An Extensible Software Transport Layer for GPU Networking

Yang Zhou

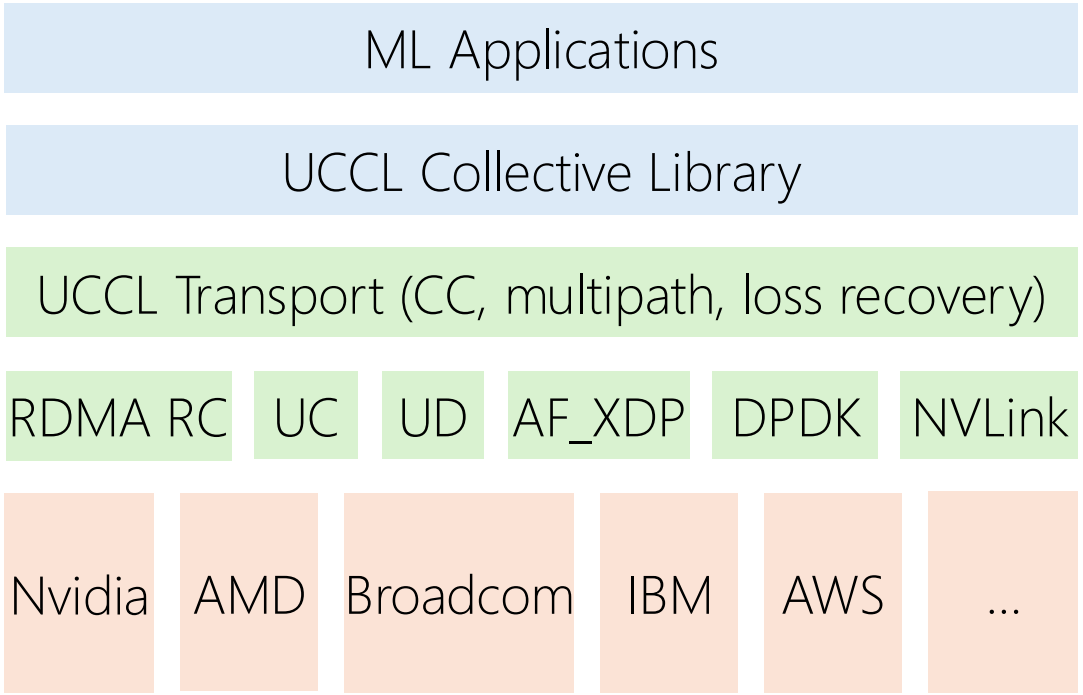
with: Zhongjie Chen, Ziming Mao, ChonLam Lao, Shuo Yang, Pravein Govindan Kannan, Jiaqi Gao, Yilong Zhao, Yongji Wu, Kaichao You, Fengyuan Ren, Zhiying Xu, Costin Raiciu, Ion Stoica

tinyurl.com/uccl-paper ⋄ github.com/uccl-project/uccl

May 29, 2025
Sky Summer Retreat

About UCCL Project

Building the fastest collective communication library (CCL)



Rearchitecting CCL layer (keep NCCL API)

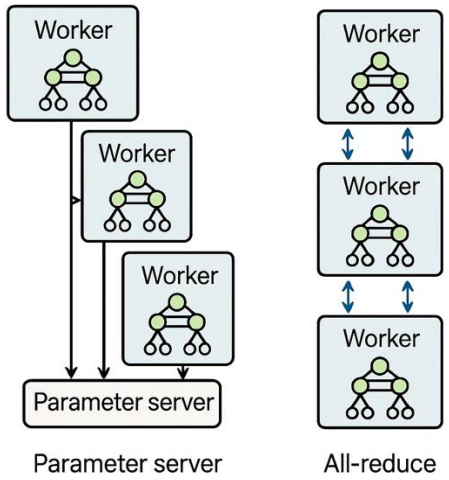
Rearchitecting transport layer (this talk)

