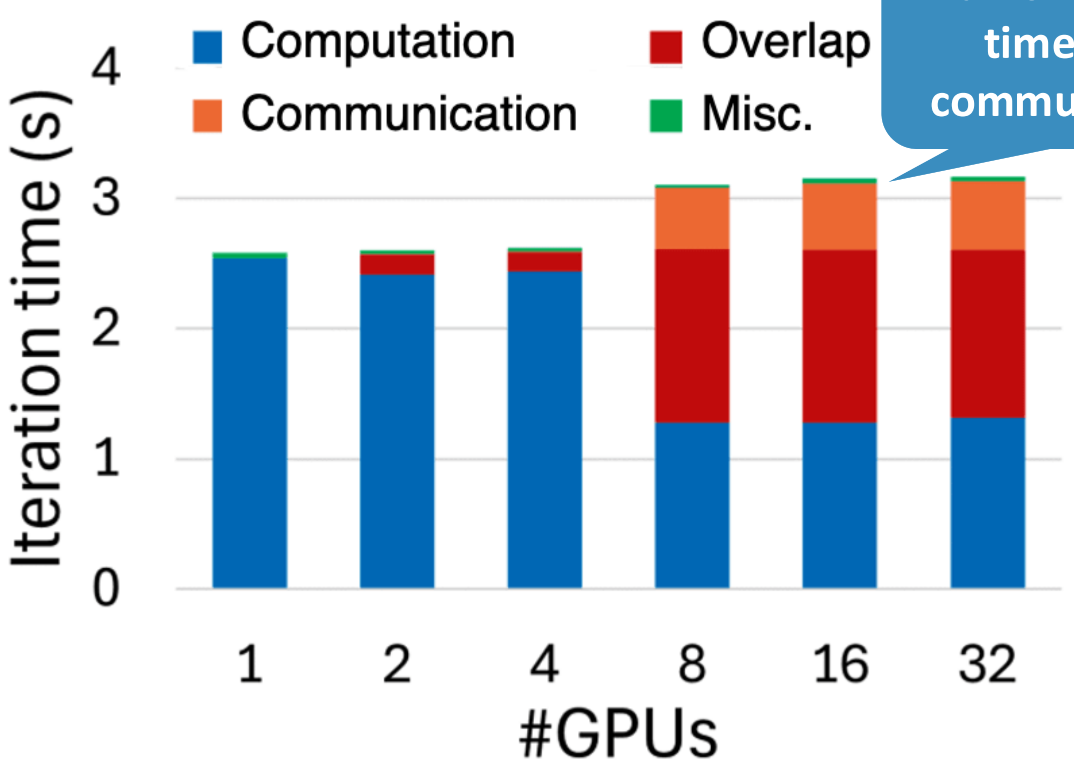
Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms

Zhiyi Hu[1], Siyuan Shen[1], Tommaso Bonato[1], Sylvain Jeaugey[2], Cedell Alexander[3], Eric Spada[3] , James Dinan[2], Jeff Hammond[2], Torsten Hoefler[1]

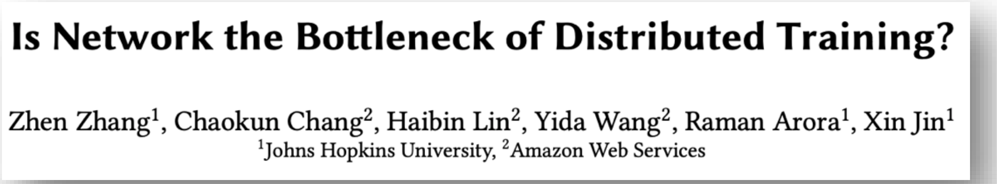
[1] ETH Zurich, [2] NVIDIA Corporation, [3] Broadcom Inc.

Motivation: GPU Communication Bottleneck



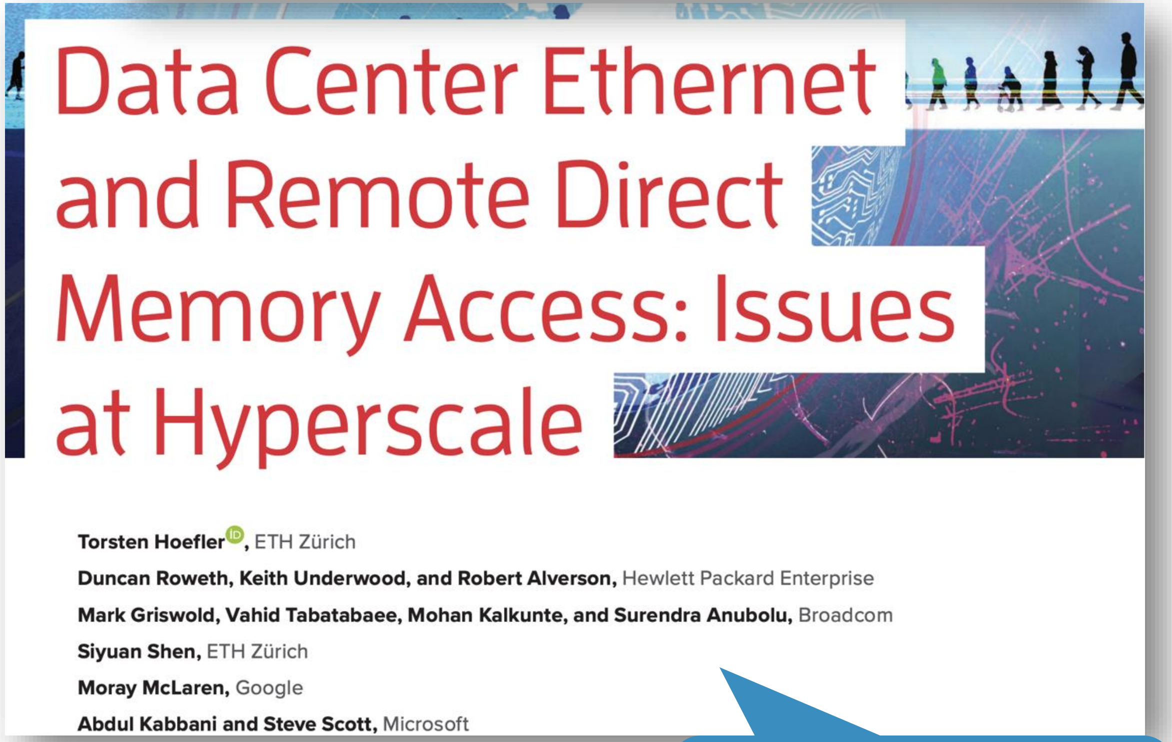
Half of iteration time is for communication

Iteration time breakdown of the full fine-tuning method on a 1B parameters Transformer decoder model [1]



Is Network the Bottleneck of Distributed Training?

Zhen Zhang[1], Chaokun Chang[2], Haibin Lin[2], Yida Wang[2], Raman Arora[1], Xin Jin[1]
[1]Johns Hopkins University, [2]Amazon Web Services



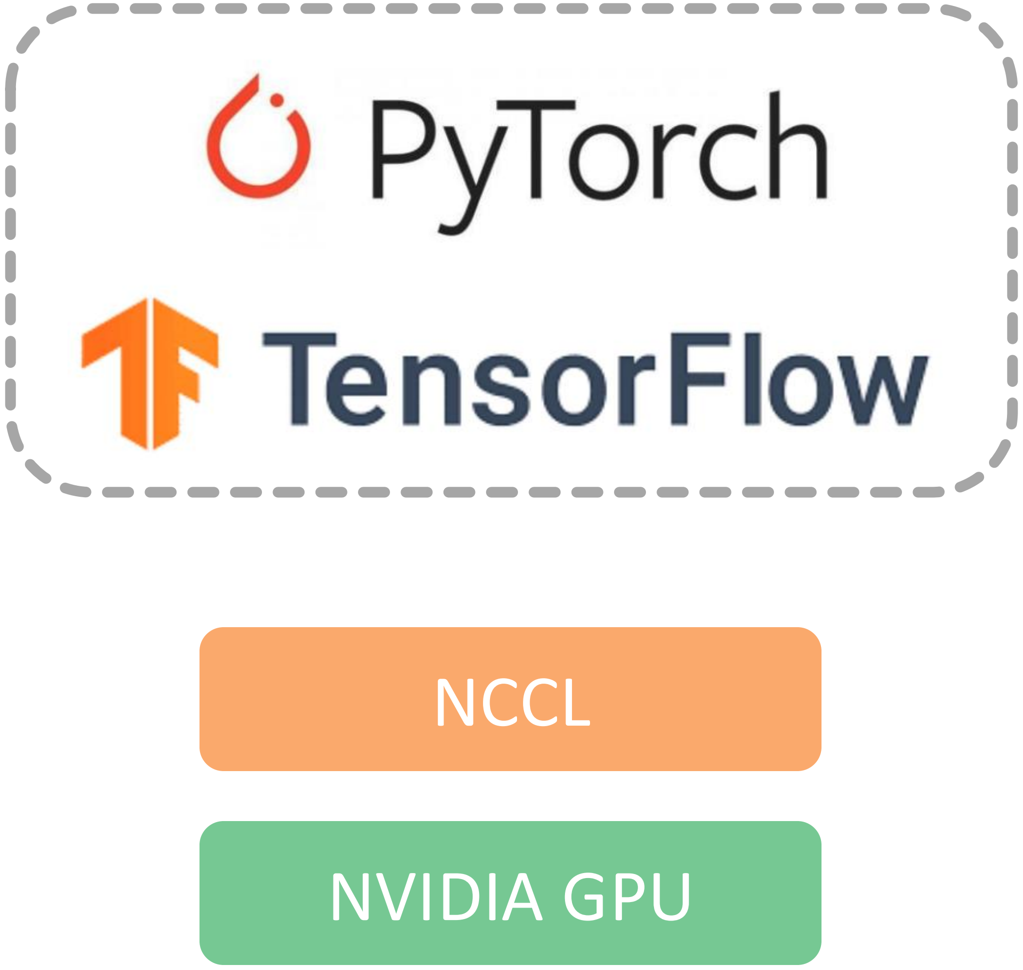
Data Center Ethernet and Remote Direct Memory Access: Issues at Hyperscale

