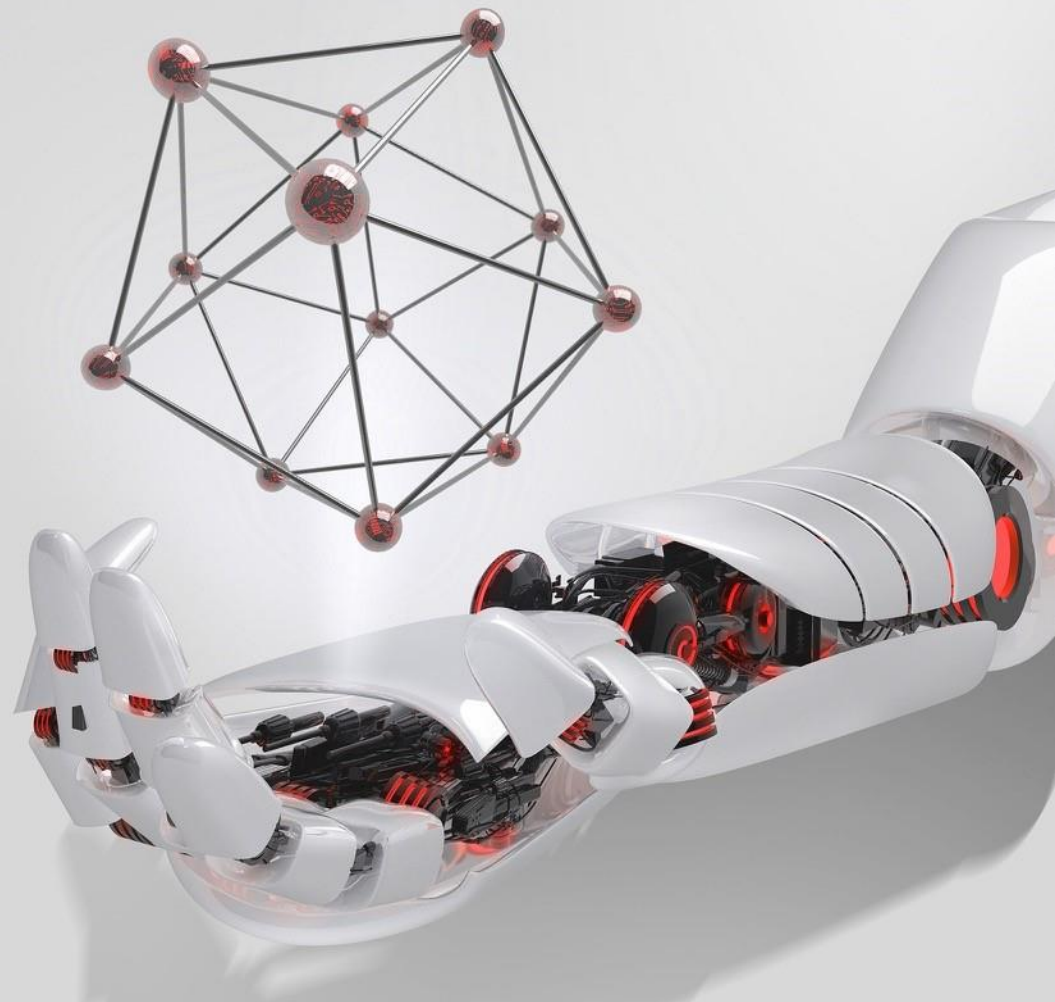


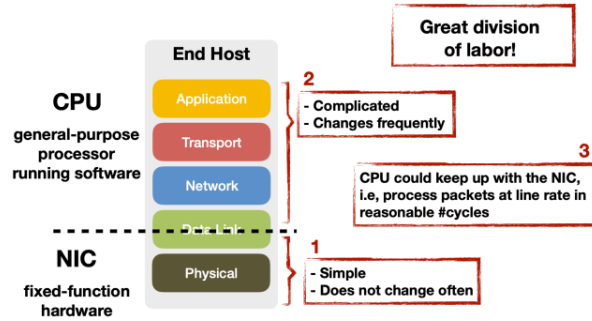
# Design Challenges of Manycore based SmartNIC & DPU

Author: Cheng Zhang  
Date : 2025-06-17

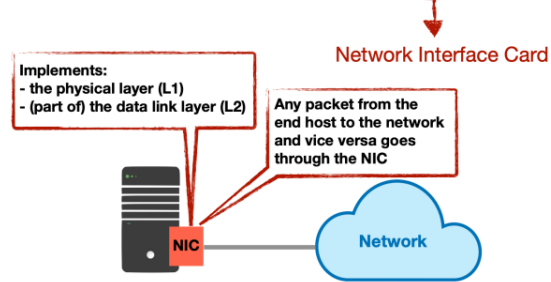


# Why Do We Need SmartNIC & DPU?

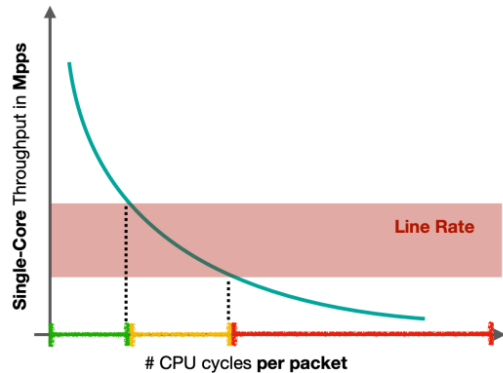
## What is a (dumb) NIC?



