



CWES52010: Connect with the Experts

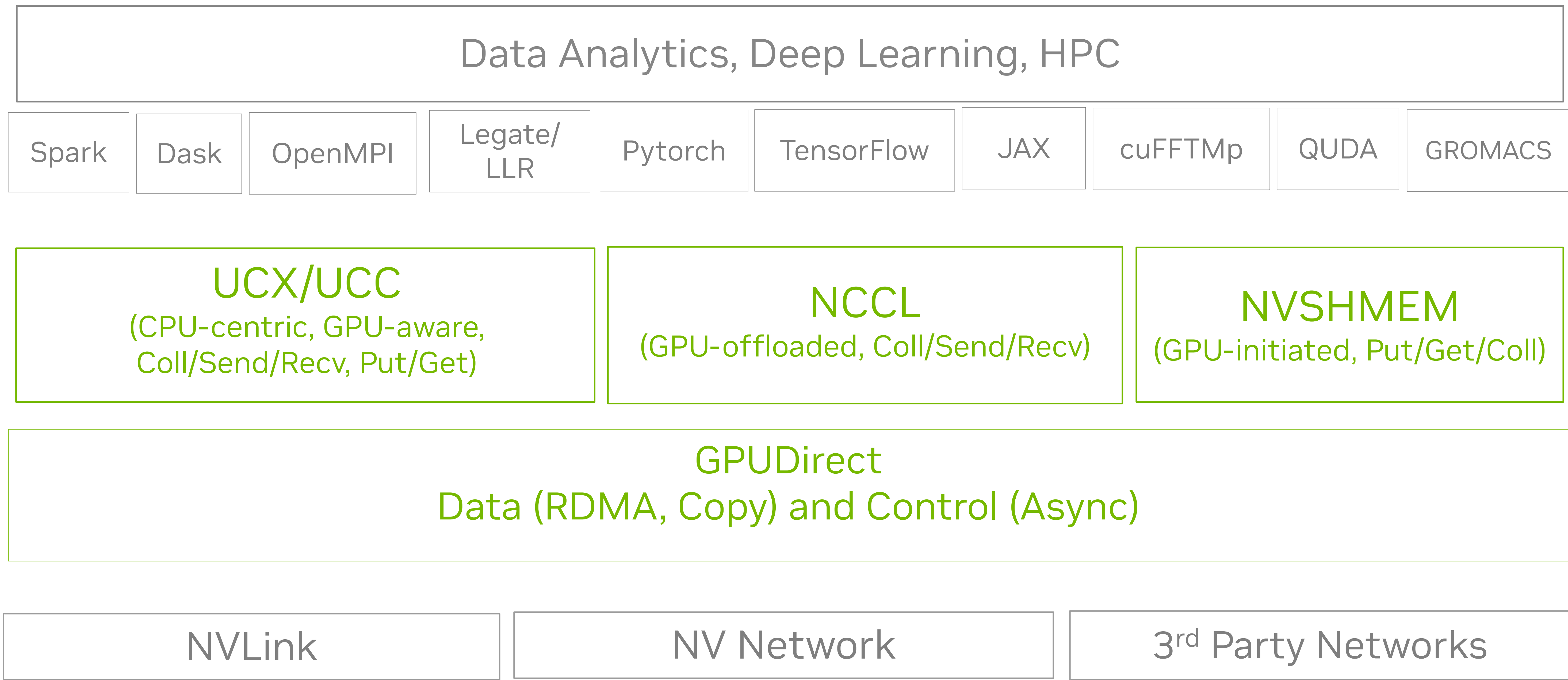
Inter-GPU Communication Techniques and Libraries

Akhil Langer, Akshay Venkatesh, David Addison, Davide Rossetti, Devendar Bureddy, Hessam Mirsadeghi, Jiri Krauss, Jim Dinan, John Bachan, Pak Markthub, Seth Howell, Sreeram Potluri, Sylvain Jeaugey

Experts from Dev. Tech, NCCL, NVSHMEM, MPI, UCX, UCC, GPUDirect, SHARP teams at NVIDIA