Torsten Hoefler, ETH Zürich
Duncan Roweth, Keith Underwood, and Robert Alverson, Hewlett Packard Enterprise
Mark Griswold, Vahid Tabatabaee, Mohan Kalkunte, and Surendra Anubolu, Broadcom
Siyuan Shen, ETH Zürich
Moray McLaren, Google
Abdul Kabbani and Steve Scott, Microsoft

Network bandwidth is not efficiently utilized

[1] N. Alnaasan et al., "Characterizing Communication in Distributed Parameter-Efficient Fine-Tuning for Large Language Models," HOTI 2024, pp. 11–19.

Motivation: NCCL is de-facto standard



Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism

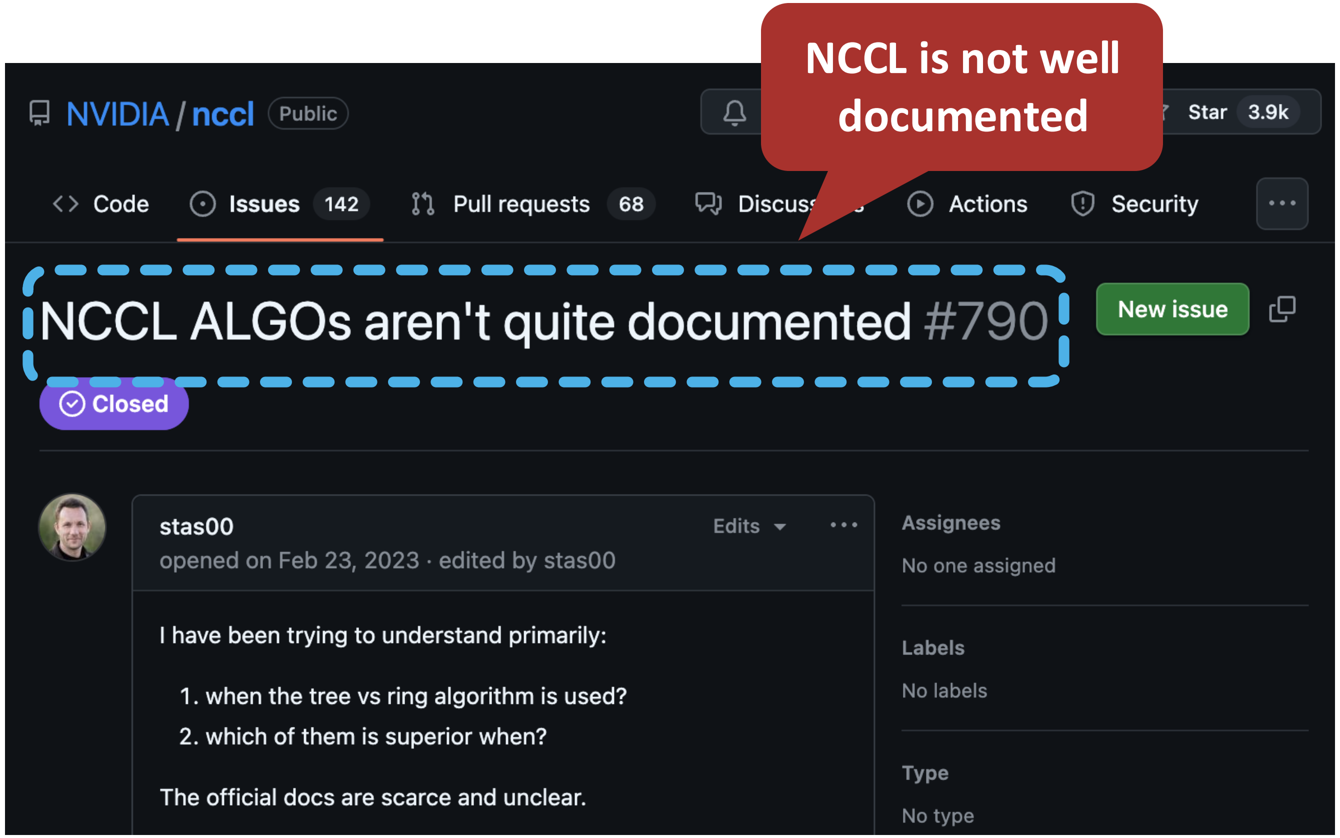
Mohammad Shoeybi[1 2] Mostofa Patwary[1 2] Raul Puri[1 2] Patrick LeGresley[2] Jared Casper[2] Bryan Catanzaro[2]

SGLang: Efficient Execution of Structured Language Model Programs

Lianmin Zheng[2*] Liangsheng Yin[3] Zhiqiang Xie[1] Chuyue Sun[1] Jeff Huang[4] Cody Hao Yu[5] Shiyi Cao[2] Christos Kozyrakis[1] Ion Stoica[2] Joseph E. Gonzalez[2] Clark Barrett[1] Ying Sheng[1*]

[1] Stanford University [2] UC Berkeley [3] Shanghai Jiao Tong University [4] Texas A&M University [5] Independent Researcher

Motivation: NCCL not well documented && Performance Tuning

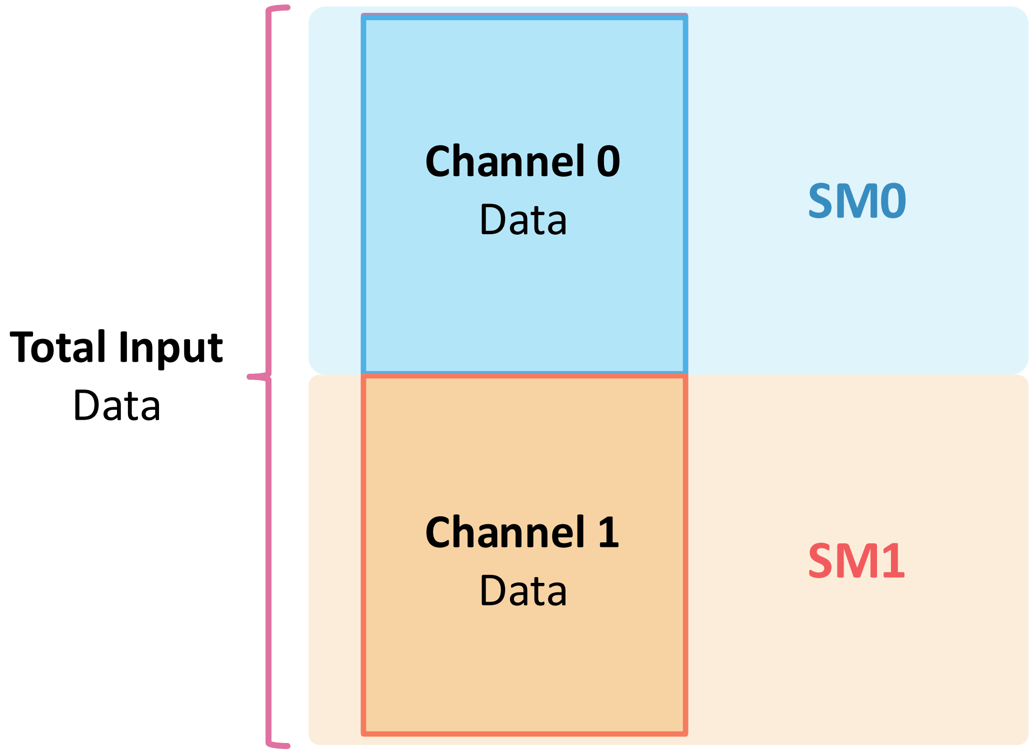


Key Parameters	*Choices*
Protocol	**Simple, LL, LL128**
Transport	**Socket, IB, GDR**
Algorithm	**Ring, Tree, NVLS, Collnet, PAT, …**