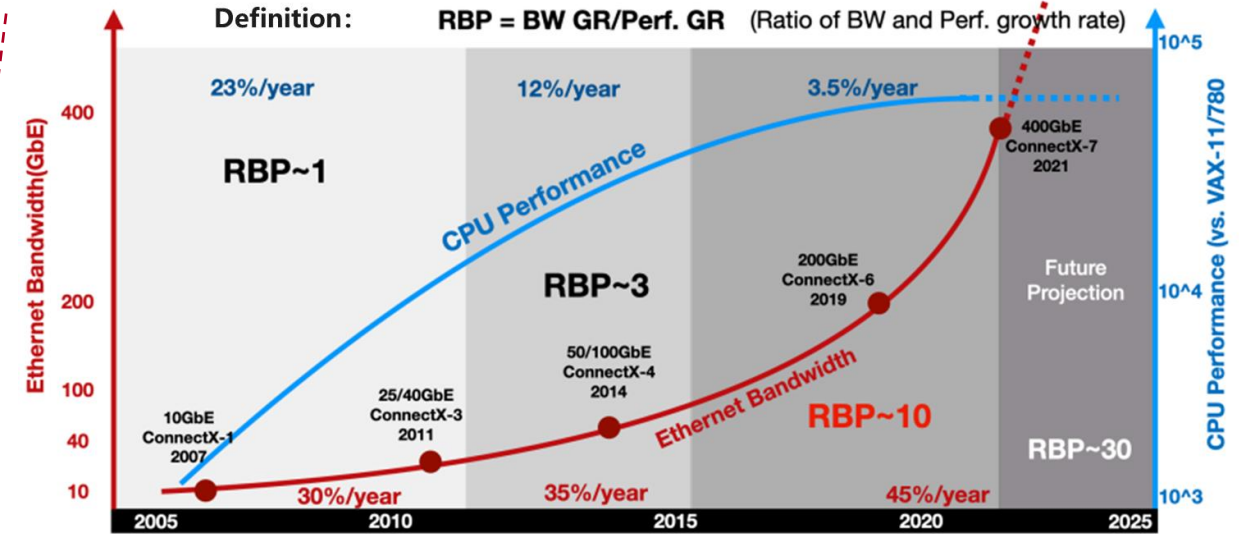
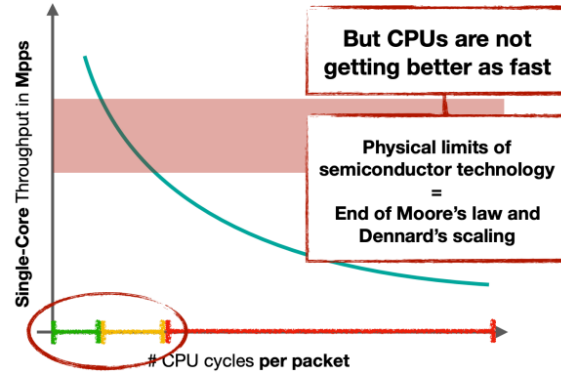
## What is a (dumb) NIC?



## Limits of Software Packet Processing



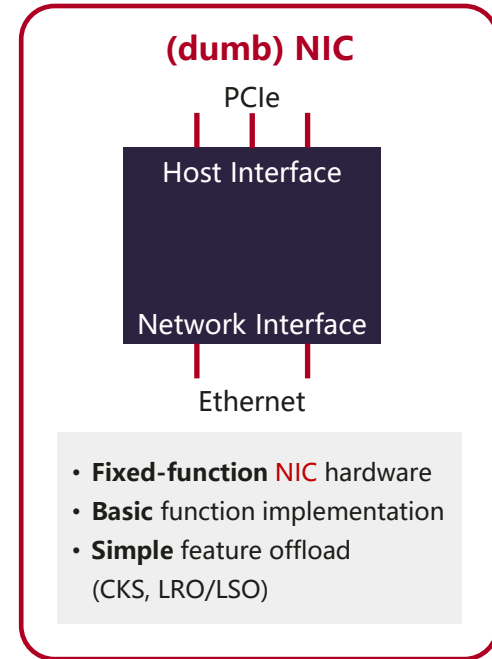
## Limits of Software Packet Processing



The **contradiction** between the rapid increase in data volume and the slowing down of Moore's Law