- Up to 3.3× higher tput over NCCL on AWS

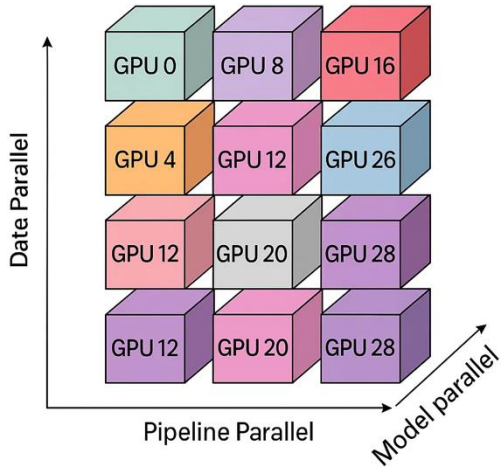
Supporting heterogeneous vendors

Open and collaborative platform: github.com/uccl-project/uccl

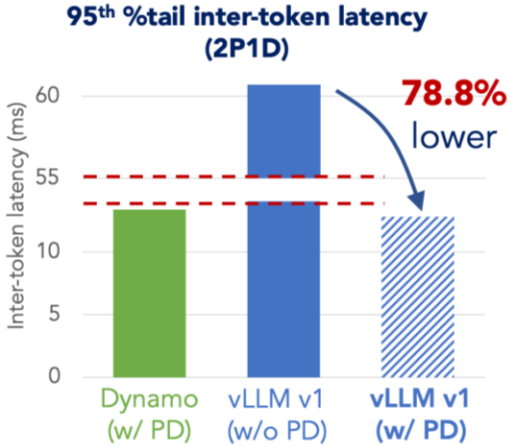
Fast Evolving ML Workloads



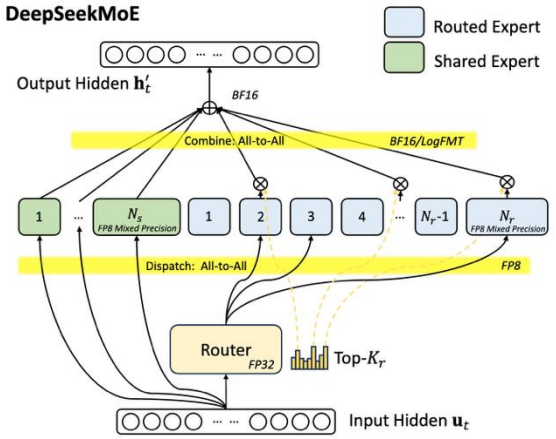
DNN training: parameter servers, allreduce