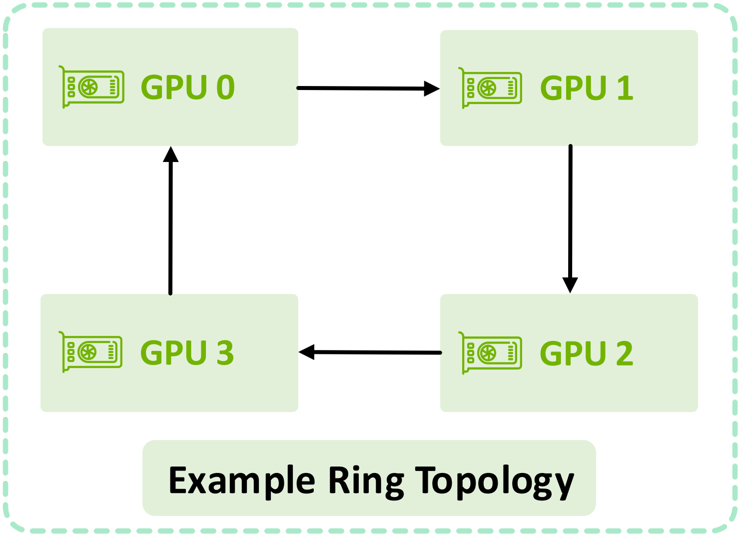
NCCL Overview – API and Execution Flow

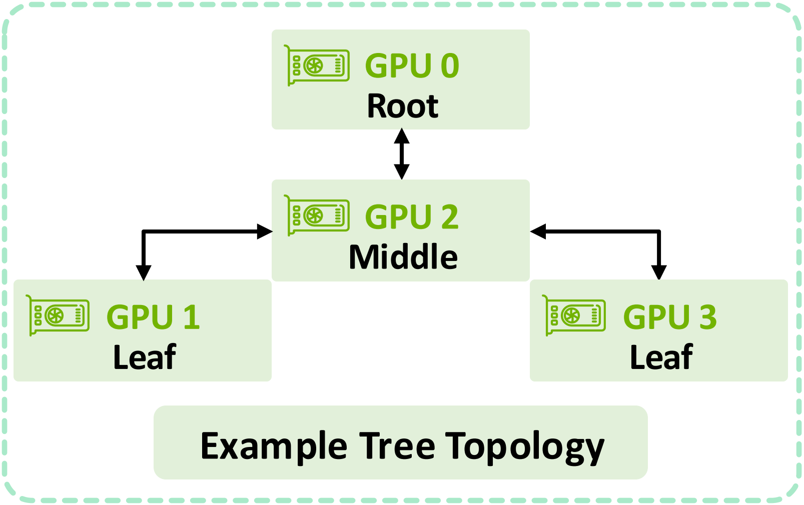

```
// Create communicator
ncclComm_t comm;
ncclCommInitRank(&comm, nranks, id, rank);

// Start group operation
ncclGroupStart();

// Collective communication call
ncclAllReduce(sendbuff, recvbuff, count, ncclFloat, ncclSum, comm, stream);

// Point-to-point communication call
ncclSend(sendbuff, count, ncclFloat, next_rank, comm, stream);
ncclRecv(recvbuff, count, ncclFloat, prev_rank, comm, stream);

// End group operation
ncclGroupEnd();

// Destroy communicator
ncclCommDestroy(comm);
```

NCCL Overview – Communication Channels & Logical Topology



Example channels with mapped streaming multiprocessors (SMs) and disjoint data





Communication Protocols

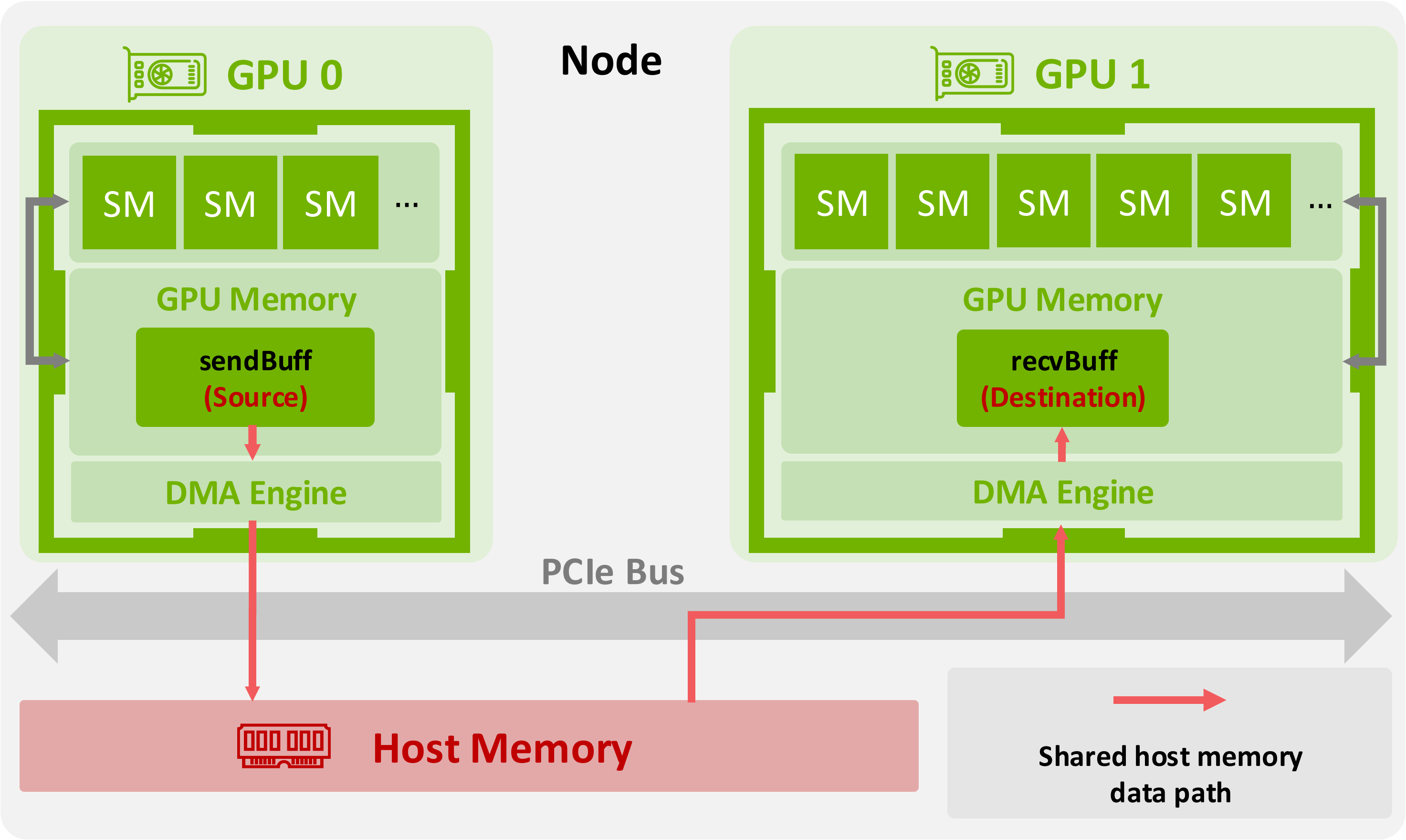
Require NVLink

	Simple	Low Latency (LL)	LL128
Design Goal	High bandwidth	Low latency	Low latency and high bandwidth
Synchronization Mechanism	Memory fence (high overhead)	Flag-based synchronization	Flag-based synchronization
Payload	Data chunks	4B data + 4B flag	120B data + 8B flag
Bandwidth Utilization	Near Peak	25 ~ 50% of peak	~95% of peak
Latency Per-hop	~ 6µs	~ 1µs	~ 2µs

