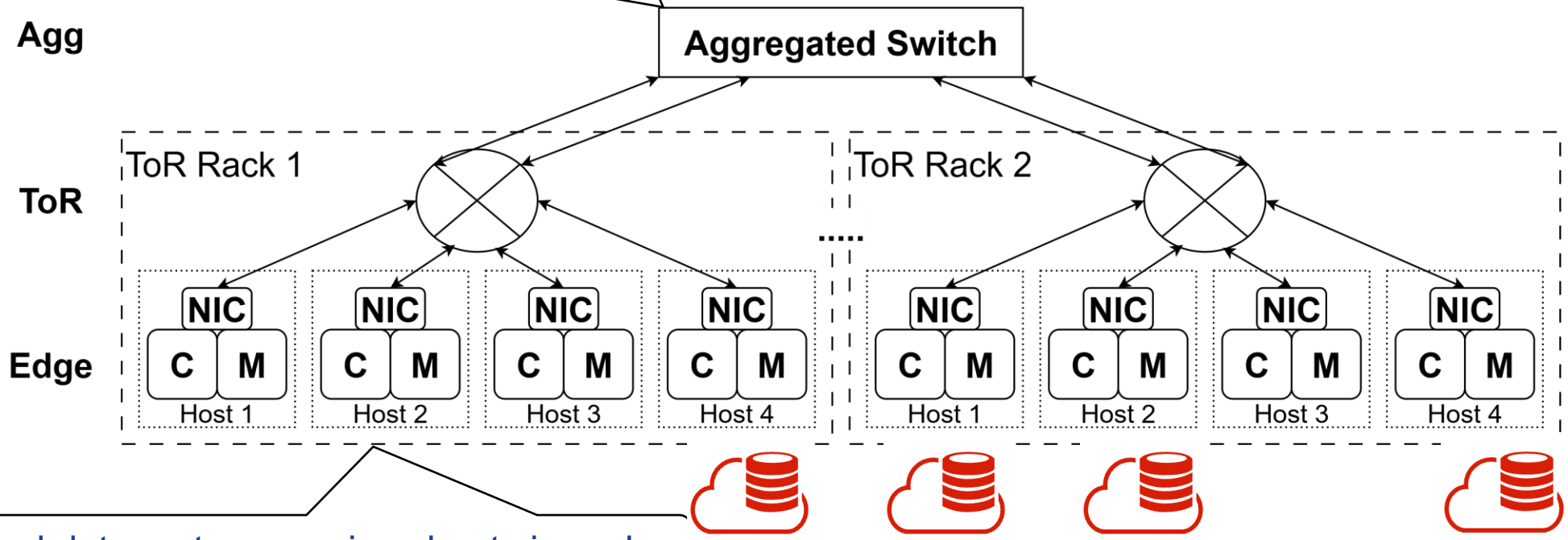
DRack: A CXL-Disaggregated Rack Architecture to Boost Inter-Rack Communication

Xu Zhang, Ke Liu, Yuan Hui, Xiaolong Zheng, Yisong Chang, Yizhou Shan, Guanghui Zhang, Ke Zhang, Yungang Bao, Mingyu Chen, Chenxi Wang

USENIX ATC '25

Background: ToR-based Racks

- Network interconnects (e.g., Ethernet) between hosts

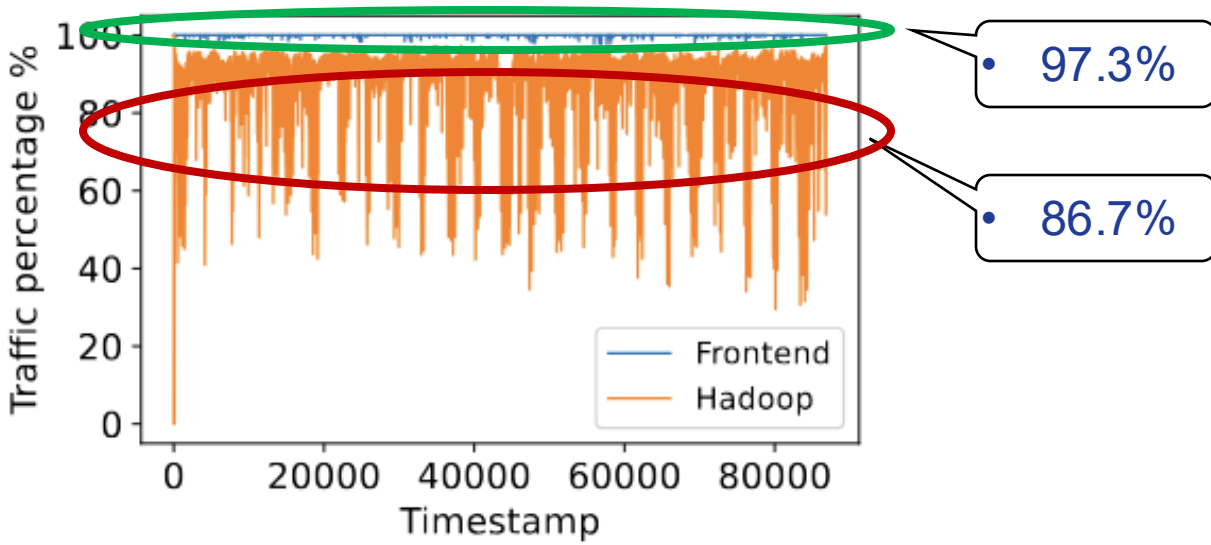


- Conventional datacenter organizes hosts in racks
- Data intensive applications are scheduled across multiple hosts and racks