LLM training: allreduce, allgather, and reduce-scatter



PD disaggregation: P2P transfer



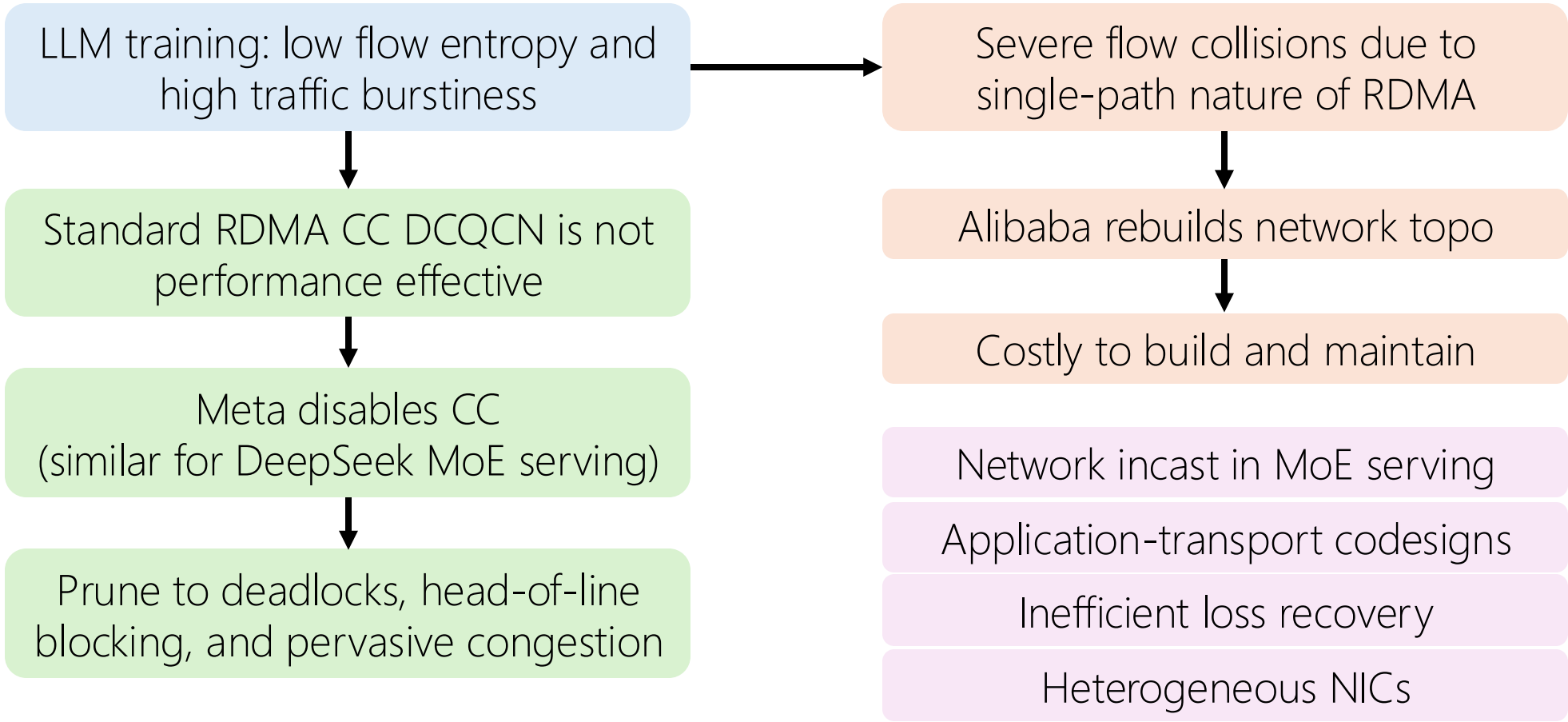
DeepSeek-V3 MoE: all-to-all like

~2015 ~2020 2024 2025 Time

Slowly Evolving Networking

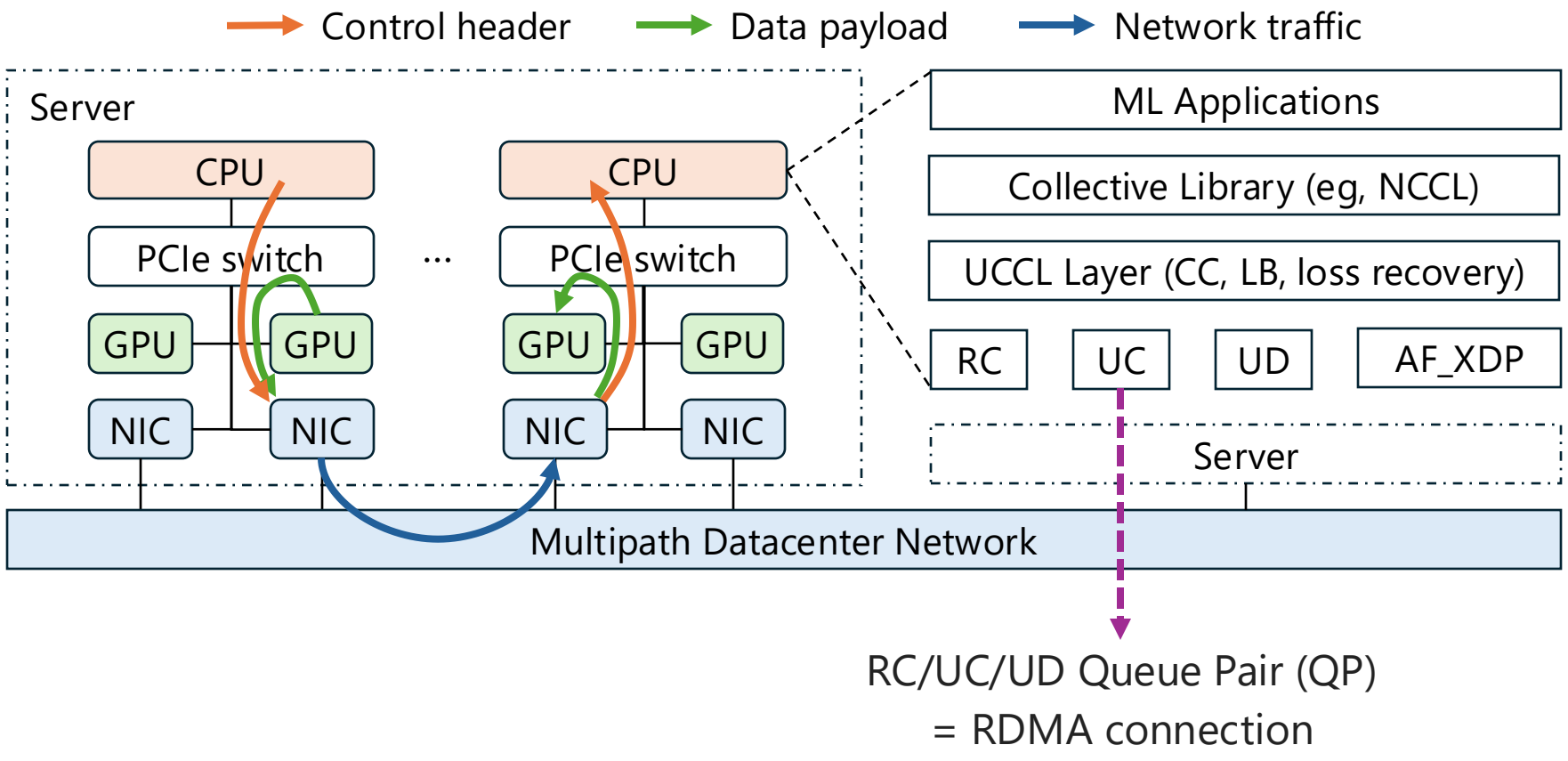
Host transport on RDMA NICs is hard to adapt to better suit ML workloads