Comparisons of NCCL communication protocols

Intra-node Communication

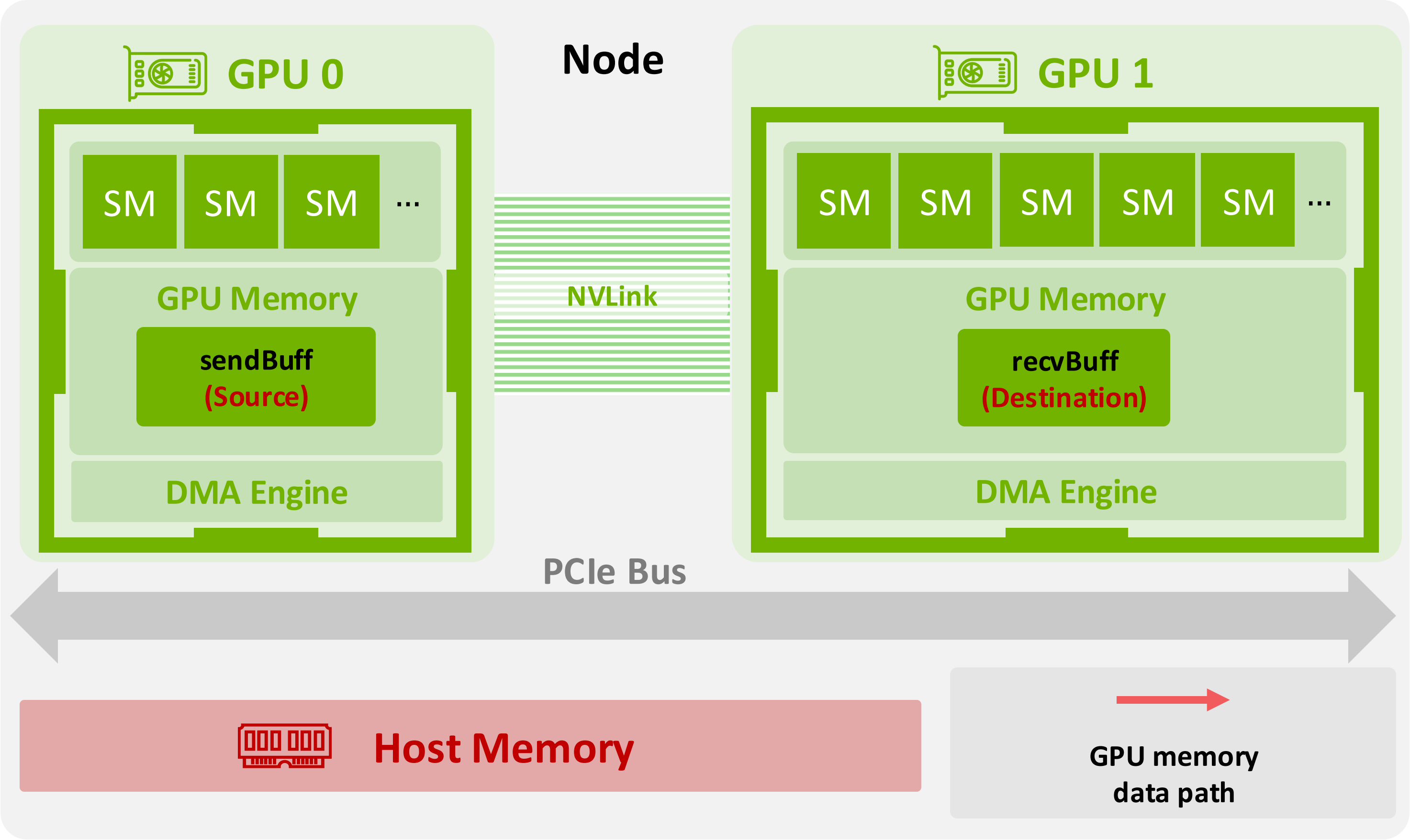
Data path: use shared host memory for data staging



Intra-node data transfer path
peer-to-peer (shared host memory)

Intra-node Communication

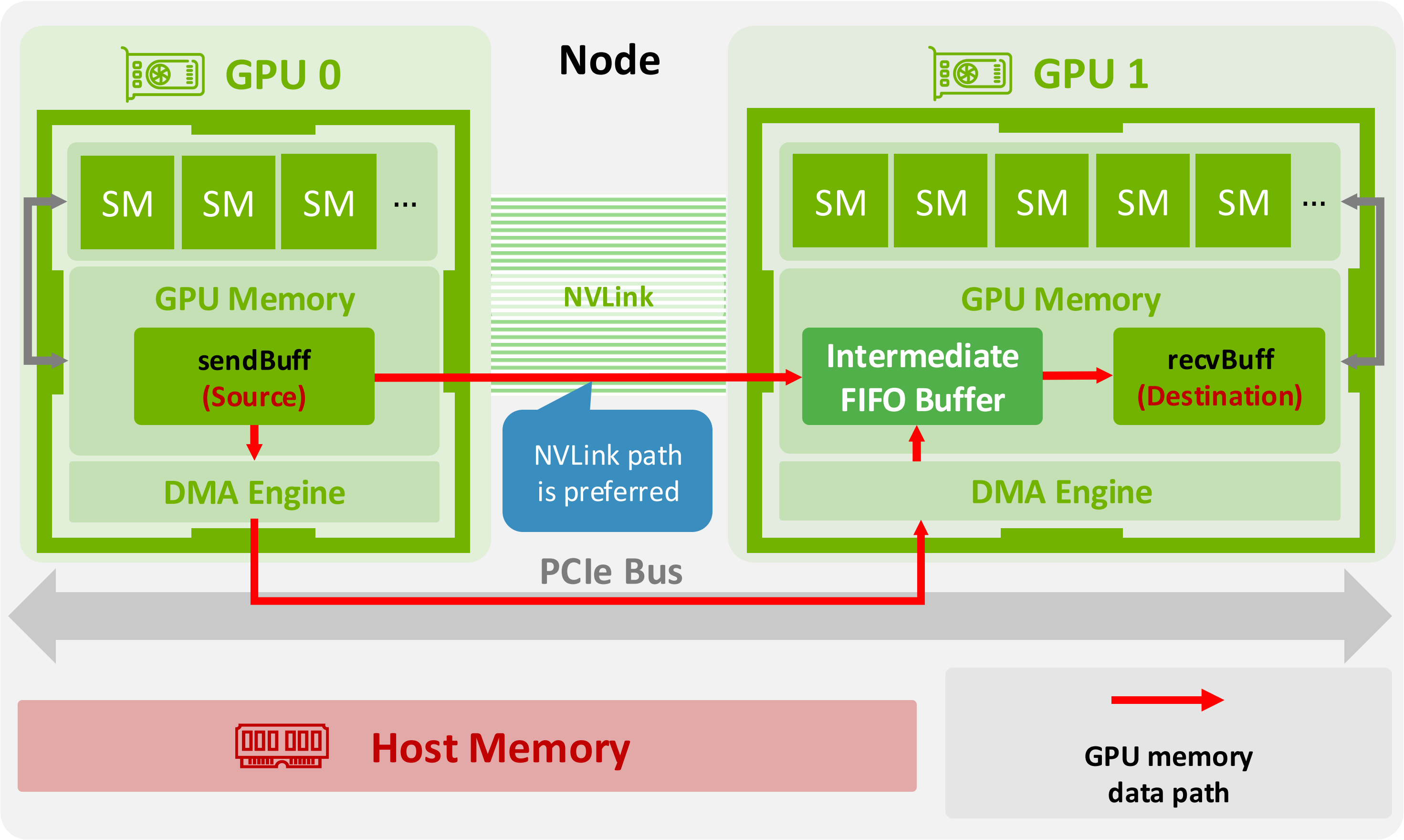
Data path: use GPU memory for data staging



Intra-node data transfer path
peer-to-peer (GPU memory)

