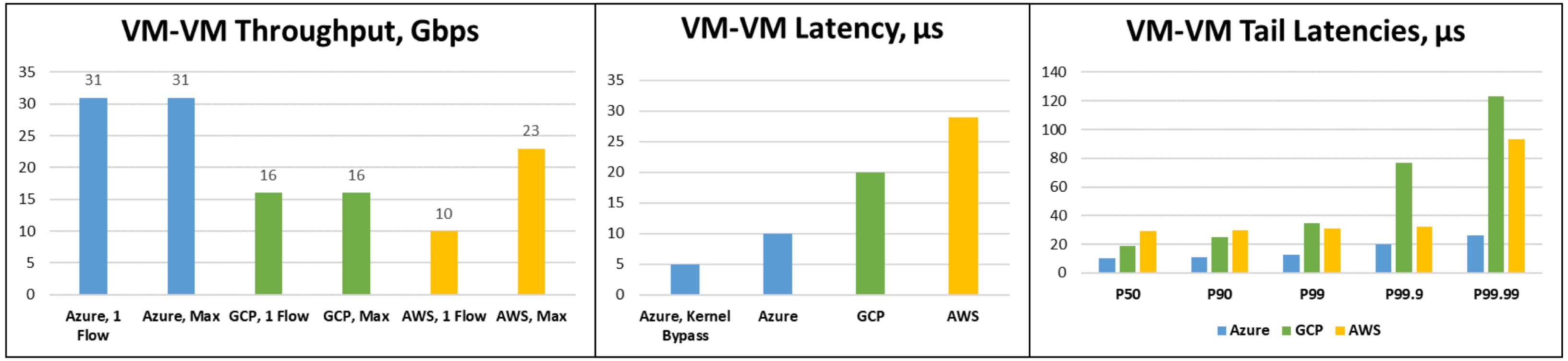


Figure 6: Performance of AccelNet VM-VM latencies vs. Amazon AWS Enhanced Networking and Google GCP Andromeda on Intel Skylake generation hardware.

Azure Accelerated Networking: SmartNICs in the Public Cloud, NSDI ’18 (Microsoft)

Challenges of FPGA-based SmartNICs

Compared architecture	FPGA challenges	Solutions
CPU/GPU/NP/SoC	Low clock frequency	Utilize the massive parallelism inside FPGA
CPU/GPU	Low DRAM memory bandwidth	Customize data path, parallel use of on-chip BRAM memory, reduce DRAM usage
CPU/GPU/NP/SoC	Hardware description language programming is complex and difficult to debug	Programming framework friendly to software developers based on high-level synthesis technology
CPU/GPU/SoC	The software and hardware ecosystem is relatively closed	Open hardware platform, programming framework and IP core
CPU/GPU/NP/SoC	The chip area is limited and not suitable for scenarios with complex logic	Separate control plane and data plane; data plane based on customized instructions
CPU	High PCIe latency when accessing main memory	Design efficient data structures and use out-of-order execution to achieve latency hiding
CPU	Limited PCIe bandwidth when accessing main memory	Design efficient data structures and use on-board cache
CPU/GPU/NP/SoC	Need to rewrite for upgrades, interrupt service	FPGA operating system that supports dynamic reconfiguration and seamless service upgrades
CPU/NP/SoC	High task switching overhead	Spatial multiplexing, not time-division multiplexing
ASIC	Some compute-intensive loads are inefficient	Harden general modules into hard cores

FPGA Programming Made Simple

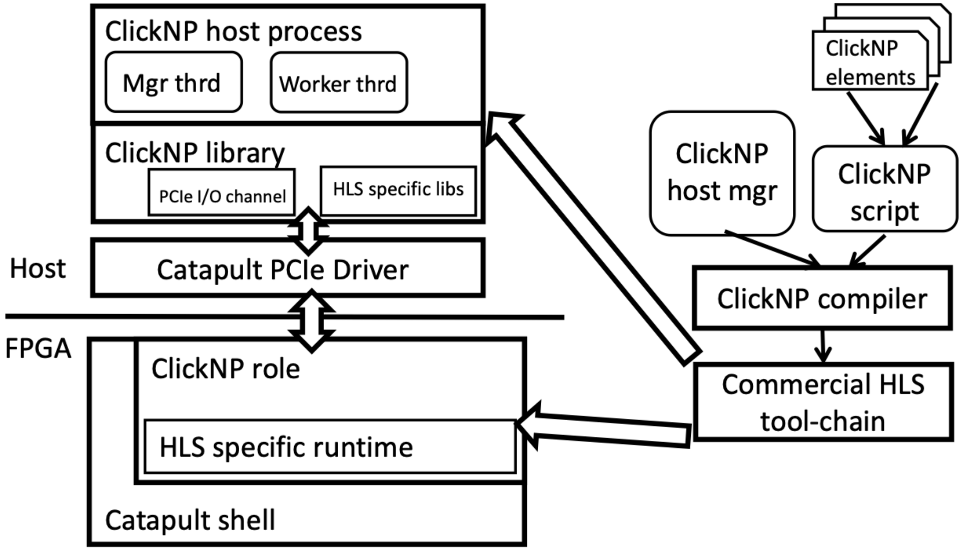


Figure 2: The architecture of ClickNP.

Table 3: Summary of ClickNP NFs.

Network Function	LoC†	#Elements	LE	BRAM
Pkt generator	665	6	16%	12%
Pkt capture	250	11	8%	5%
OpenFlow firewall	538	7	32%	54%
IPSec gateway	695	10	35%	74%
L4 load balancer	860	13	36%	38%
pFabric scheduler	584	7	11%	15%

† Total line of code of all element declarations and configuration files.

```
.element Count <1, 1> {
  .state{
    ulong count;
  }
  .init{
    count = 0;
  }
  .handler{
    if (get_input_port() != PORT_1) {
      return (PORT_1);
    }
    flit x;
    x = read_input_port(PORT_1);
    if (x.fd.sop) count = count + 1;
    set_output_port(PORT_1, x);

    return (PORT_1);
  }
  .signal{
    ClSignal p;
    p.Sig.LParam[0] = count;
    set_signal(p);
  }
}
```

(a)

```
Count :: cnt @
Tee :: tee
host PktLogger :: logger

from_tor -> cnt -> tee [1] -> to_tor
tee [2] -> logger
```

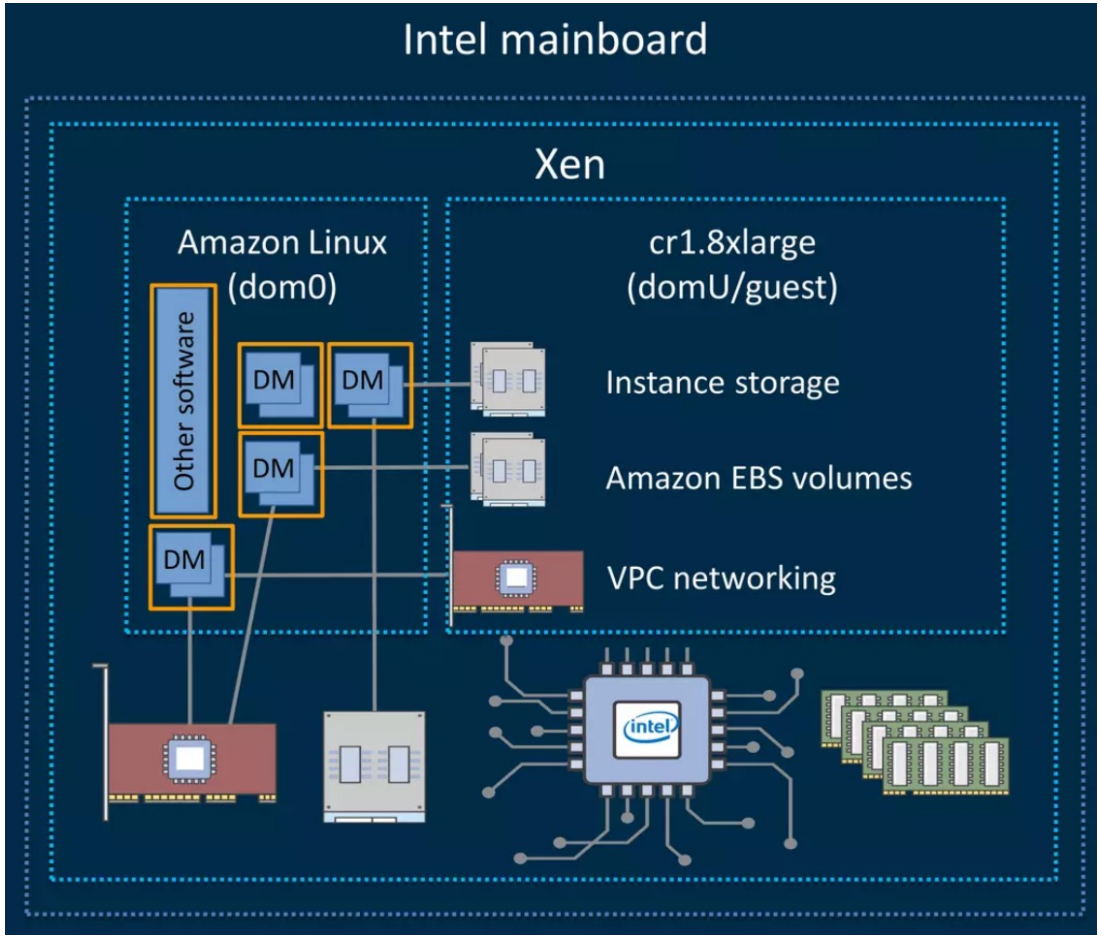
(b)

ClickNP: Highly Flexible and High Performance Network Processing with Reconfigurable Hardware, SIGCOMM '16

When to Use FPGA – the 10/100/1000 Rule

- 10 – 10 years of workload lifetime
  - If the workload is shifting too fast, FPGA is not as agile as CPUs
- 100 – 100 lines of C++ code
  - If the workload is too complicated, an FPGA implementation consumes too much area
- 1000 – 1000 servers
  - If the workload is too lightweight, the FPGA development cost is hard to amortize

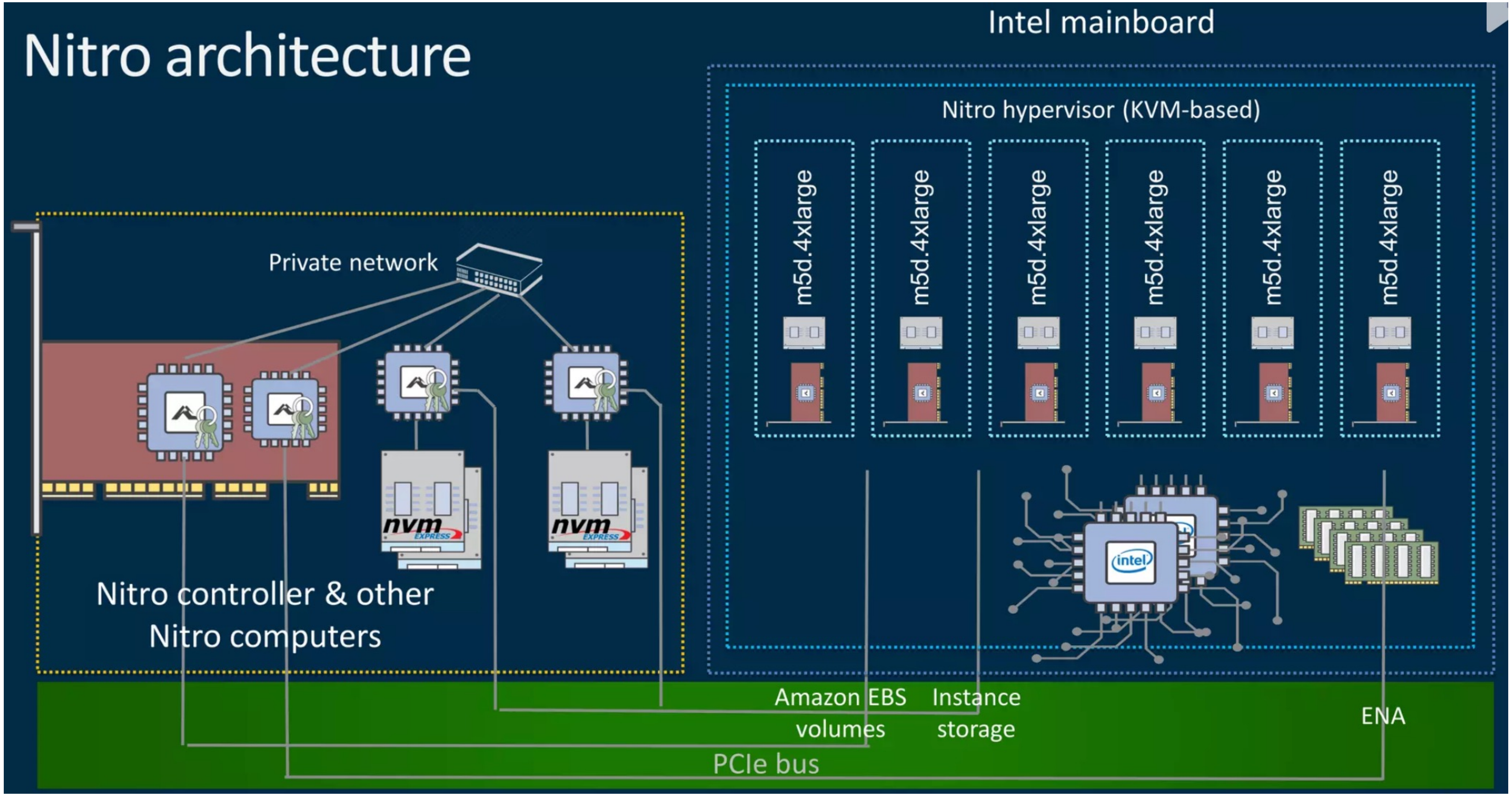
ASIC-based On-Path SmartNICs: AWS Nitro



- Traditional: Hypervisor (dom0) consumes several cores
  - Network virtualization (VPC)
  - Storage virtualization
    - Local instance storage
    - Network-attached EBS volumes
  - Management

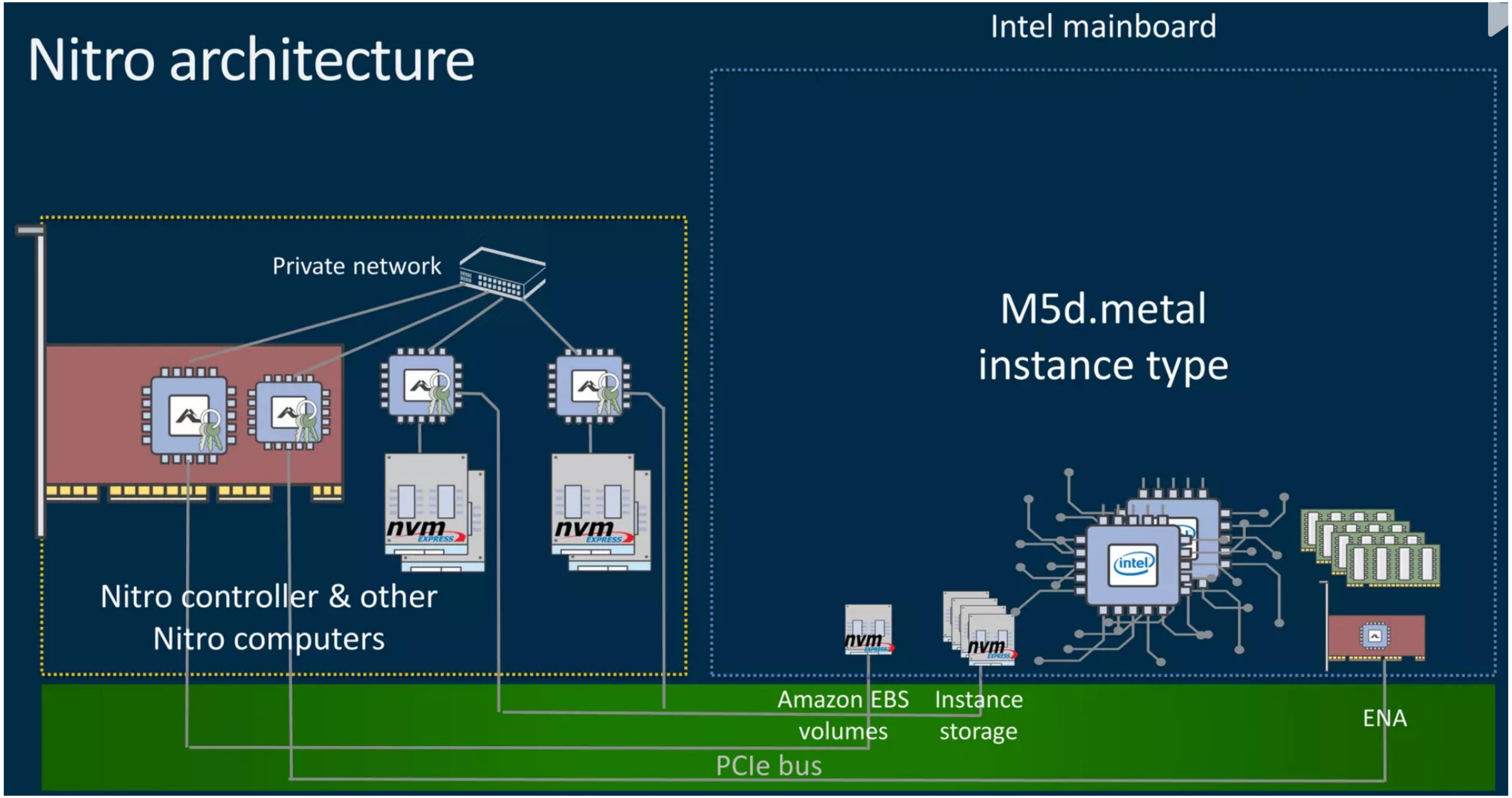
Security benefits of the Nitro architecture - SEP401-R - AWS re:Inforce 2019