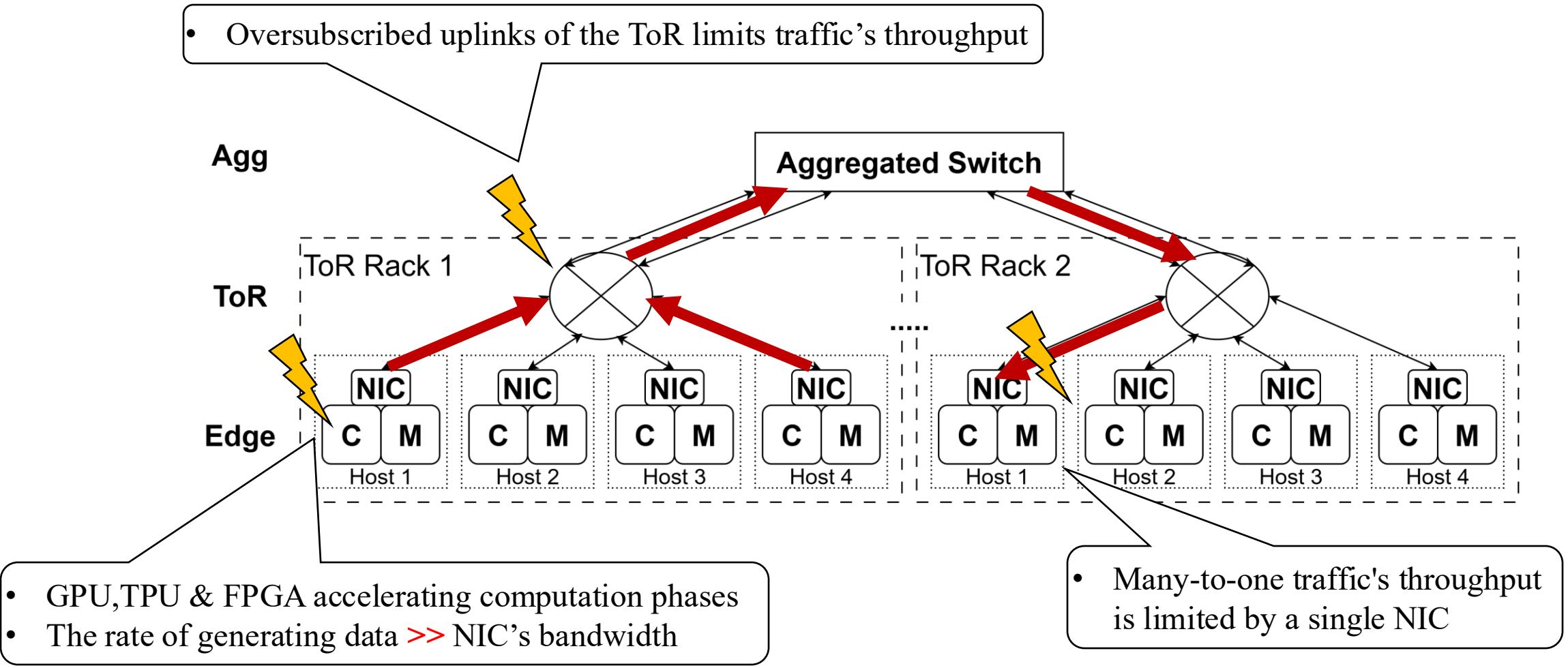
Motivation：Rising and Inevitable Inter-Rack Traffic

Experiment: a public dataset from Facebook

- Significant demand for inter-rack communication
  - Service-based rack organization
  - Resource fragmentation
  - Large scale applications

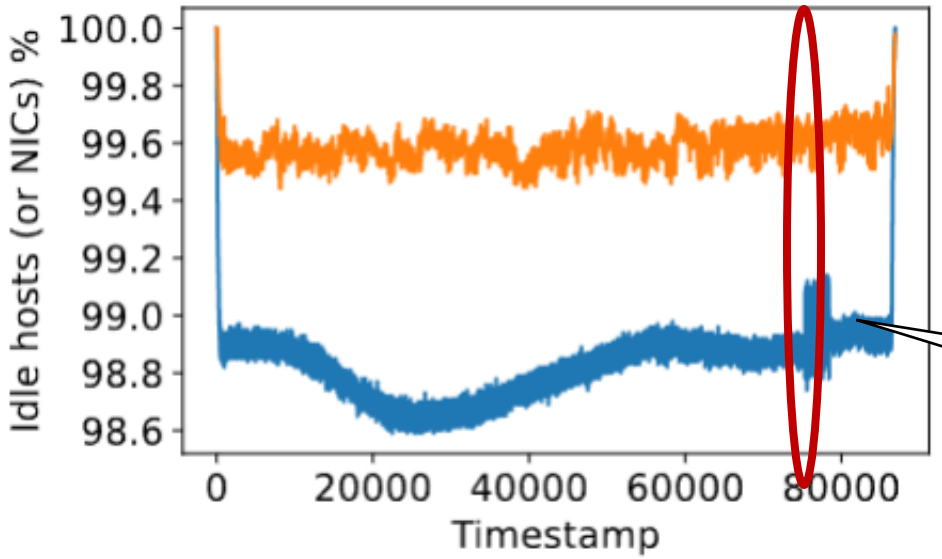


Motivation： Inter-Rack Comm. Efficiency is Crucial



Insight

The bandwidth utilization of existing NICs within a rack remains low

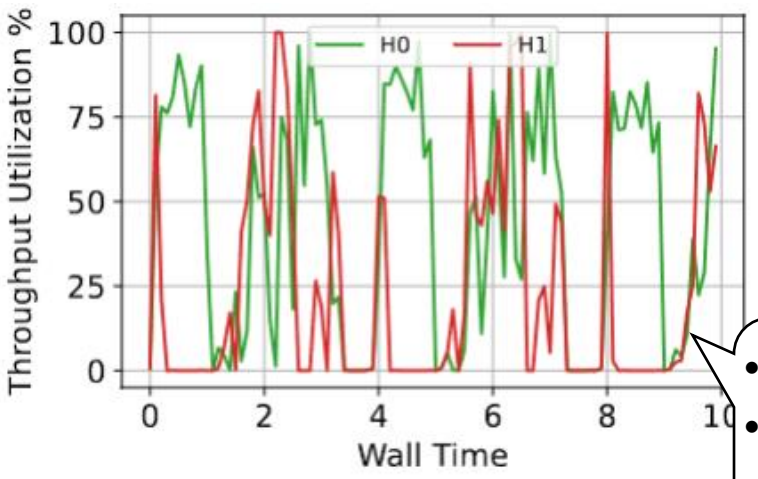


- Over 90% of hosts (NICs) are not sending or receiving in 1s

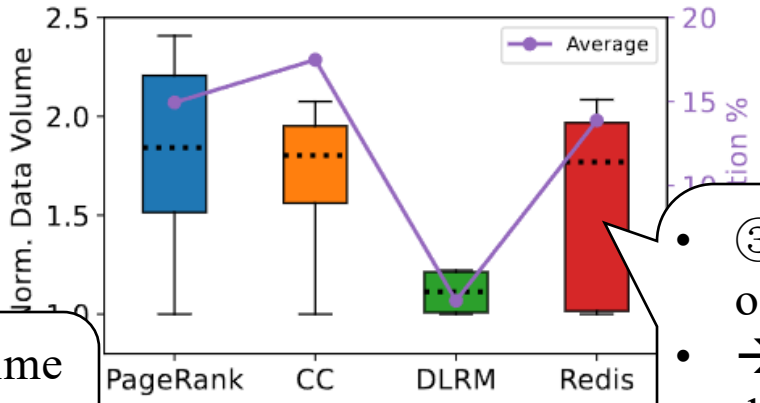
Insight

Experiment: running data-intensive apps over 8 servers

- Three main reasons
  - ① A host running a non-distributed job → underutilized NICs
  - Distributed jobs:



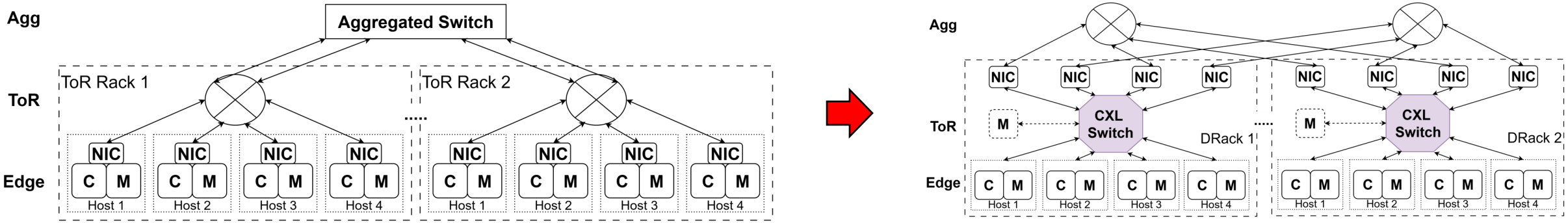
- ② Varied computation time
- → hosts communicating at different time



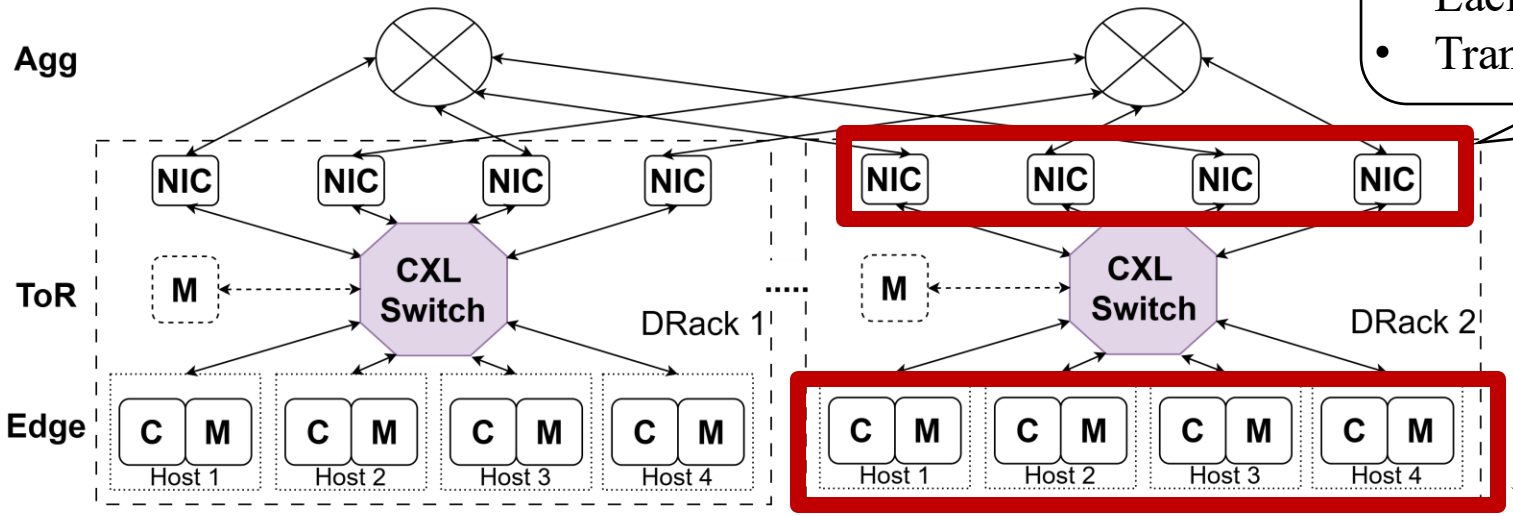
- ③ The uneven distribution of data between hosts
- → NICs processing a disproportionate amount of data

Design Principle

- A new rack architectural concept – Disaggregated Rack (DRack)
  - Leveraging NIC pool and memory pool



Design Principle



NIC Pool

- Shared by all hosts
- Each NIC exposes one virtual NIC (vNIC) to each host
- Transmitting inter-rack traffic using all NICs

❖ NIC pool's bandwidth >> single PCIe link or host's memory bandwidth

❖ CXL Fabric supports shared memory pool!

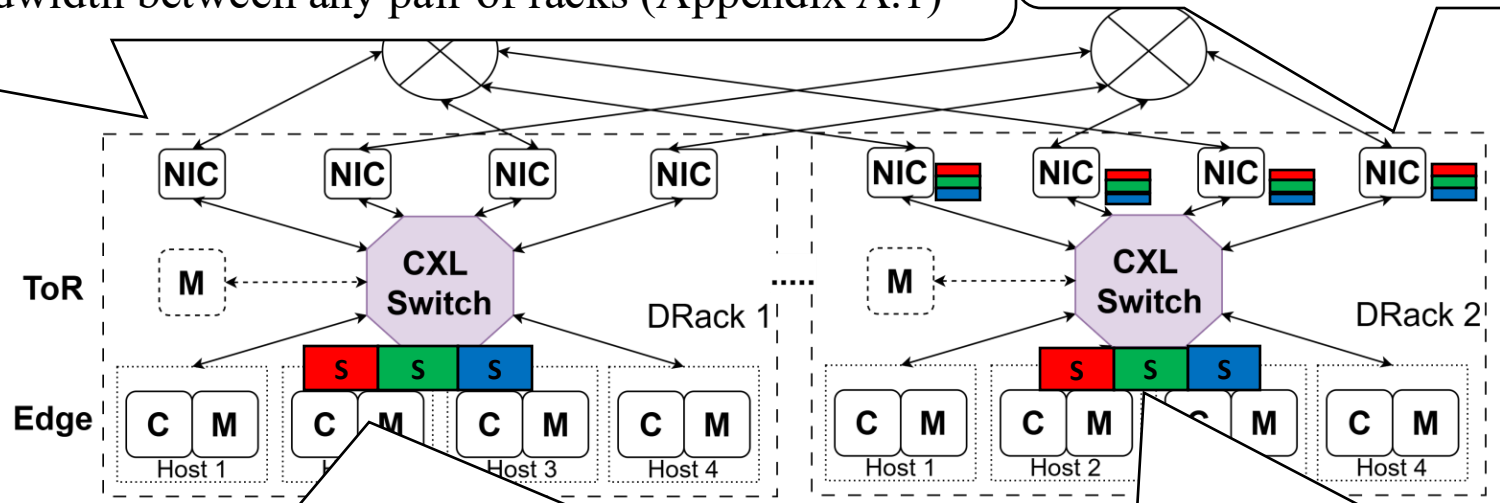
Memory Pool

- system-integrated memories from all host
- Providing high bandwidth network buffers

Benefits

Oversubscribed network core
- NIC pool acting as the uplinks of ToR tier
- A fully bisectional bandwidth between any pair of racks (Appendix A.1)