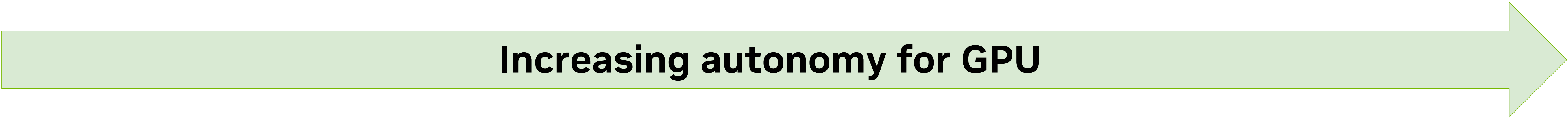
GPU COMMS STACK

Technology | Libraries



NVIDIA GPUDirect™

Variants on who decides what to do and when to do it



GPUDirect	GPUDirect Async	
CPU initiated	CPU prepared, GPU triggered	GPU kernel initiated
GPU: GPUDirect Peer-Peer P2P 3 rd Party: GPUDirect RDMA GDR	Stream triggered GDA-ST Graph triggered GDA-GT Kernel triggered GDA-KT	Kernel Initiated GDA-KI

Preparation: create work request for an IO device, e.g. work queue entry on a cmd queue
Triggering: hand off work request to an IO device, e.g. ring a doorbell
Initiated: Preparation + Triggering