Intra-node Communication

Data path: use GPU memory for data staging



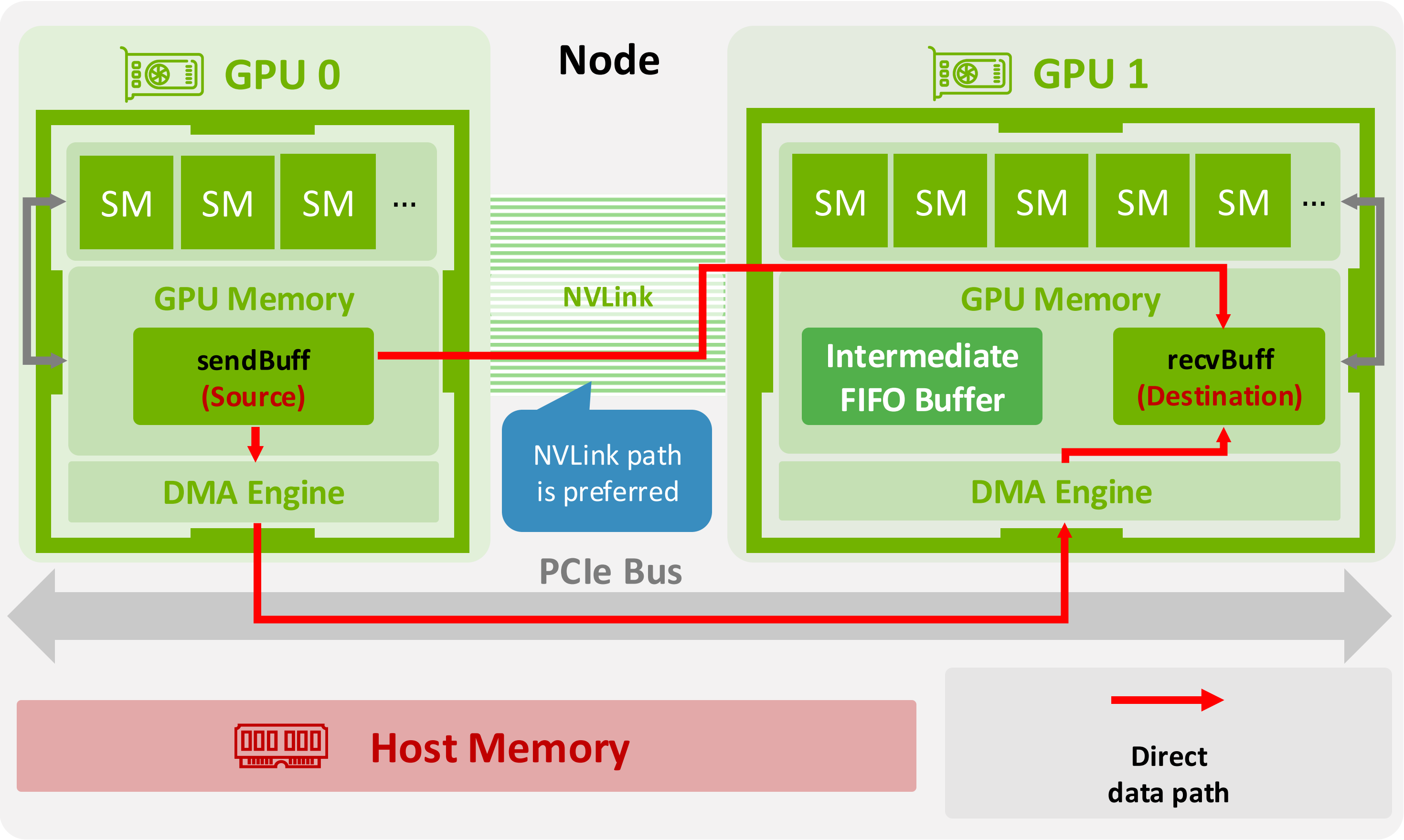
Intra-node data transfer path
peer-to-peer (GPU memory)

Intra-node Communication

Data path: direct, no intermediate buffer for data staging

P2P_DIRECT mode is enabled

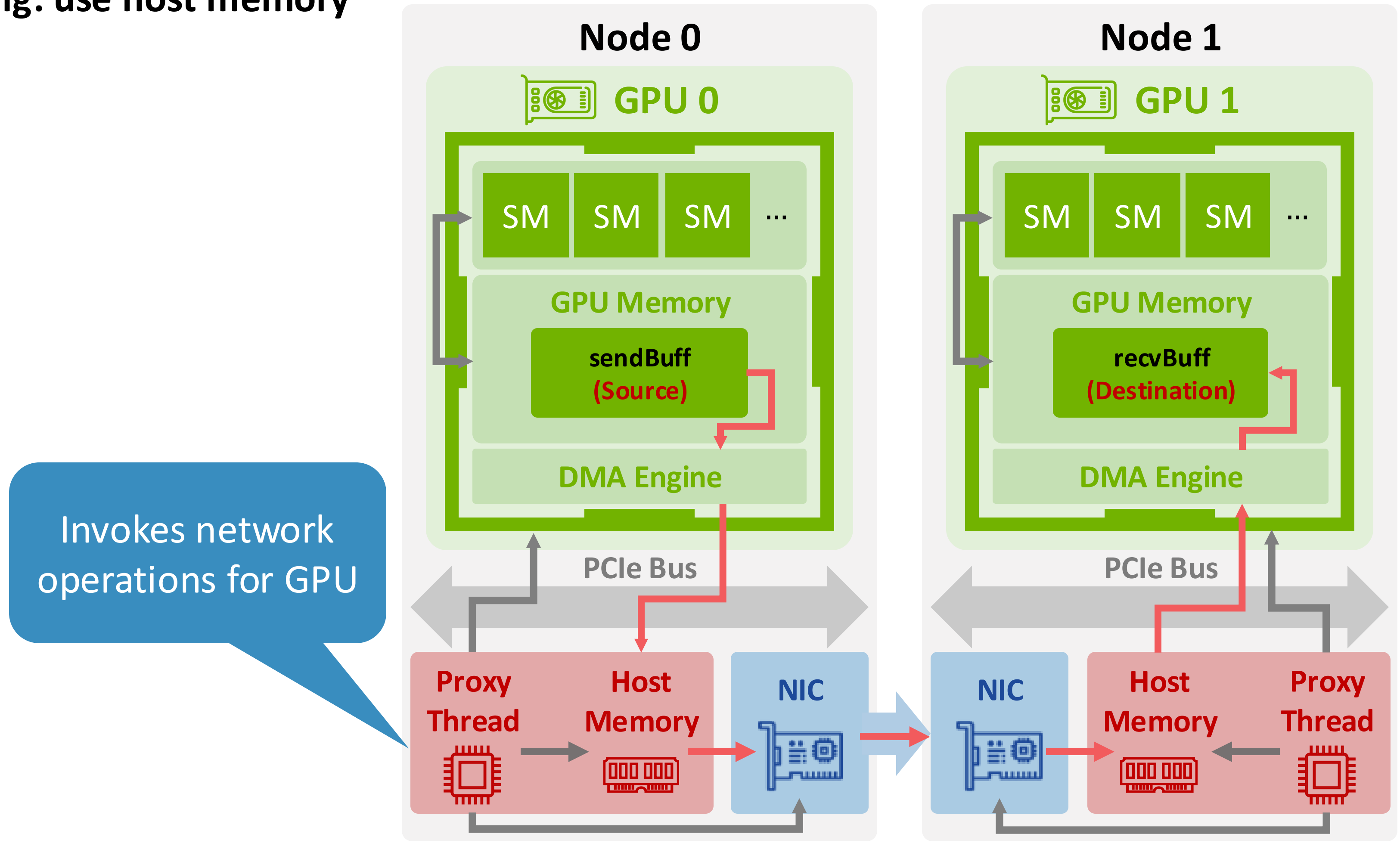
GPU 0 and GPU 1 ***must*** belong to the same process



Intra-node data transfer path peer-to-peer (Direct)