NIC egress bottleneck
- Each network stream leveraging all NICs



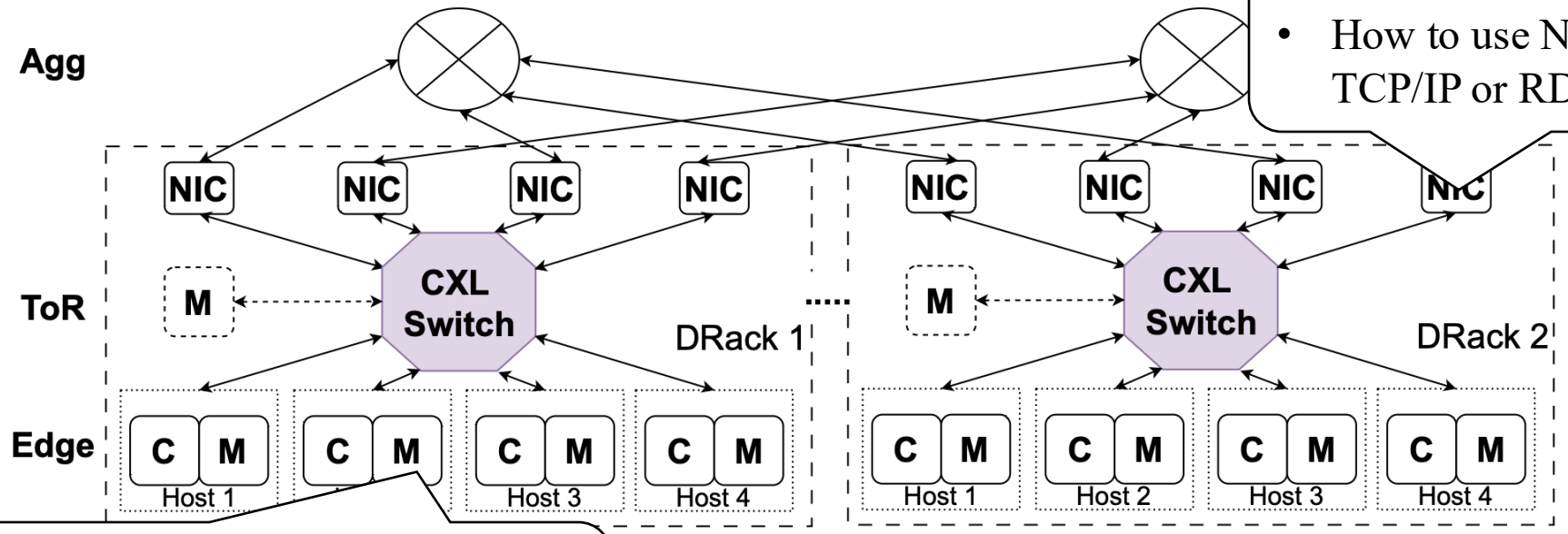
NIC ingress bottleneck
- Network packets received by all NICs and DMA to memory pool

- In the computation stage, generated intermediate data is stored in memory pool

❖ DRack is orthogonal to and can improve existing job schedulers
- Shown in Section 2.4

Challenges

- There are 2 challenges to overcome to ensure DRack's efficiency and practicality



Compatibility
- How to use NIC pool based on the TCP/IP or RDMA stack?

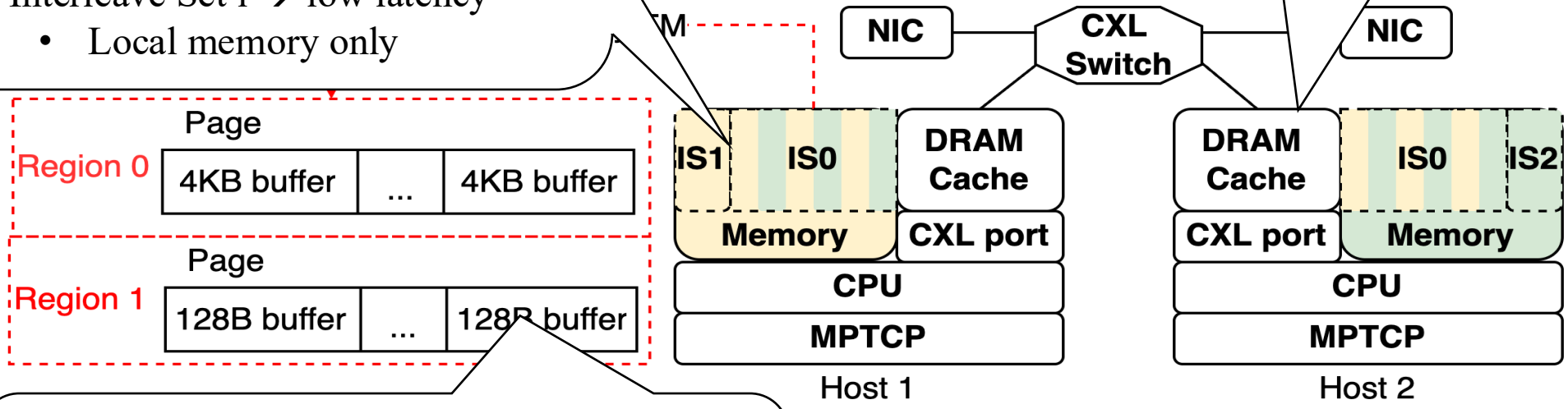
Latency
- Accessing memory pool via CXL exhibit at least 2.7x latency than accessing local memory

Design Details

Challenge 1 - Latency

Memory Pool
- Interleave Set 0 → high bandwidth
  - Across all hosts' memory
  - 256B interleave granularity
- Interleave Set i → low latency
  - Local memory only

DRAM Cache
- hide the latency of accessing IS0



Data Placement
- IS0 stores network buffers or user buffers
- ISi stores latency-critical metadata
  - E.g., descriptor queues

Challenge 2 - Compatibility

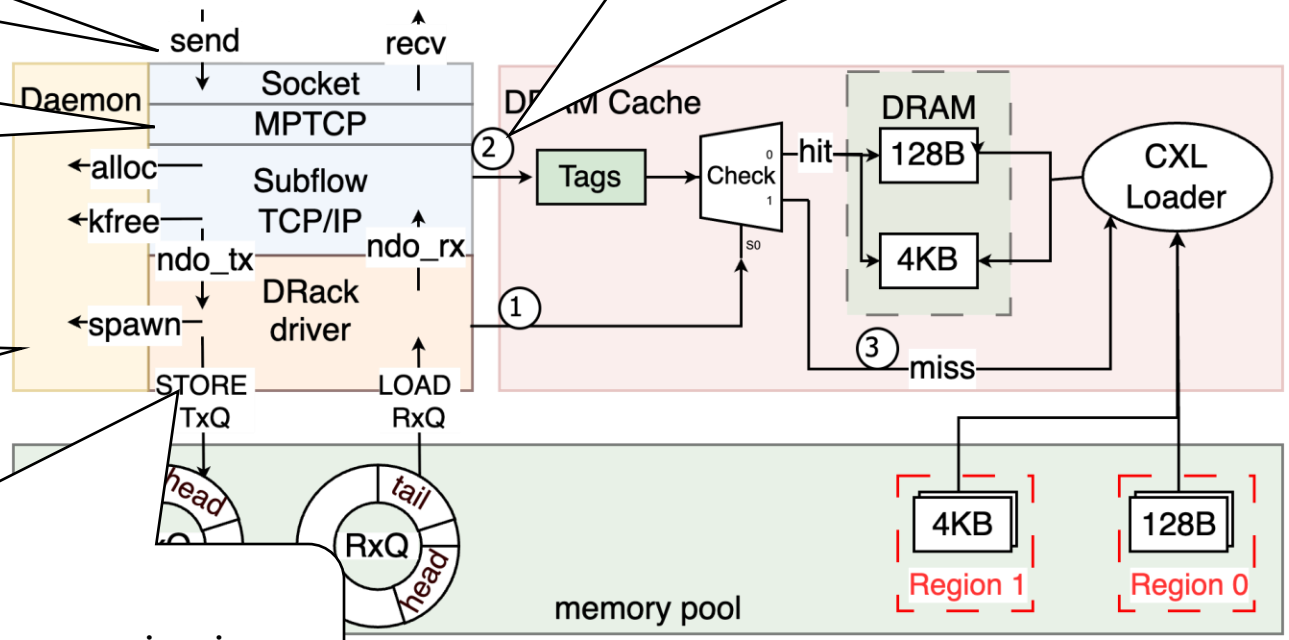
- Socket programming model based on TCP/IP