AWS Nitro Removes Hypervisor CPU Cost



Security benefits of the Nitro architecture - SEP401-R - AWS re:Inforce 2019

AWS Nitro Enables Bare-Metal Instance



Security benefits of the Nitro architecture - SEP401-R - AWS re:Inforce 2019

AWS Nitro Provides Security Benefits

Integrity: Nitro system

Nitro controller is the root of trust

Nitro controller boots from completely private SSD

Boot process formally verified by AWS Automat
https://link.springer.com/chapter/10.1007/978-

Conducts various integrity checks of Nitro computers

Continues on with mainboard boot

When necessary, secure software updates for all components using secure channels, signed binaries

Integrity: M5d boot process

Mainboard cannot update firmware

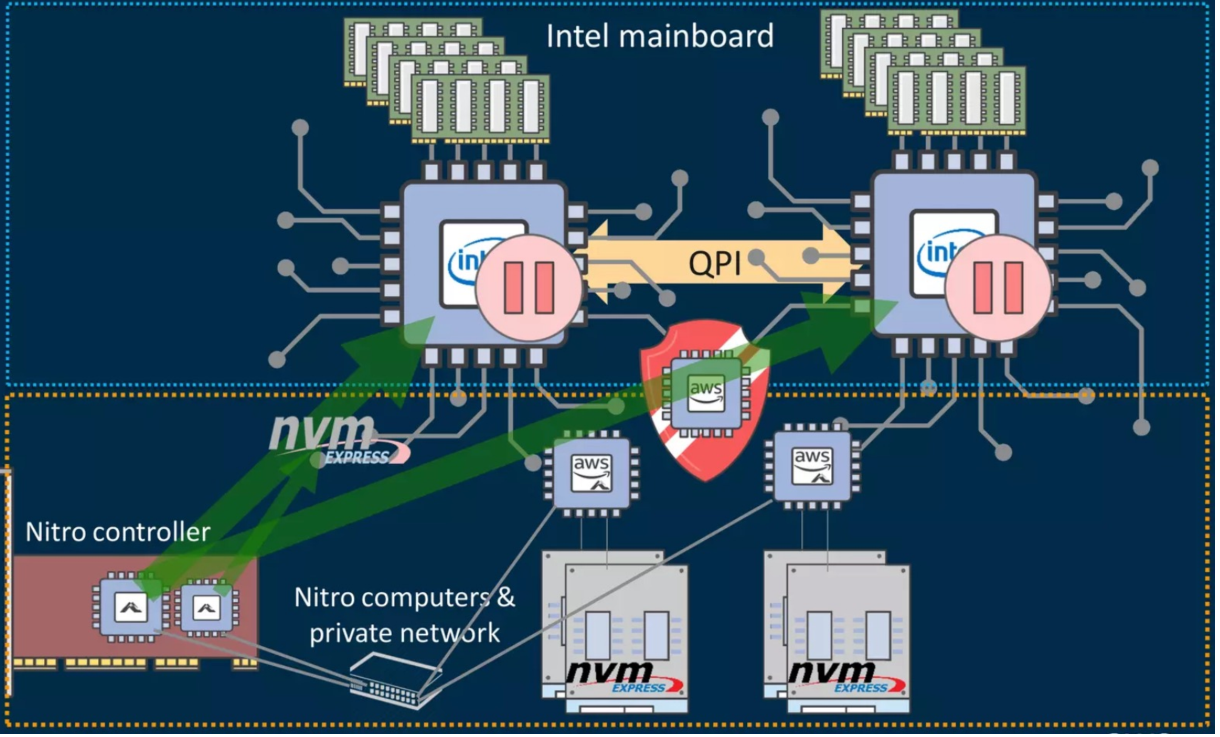
But…

Hold mainboard in reset during power-up

Validate all firmware; if valid, continue

Either inject known-good hypervisor

Or boot customer OS/hypervisor AMI from pseudo-NVMe (EBS) volume



Security benefits of the Nitro architecture - SEP401-R - AWS re:Inforce 2019

AWS Nitro Provides Security Benefits

Passive communications design

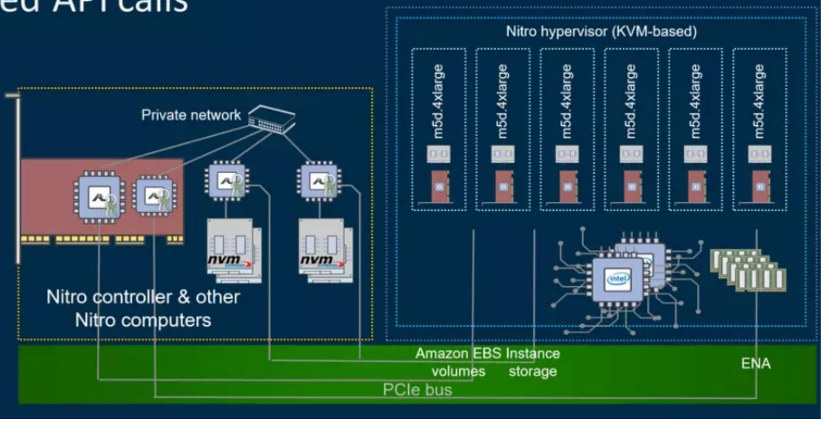
Hypervisor awaits commands from Nitro controller

- Sent via trusted communications channel
- Never initiates communications with the controller
- Not connected to the network at all

Nitro controller awaits commands from the external control plane

- Listens on network substrate awaiting encrypted, authenticated API calls
- Never initiates outbound connections

Outbound communications from either layer are a clear sign of compromise and are treated accordingly



Security benefits of the Nitro architecture - SEP401-R - AWS re:Inforce 2019

AWS Nitro Provides Security Benefits

Additional confidentiality benefits

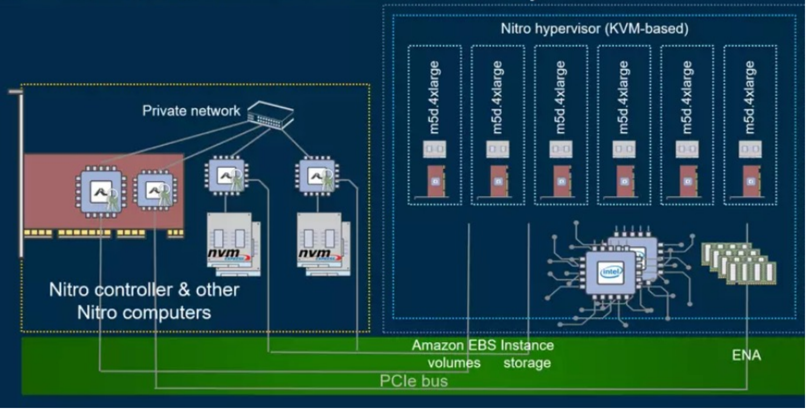
No Dom0 in Nitro hypervisor—greater simplicity and safety

No SSH or other interactive modes anywhere—no direct human access

All access via 100% AuthN/AuthZ APIs with logging/auditing
—no APIs for memory access

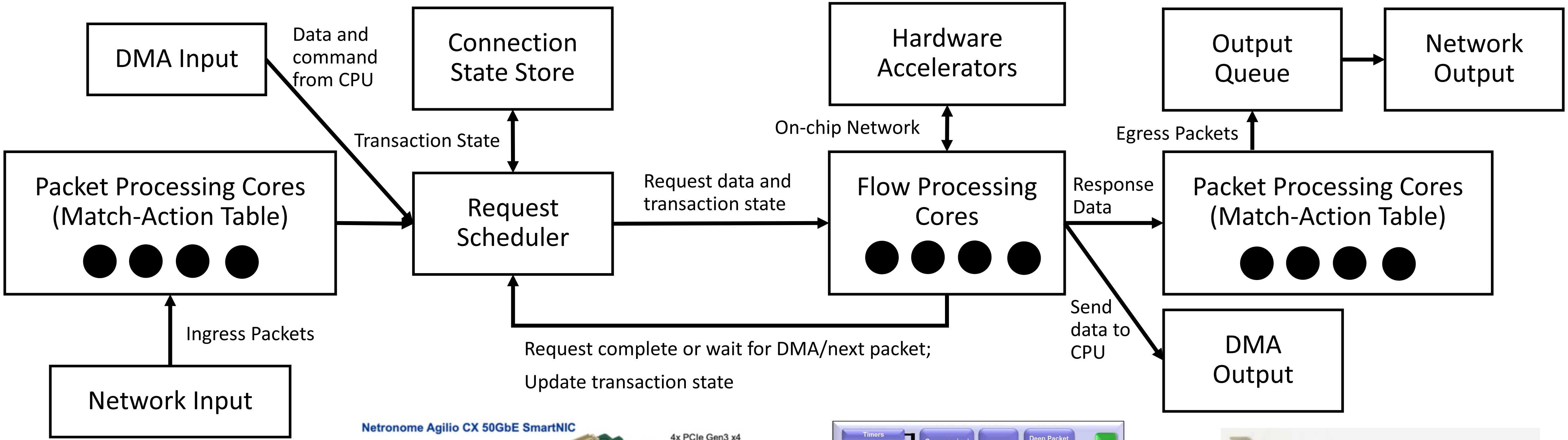
Only the Nitro controller has access to the physical Amazon EC2 network; the mainboard does not

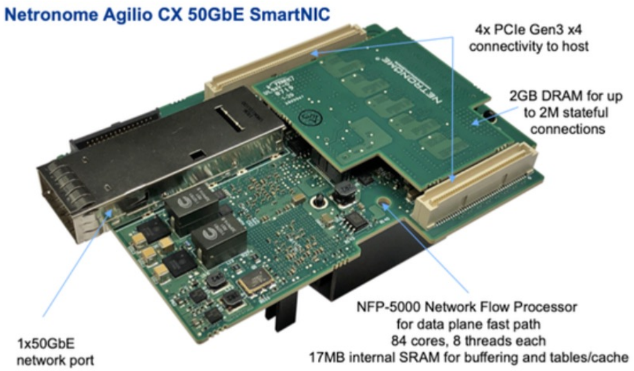
End-to-end Nitro system is developed, deployed, and managed by DevSecOps process



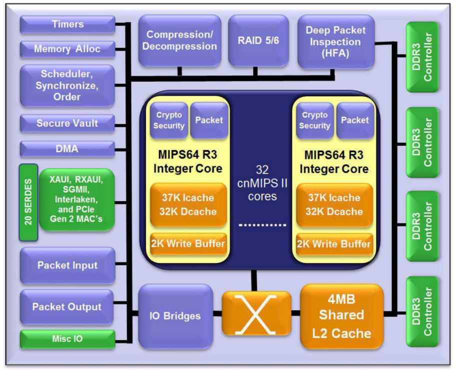
Security benefits of the Nitro architecture - SEP401-R - AWS re:Inforce 2019

NP-based On-Path SmartNICs





Netronome Agilio CX SmartNIC



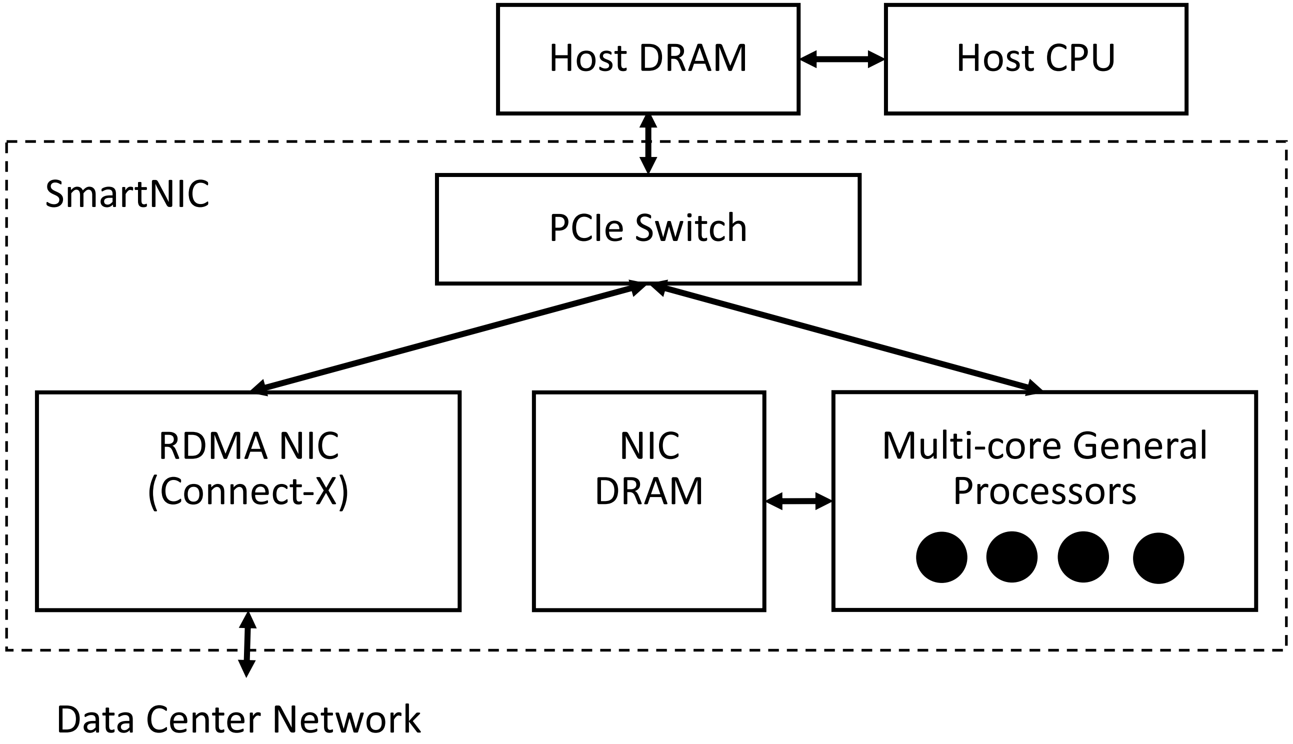
Cavium OCTEON II 68xx

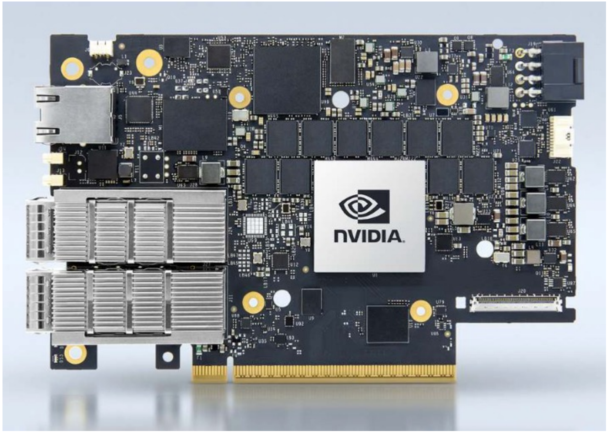
Huawei Hi1822 SmartNIC

NP-based SmartNICs: Hard to Program

- Need to read 1000+ pages of documents
  - 10+ data structure and 10+ packet processing accelerators on chip
- An NP-based RDMA implementation has 10K+ lines of C code
  - Limited instruction cache: cache thrashing if not optimized
  - Need to carefully arrange the hot paths to make sure it fits in the cache
- Need to pipeline accesses to DMA and data structures on chip
  - A classical latency hiding problem: the trade-off between polling and context switch
  - When the flow processing cores waits for data structure access and locks, if it switches context to other QP contexts, loading the context takes time
  - Typical latencies: on-chip data structure access < context switch < DMA