Inter-node Communication

Data staging: use host memory

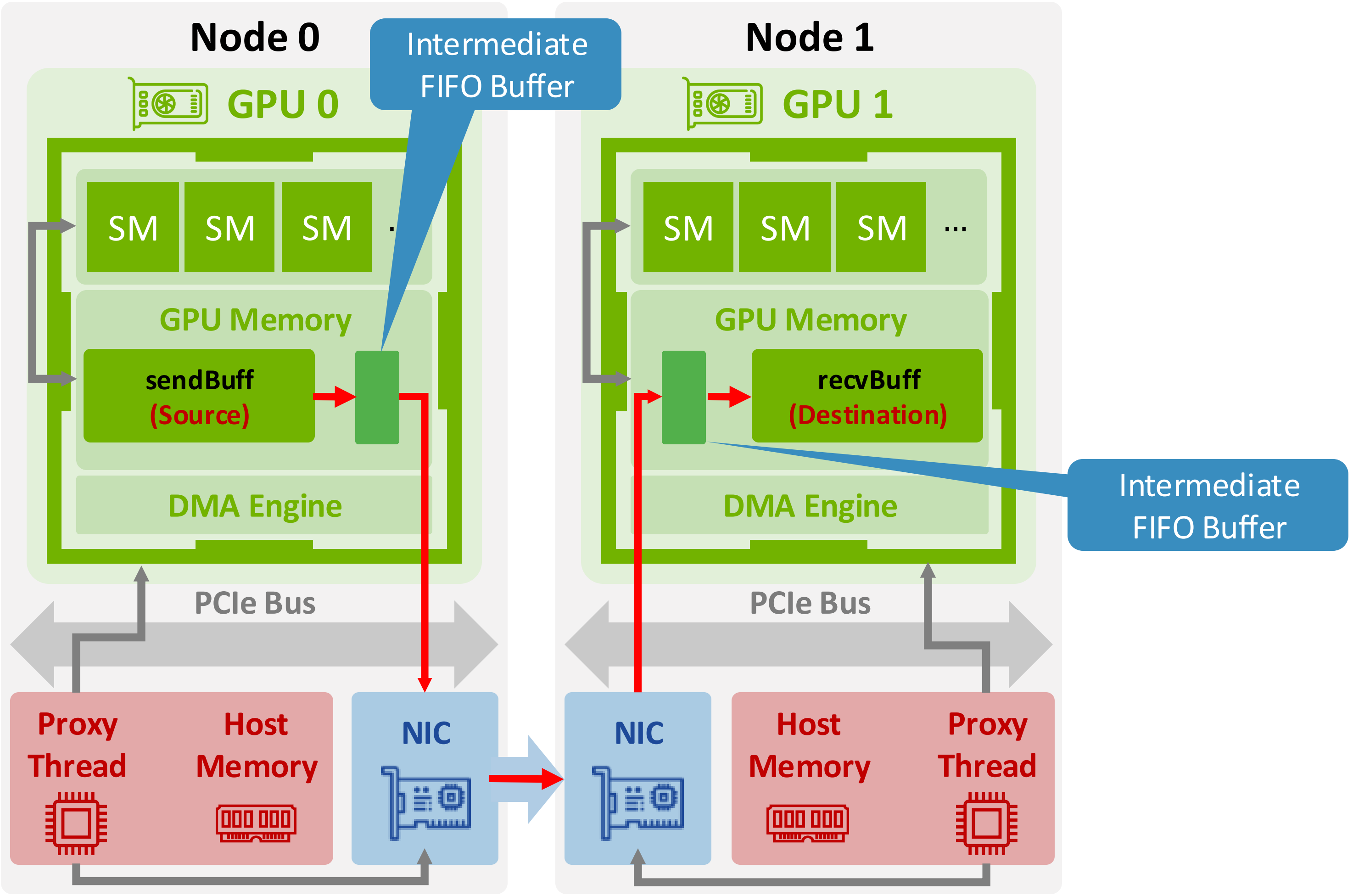


Inter-node data transfer path

TCP(Socket) and RDMA (IB, without GDRDMA optimization)

Inter-node Communication

Data staging: use GPU memory



Inter-node data transfer path
IB (with GDRDMA Optimization)

Collective Algorithms and Primitives

NCCL version 2.19.1

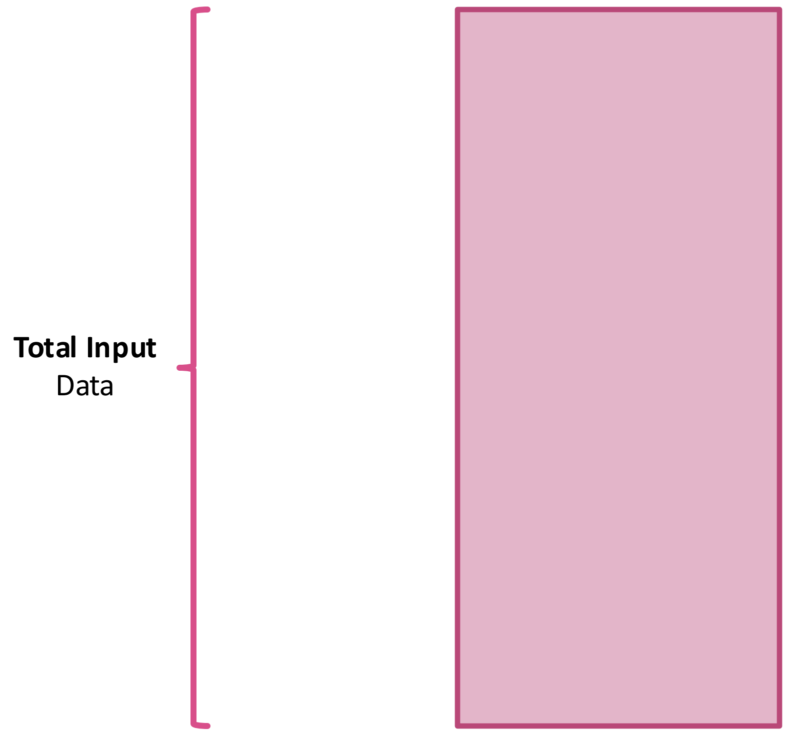
	AllReduce			Broadcast			Reduce			ReduceScatter			AllGather		
Algorithm	Simple	LL	LL128	Simple	LL	LL128	Simple	LL	LL128	Simple	LL	LL128	Simple	LL	LL128
Ring	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Tree	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
CollNet Direct	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
CollNet Chain	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
NVLS	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗
NVLS Tree	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

Legend: ✓= Supported, ✗= Not supported.

Supported algorithms and protocols for NCCL collective operations

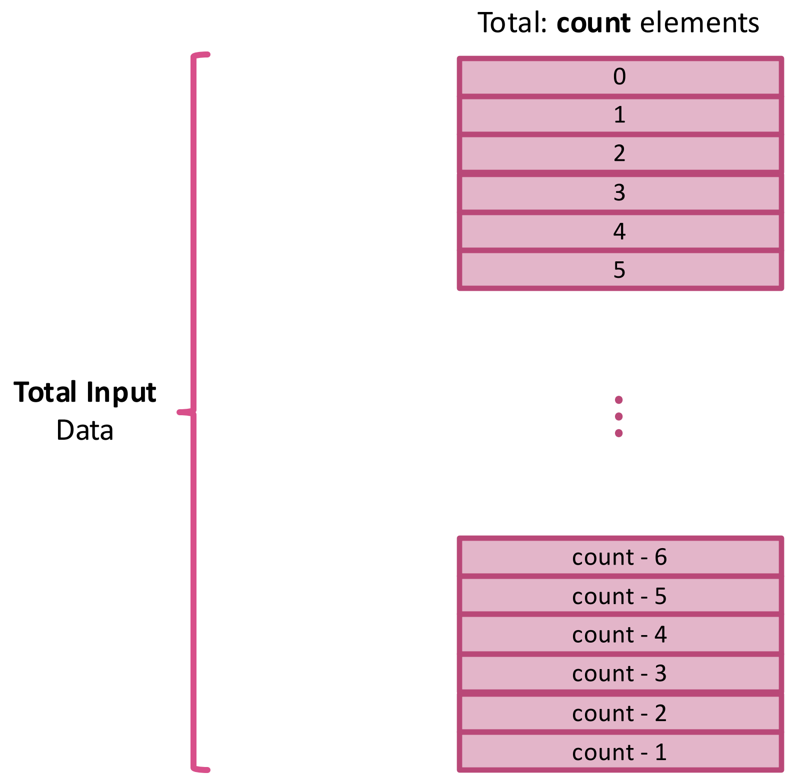
More and more algorithms (e.g., PAT) are coming out!