MPTCP
- Splitting a flow into multiple subflows
- Distributing subflows among vNICs

Daemon
- Apply & manage memory pool

DRack Driver
- Orchestrating inter- and intra-rack communication
- Updating vNICs' metadata bypass DRAM cache

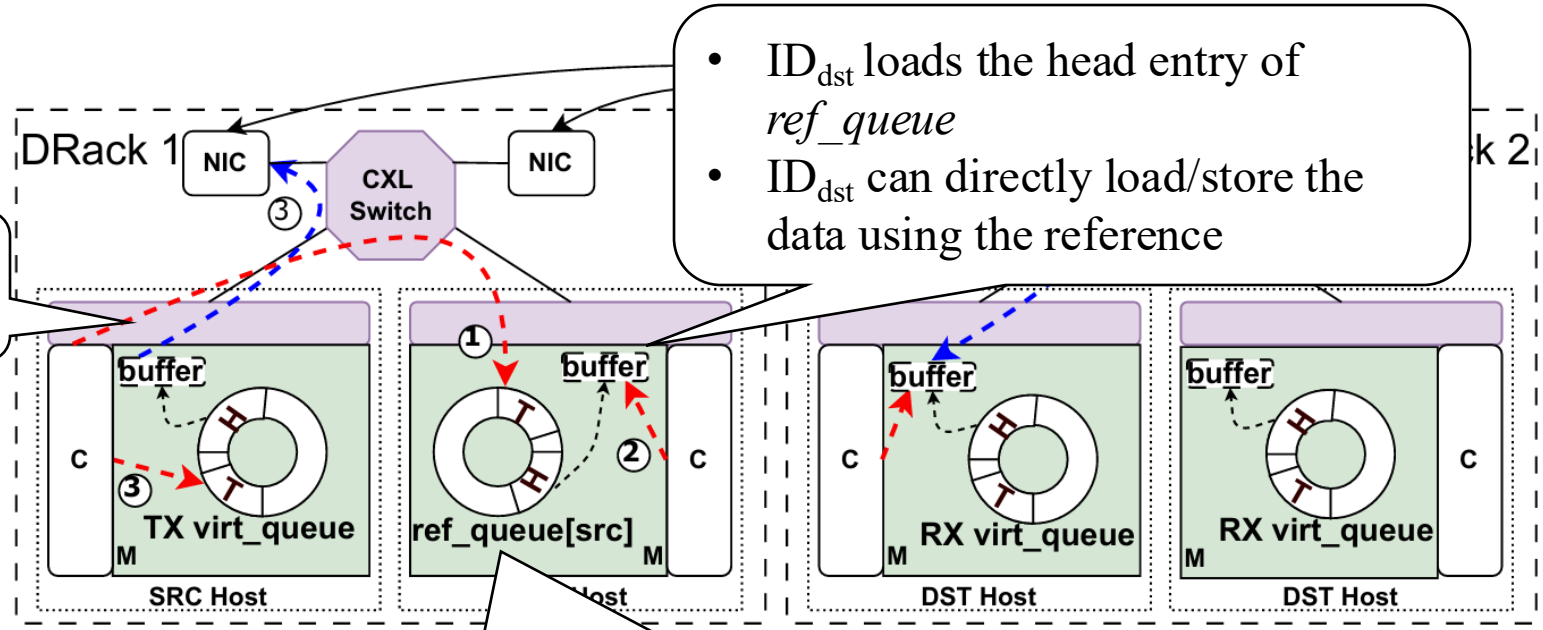
- network buffers located at IS0 are cacheable



Intra-DRack Data Transfer

❖Pass-by-Reference

- ID_{src} stores the reference into the tail entry of ID_{dst} 's *ref_queue*



- ID_{dst} loads the head entry of *ref_queue*
- ID_{dst} can directly load/store the data using the reference

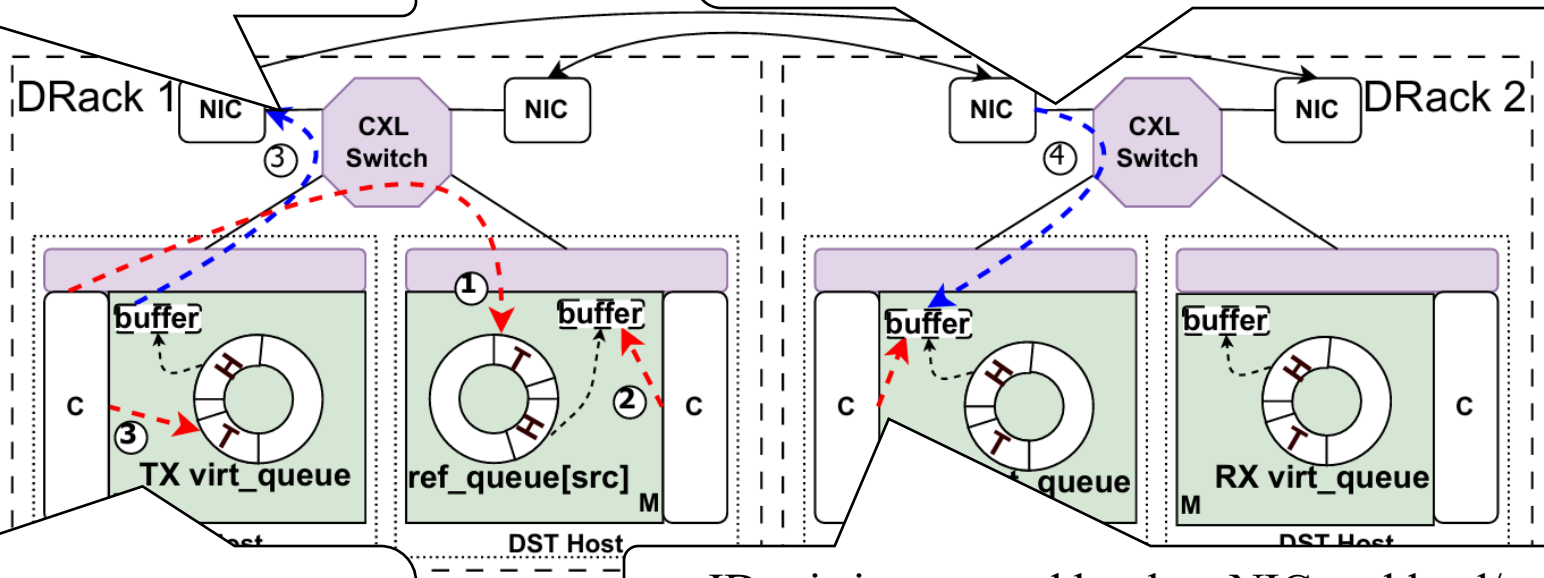
- ID_{dst} allocates a separate *ref_queue* for every ID_{src}
- Stored in IS_{dst}