SoC-based Off-Path SmartNICs (aka. DPU)





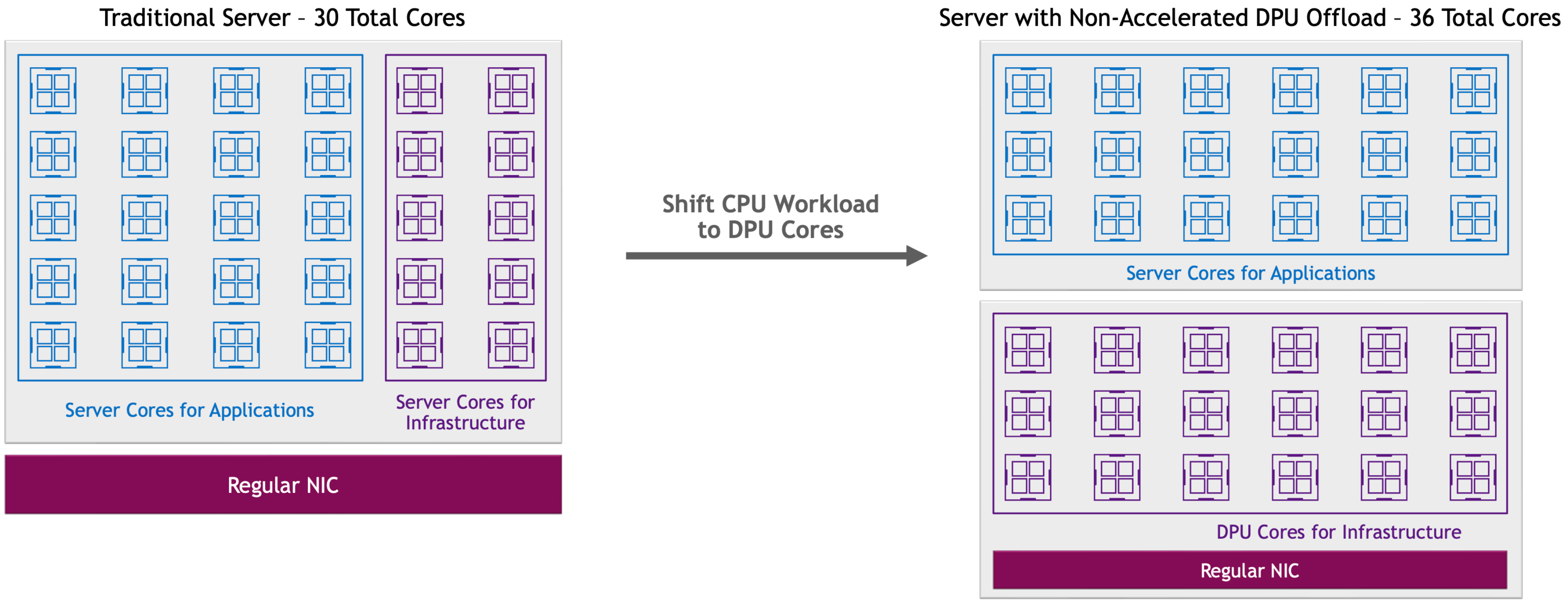
Mellanox BlueField

Off-Path vs. NP-based On-Path SmartNICs

Comparison item	Multi-core general-purpose processor (SoC)	Network processor (NP)
Instruction type	Standard ARM / MIPS instruction set	Extended ARM / MIPS instruction set
Operating system	General-purpose operating system (such as Linux)	No operating system or customized operating system
Operating system, paging, etc.	Supported	Generally not supported
Context switch and scheduling	Software operating system	Hardware
Locks, timers, etc.	Software	Hardware
On-board/core communication	Shared memory	Custom data path
Packet buffer	Off-chip DRAM	On-chip high-speed cache
Packet processing framework	General (such as DPDK)	Dedicated
Multi-core queuing model	d-FCFS (hardware dispatch)	c-FCFS (hardware scheduling)
Average processing latency	About 5 μs	Less than 2 μs
Single-core processing capacity	About 3 M pps	About 1 M pps
Number of processor cores	About 8	About 64
Total packet processing capacity	About 24 M pps	About 64 M pps
Power consumption	10 W to 20 W	

DPU is Not Yet Another CPU

NAIVELY MOVING WORKLOADS TO NIC CPUS DOESN’T WORK



18 DPU Cores Replace 12 Server CPU Cores —
No Gain in Performance or Efficiency
Not compatible for higher bandwidth without requiring significant system modification

NVIDIA DATA CENTER PROCESSING UNIT (DPU) ARCHITECTURE, HotChips 2021

DPU Must Include Hardware Acceleration

NVIDIA DPU SYSTEM ARCHITECTURE

Server Class CPU subsystem

- Data center operating system control plane
- Isolated memory subsystem optimized for networking

NIC subsystem

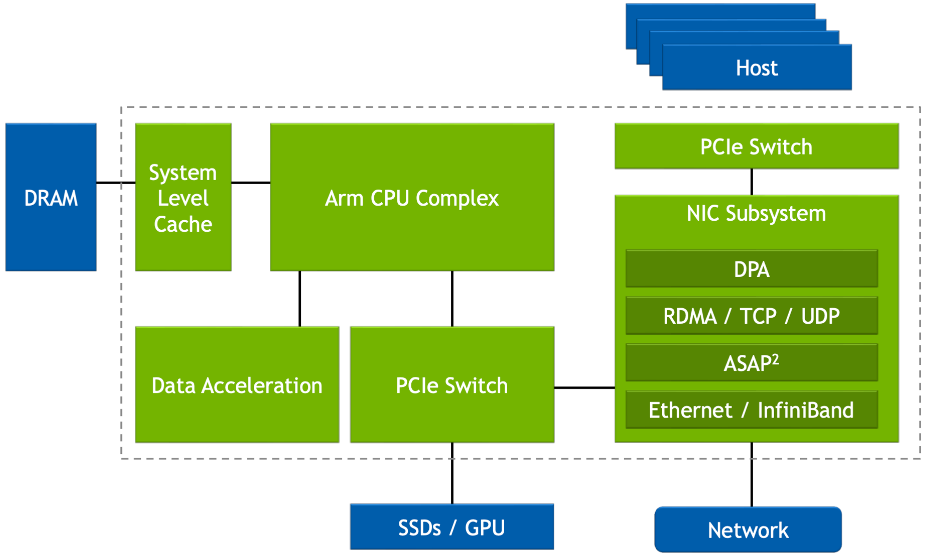
- Isolated boot domain, real time OS
- Accelerating data path at line rate

PCIe subsystem

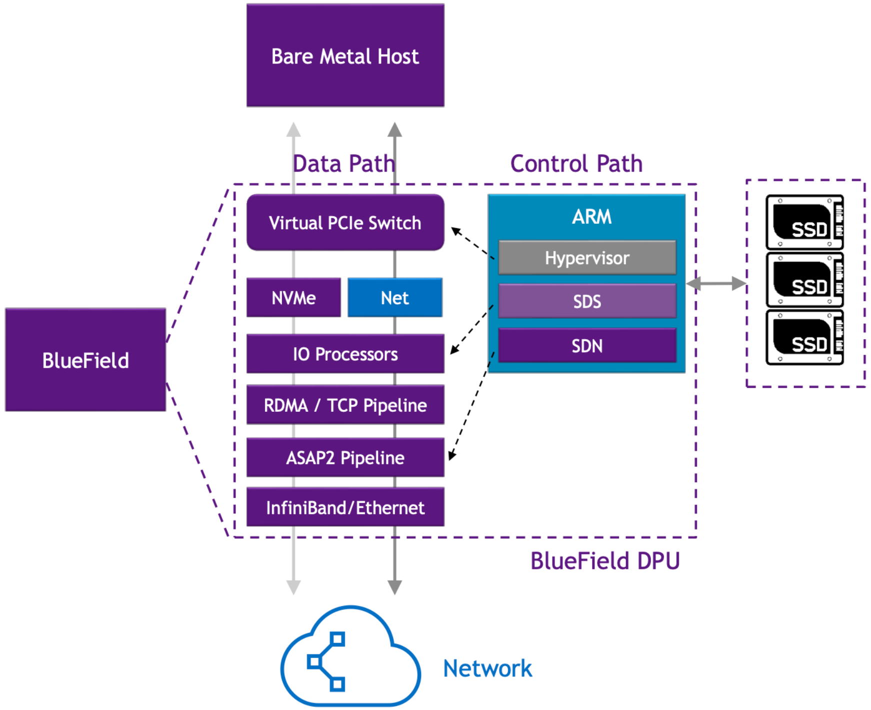
- Flexible EP/RP assignment, PCIe switching, NTB, p2p communication, emulated devices, optimized for IO

Data acceleration

- Accelerating ARM workload



Programmable Data Path



DPU: A convergence of Off-Path and NP-based On-Path SmartNICs

NVIDIA DATA CENTER PROCESSING UNIT (DPU) ARCHITECTURE, HotChips 2021

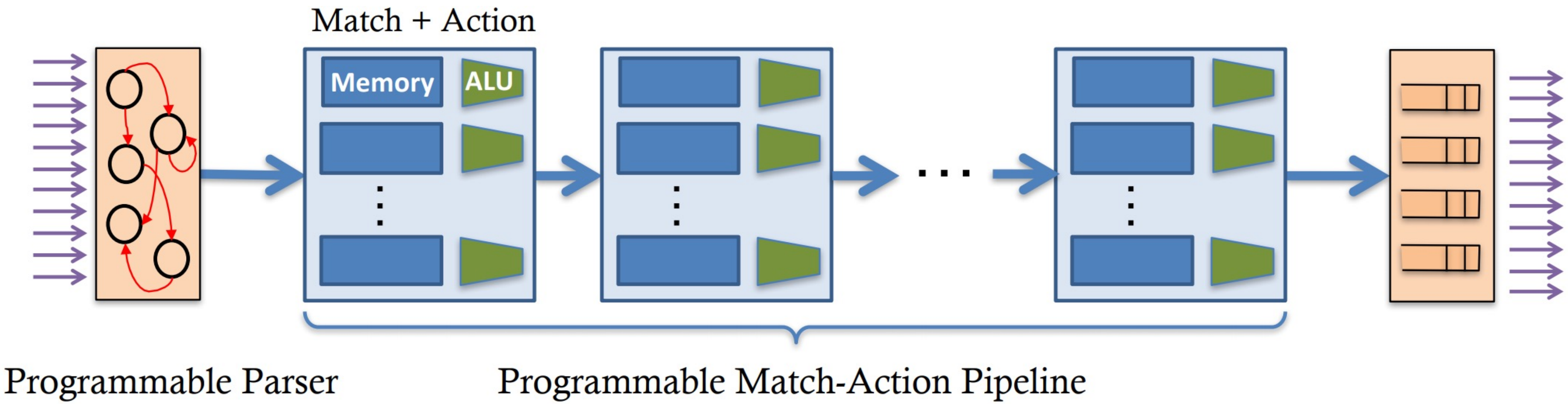
How to Choose SmartNICs?

- ASIC and FPGA are similar.
  - FPGA is more programmable.
  - ASIC is cheaper in extremely large scale (1M+ hosts).
  - Suitable for large cloud providers.
- NP-based SmartNICs and DPUs are similar.
  - Only offloading tasks to generic processors does not work.
  - Hardware acceleration adds complexity to programming, but essential for performance.
  - Suitable for small-to-medium scale deployments.

Contents

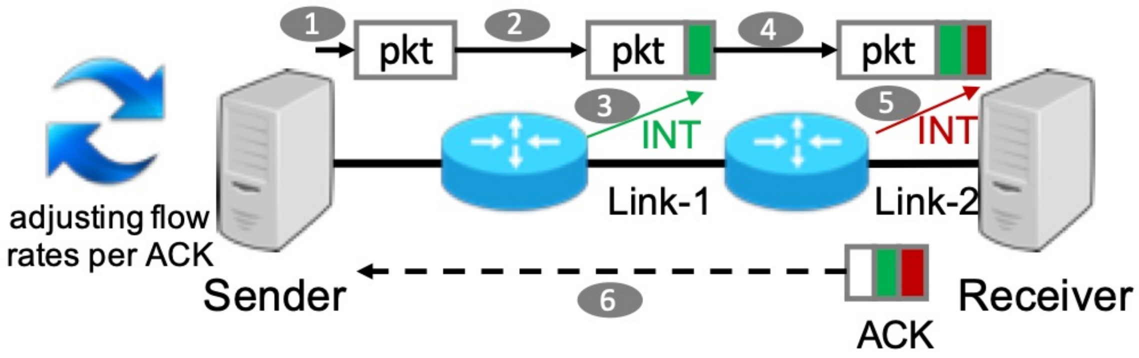
- Trend 1: Intelligent Network Devices
  - SmartNIC: FPGA, ASIC, NP and DPU
  - Programmable Switch
- Trend 2: Fast Interconnect
  - NVLink and CXL: Direct P2P with Memory Semantics
  - Convergence of AI and Cloud Networking

Programmable Switch: P4



	Server	Switch
Example	[NetBricks, OSDI '16]	Barefoot Tofino
Packets per second	~30 million	**> 1 billion**
Bandwidth	10-100 Gbps	**6.5 Tbps**
Processing delay	10-100 us	**< 1 us**
Visibility	Narrow	**Wide**
Memory capacity	**Large**	Small
Programmability	**High**	Low

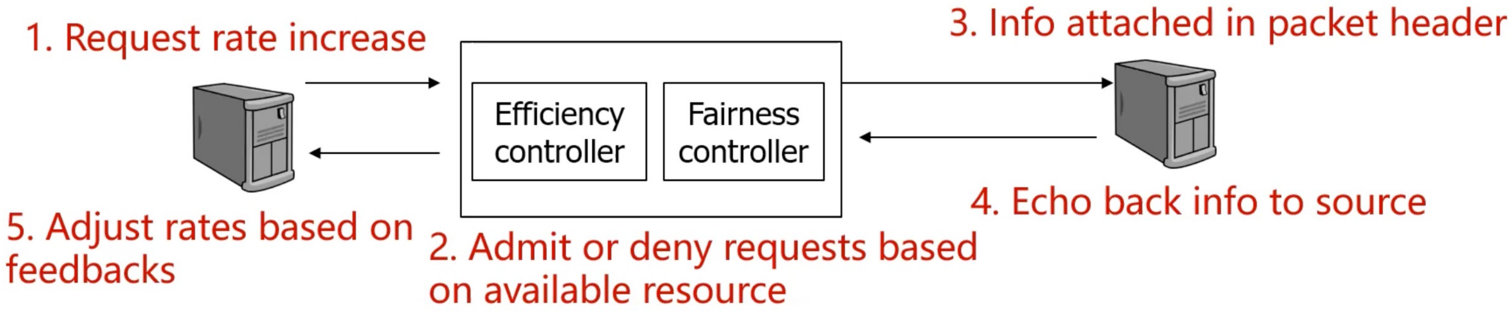
Use Case 1: NIC + Switch Congestion Control



HPCC: High Precision Congestion Control, SIGCOMM '19

Confined AQM: end-to-end active credit-based congestion control

- Separated *efficiency* and *fairness* controller at switches
- **Efficiency controller**: ensure high utilization of bandwidth, but no over-utilization -> zero-queue and zero-drop
- **Fairness controller:** ensure QoS and fair bandwidth distribution