UCX: UNIFIED COMMUNICATION X

Overview

Unified API for data movement

Tagged messaging, Active Messages, RMA/Atomics

Applications

HPC

CUDA-aware MPI – OpenMPI/MPICH

RAPIDS

DASK, SPARK

Legate

Omniverse

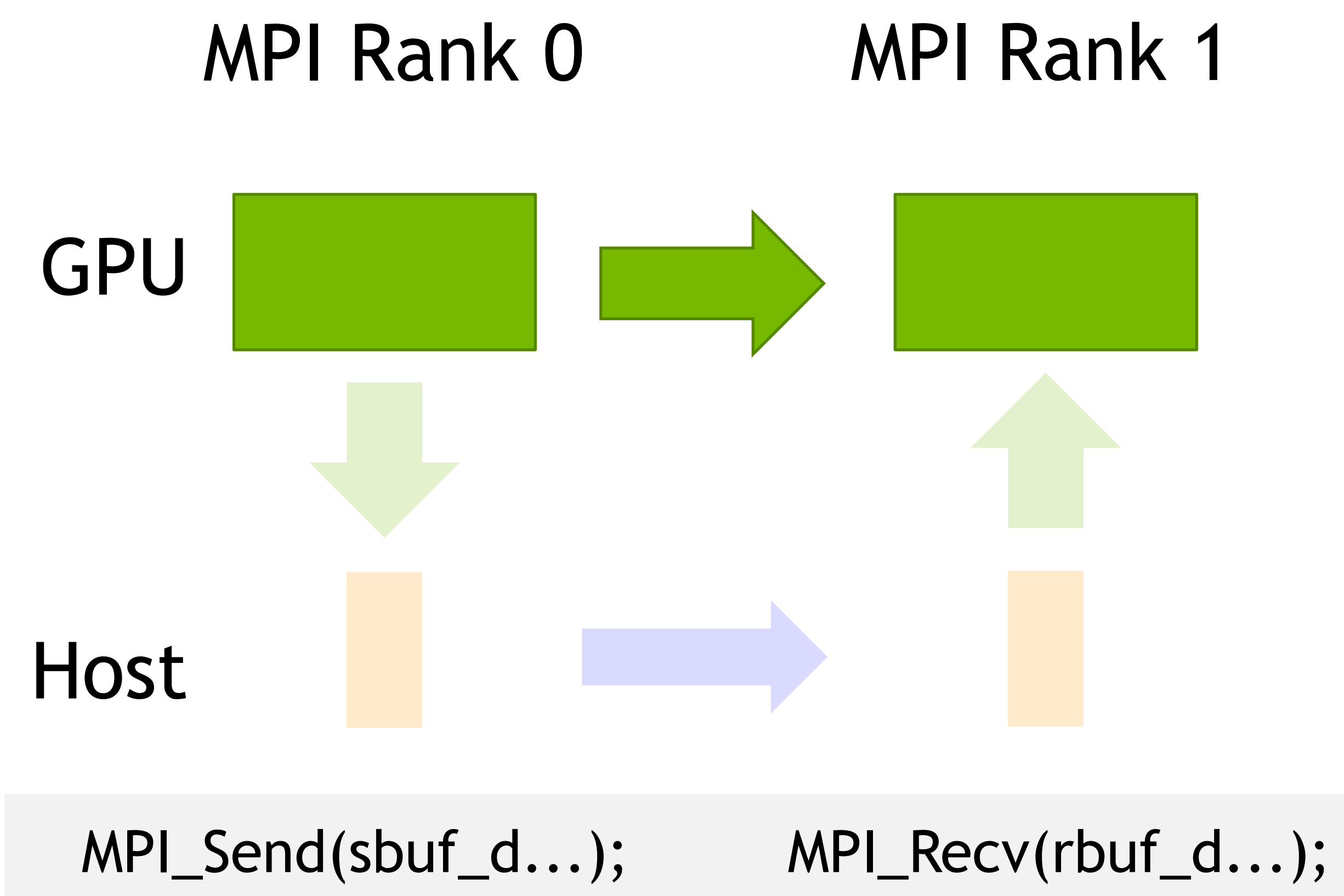
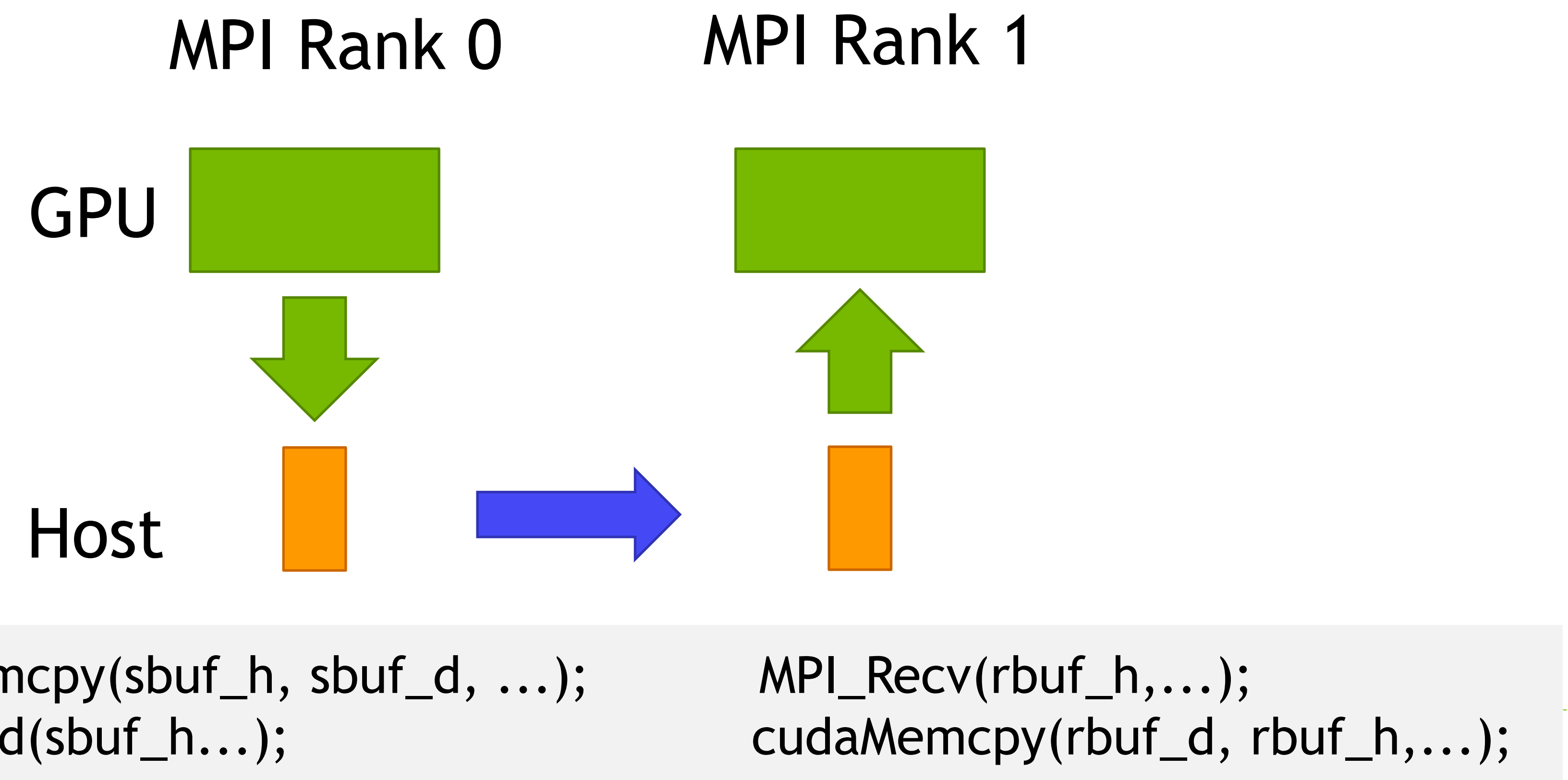
Recent Enhancements