- Hardware changes are time-consuming



Overarching Problem: Network Extensibility

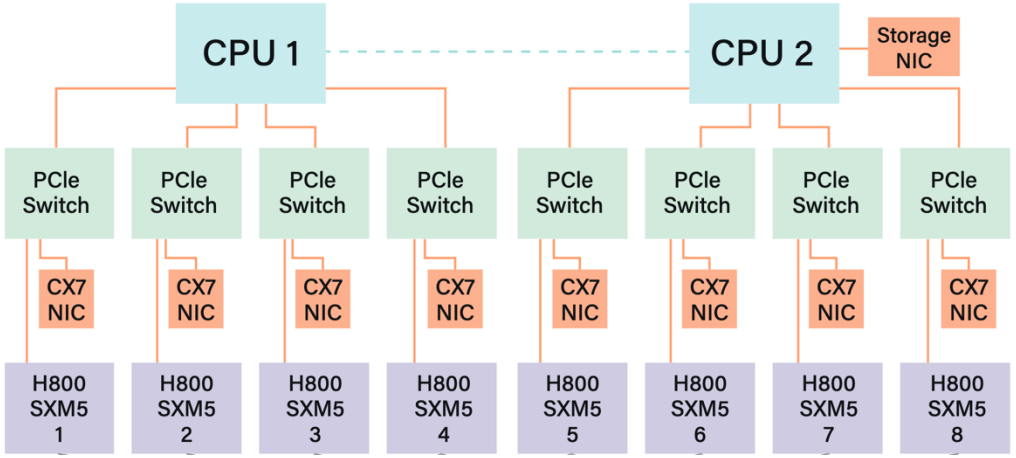
UCCL approach: a software-only extensible transport for GPU networking



UCCL Key Challenges

- How to **decouple** the data and control paths for existing RDMA NICs?
  - Eg, Nvidia NICs, Broadcom NICs, AWS EFA NICs

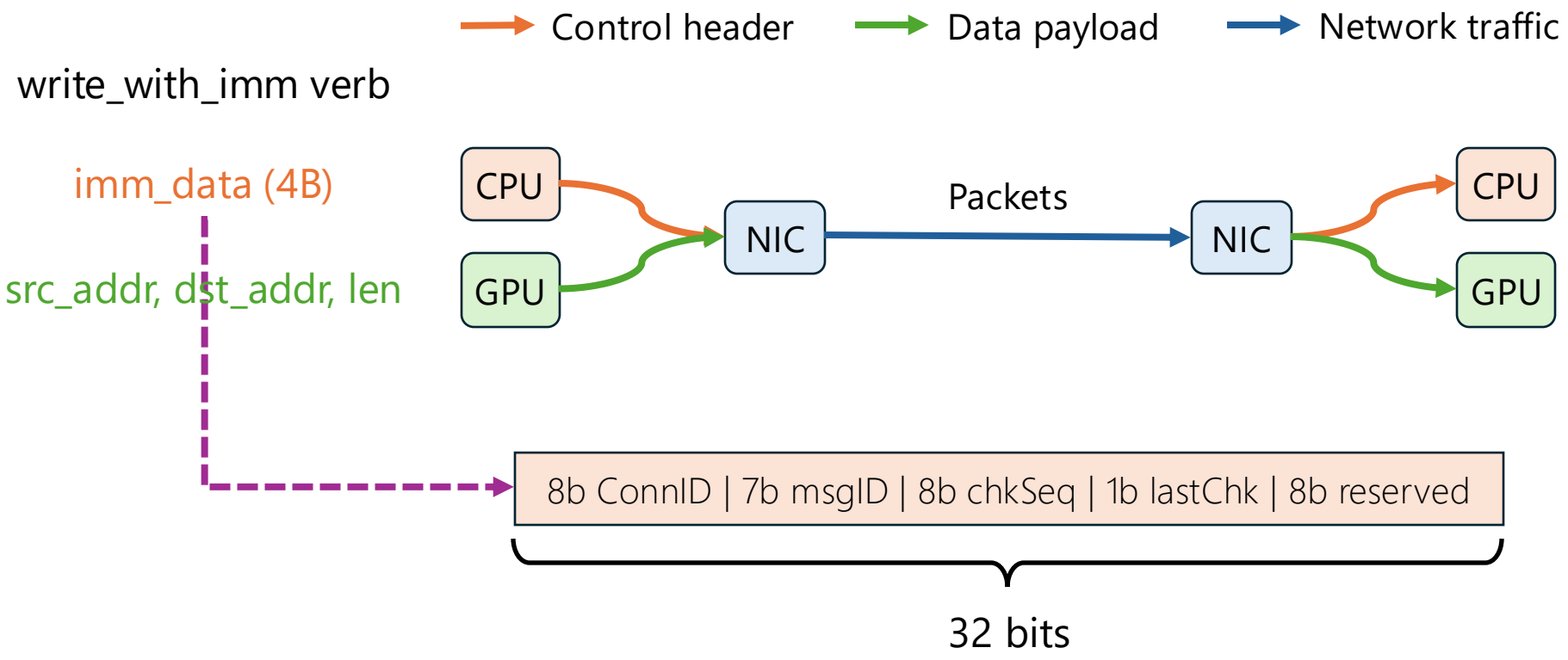
- How to achieve hardware-level **performance** for software control path?
  - Eg, 3.2 Tbps inter-server bandwidth