Towards Compute-Native Networking, APNet '21

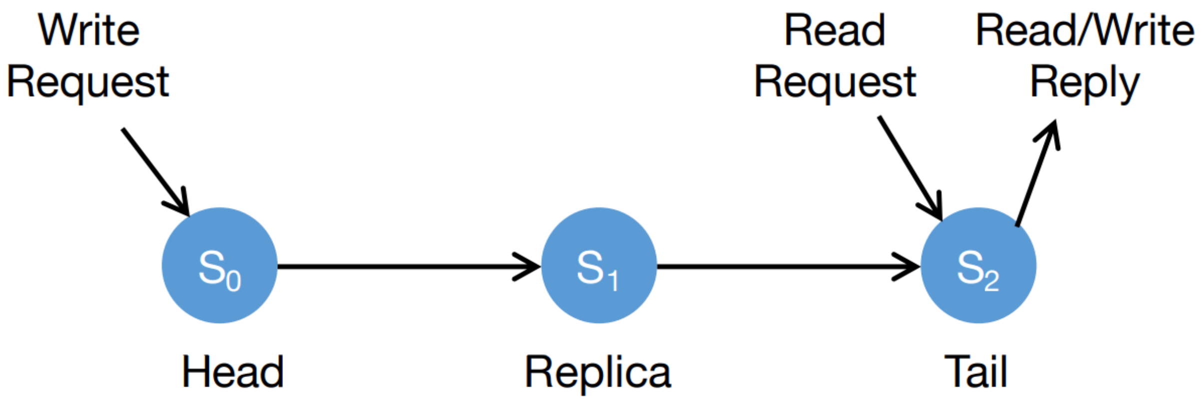
Use Case 2: Sub-RTT Coordination on Switches

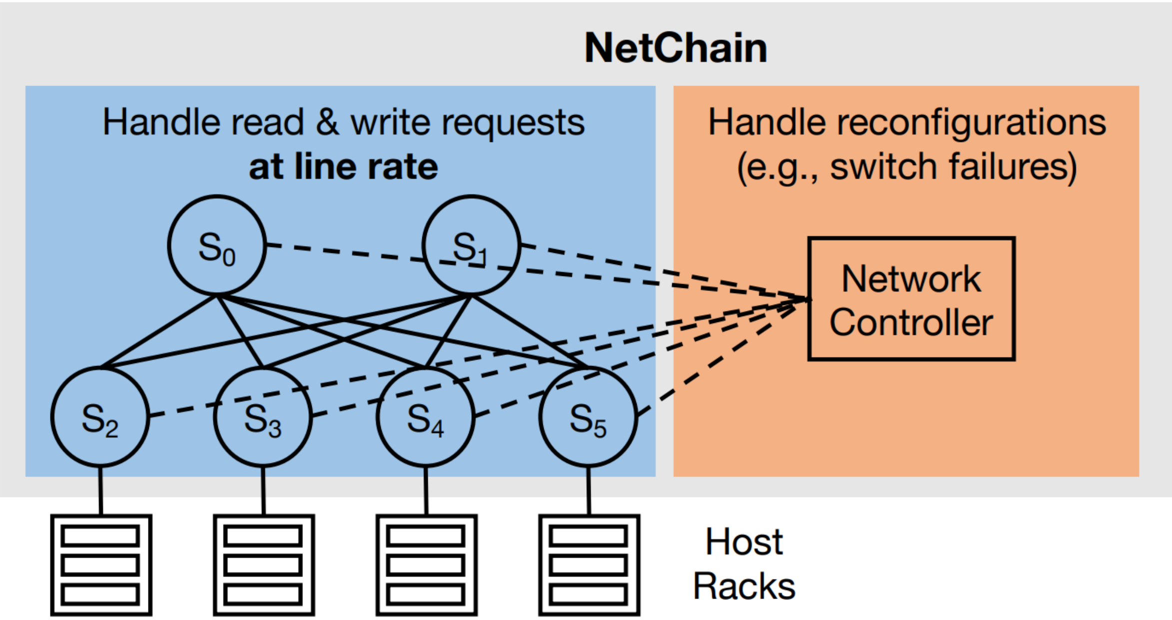
- High throughput
- Low latency

Directly from high-performance switches

- Strong consistency
- Fault tolerance

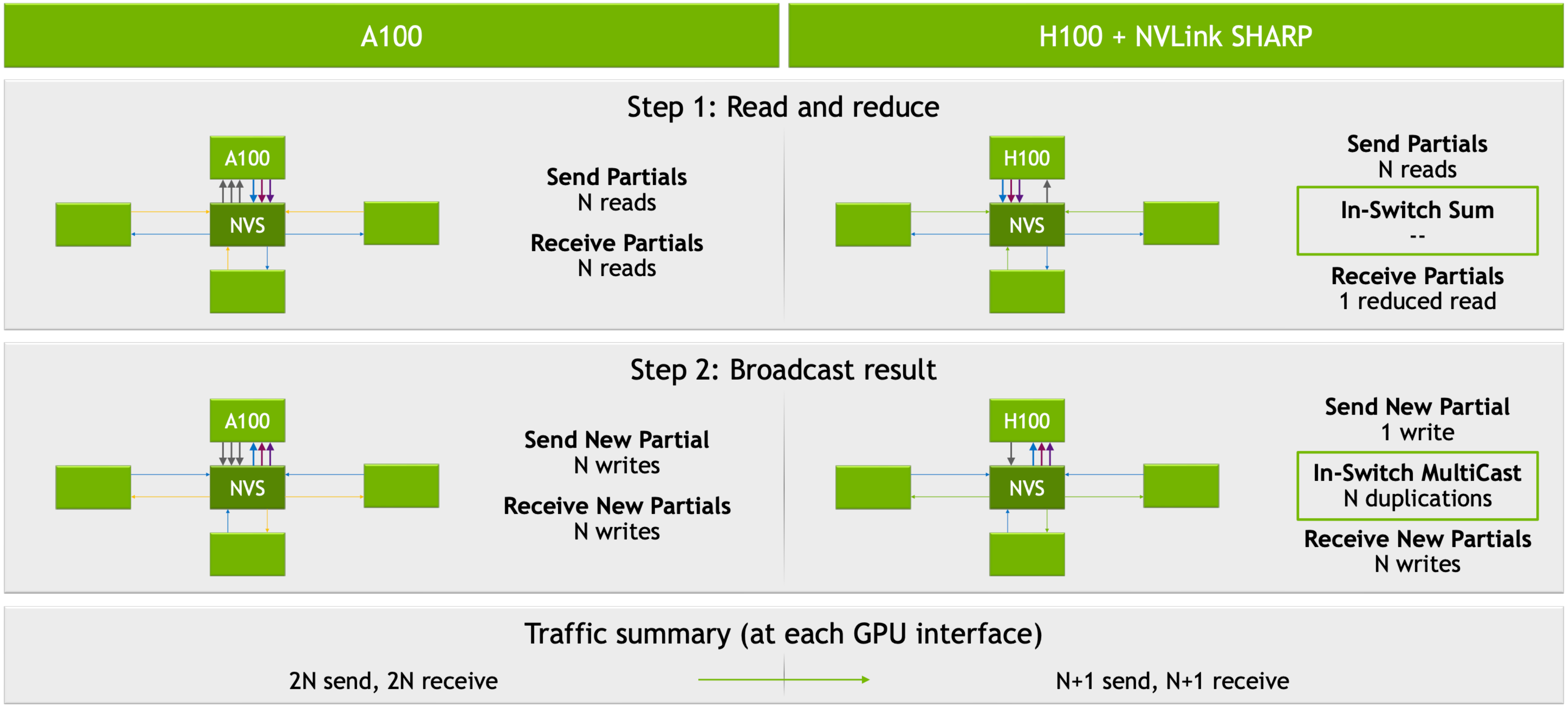
Chain replication in the network





NetChain: Scale-Free Sub-RTT Coordination, NSDI '18

Use Case 3: SHARP for AI Param Aggregation



~2x effective NVLink bandwidth

THE NVLINK-NETWORK SWITCH: NVIDIA'S SWITCH CHIP FOR HIGH COMMUNICATION-BANDWIDTH SUPERPODS, HotChips '22

Why Intel Stopped Tofino

McKeown, Nick Jan 29, 2023, 7:02:49 PM

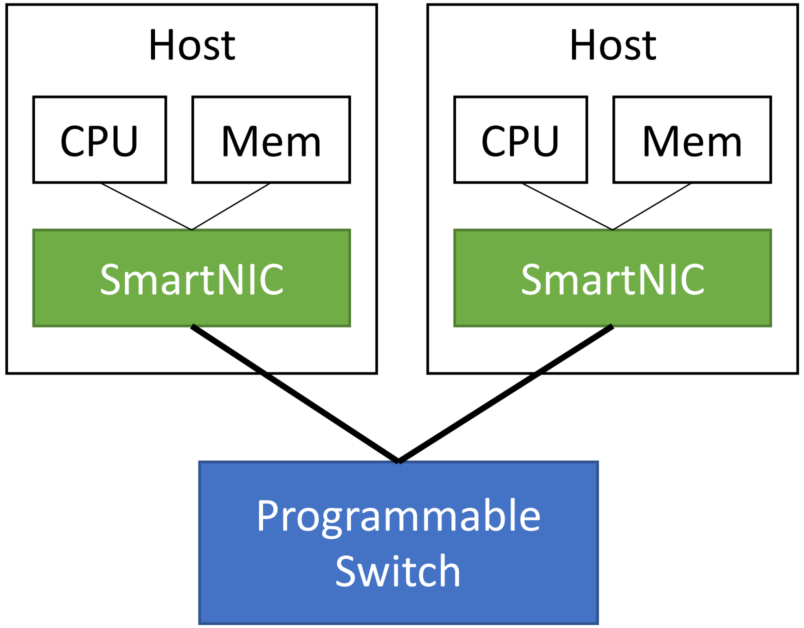
to P4-an...@lists.p4.org, P4-di...@lists.p4.org, p4-d...@lists.p4.org, p4-...@lists.p4.org, p4-...@lists.p4.org

Dear P4 Community:

Since its introduction a decade ago, P4 has led to a Cambrian explosion of ideas including new protocols, new applications like in-band telemetry, and new testing, validation, and formal verification techniques. P4 has become the industry standard for programming and specifying forwarding behavior. As a measure of success, one in four papers published at ACM SIGCOMM '22 — the top conference for networking research — are built on P4 in some way.

As you may know, Intel recently announced that it will stop development of the next-generation Intel® Tofino® Intelligent Fabric Processor (IFP) products currently on its roadmap. However, we will continue to sell and support our existing Tofino® products. Intel Tofino® IFPs proved to the world that you can build fully programmable switches without compromising on performance. Tofino's program independent switch architecture (PISA) will have a lasting effect on how packet-processing pipelines are built; it has already influenced programmable products at the edge such as SmartNICs and IPUs.

SmartNIC or Programmable Switch?



	SmartNIC	Programmable Switch
Congestion control	DCQCN, TIMELY, MP-RDMA, IRN	HPCC, pFabric, DeTail, CP, NDP
Load balancer	VFP	SilkRoad
Key-value store	Pilaf, FaRM, DrTM, FaSST, KV-Direct	SwitchKV, NetCache, IncBricks
Aggregation	NetAgg, CamCube	SHARP, DAIET, SwitchML, ATP
Lock	DSLR	NetLock
Coordination / Replication	Consensus in a Box, DARE, APUS, Derecho, Mu	NetChain, NetPaxos, SpecPaxos, NOPaxos, Eris
Programming system	Floem, iPipe, StRoM, ClickNP, FairNIC, λ-NIC	SNAP, Frenetic, P4, P4visor, μP4, Domino, Lyra, Gallium

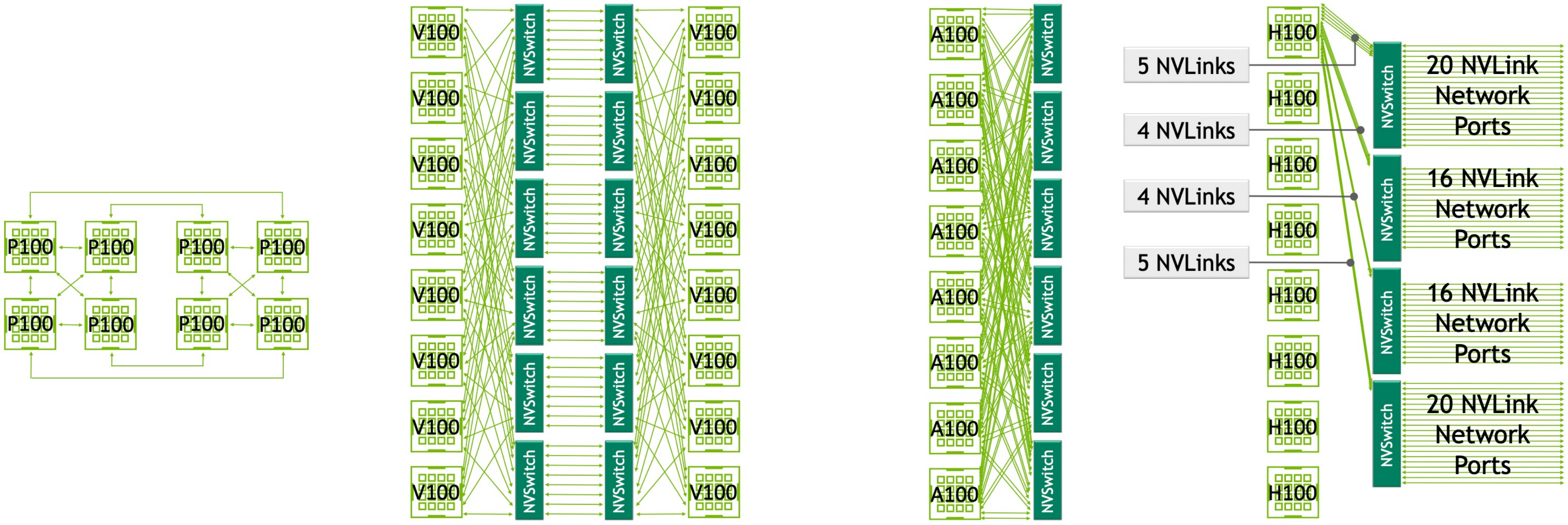
Contents

- Trend 1: Intelligent Network Devices
  - SmartNIC: FPGA, ASIC, NP and DPU
  - Programmable Switch
- Trend 2: Fast Interconnect
  - NVLink and CXL: Direct P2P with Memory Semantics
  - Convergence of AI and Cloud Networking