Hopper support

Reduced registration overheads

Improved topology detection

Improved performance for managed memory



UCC: UNIFIED COLLECTIVE COMMUNICATION

Overview

Unifies Software and Hardware collectives

Software currently supported as UCP TL and hardware supported as SHARP TL

Unifies CPU and GPU collectives

Currently provides native support or via NCCL TL (NVIDIA GPUs) or RCCL TL (AMD GPUs)

Unifies and satisfies the needs of MPI, OpenSHMEM, and PyTorch

MPI (supported via Open MPI)

OpenSHMEM (supported via OSHMEM/Open MPI)

Native support for PyTorch

(<https://github.com/pytorch/pytorch/blob/master/torch/csrc/distributed/c10d/ProcessGroupUCC.hpp>)

Not only satisfies but provides many optimized algorithms and implementations

Optimized algorithms for latency, bandwidth, and scalability

NCCL Overview

NCCL provides primitives for GPU-to-GPU communication, optimized for all platforms.

It is implemented as a library launching **CUDA kernels** on the GPU, interleaved with compute kernels via CUDA stream semantics.

Initialization

CPU-side communication

Socket-based out-of-band bootstrap

Fault tolerance

GTC '21

Hardware topology detection

Graph with GPUs, CPUs, NICs, PCI switches, NVLinks and NVSwitches.

Graph search for collective algorithms.

GTC '20

Collective communication algorithms

Ring, Tree, Collnet (Chain/Direct), **NVLink SHARP**.

GTC '22

Point-to-point communication

Send/Receive semantics

Communication schedule

Aggregation

GTC '22

NCCL Recent features

H100 support

NVLink SHARP support, “NVLS” algorithm.