Iterative Execution and Communication Primitives



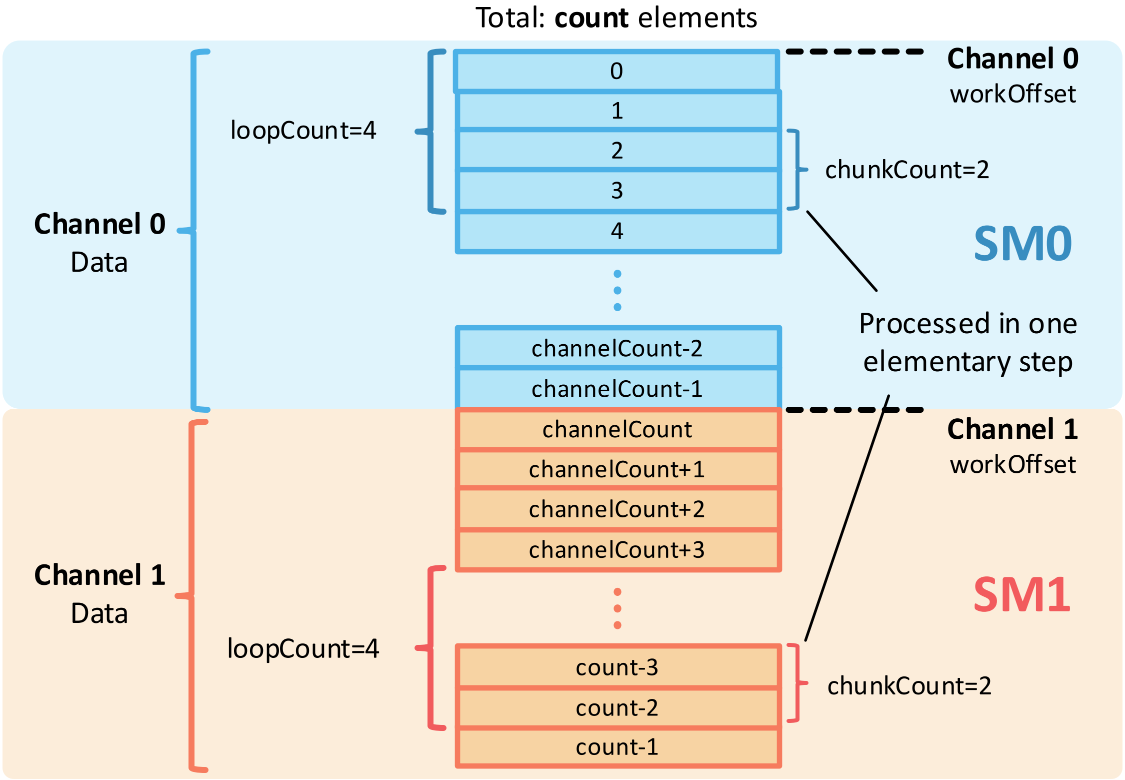
Iterative execution for data chunks

Iterative Execution and Communication Primitives



Iterative execution for data chunks

Iterative Execution and Communication Primitives



Iterative execution for data chunks

Step Index	NCCL Primitive
0	`send`
1 to $k-2$	`recvReduceSend`
$k-1$	`recvReduceCopySend`
k to $2k-3$	`recvCopySend`
$2k-2$	`recv`

Steps in one loop iteration of NCCL Ring AllReduce

NCCL Collective Algorithm Example: Ring Allreduce

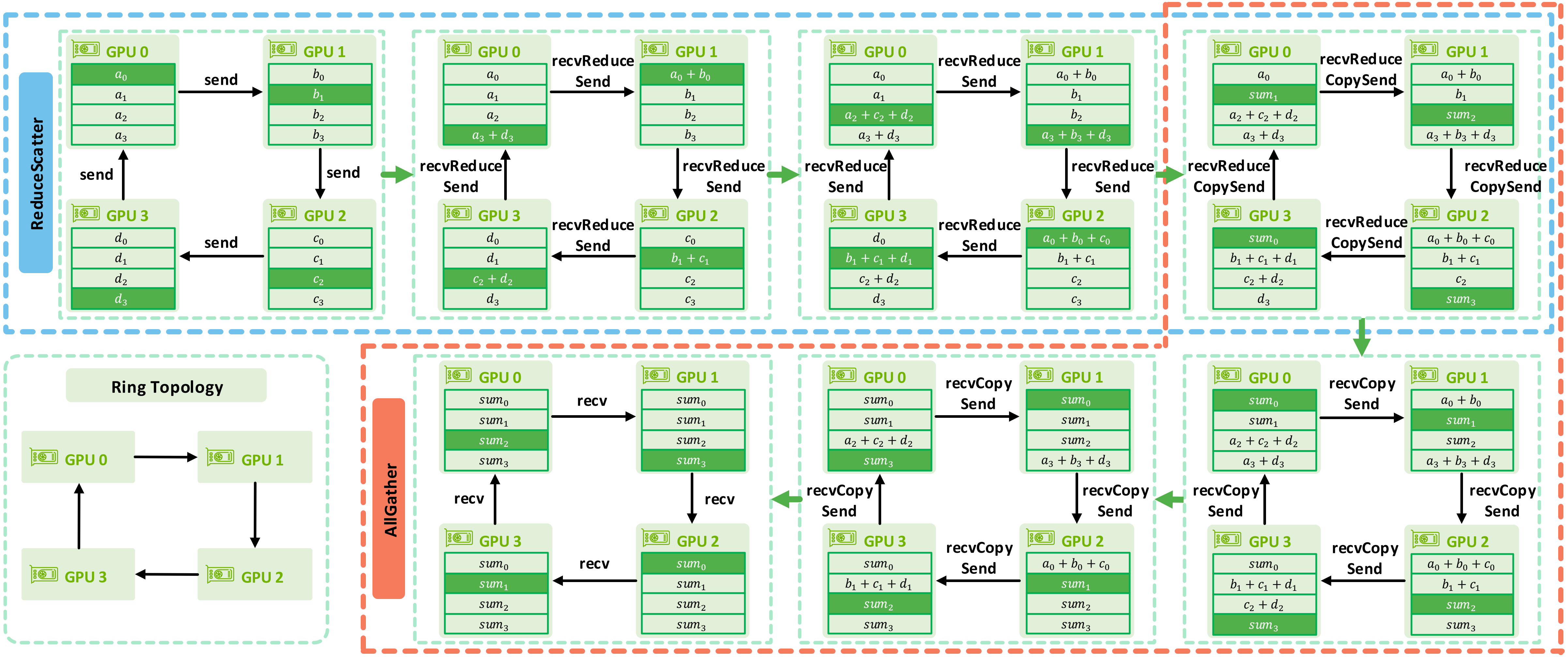
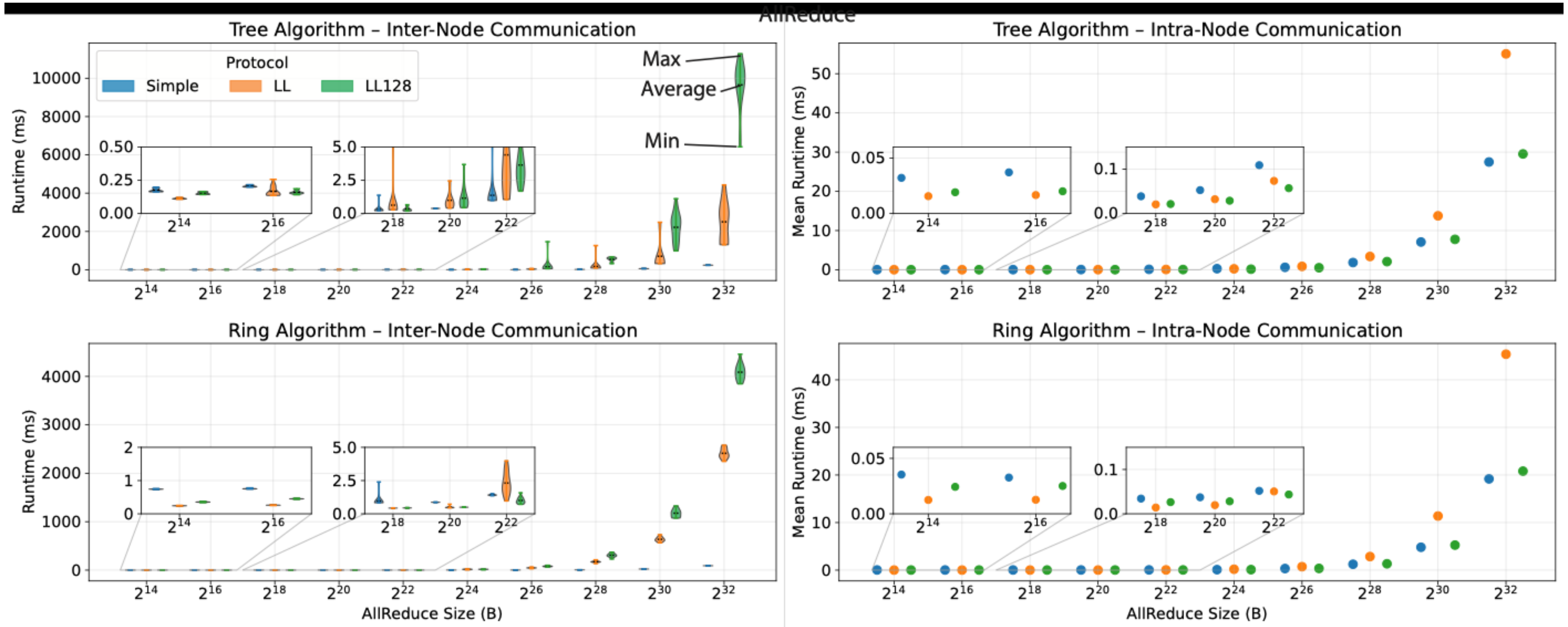