Trend 2: Fast Interconnect

NVLINK-ENABLED SERVER GENERATIONS

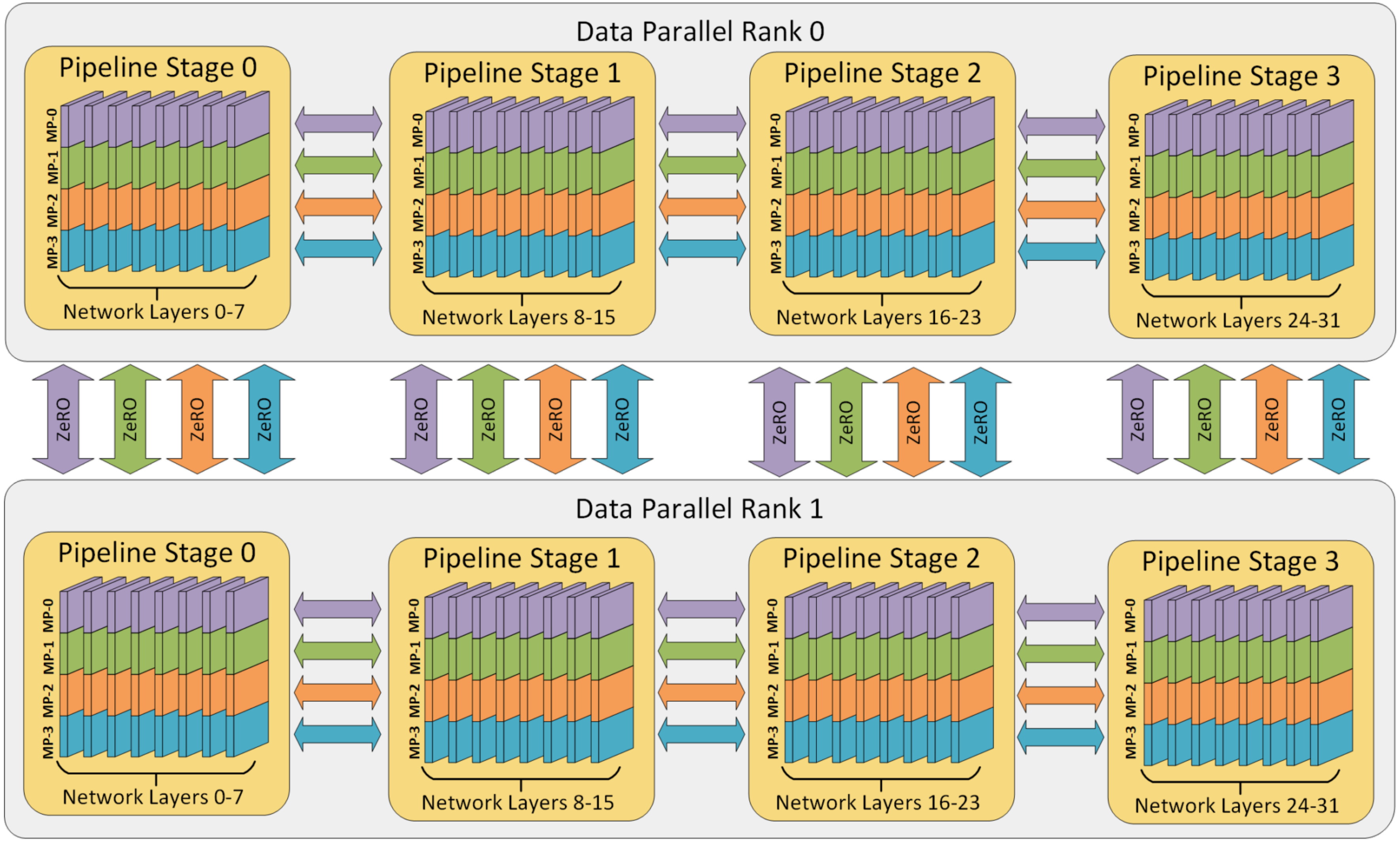
Any-to-Any Connectivity with NVSwitch



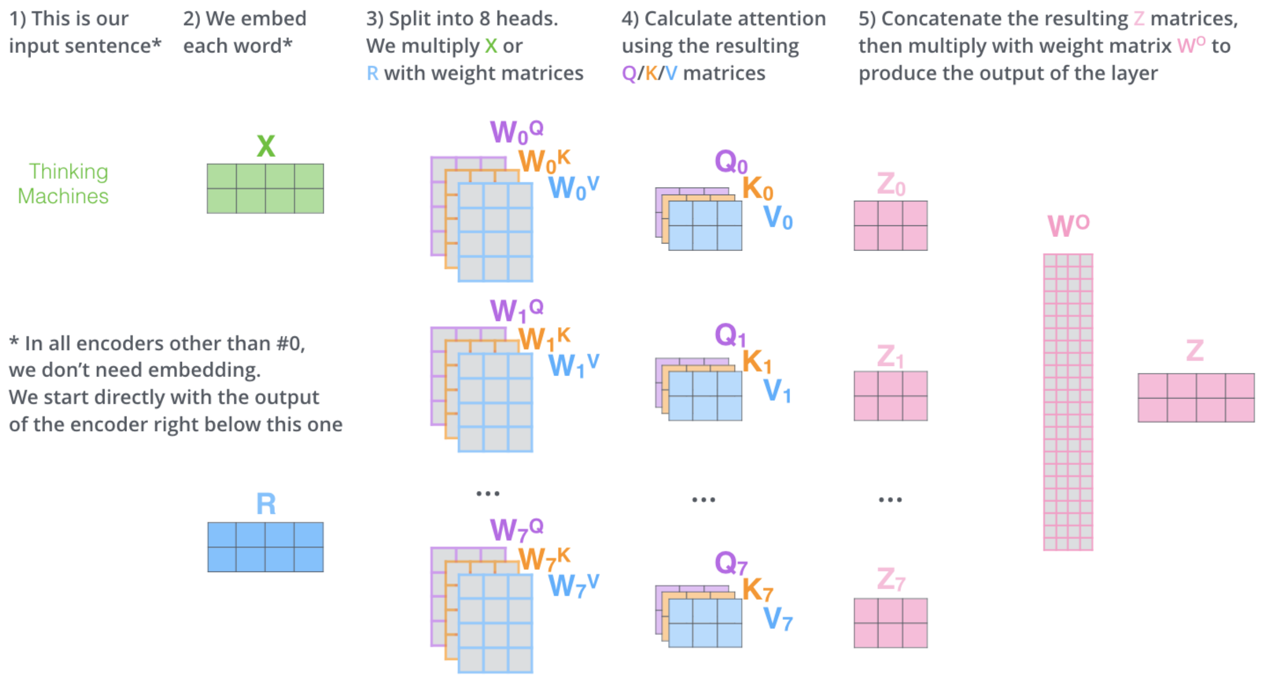
2016	2018	2020	2022
DGX-1 (P100)	DGX-2 (V100)	DGX A100	DGX H100
140GB/s Bisection BW 40GB/s AllReduce BW	2.4TB/s Bisection BW 75GB/s AllReduce BW	2.4TB/s Bisection BW 150GB/s AllReduce BW	3.6TB/s Bisection BW 450GB/s AllReduce BW

THE NVLINK-NETWORK SWITCH: NVIDIA’S SWITCH CHIP FOR HIGH COMMUNICATION-BANDWIDTH SUPERPODS, HotChips ‘22

Why LLM Training Needs High Bandwidth



Tensor Parallelism Needs High Bandwidth



Back-of-envelope estimation for tensor parallelism in attention computation:

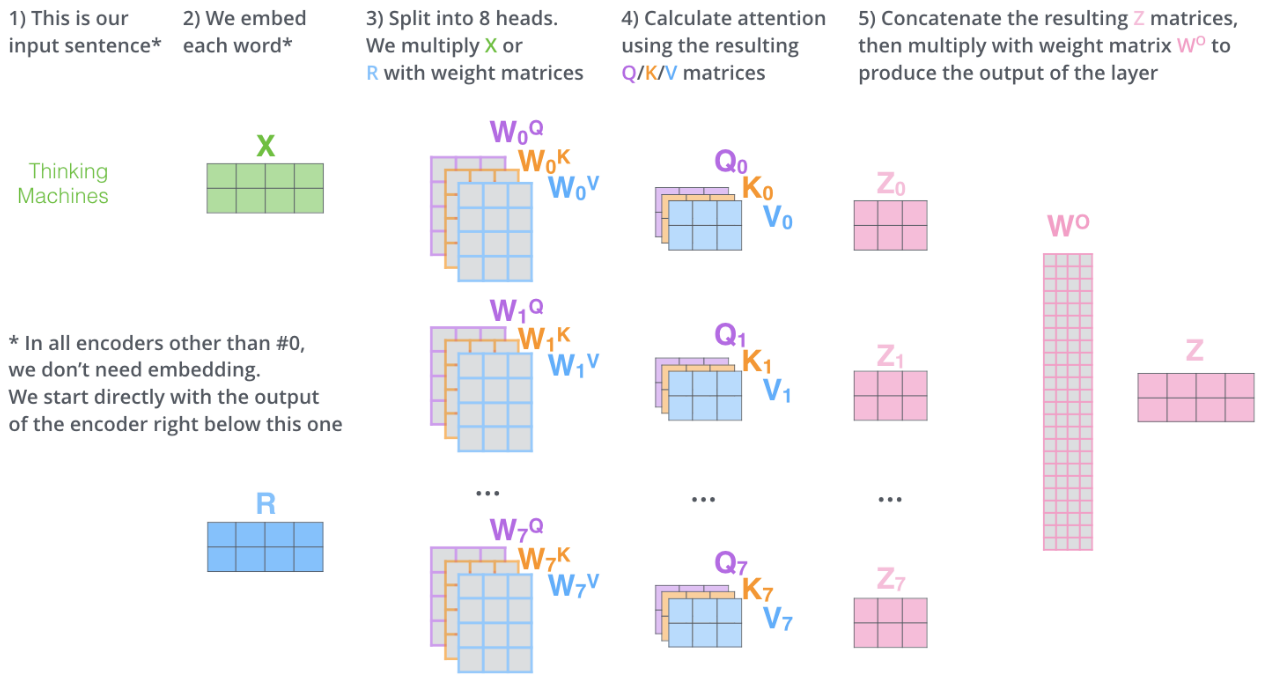
- **Computation cost** is roughly:
  3 * batch size * token length * (embedding size * embedding size / num GPUs) * 2 flops

- **Communication cost** is roughly:
  Batch size * token length * embedding size * 2 bytes

- **Computation / Communication = 3 * embedding size / num GPUs**
- Example: LLaMA-2 70B, embedding size = 8192

- Computation FLOPS of H100: 989T Tensor flops for 16 bit
- If we use 8 GPUs for tensor parallelism and assume 100% FLOPS utilization:
  - Bandwidth >= Computation / (3 * embedding size / num GPUs) = 989T / (3 * 8192 / 8) = 321 GB/s
  - **Bi-direction bandwidth >= 642 GB/s**
- That’s why we need 900 GB/s bandwidth for NVLink

Tensor Parallelism Also Needs Low Latency

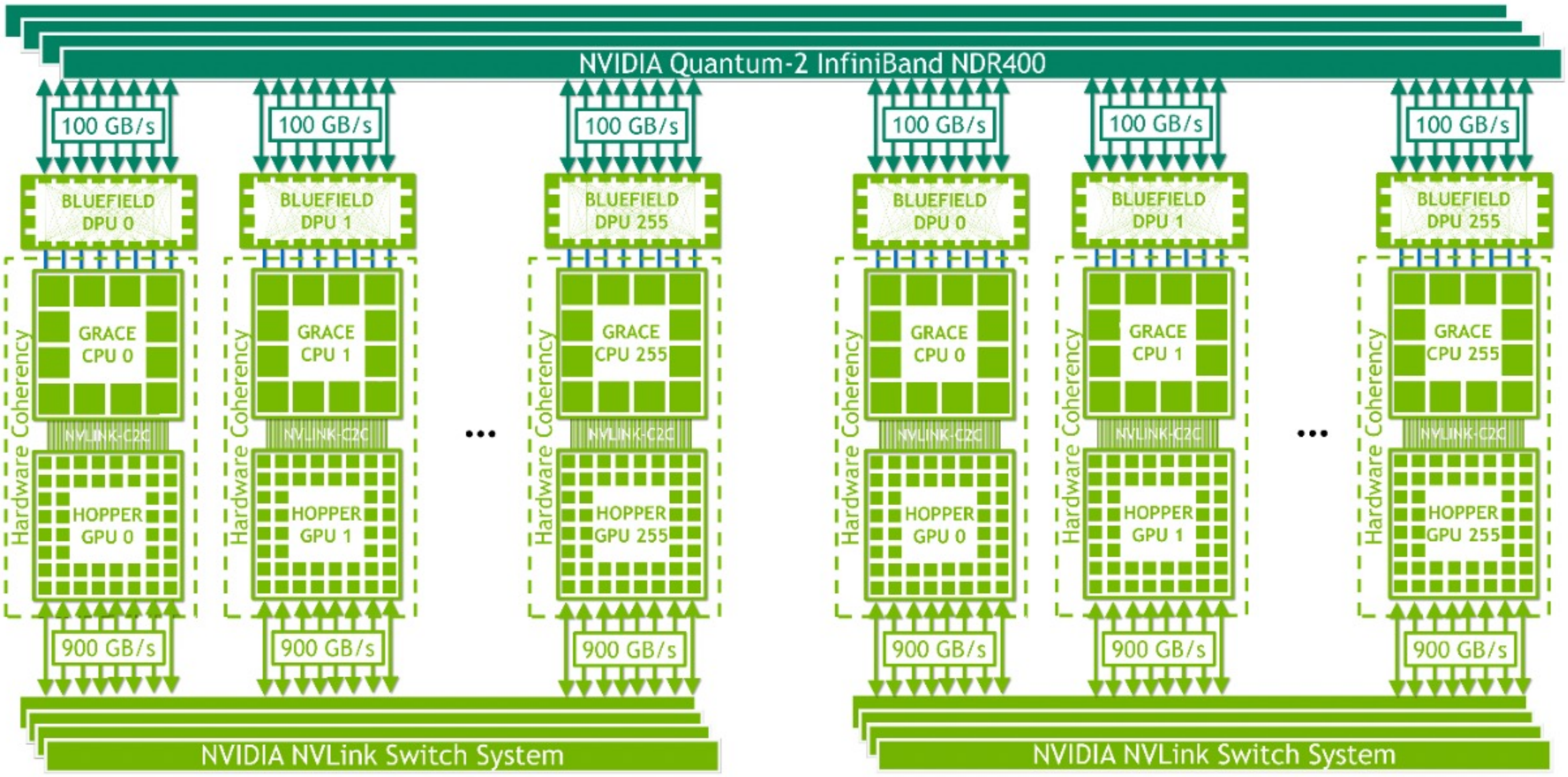


Taking latency into account:

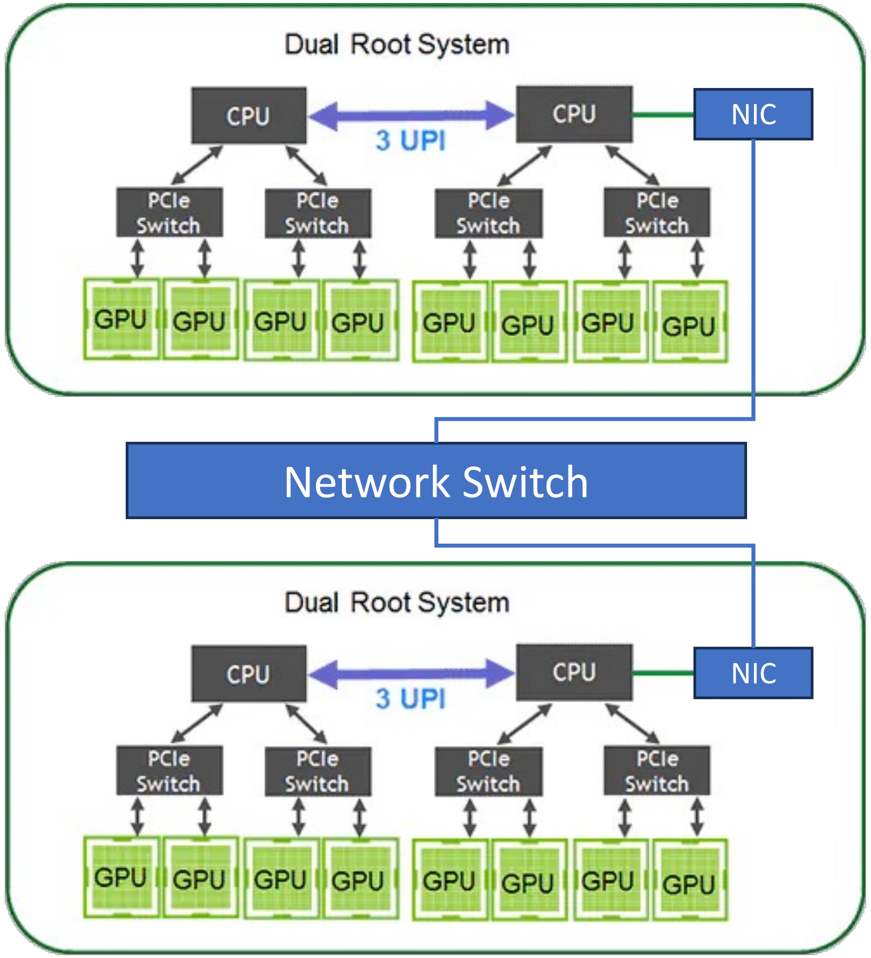
- **Communication time** is roughly: (Batch size * token length * embedding size * 2 bytes / bandwidth) + latency
- **Computation time** is roughly: 3 * batch size * token length * (embedding size * embedding size / num GPUs) * 2 flops / computation flops
- Example: LLaMA-2 70B, embedding size = 8192, token length = 4096, batch size = 1

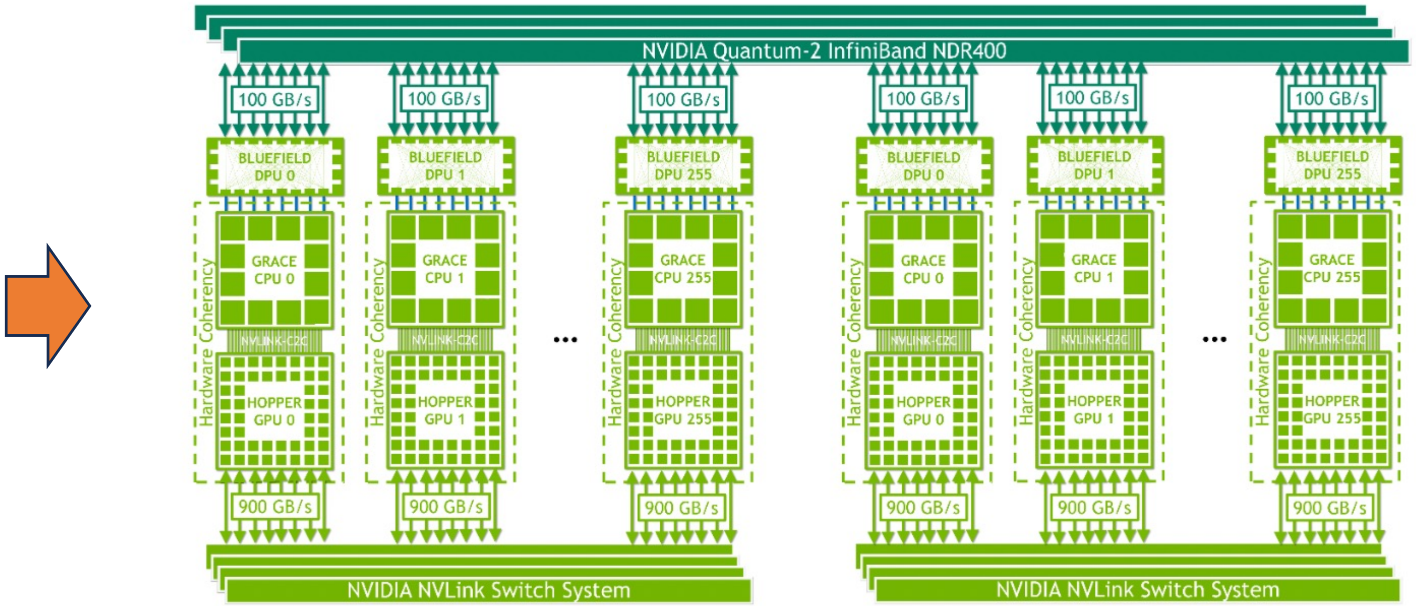
- Communication time = (1 * 4096 * 8192 * 2 / 450G) + latency = 14 us + latency
- Computation time = 3 * 1 * 4096 * (8192 * 8192 / 8) * 2 / 989T = 19 us
- To achieve 100% FLOPS utilization: **Latency <= 19 us – 14 us = 5 us**
- **NVLink latency is <1 us** when GPU P2P is enabled.
- However, if we use **CPU as a proxy among GPUs**, the latency would be **>10 us**, and the throughput also suffers.