Cited from MIT 6.829 Slides

# Different Types of NICs

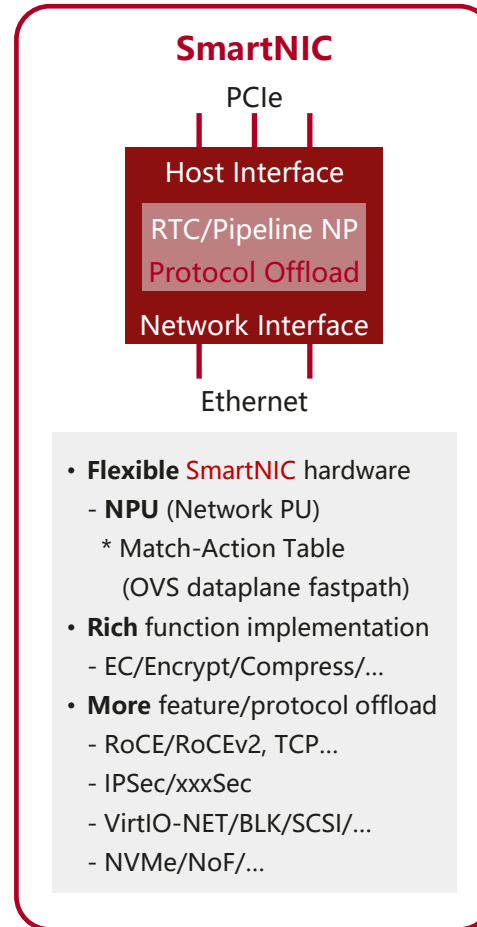


**Traditional NIC**

Simple Function Implementation

With the application of tunneling protocols such as VXLAN and virtual switching technologies such as OpenFlow and OVS,

the complexity of network processing is gradually increasing, requiring more CPU resources to be consumed.

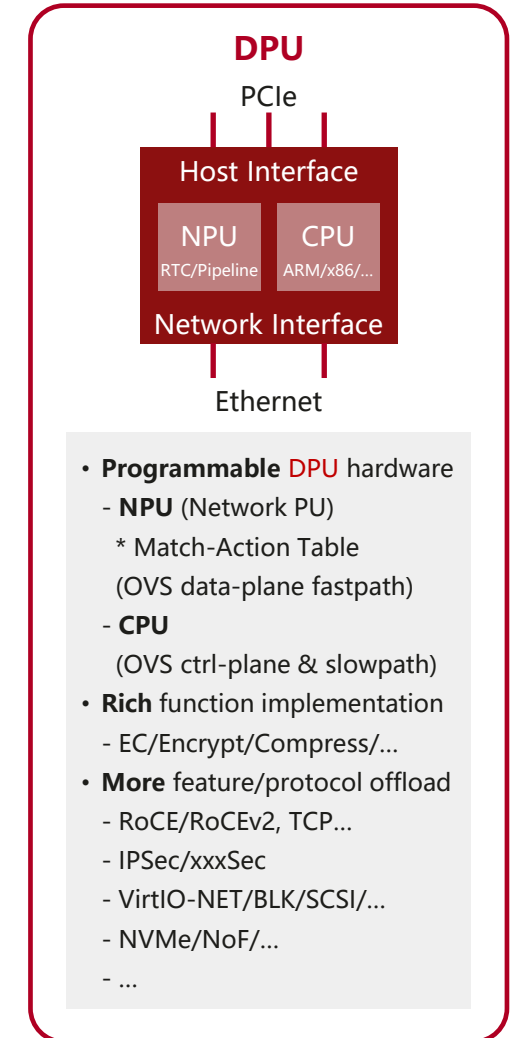


**SmartNIC**

Data-Plane Offload

With the continuous improvement of data center network speed, hosts still consume a large amount of valuable CPU resources to classify, track, and control traffic.

How to achieve "zero consumption" of host CPUs has become the next research direction for cloud vendors.