Inter-DRack Data Transfer

❖Pass-by-Value

- vNIC DMA read descriptors and TX Buffers

- Each vNIC DMA writes the payload to the RX Buffers specified by the descriptors in the RX *virt_queue*



- Each vNIC has a pair of RX/TX *virt_queue*
- ID_{src} stores descriptors to TX *virt_queue* and ring vNIC's doorbell via MMIO

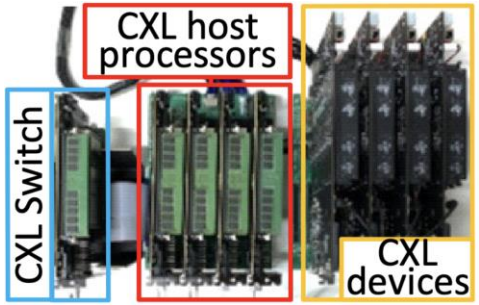
- ID_{dst} is interrupted by the vNIC and load/store RX Buffers

Implementation

Software-Hardware co-Simulation Prototype

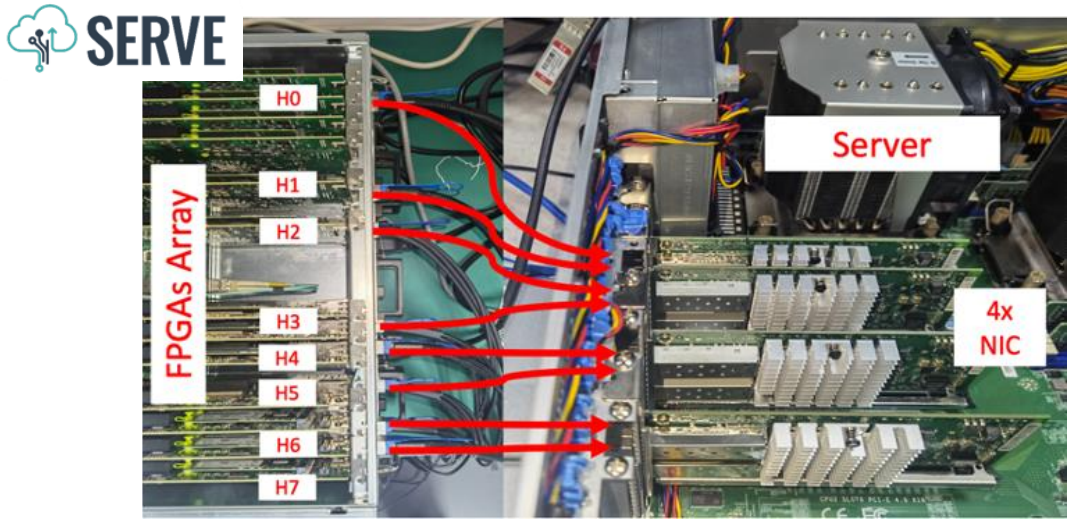
❖ Simulation requirements

- Multiple hosts running real-world apps, flexible network topology



DirectCXL ATC'22

	Software simulator	FPGA-based implementation
Speed	✗	✓
Flexibility	✓	✗
Development	✓	✗

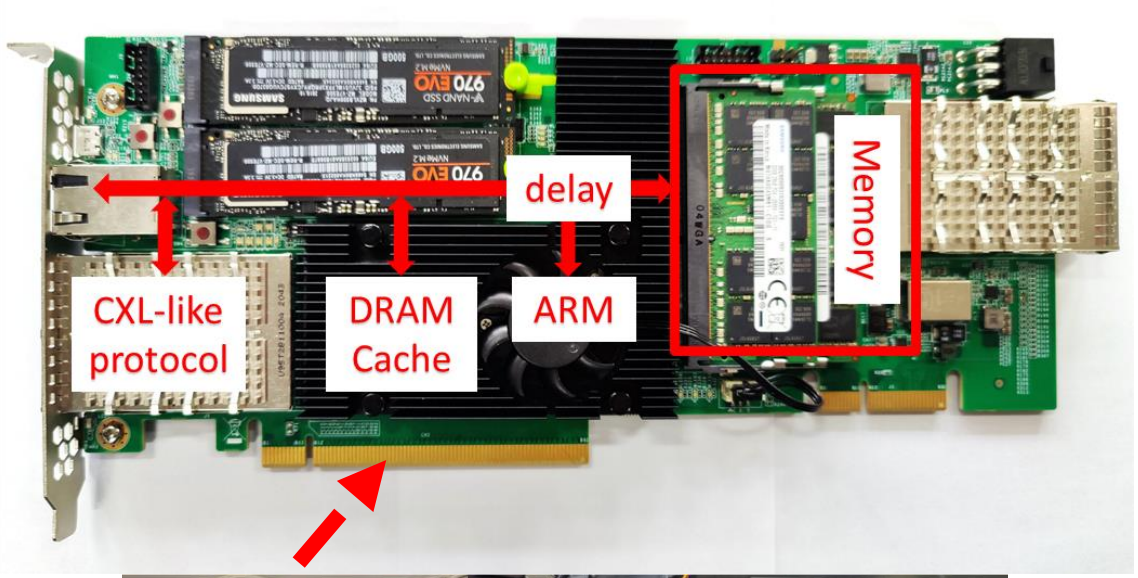


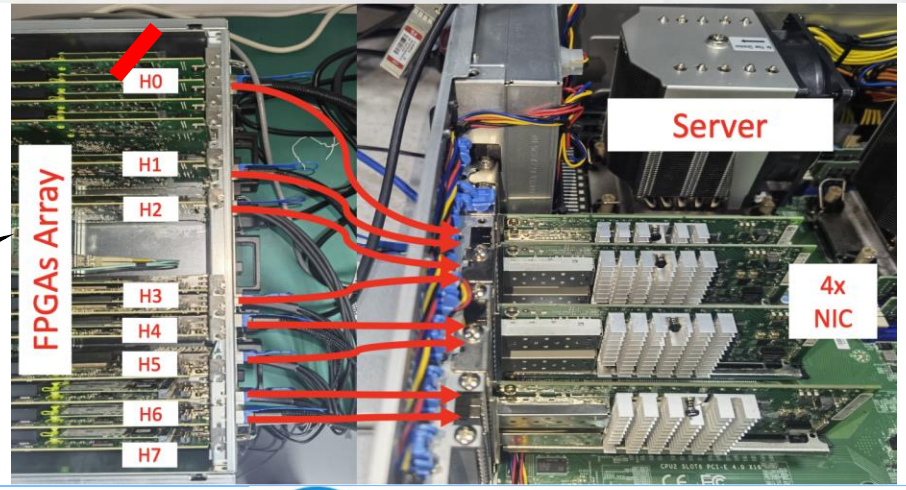
Our approach

- Software simulator → flexible network topology
- MPSoC-FPGA → hosts supporting CXL