Direct Peer-to-Peer Interconnect

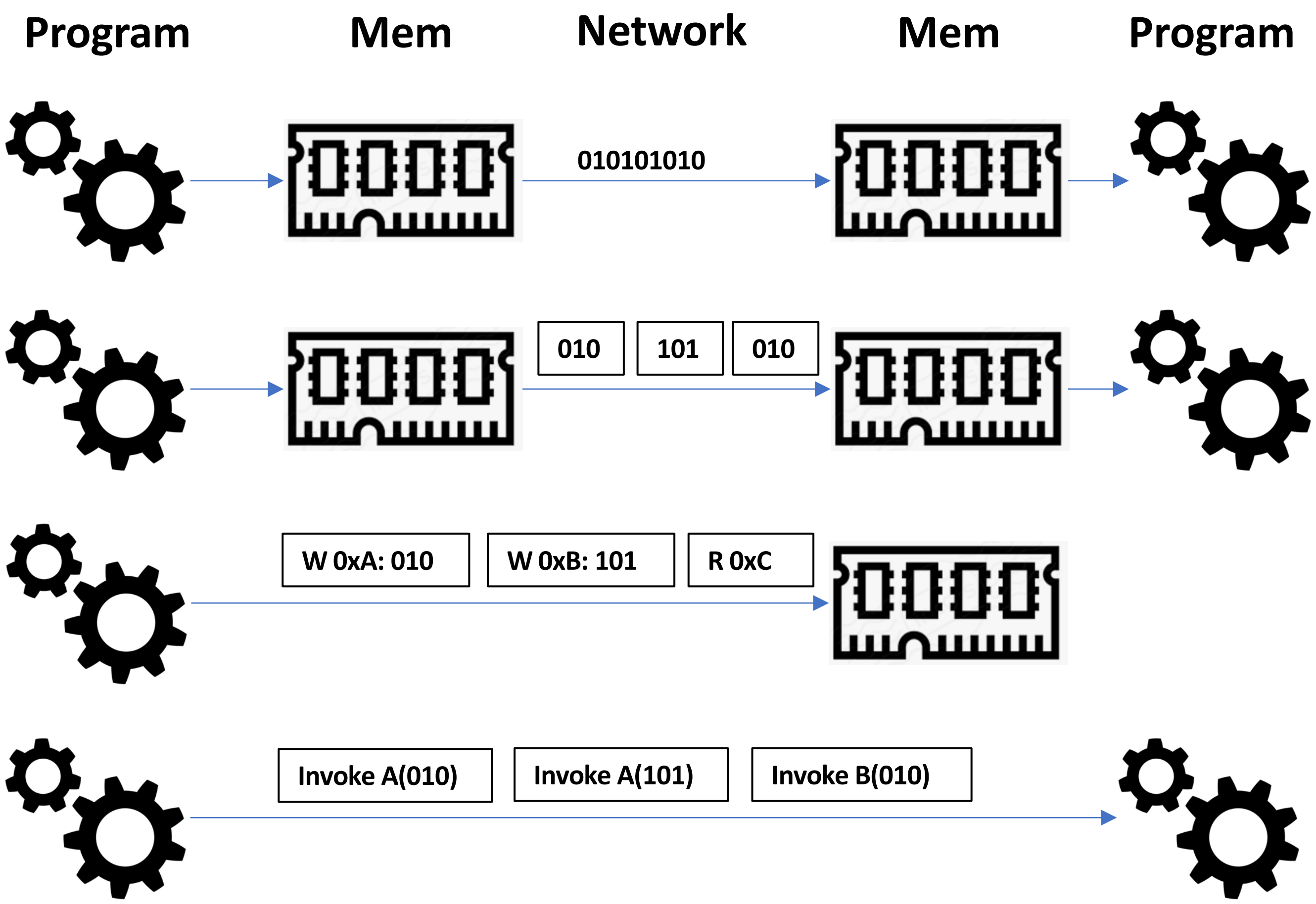


Convergence of Intra- and Inter-host Network



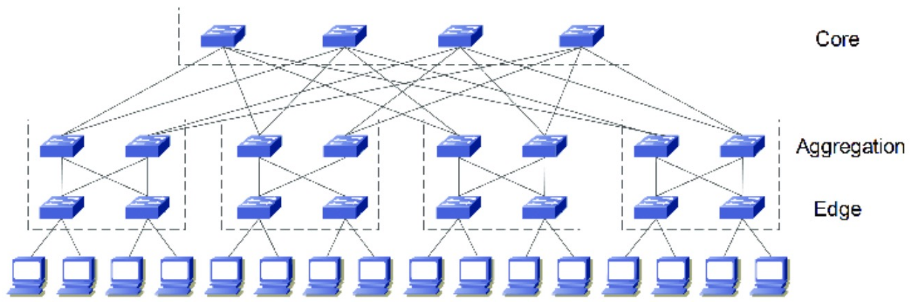


Memory Semantics



Semantics	Example	
Byte Stream	Socket send/recv	
Message	ZeroMQ / Kafka RDMA send/recv	Memory Semantics
Remote Memory Access	RDMA read/write /atomic CXL Load/Store	Memory Semantics
RPC	gRPC, Thrift	Memory Semantics

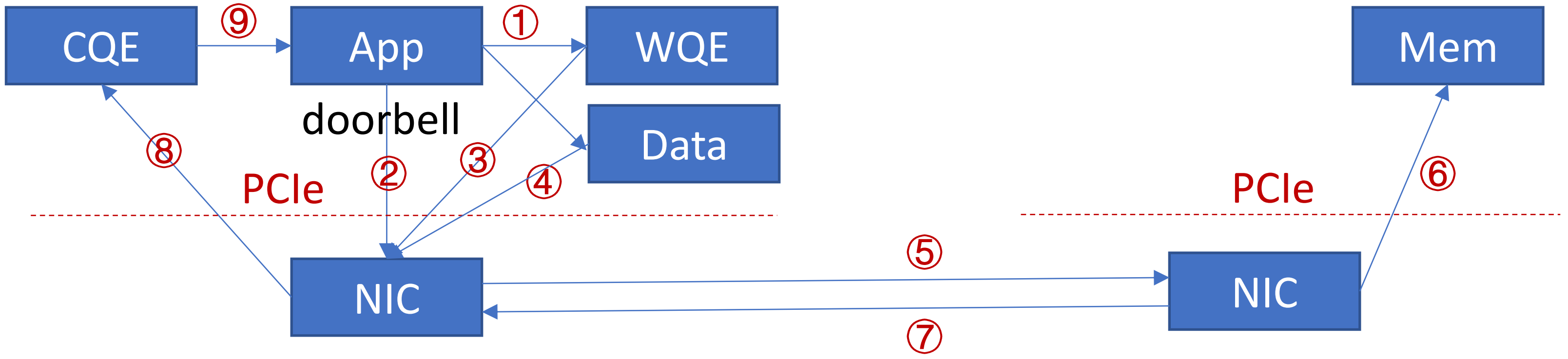
Exploiting Parallelism with Memory Semantics



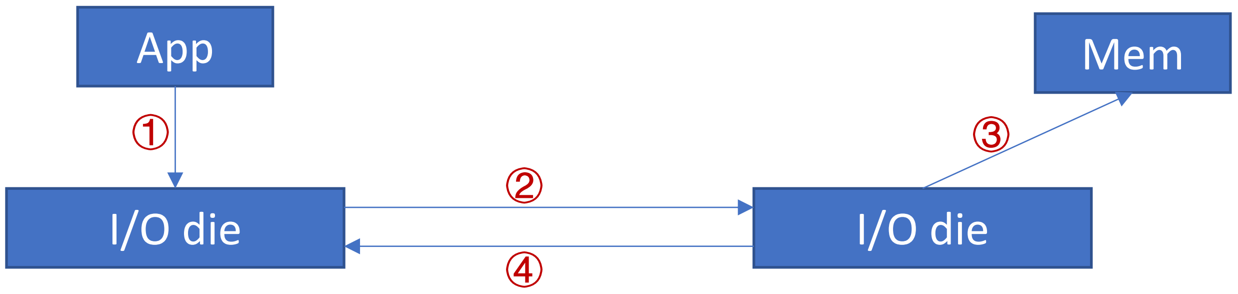
- RDMA is strictly in-order communication
  - Hard to utilize multiple network paths due to reordering cost at receiver
  - One lost packet blocks subsequent transactions from delivery
  - Hard to support page faults because a slow page-fault memory access would block all subsequent accesses
- Many memory accesses can be executed out-of-order
  - Example: transferring multiple large tensors
  - Parallelize transfer over multiple network paths to improve bandwidth
  - A lost packet only blocks one transaction
  - Page-fault and other slow memory accesses can be processed out-of-order

Load/Store vs. Read/Write

- Read/Write is **asynchronous** remote memory access
- Requires multiple PCIe RTTs, min latency 1.6 us



- Load/Store is **synchronous** remote memory access. CPU accesses network directly.
- No PCIe, No WQE, CQE or doorbell, latency < 0.5 us



Load/Store is Not a Panacea

	Load/Store	Read/Write
Programming	Sync	Async
Granularity	Cache line	User-specified message size
Latency	Low	High
Access efficiency of large data blocks	Low	High
Application transparency	Application imperceptible, can be used to extend local memory, achieving memory pooling	The application needs explicit access to remote memory; if used for memory expansion, the application needs to be modified
Hardware requirements	High, requires the NIC to work closely with the CPU	Low, the NIC can be in a detached form
Reliability	Large blast radius, a node failure will affect all nodes using the remote memory of that node; store instruction fault is difficult to capture	Easy to capture asynchronous remote access exceptions through the application, reducing the blast radius to the affected application
Cache coherence	Depends on whether the hardware supports it, but the overhead of hardware supporting cache coherence is high at a large scale	Not supported, the software explicitly copies between remote and local memory. In the case of sharing, it needs to coordinate with distributed locks to ensure consistency

GPU Comm.: From Load/Store to Read/Write

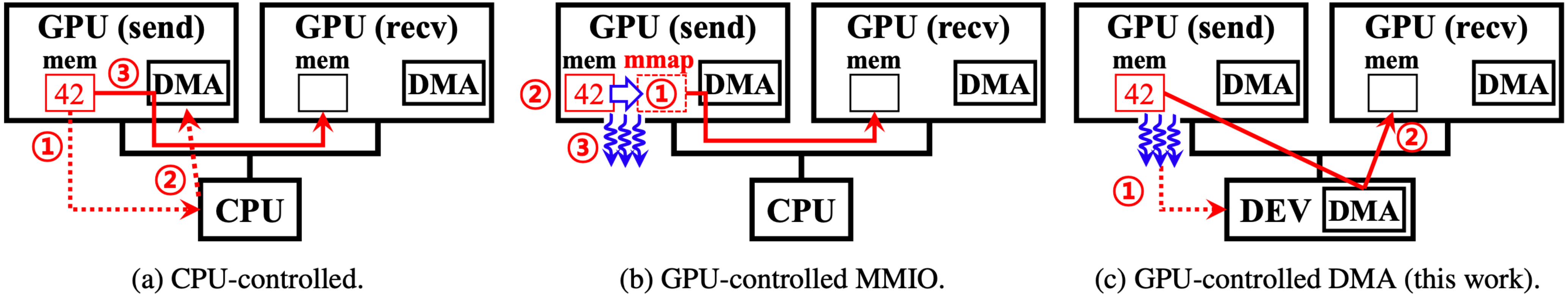
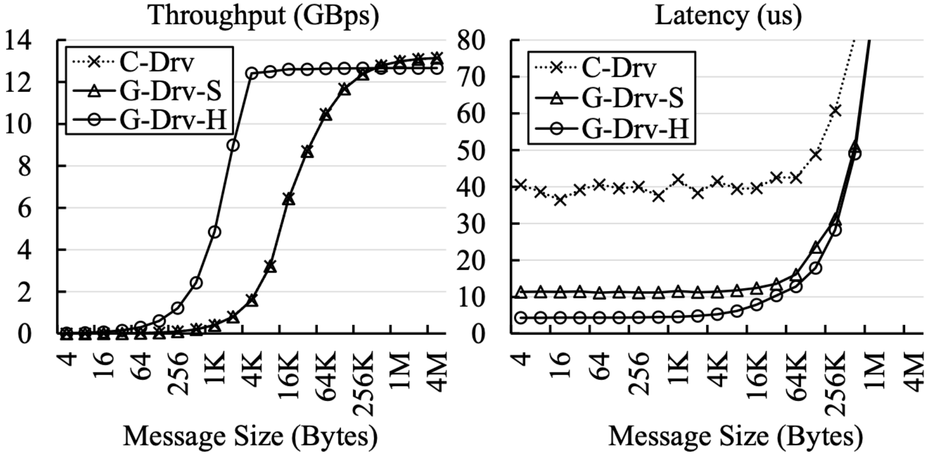


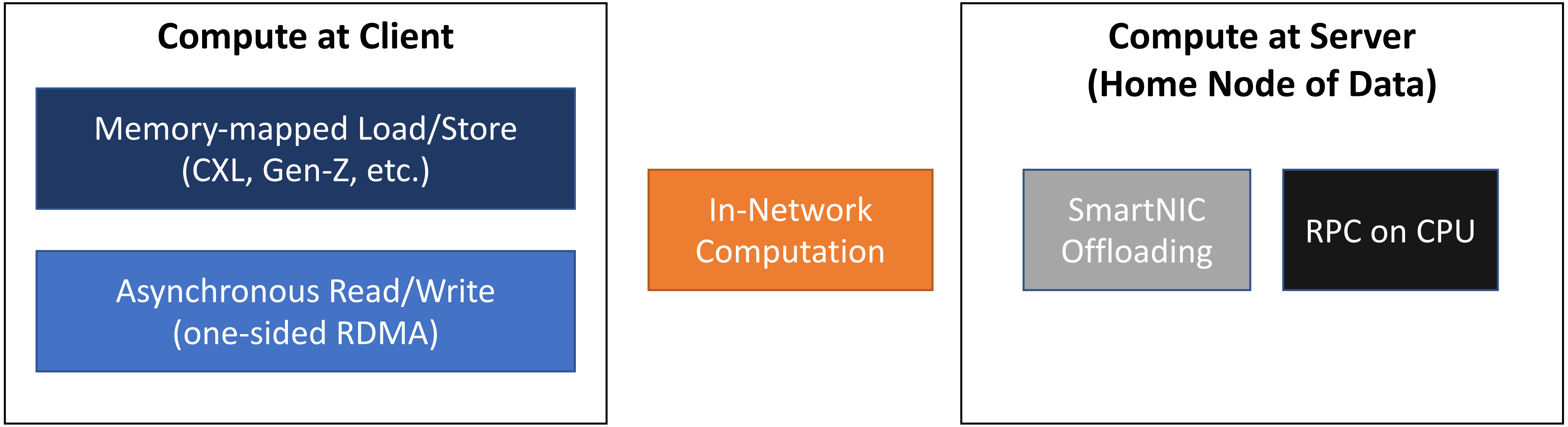
Figure 3: Comparison between CPU-controlled and GPU-controlled communication – the latter has two different approaches, which leverage (b) MMIO (like NCCL) or (c) directly initiated DMA (this work). DEV refers to any kinds of devices that can implement our DMA engine.