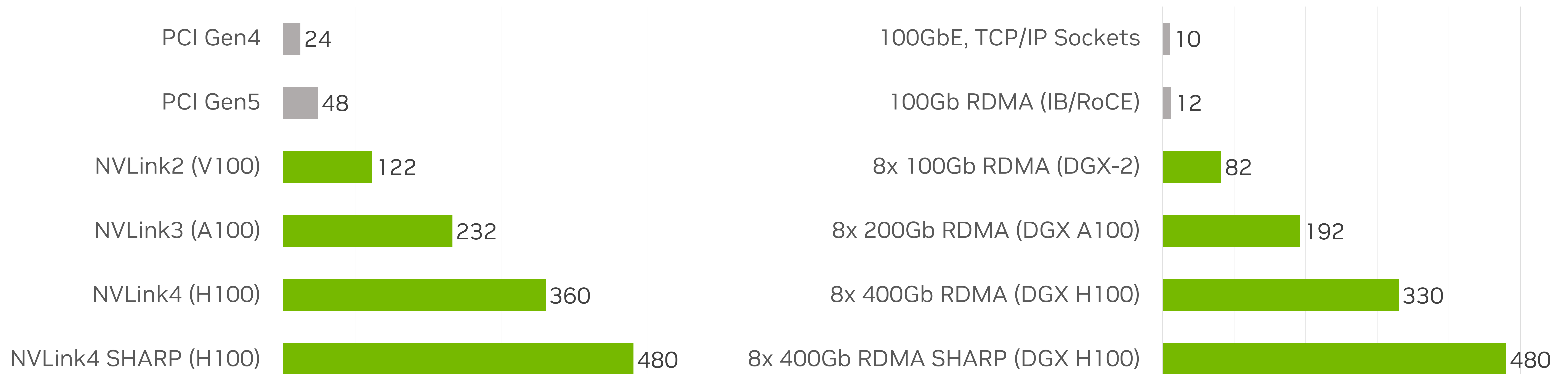
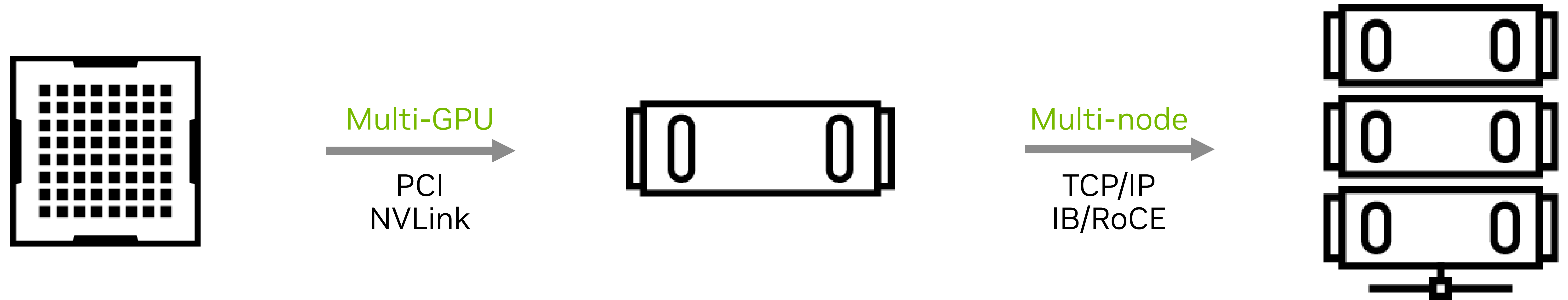
Improved fault tolerance

Ability to abort init/destroy operations, and restart NCCL without restarting the application.