Illustration of Ring AllReduce algorithm

Benchmarking Results

Alps Supercomputer (CSCS)

- Grace Hopper Superchips (GH200)
- 150 GB/s intra-node communication
- 25 GB/s Cray Slingshot
- Dragonfly topology



Benchmarking results for all NCCL collectives in the paper

Impact and Outlook – ATLAHS Toolchain

Nominated as best student paper in SC'25

ATLAHS: An Application-centric Network Simulator Toolchain for AI, HPC, and Distributed Storage

Siyuan Shen*
siyuan.shen@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

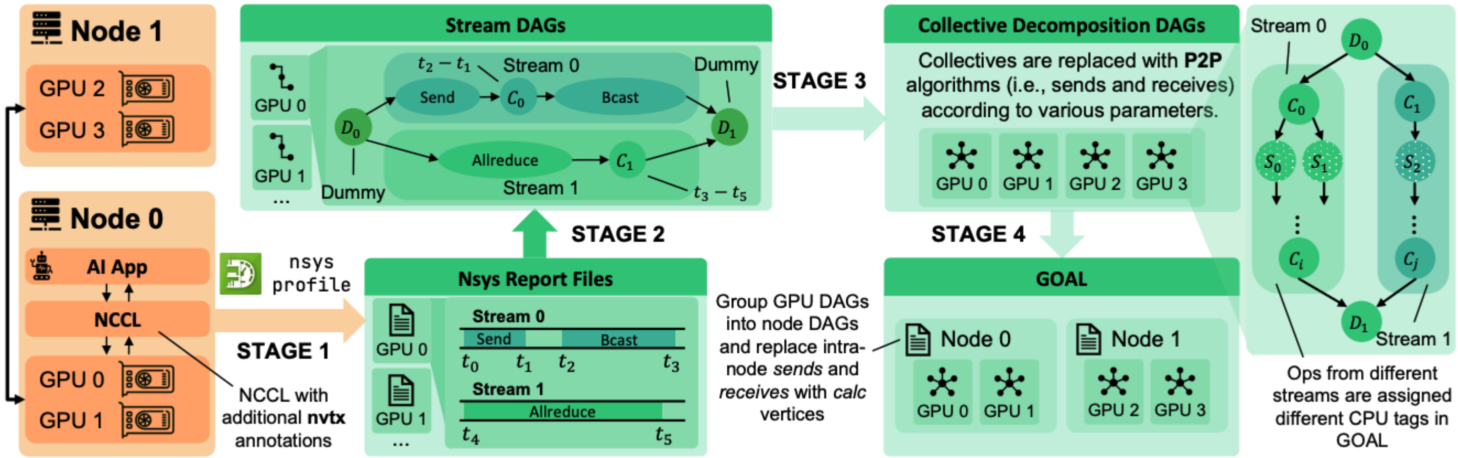
Tommaso Bonato*
tommaso.bonato@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Zhiyi Hu
zhiyihu@student.ethz.ch
ETH Zürich
Zürich, Switzerland

Pasquale Jordan
pasquale.jordan@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Tiancheng Chen
tiancheng.chen@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Torsten Hoefler
torsten.hoefler@inf.ethz.ch
ETH Zürich
Zürich, Switzerland



Paper link:

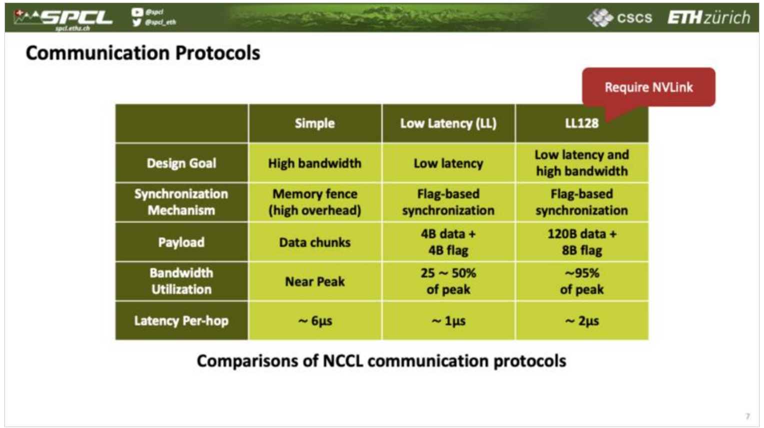
https://arxiv.org/pdf/2505.08936

https://github.com/spcl/atlahs

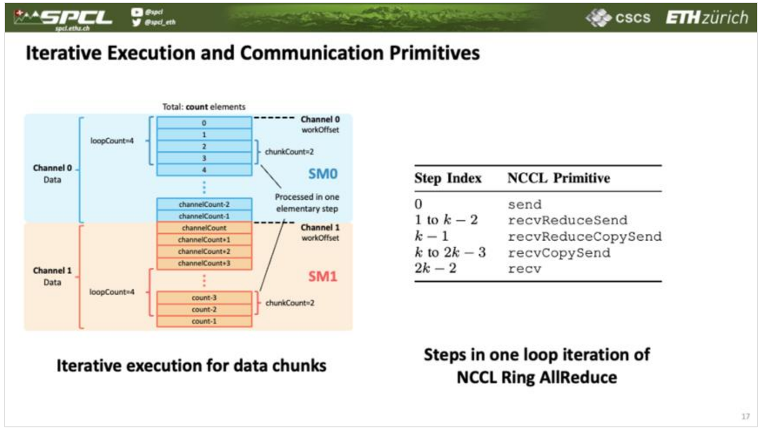
Used insights from this work to generate accurate schedules for NCCL Workloads

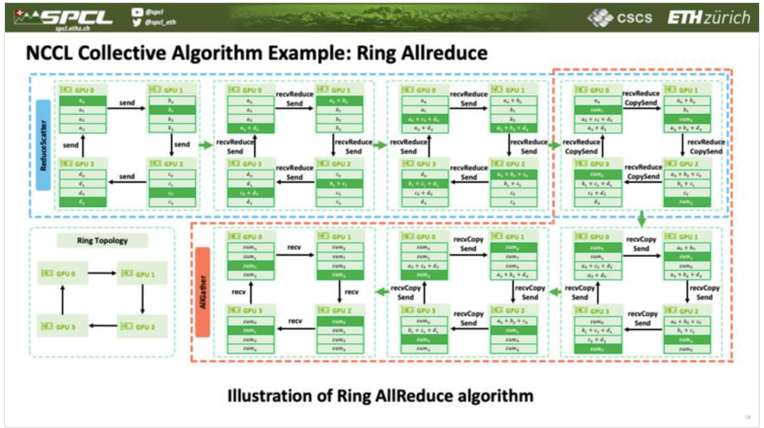
Achieved ~95% accuracy when simulating large-scale AI workloads

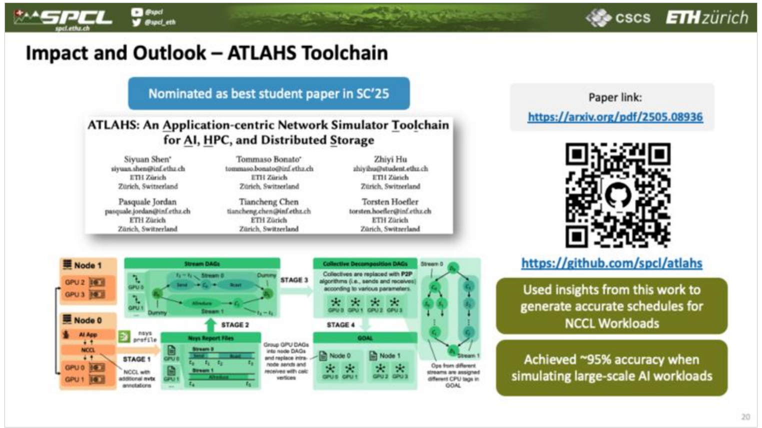
Conclusions



	Simple	Low Latency (LL)	LL128
Design Goal	High bandwidth	Low latency	Low latency and high bandwidth
Synchronization Mechanism	Memory fence (high overhead)	Flag-based synchronization	Flag-based synchronization
Payload	Data chunks	4B data + 4B flag	120B data + 8B flag
Bandwidth Utilization	Near Peak	25 ~ 50% of peak	~95% of peak
Latency Per-hop	~ 6µs	~ 1µs	~ 2µs







More of SPCL's research:

- youtube.com/@spcl — 180+ Talks
- twitter.com/spcl_eth — 1.4K+ Followers
- github.com/spcl — 3.8K+ Stars

… or spcl.ethz.ch

Many more results and analysis in the paper:

https://arxiv.org/pdf/2507.04786