Technique #1: Decoupling Control & Data Path

Leveraging UC/RC QPs + RDMA write with immediate

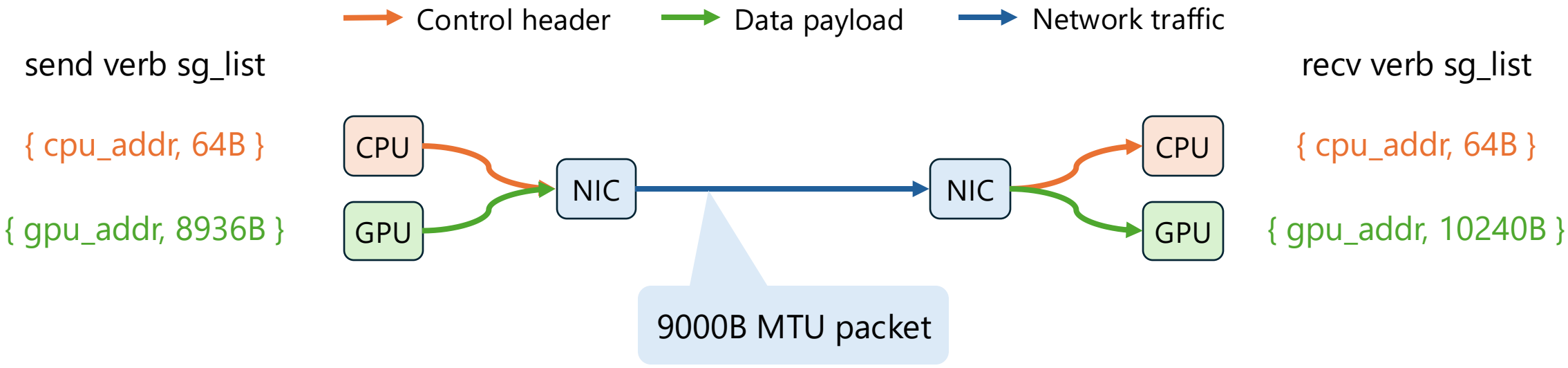
- Eg, for Nvidia and Broadcom NICs (that support UC or allow disabling RC's CC logic)



Technique #1: Decoupling Control & Data Path

Leveraging UD QPs + send/recv with scatter-gather list

- Eg, for AWS EFA NICs (that cannot disable RC's CC logic)



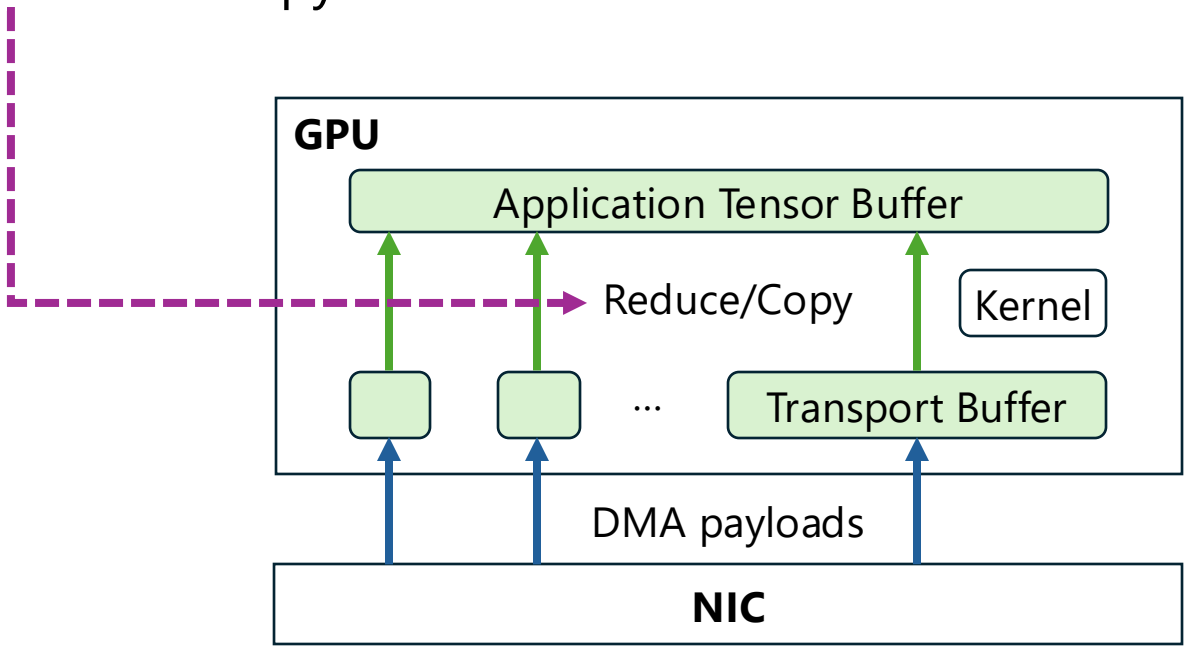
Technique #1: Decoupling Control & Data Path