Communicator configuration API

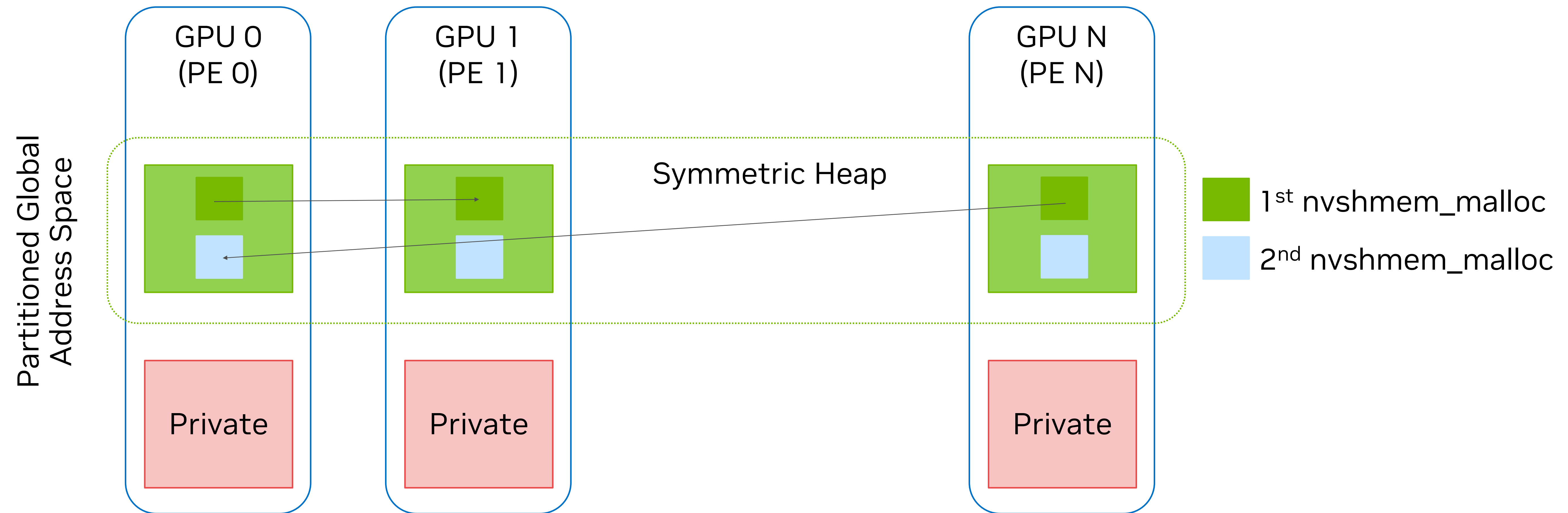
Allow to specify communicator settings through a config structure instead of environment variables: blocking mode, maximum number of CTAs to launch, network type to use, ...

Collective Communication Bandwidth



NCCL Tests Allreduce Bus Bandwidth in GB/s

NVSHMEM Symmetric Memory Model



Symmetric objects are allocated collectively with the same size on every PE
Symmetric memory: *nvshmem_malloc(...)*; Private memory: *cudaMalloc(...)*

Read: *nvshmem_get(...)*; Write: *nvshmem_put(...)*; Atomic: *nvshmem_atomic_add(...)*
Flush writes: *nvshmem_quiet()*; Order writes: *nvshmem_fence()*