MPSoC FPGAs Simulating Hosts

- **Quad-core ARM CPU running real-world apps fast**
- **MPSoC exports the CPU's load/store to FPGA logic via HP/HPC ports**
- **FPGA IPs**
  - DRAM cache
  - CXL.mem/.io-like protocol
  - Delay → insert latency to local memory

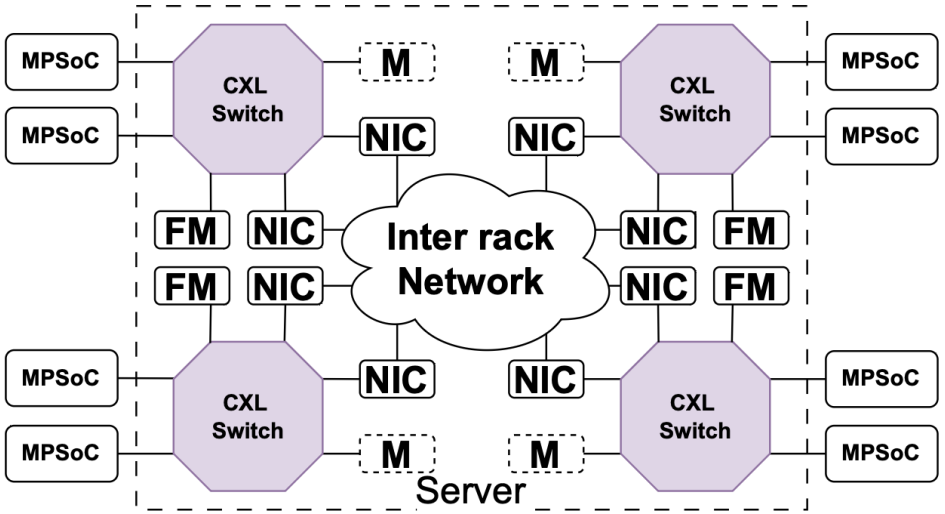


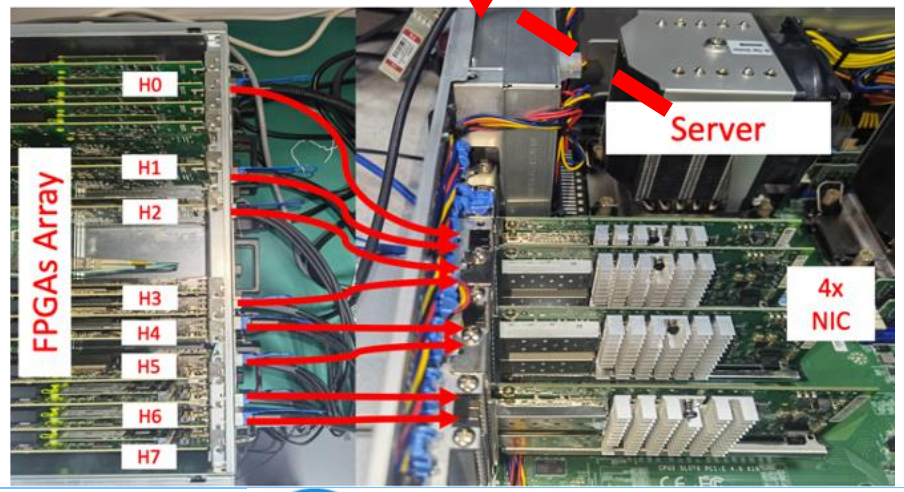


- 8 MPSoC FPGAs link to a server

Software Simulating Network Topology

- **Software simulator based on DPDK**
  - CXL switch, NICs, inter-rack Ethernet network
  - Routing CXL.mem flits among MPSoCs
  - Routing CXL.io flits between MPSoCs and NICs
  - Routing Ethernet packets among NICs



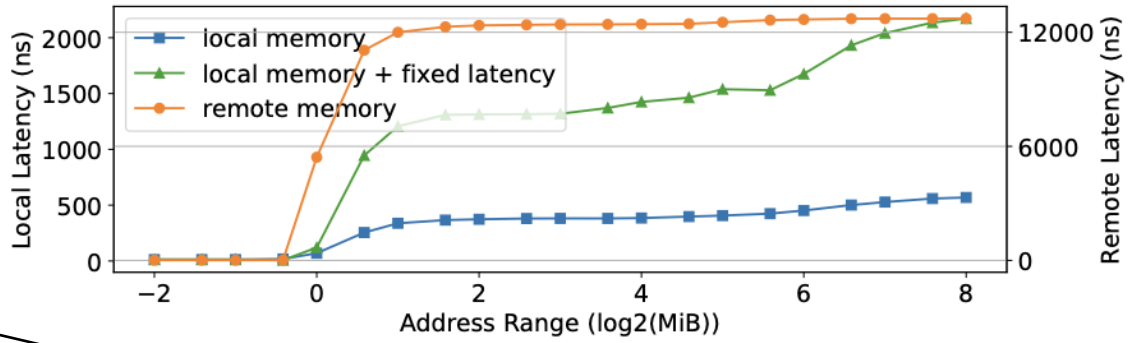


Evaluation

Platform Parameters

Memory pool's latency

- One-layer CXL switch is expected to add 170 to 270 ns
- $2.7 < Latency_{remote} : Latency_{local} < 6.4$



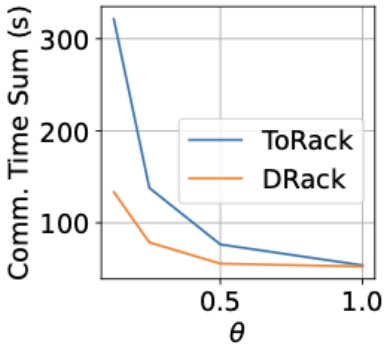
- To illustrate the minimal performance gain achieved by DRack