ARK: GPU-driven Code Execution for Distributed Deep Learning, NSDI '23

Figure 8: Performance comparison between the CPU-controlled communication (C-Drv) and the GPU-controlled DMA engines (G-Drv-S (software) and G-Drv-H (hardware)) over PCIe v3.

Which Semantics to Pick?



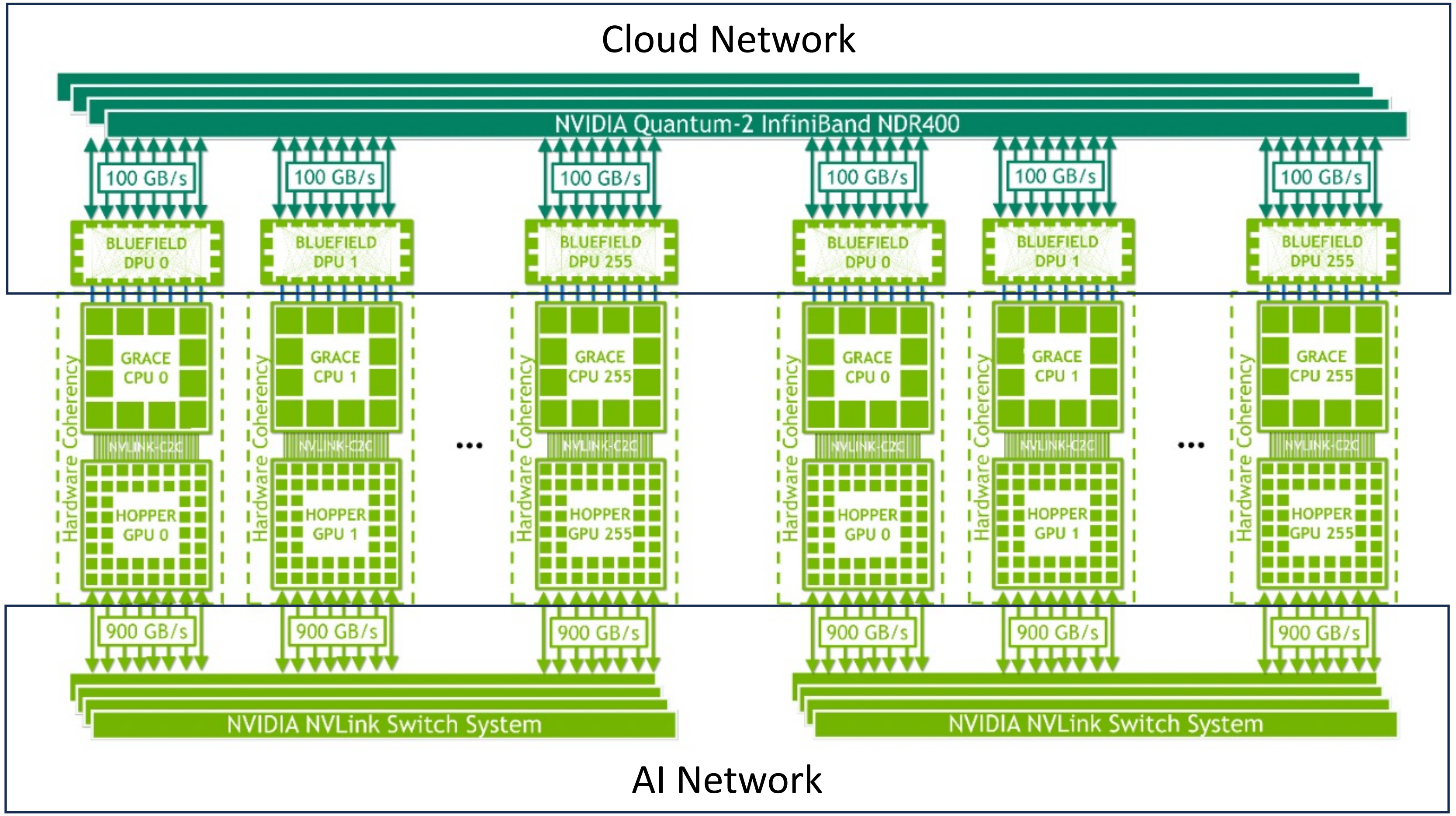
- **Load/Store**: low overhead per operation, but synchronous; may have cache and buffer
- **Read/Write**: high overhead per operation, but each op can transfer a large block, and many ops can work in parallel

- **Switch (in-network):** high throughput but low programmability and buffer size
- **SmartNIC**: high parallelism but high PCIe latency and low buffer size
- **RPC on CPU**: close to memory, easy to program but high cost

Contents

- Trend 1: Intelligent Network Devices
  - SmartNIC: FPGA, ASIC, NP and DPU
  - Programmable Switch
- Trend 2: Fast Interconnect
  - NVLink and CXL: Direct P2P with Memory Semantics
  - Convergence of AI and Cloud Networking

Convergence of AI and Cloud Networking



Convergence of AI and Cloud Networking

DPU ENABLES CLOUD-NATIVE SUPERCOMPUTING

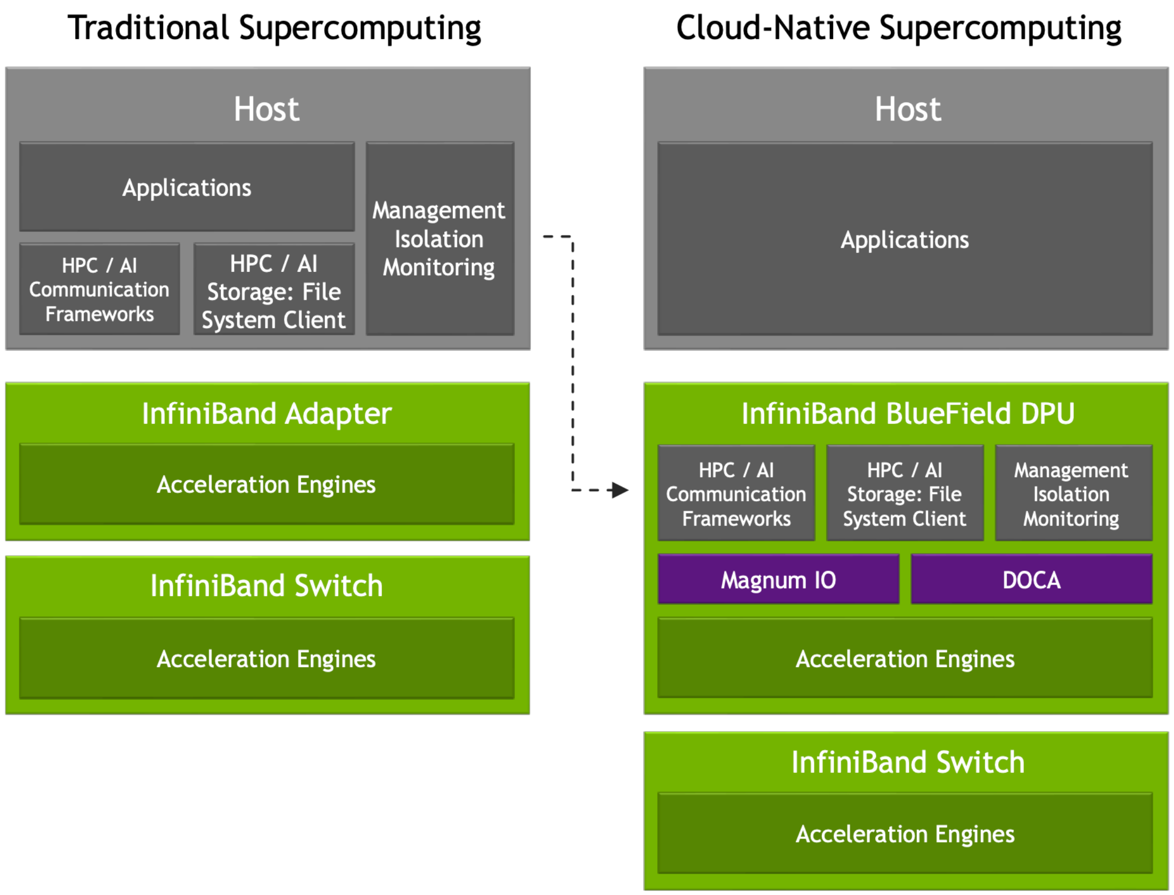
Multi-Tenancy with Zero-Trust Security

Collective offload with UCC accelerator

Smart MPI progression

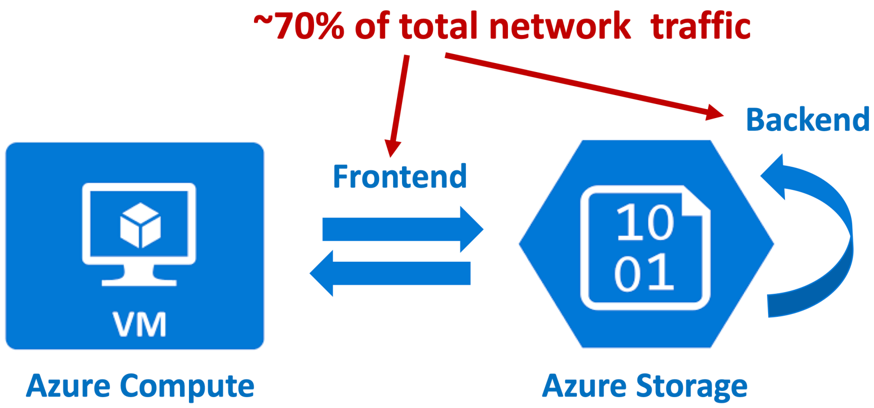
User-defined algorithms

1.4X higher application performance



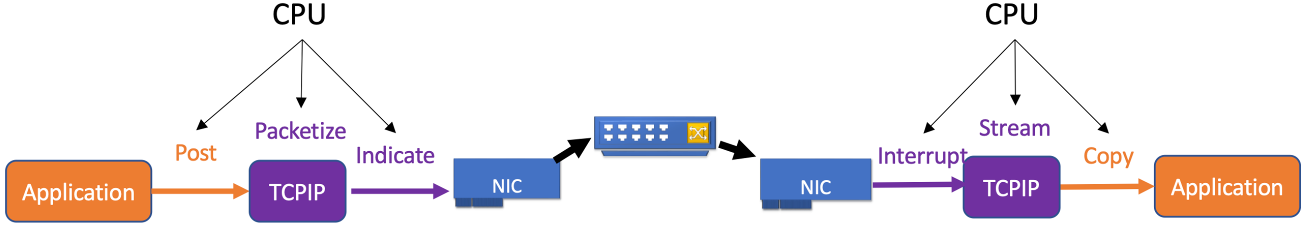
NVIDIA DATA CENTER PROCESSING UNIT (DPU) ARCHITECTURE, HotChips 2021

Region-scale RDMA for Disaggregated Storage

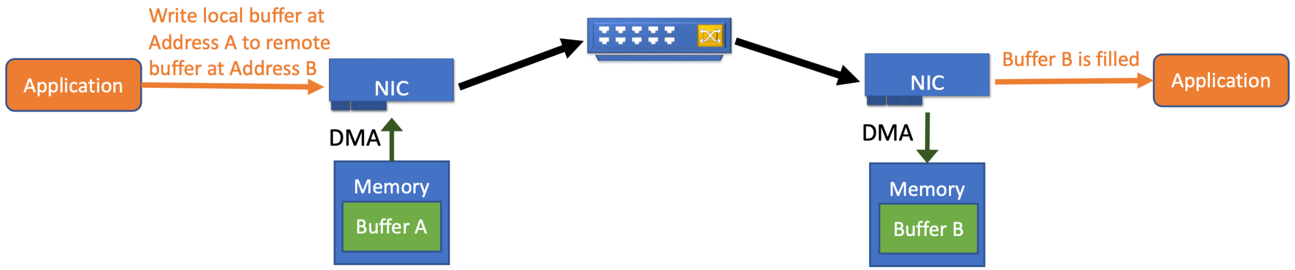


70% of Azure traffic is RDMA for disaggregated storage

Application: "Write the block of data at local disk w, address x to remote disk y, address z"



TCP: Waste of CPU, high latency due to CPU processing



RDMA: NIC handles all transfers via DMA

Empowering Azure Storage with RDMA, NSDI '23

Region-scale RDMA for Disaggregated Storage

RDMA Benefits

For Azure

Core savings

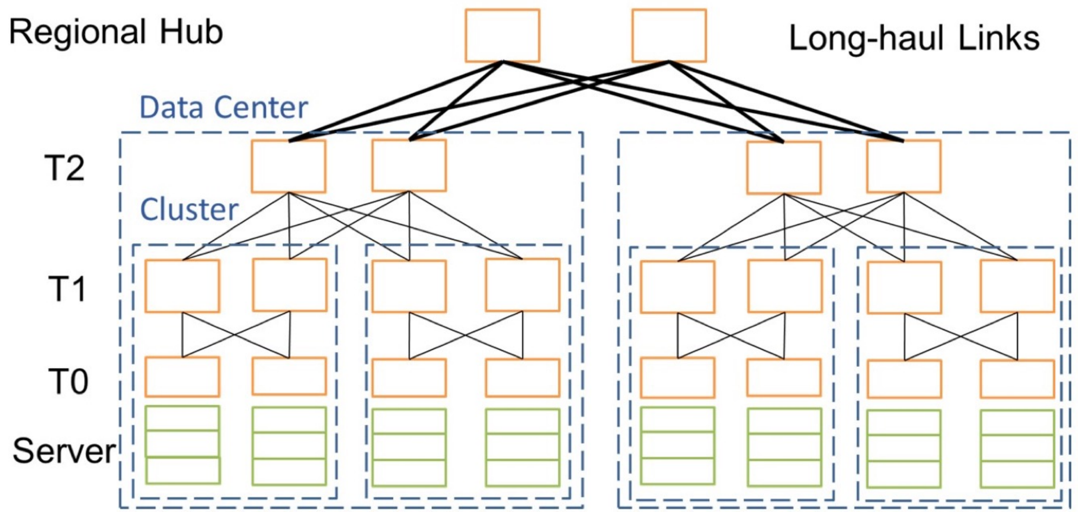
Sell freed-up CPU cores in compute

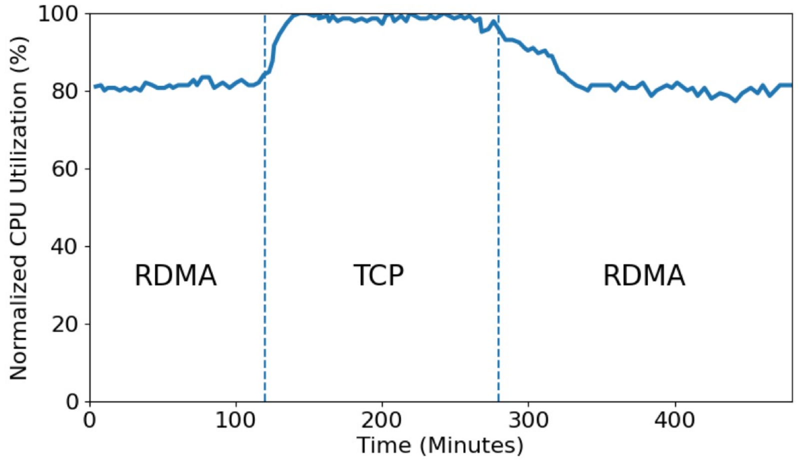
Buy cheaper servers in storage

For Customers

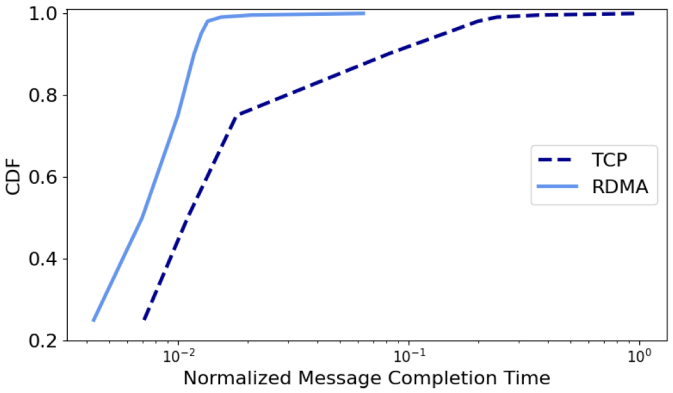
Lower IO latency and jitter

Empowering Azure Storage with RDMA, NSDI '23





RDMA reduces CPU utilization



RDMA reduces storage latency

Region-scale RDMA for Disaggregated Storage

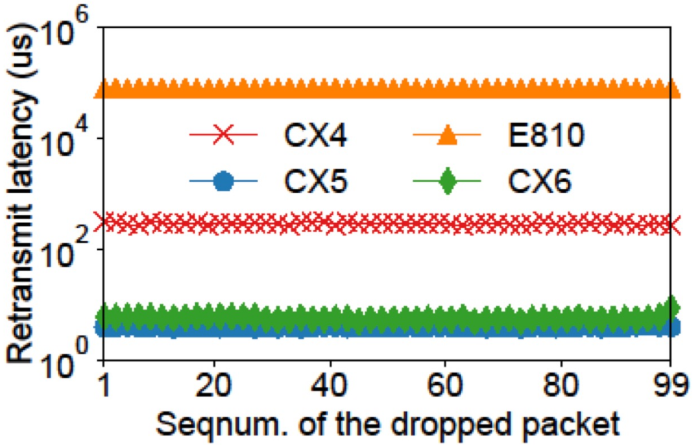
- Challenges:
  - PFC Storm caused by malfunctioning NICs and switches
  - Interoperability of heterogenous NICs and switches
  - Scaling PFC and congestion control over long-haul (~100 km) links between AZs