Synchronize: *nvshmem_barrier()*; Poll: *nvshmem_wait_until(...)*

NVSHMEM Latest Features

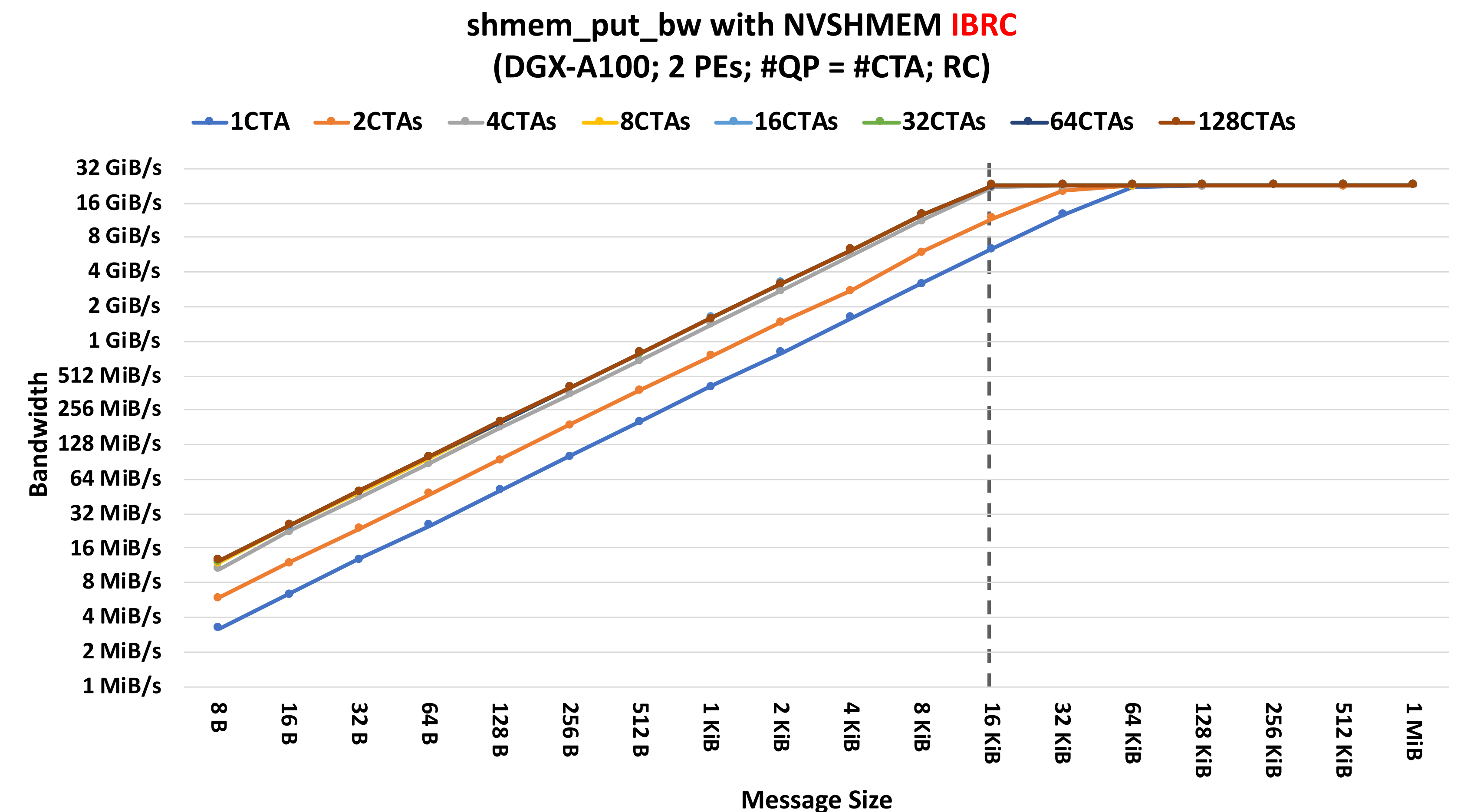
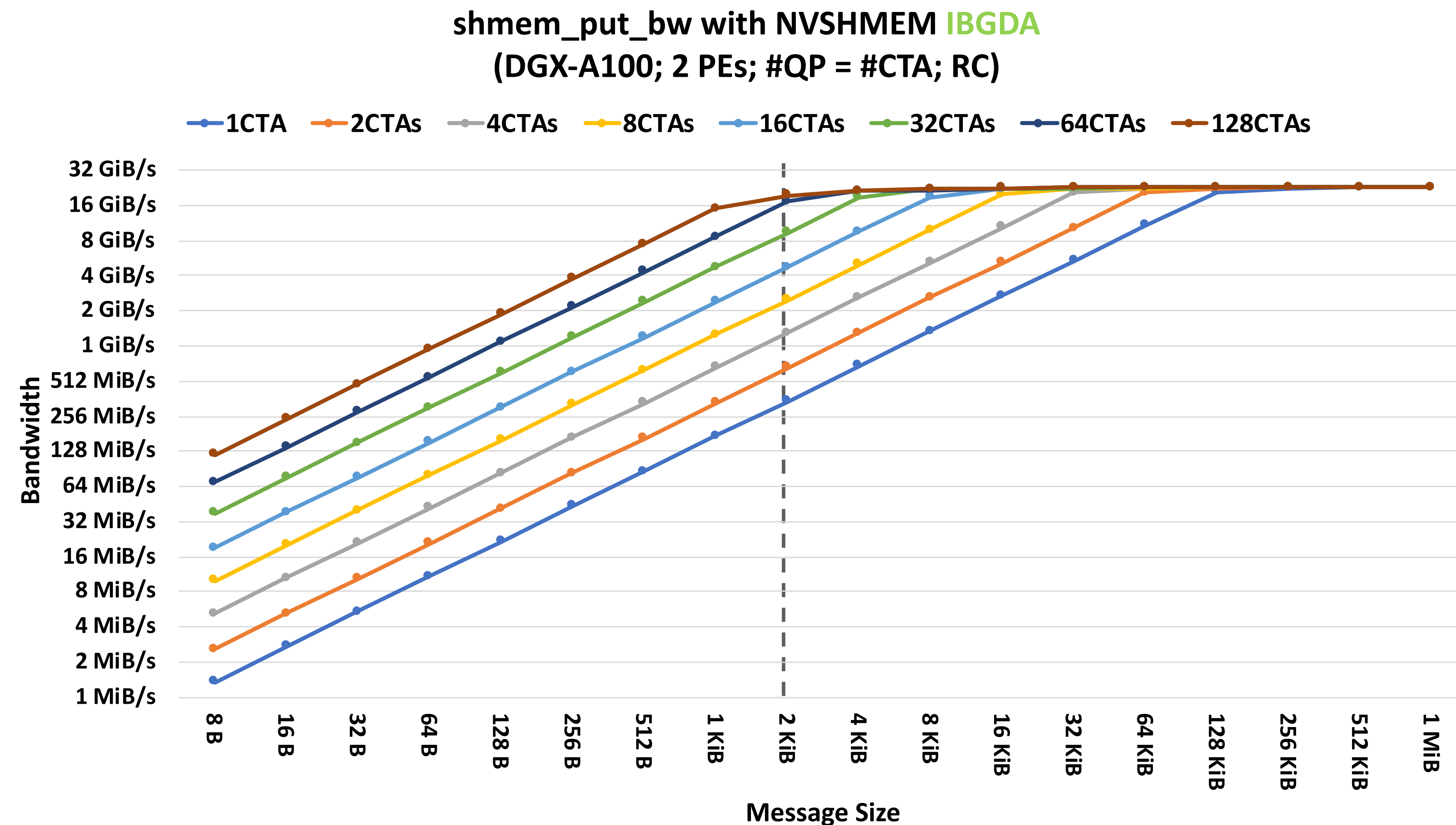
Announcing NVSHMEM 2.9.0