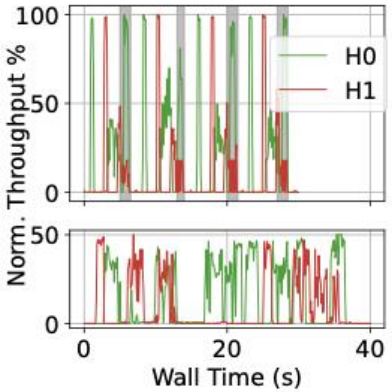
NIC's bandwidth

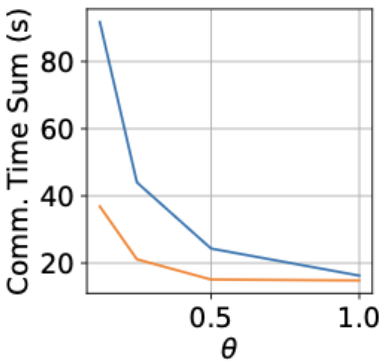
- $0 < bandwidth < maximum\ packets\ processing\ throughput\ of\ CPUs\ (C)$
- $0 < \theta = \frac{bandwidth}{C} < 1$

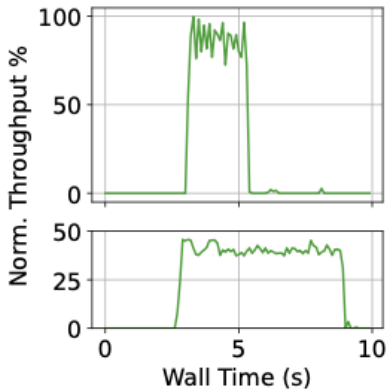
- Varying θ in experiments

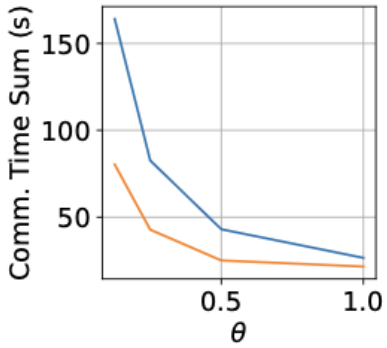
Throughput-sensitive Applications

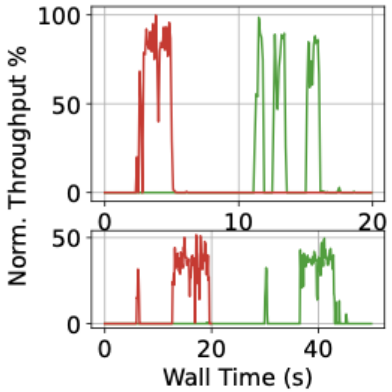












❖ Graph processing

- Setup: LiveJournal online social network; Gemini
- Cause: Hosts finish iterating and updating their subgraph asynchronously

❖ DNN Training

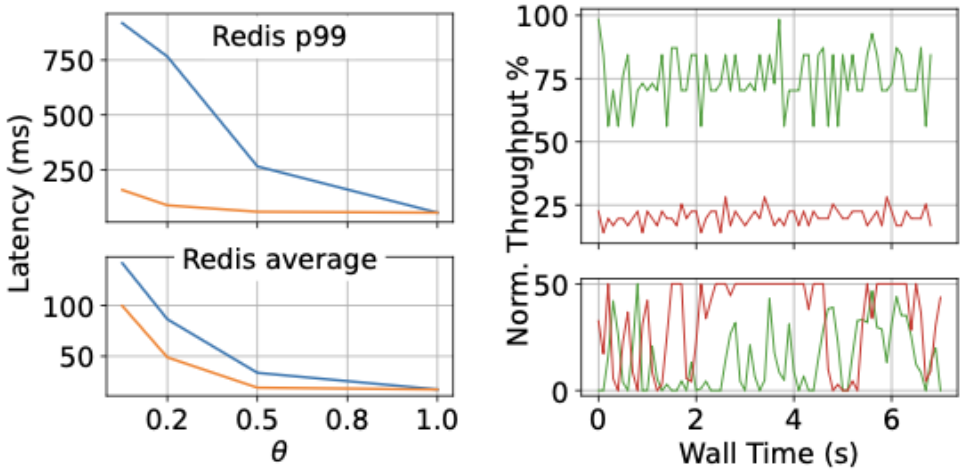
- Setup: ResNet18; all-reduce
- Cause: only one host sends/receives data across racks

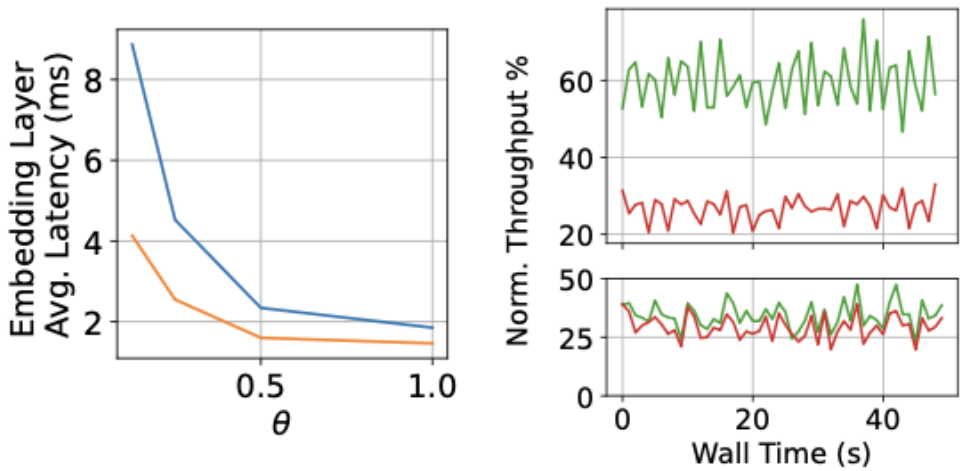
❖ LLM Training

- Setup: TinyStories-33M; parameter server
- Cause: workers and PSs send and receive data in an interleaved manner

DRack reduces the communication time by an average of 37.4%

Latency-sensitive Applications





❖ Key-Value Store

- Setup: Redis; Memtier_benchmark
- Cause: key skewness → the host with hot keys serves imbalanced requests

❖ DLRM Inference

- Setup: uniformly distributed embeddings
- Cause: frequent access to hot embeddings

DRack reduces the latency by an average of 33.3%

Other Results

1. **performance gain if existing job schedulers use Drack**
2. **Performance breakdown**
    1. NIC pool, memory pool, DRAM cache, and MPTCP
3. **Intra-rack communication latency**
4. **Inter-rack communication throughput**

… and more!

Conclusion

- Existing ToR-based racks failed to provide high inter-rack network bandwidth and NICs' utilization rate is low.
- DRack proposes **NIC pool** and **memory pool** to provide aggregated inter-rack bandwidth for hosts.
- We built a quad-rack **8-MPSoC-FPGA** based prototype and validated its performance with both microbenchmarks and real-world applications.

DRack: A CXL-Disaggregated Rack Architecture to Boost Inter-Rack Communication

Thanks & Questions?

Contact: ① zhangxu19s@ict.ac.cn
② liuke@ict.ac.cn