- Solutions:
  - PFC Watchdog on switches and SmartNICs to distinguish PFC Storm from congestion PFCs
    - Minimize PFC generation using per-flow E2E congestion control, BUT keep PFC to allow fast start and lower tail latency
  - Fine-tune DCQCN for NIC inter-op; Switch – SONiC: unified software stack
  - Jointly tune DCQCN params with switch buffers; sparse ECN marking; DCQCN does not suffer from RTT unfairness

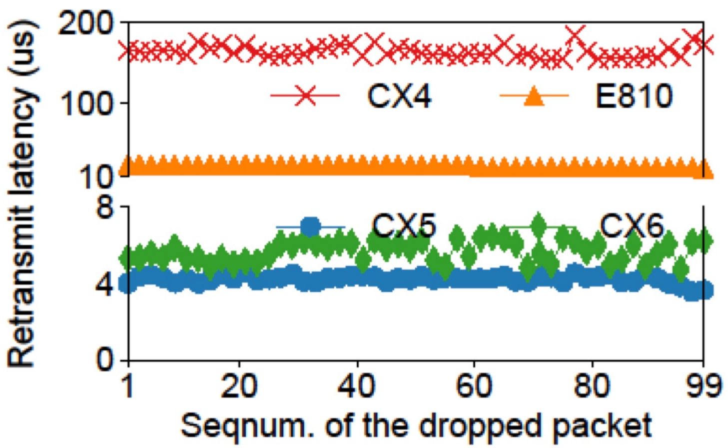
Empowering Azure Storage with RDMA, NSDI ‘23

RDMA: The Devil is in the Details

Retransmission latency



READ



WRITE

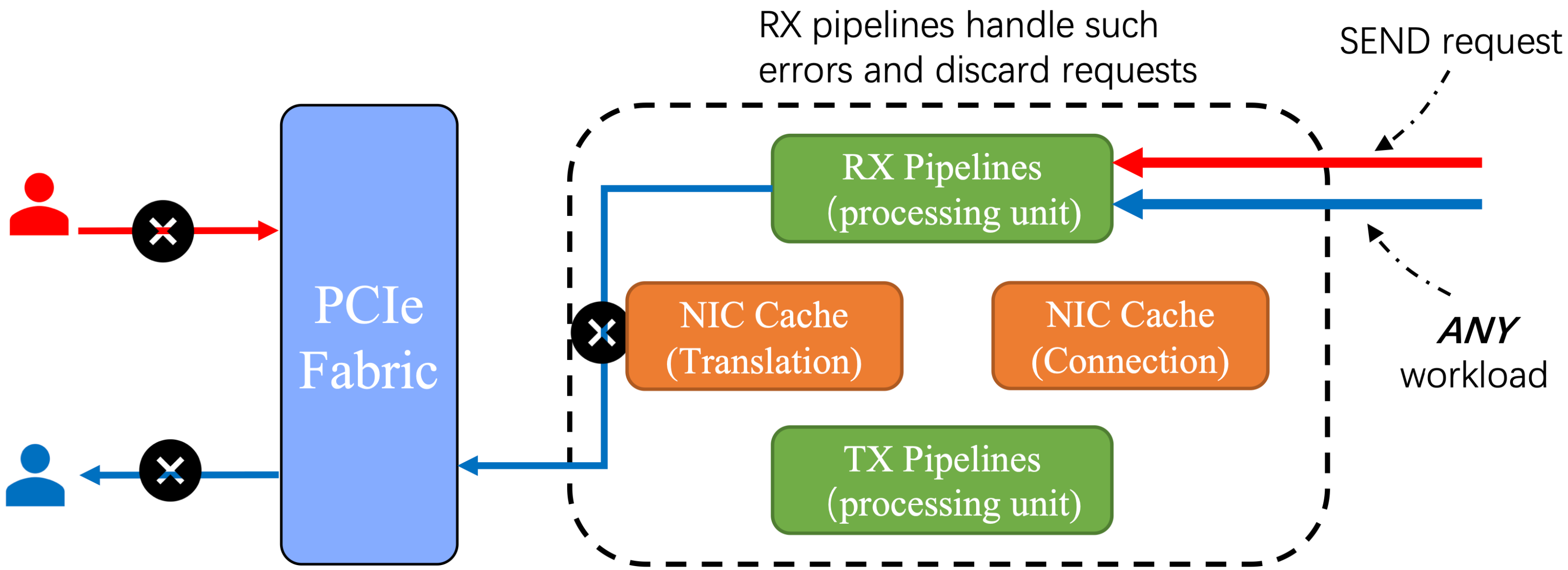
Significant improvement from NVIDIA CX4 Lx to CX5 and CX6 Dx

Intel E810 cannot efficiently recover lost READ packets

Lumina: Understanding the Micro-Behaviors of Hardware Offloaded Network Stacks, SIGCOMM ‘23

Performance Isolation Problem of RDMA NICs

RNR causes severe RX pipelines contention



	Victim Bandwidth	Attacker Bandwidth
w/o RNR error	97.07 Gbps	\
w/ RNR error	0.018 Gbps	0 Gbps

Husky: Understanding RDMA Microarchitecture Resources for Performance Isolation, NSDI ‘23

Lessons from Azure RDMA Deployment

- Failovers are very expensive for RDMA
- Host network and physical network should be converged
- Switch buffer is increasingly important and needs more innovations
- Cloud needs unified behavior models and interfaces for network devices
- Testing new network devices is critical and challenging

The Congestion Challenge

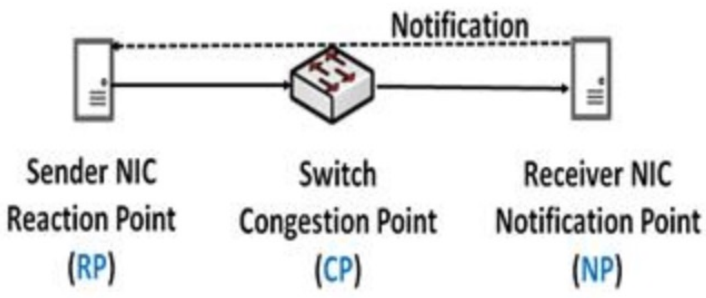
Network congestion

- Congestion within network
- Adaptive routing can help

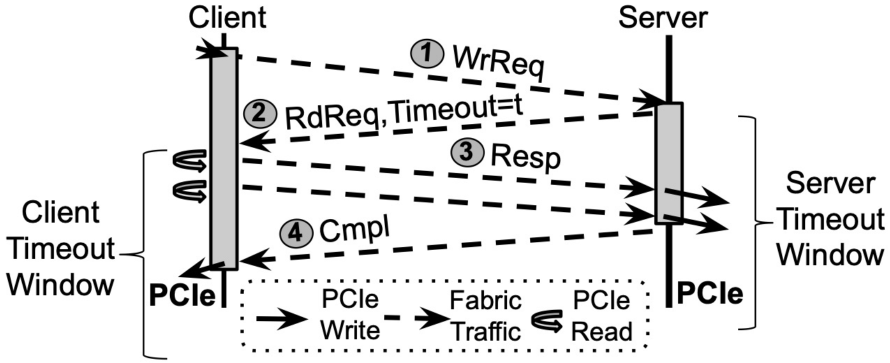
Endpoint congestion

- Caused by the endpoints
- Network routing cannot help (can make it worse)

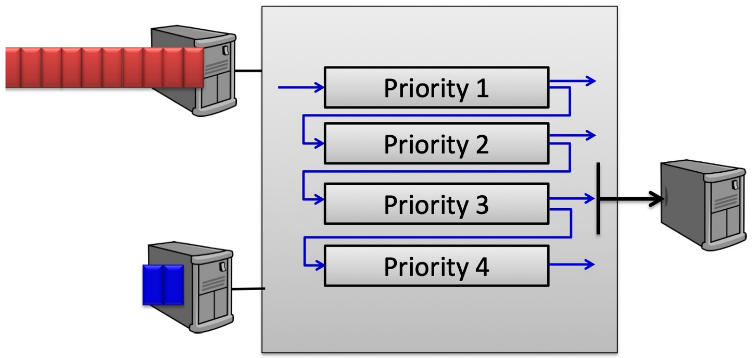
Predictable RDMA Network for AI/HPC



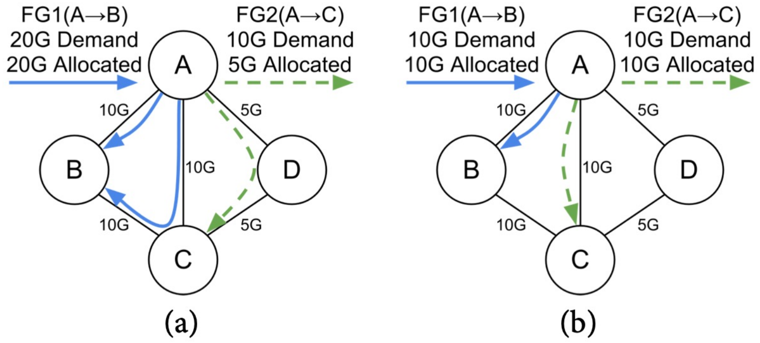
1. Congestion Control
(DCQCN)



2. End-to-end Flow Control
(1RMA)



3. Flow Scheduling
(PIAS)



4. Traffic Engineering
(B4 in WAN, ? in datacenters)

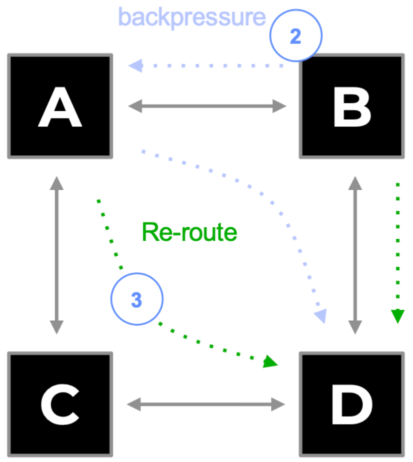
Predictable RDMA Network for AI/HPC

Conventional Network

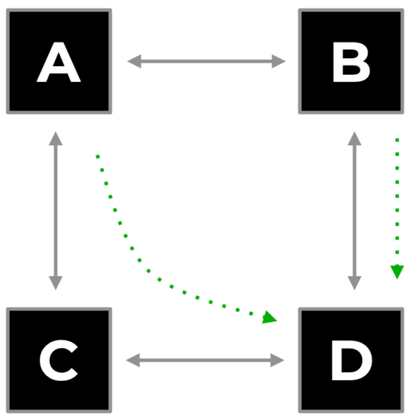
- Commonly done based on network backpressure
- Reactive approach makes the routing decision difficult, increases latency, and increases hardware complexity
- Network latency is **unpredictable**

Software-scheduled Network

- Avoids congestion
- Enables maintaining a deterministic TSP architecture to scale to a multi-node deterministic network execution



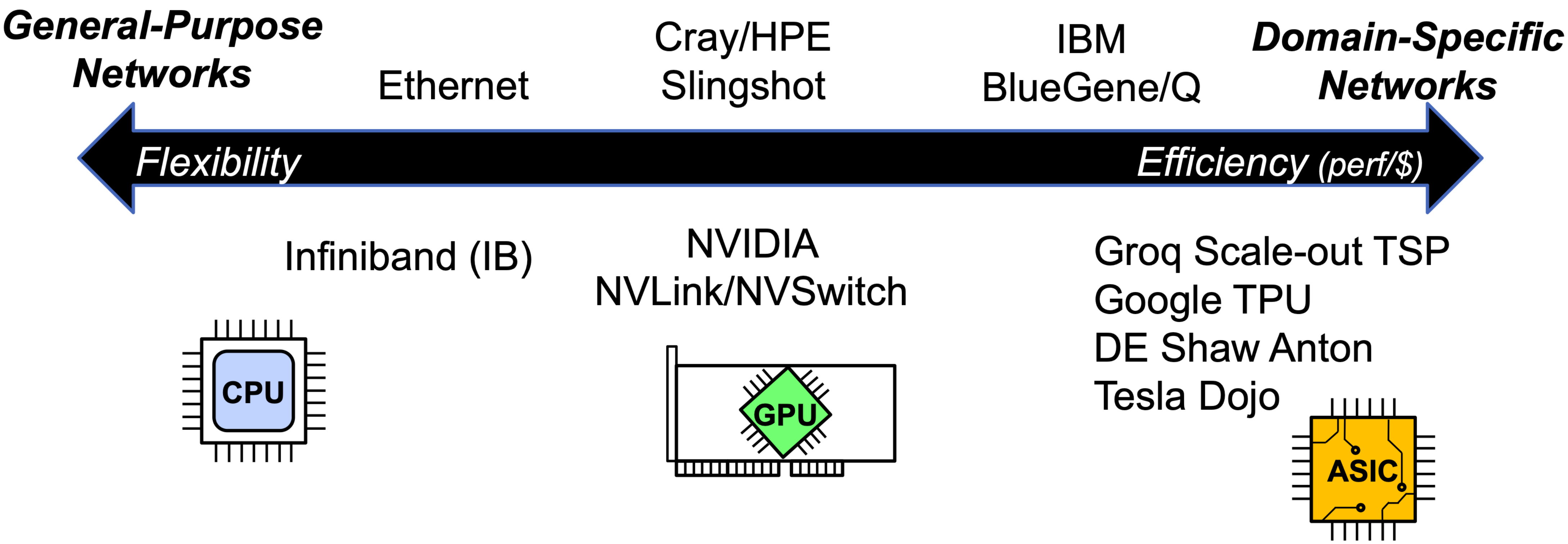
Traditional Non-deterministic Network



Software-scheduled Network

Domain-Specific Interconnection Networks in the Era of Domain-Specific AI Supercomputer, APNet '23 Keynote

General-Purpose vs. Domain-Specific



Domain-Specific Interconnection Networks in the Era of Domain-Specific AI Supercomputer, APNet ‘23 Keynote

My View on the Debate

- Programmability and scalability first
  - Programmability promotes ecosystem
  - Scalability enables a unified architecture for many scenarios
  - Identify bottlenecks in real systems before optimizing performance
- Consider GPUs vs. DSAs
  - DSAs have higher performance but CUDA has the best ecosystem
  - The price of H100 ($30K~$40K) is 15~20x of its manufacturing cost ($2K)

Summary

- Trend 1: Intelligent Network Devices
  - SmartNIC: FPGA, ASIC, NP and DPU
  - Programmable Switch
- Trend 2: Fast Interconnect
  - NVLink and CXL: Direct P2P with Memory Semantics
  - Convergence of AI and Cloud Networking
- Where should we put network intelligence?
  - The AI era is coming, so everything is going to be smart
  - SmartNICs for virtualization, switches for In-Network Telemetry, direct P2P among xPUs with memory semantics
  - Programmability and scalability first

Thanks!