Leveraging UD QPs + send/recv with scatter-gather list

- Eg, for AWS EFA NICs (that cannot disable RC's CC logic)

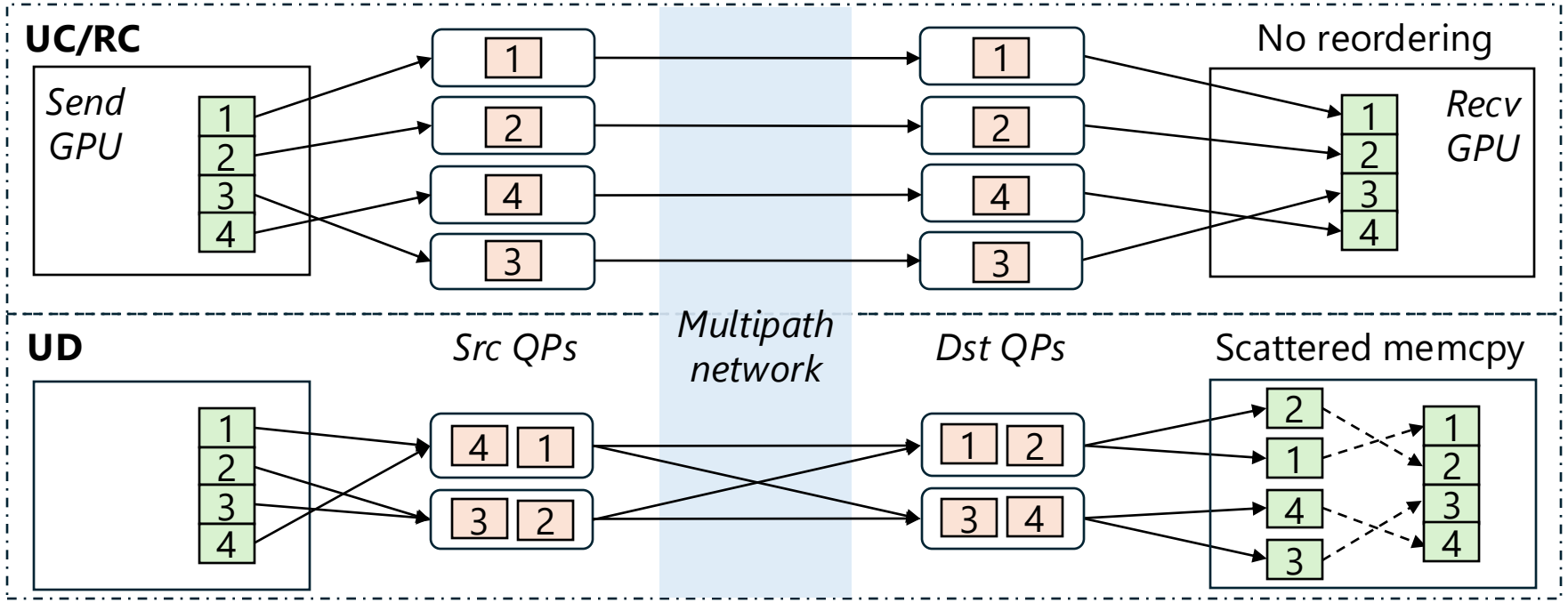
Handling out-of-order packet delivery

- Fusing scattered memcpy at the receiver GPU

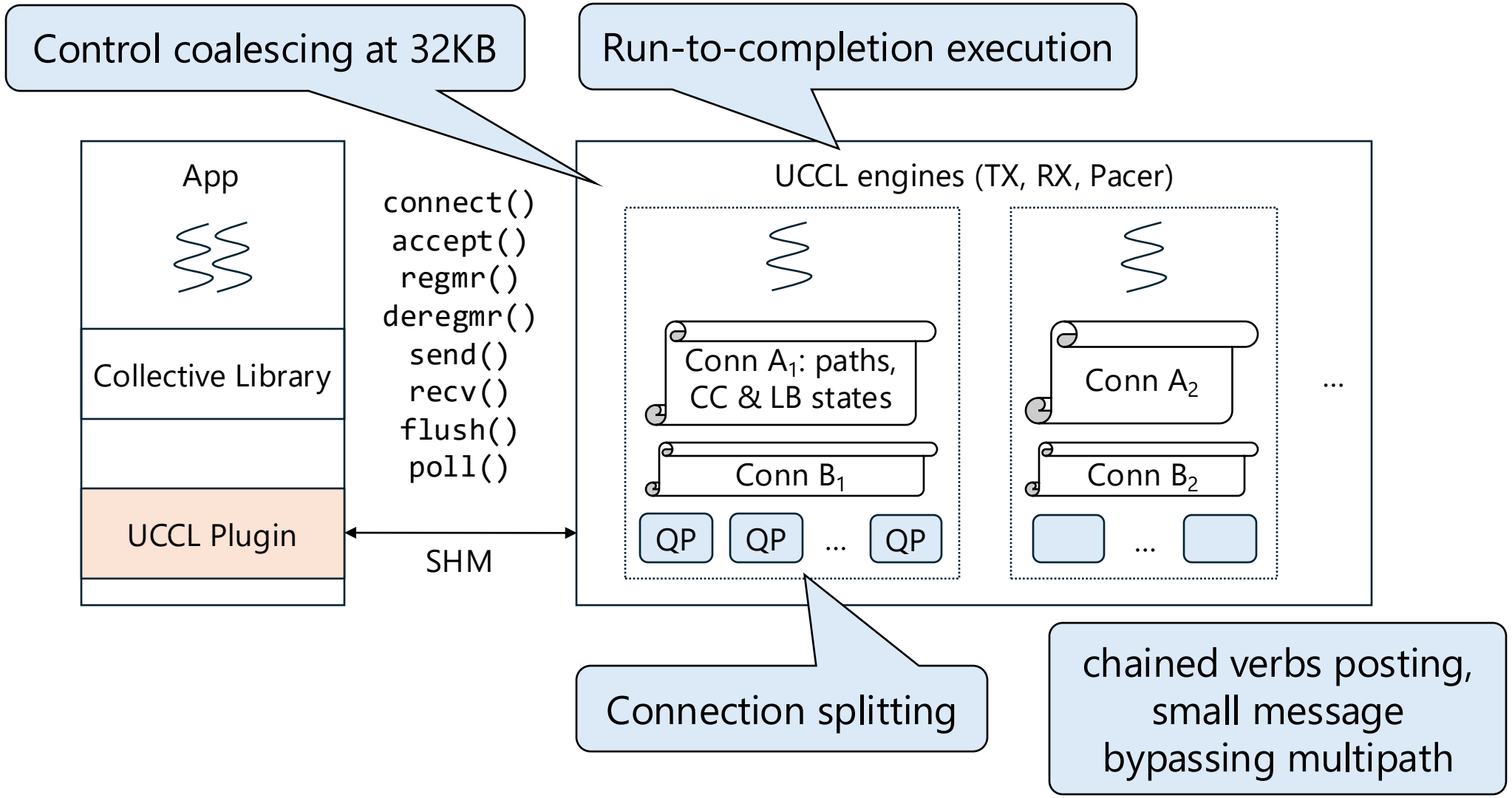


Technique #1: Decoupling Control & Data Path

Multipathing with packet spraying



Technique #2: Efficient Software Transport

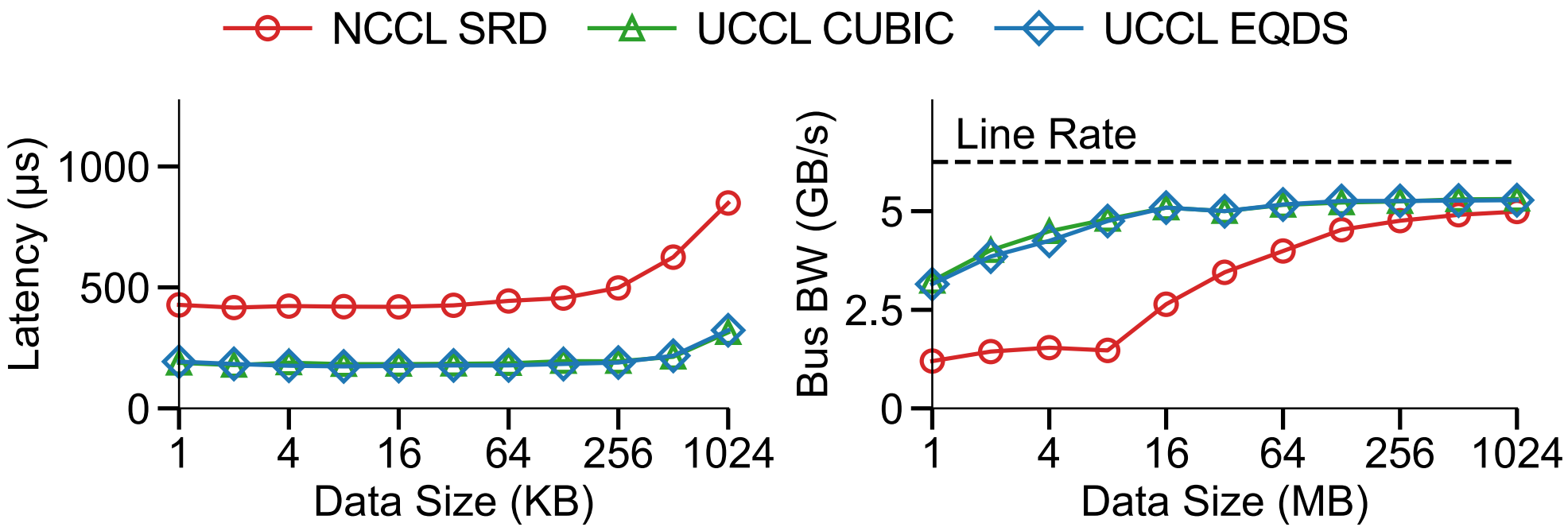


Implementation & Feature Support

- 27k LoC in C++
  - Drop-in replacement for NCCL applications
  - Packet spraying with 256 paths
  - Latency-based CC, receiver-driven CC
  - Efficient loss recovery by selective repeat
- Support both Nvidia and AMD GPUs