**DPU**

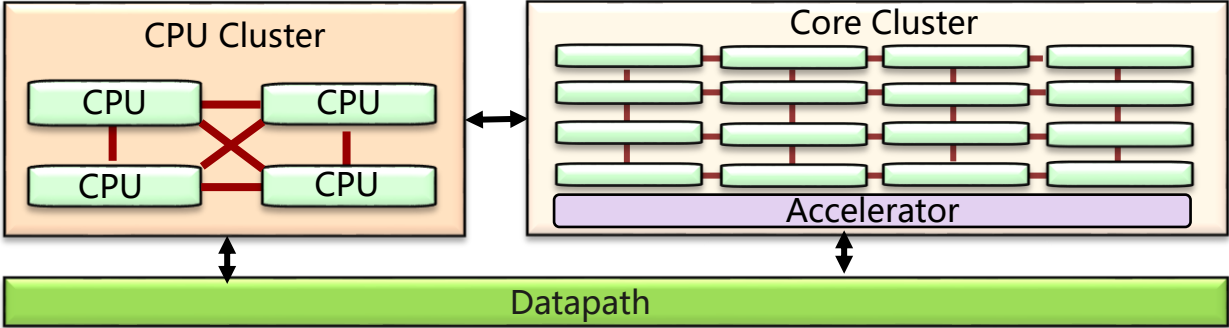
Data-Plane + Control-Plane Offload

# Mainstream Products: Architecture Classification

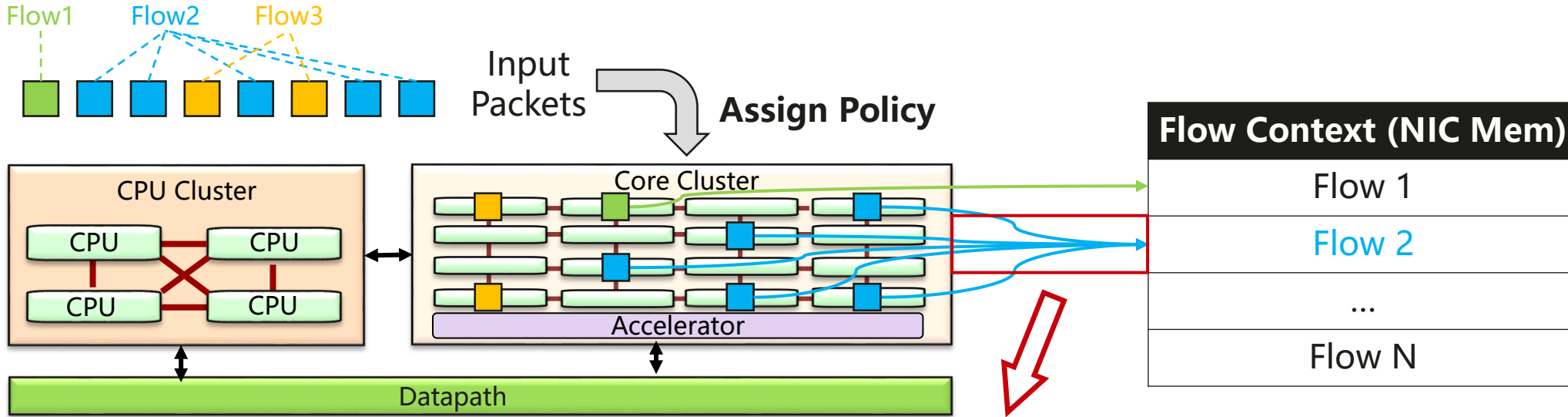
	Data-plane Architecture				
	ASIC	FPGA	NPU-Pipeline	Many Core	ASIC+NPU
SmartNIC	NV ConnectX-7	NetFPGA		Netronome CX/LX Marvell OCTEON	Huawei NV ConnectX-8
DPU	NV Bluefield1/2 BRCM Stringray	Intel FPGA IPU Xilinx Alveo	AMD Pensando Intel ASIC IPU	Huawei Netronome FX	NV Bluefield3

**Manycore based architecture** improve its *flexibility* and adapt to a wide range of application scenarios!

# A Reference Architecture of Manycore based DPU

	Reference Architecture
<b>Target Scenario</b>	<ul style="list-style-type: none"><li>• Storage</li><li>• Smart Compute</li><li>• Cloud Compute</li></ul>
<b>Architectural Description</b>	<ul style="list-style-type: none"><li>• Data Plane: programmable core cluster but without strong cache coherence</li><li>• Control Plane: General CPU Core</li></ul>
<b>Architecture</b>	 <p>The diagram illustrates the reference architecture of a Manycore based DPU. It features three main components: a CPU Cluster, a Core Cluster, and a Datapath. The CPU Cluster (orange box) contains four green CPU blocks arranged in a 2x2 grid, connected by red lines in a mesh topology. The Core Cluster (yellow box) contains a 4x4 grid of green blocks, with a purple Accelerator block at the bottom. A horizontal double-headed arrow connects the CPU Cluster and the Core Cluster. Both the CPU Cluster and the Core Cluster are connected to a green Datapath bar at the bottom via vertical double-headed arrows.</p>

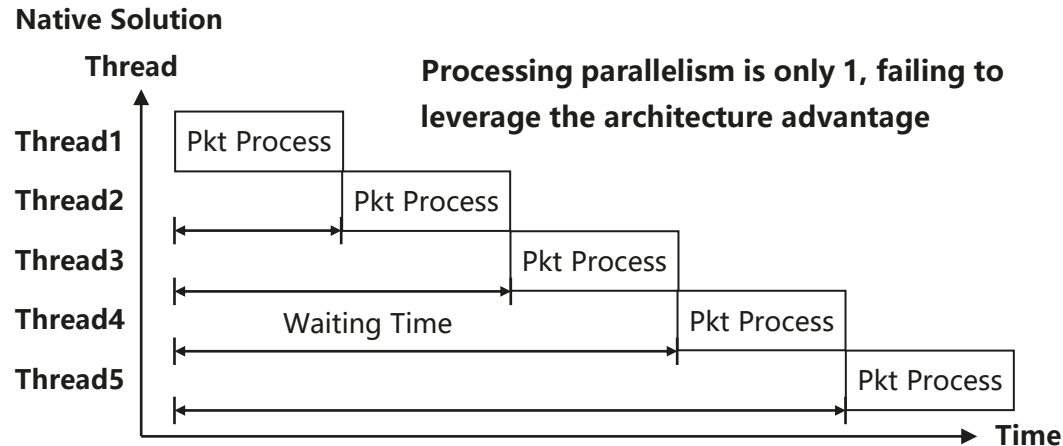
# Challenge 1: Shared Context Processing on Manycore