New Features Added Since NVSHMEM 2.5.0 / GTC Spring 2022

- InfiniBand GPUDirect Async (IBGDA) Transport
- HPC SDK Integration
 - Library versioning
 - CUDA minor version compatibility support
 - Modular transport, bootstrap support
- CMake build system
- Collective Optimizations
- Inlined Device Code
- DMABUF support
 - Compatibility with upstream ibverbs
 - nv_peer_mem module not needed
- Slingshot-11 support
 - Beta support
 - Ongoing work on stability and performance

GPU-Initiated (IBGDA) vs Proxy Communication Performance

Full Bandwidth at 2kB versus 16kB Transfer Sizes



- IBGDA (left) performance scales with the number of thread blocks (CTAs); Proxy (right) scaling limit at 8 CTAs
- IBGDA block-put throughput up to 5.75x higher with 128 CTAs; IBRC block-put throughput up to 1.8x higher with 1 CTA
- Tradeoff: IBGDA provides high throughput when CTAs communicate in parallel, proxy has low overhead op handoff



Companion Presentations

- [Magnum IO sessions](#) (Select Magnum IO under Product and Technologies Section in GTC Catalog)
- Connect with the Experts: Inter-GPU Communication Techniques and Libraries [CWES52010]
- How to Streamline Shared Memory Space With the NVSHMEM Communication Library [S51705]
- Scaling Deep Learning Training: Fast Inter-GPU Communication with NCCL [S51111]
- Blending Accelerated Programming Models in the Face of Increasing Hardware Diversity [S51215]
- Overcoming IO Bottlenecks in GPU-Accelerated HPC and AI Applications by Bypassing System Memory [PS51246]
- Designing the Next Generation of AI Systems [S51839]
- Accelerating Data Movement Between GPUs and Storage or Memory [S51142]