  - Future: AWS Trainium
- Support a variety of NIC vendors:
  - RDMA: Nvidia, Broadcom, AWS EFA
  - Non-RDMA: Nvidia, AWS ENA, IBM VirtIO

Evaluation: 4 AWS p4d (all-to-all)

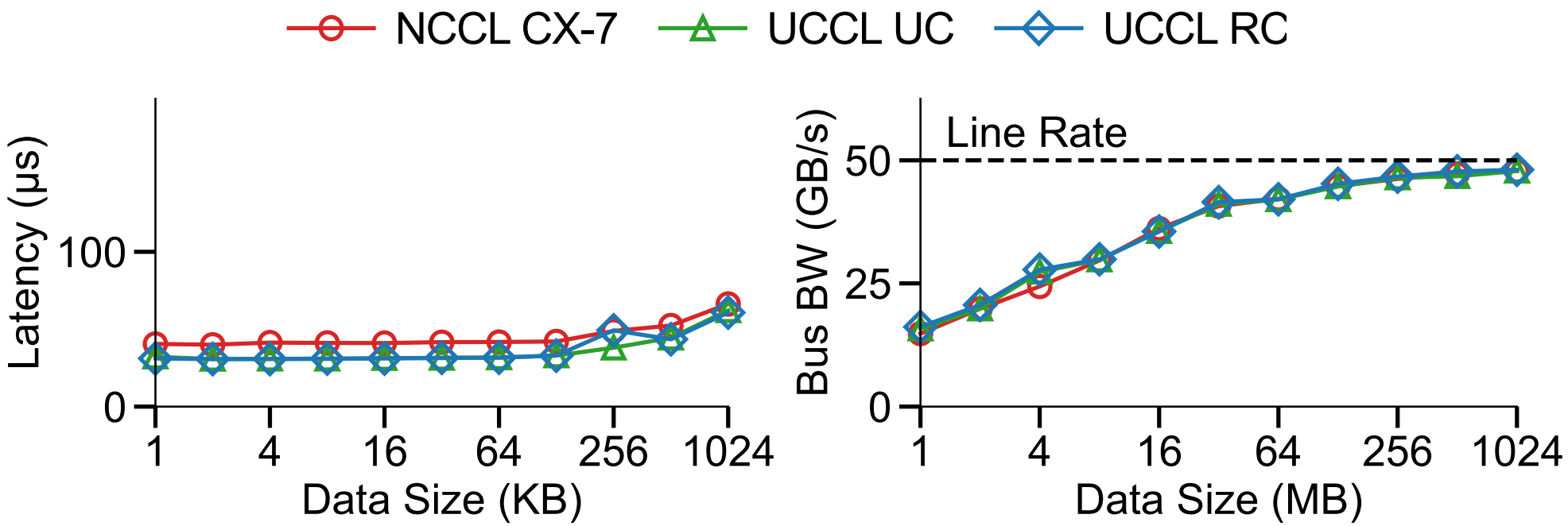
- 4×100G EFA NICs per node, Fattree over Ethernet
  - NVLink disabled to emulate larger testbed



UCCL achieves up to 3.2× higher performance over NCCL on AWS

Evaluation: 2 HGX (all-to-all)

- 8×400G Nvidia CX-7 NICs per node, same rack over InfiniBand
  - NVLink disabled to emulate larger testbed



UCCL matches NCCL performances on ASIC-based NICs

Dev Plan

- Dynamic membership with GPU servers joining and exiting
- GPU-initiated P2P communication (eg, IBGDA)
  - For MoE all-to-all and PD disaggregation
  - Generic to NIC vendors like AWS EFA and Broadcom, and GPU vendors like AMD
- Rearchitecting NCCL to unleash network hardware capability
  - Scalable and efficient CPU proxy
  - Low-cost async collectives with compute-communication ordering guarantee
  - Device kernels in vendor-agnostic Triton language

- We would like to hear about your feature needs!

Conclusion

UCCL: building the fastest collective communication library

- Network transport layer, CCL layer, heterogeneous vendors, and more
- Open and collaborative platform---talk with us in the poster session

tinyurl.com/uccl-paper

github.com/uccl-project/uccl

Thank you!

yangzhou.rpc@gmail.com/berkeley.edu