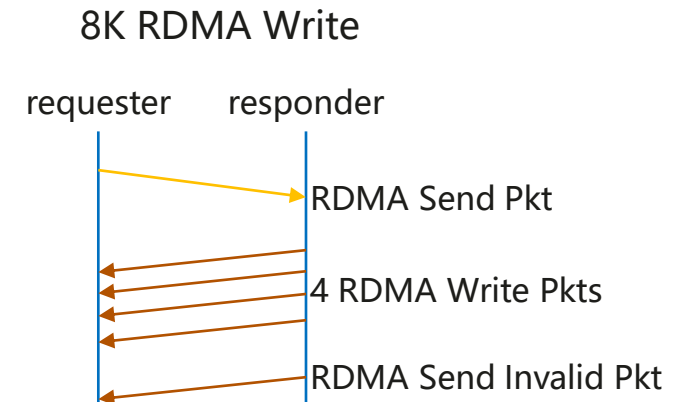
## Competition & Conflict

Multiple cores/threads which process different packets of the same flow will access the same context

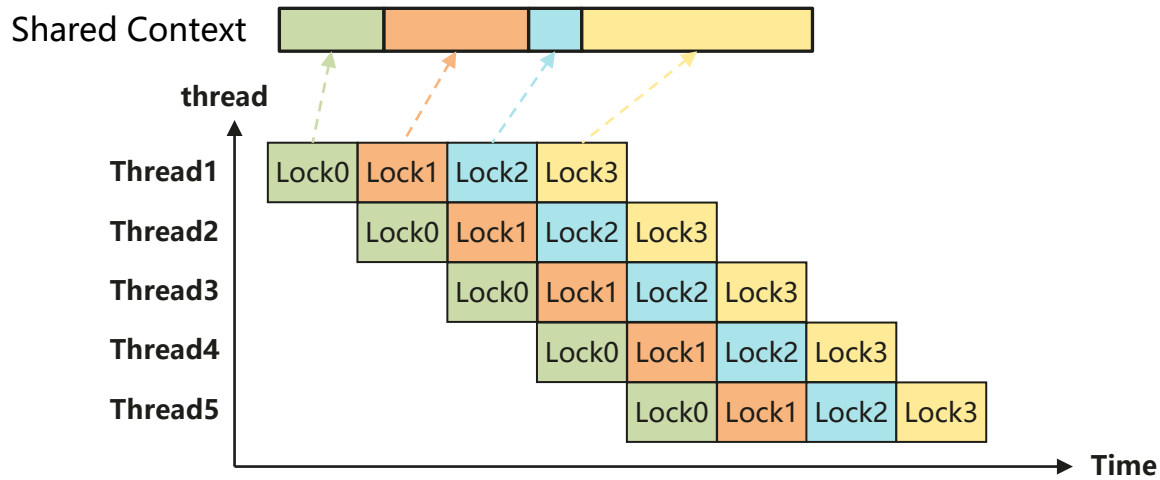
### Example



### RDMA Write Process

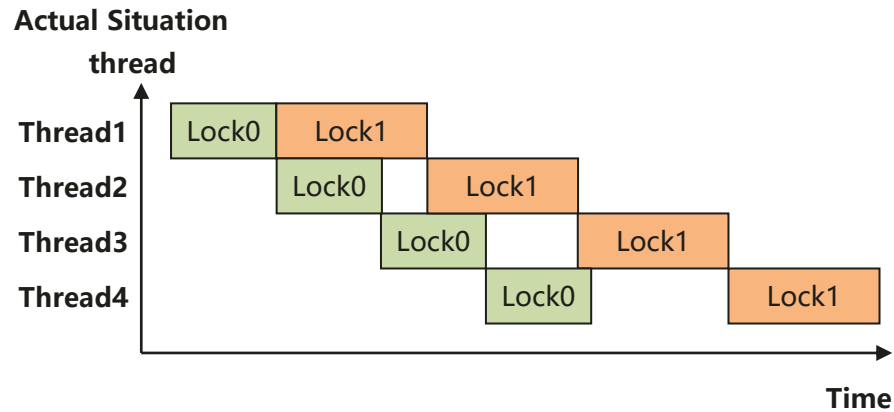


# Challenge 1: Shared Context Processing on Manycore



## Common Ideas: Using multiple small locks

So that multiple cores/threads can process different packets of the same flow in parallel.

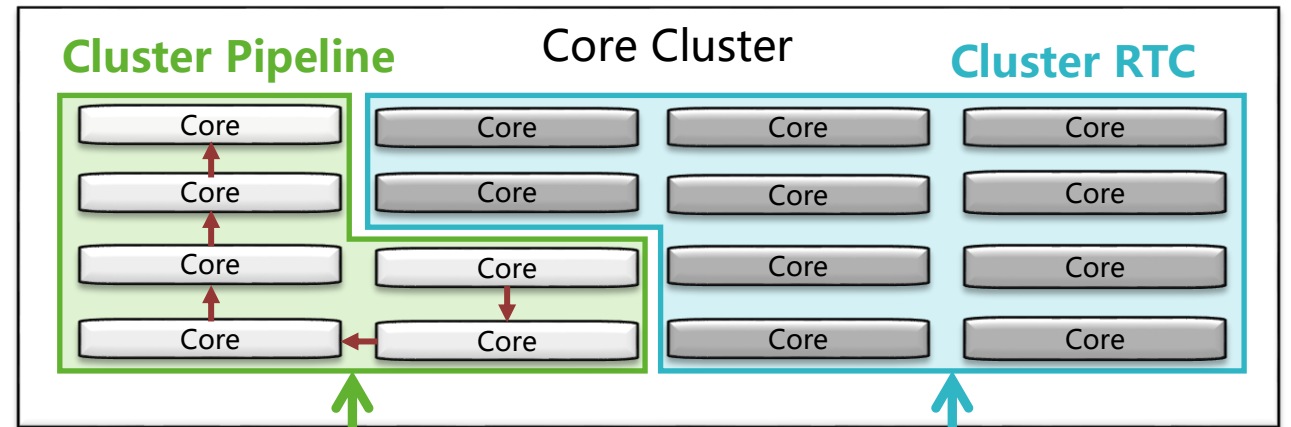


- **Observation 1:** You need more finer locks
  - The maximum parallelism for a single flow is limited by number of locks.
- **Observation 1:** You need more time for operating locks
  - High latency due to long waiting time for packets in lock queues due to empty bubbles.
- **Possible Explorations:**
  - Domain specific locks.
  - Closer coupling of locks and operations.

# Challenge 2: One Architecture Meets All

- **Observation 1:** Different stateful applications may have different sensitivities to different metrics.
  - Some applications are more sensitive to throughput, while others are more sensitive to latency.
- **Observation 2:** RTC/PIPELINE processing model may have different affinities.
  - Pipeline model is high throughput oriented, while RTC model is low completion latency.
- **Possible Exploration :** Employ flexibility NOC and cores to meet different metrics.

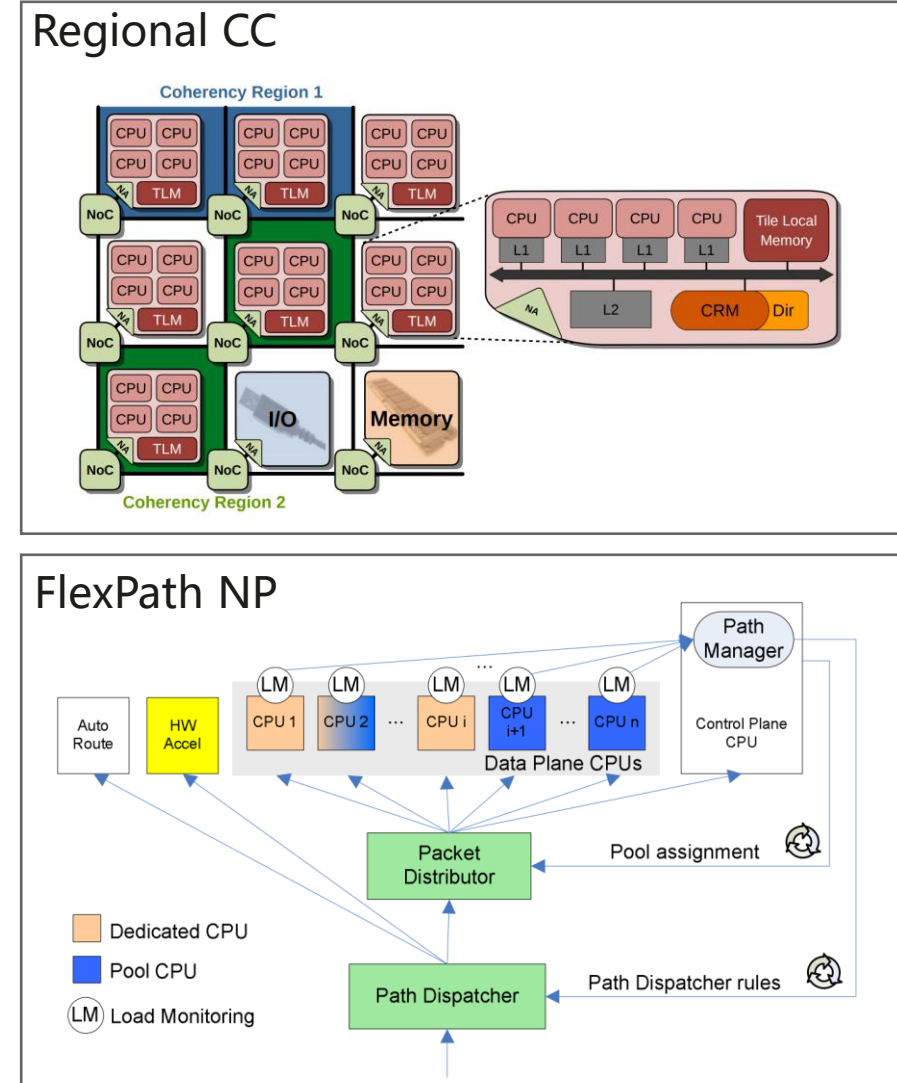
Application Scenario	Low Latency	High Throughput	Flexibility	Large Flow Number
HPC	✓	○	○	✓
Distributed Database	✓	○	○	○
Distributed Storage	✓	○	✓	✓
Virtualization / Cloud	○	✓	✓	✓
AI LLM	○	✓	○	✓





# Challenge 2: One Architecture Meets All

- **Observation 1:** How to mitigate context synchronizations?
  - context access/passing accounts for non-negligible processing latency.
  - Some contexts show locality shared among physical cores.
- **Possible Exploration:**
  - Faster and reconfigurable contexts swapping mechanisms.
- **Observation 2:** How to flexibly and adaptively schedule cores?
  - Core scheduler: Allocating cores based on concurrent latency/bandwidth sensitive workloads.
  - Packet dispatcher: Distribute packets to appropriate cores for application processing.
- **Possible Exploration:**
  - Adaptive core scheduler and flexible packet dispatcher like FlexPath NP.



- Ohlendorf, R., Meitinger, M., Wild, T., & Herkersdorf, A. (2010). FlexPath NP—Flexible, Dynamically Reconfigurable Processing Paths in Network Processors. *Dynamically Reconfigurable Systems: Architectures, Design Methods and Applications*, 355-374.
- A. Srivatsa, S. Rheindt, T. Wild and A. Herkersdorf, "Region based cache coherence for tiled MPSoCs," *2017 30th IEEE International System-on-Chip Conference (SOCC)*, Munich, Germany, 2017, pp. 286-291

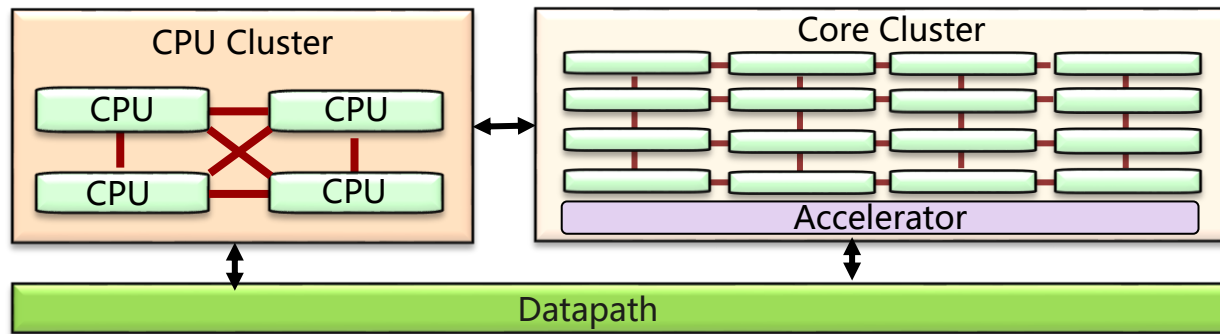
# Challenge 3: Achieving Ultra High Bandwidth

➤ **Observation 1:** Non-linear scale problem for manycore processing.

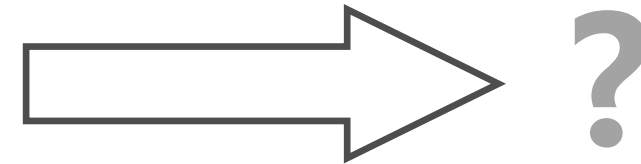
- But for the future NIC how to achieve 1.6 Tbps+ processing performance with manycore architecture?
- Especially, it is the most challenge to support line-rate stateful applications.

➤ **Possible Exploration:** does it still should be defined as a traditional NIC?

- Merging high bandwidth Eth/PCIe switching TOR with NIC.



From 400Gbps, 800Gbps, 1.6Tbps



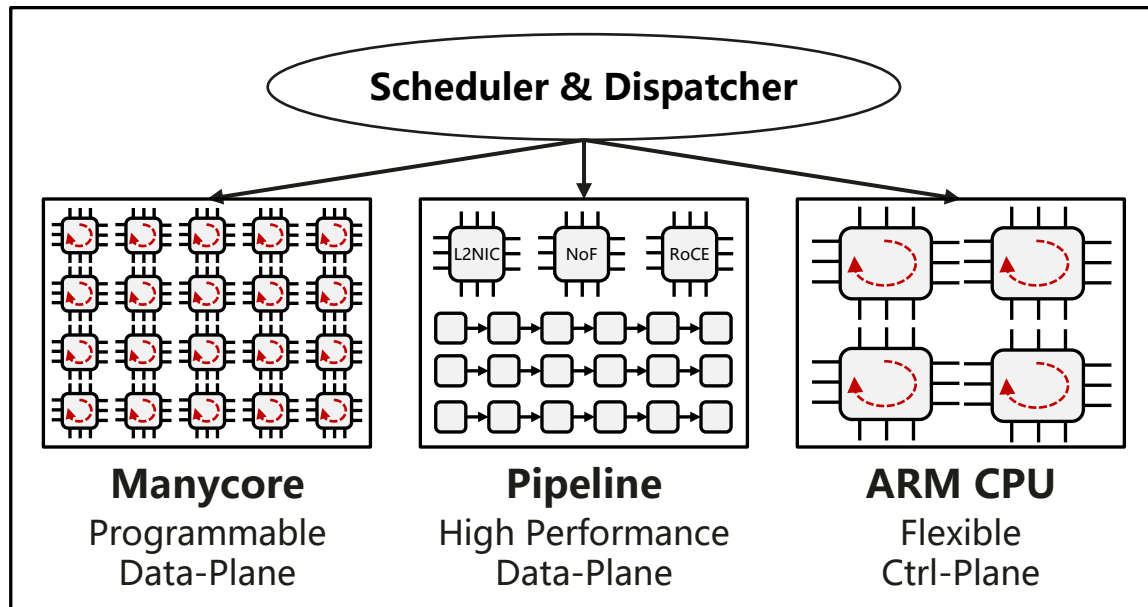
To 3.2+Tbps

# Challenge 4: Architectural Evolution for Future

- **Observation 1:** how to evolve the architecture, enhance low latency and high bandwidth capabilities without losing existing programmable capabilities?

## Possible Exploration 1:

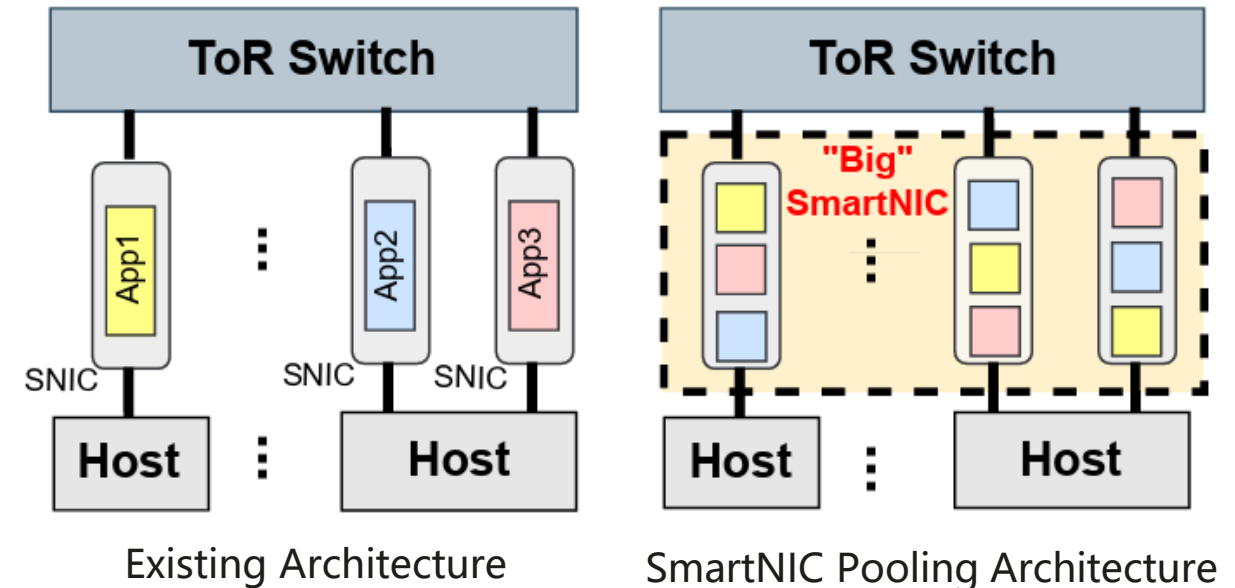
### To Heterogeneous based Multi-processor



Recent Commercial DPU products employs similar approaches

## Possible Exploration 2:

### To Homogenous based Pooling System



Recent cloud vendors employs similar approaches  
(e.g., Azure employs SmartNIC pooling for NF offloading [NSDI '23])

**Q & A**