HCCL: Collective Communication for Meta Training and Inference Accelerators

Wes Bland, Meta
On behalf of the entire MTIA Comms team (see paper for full author list)

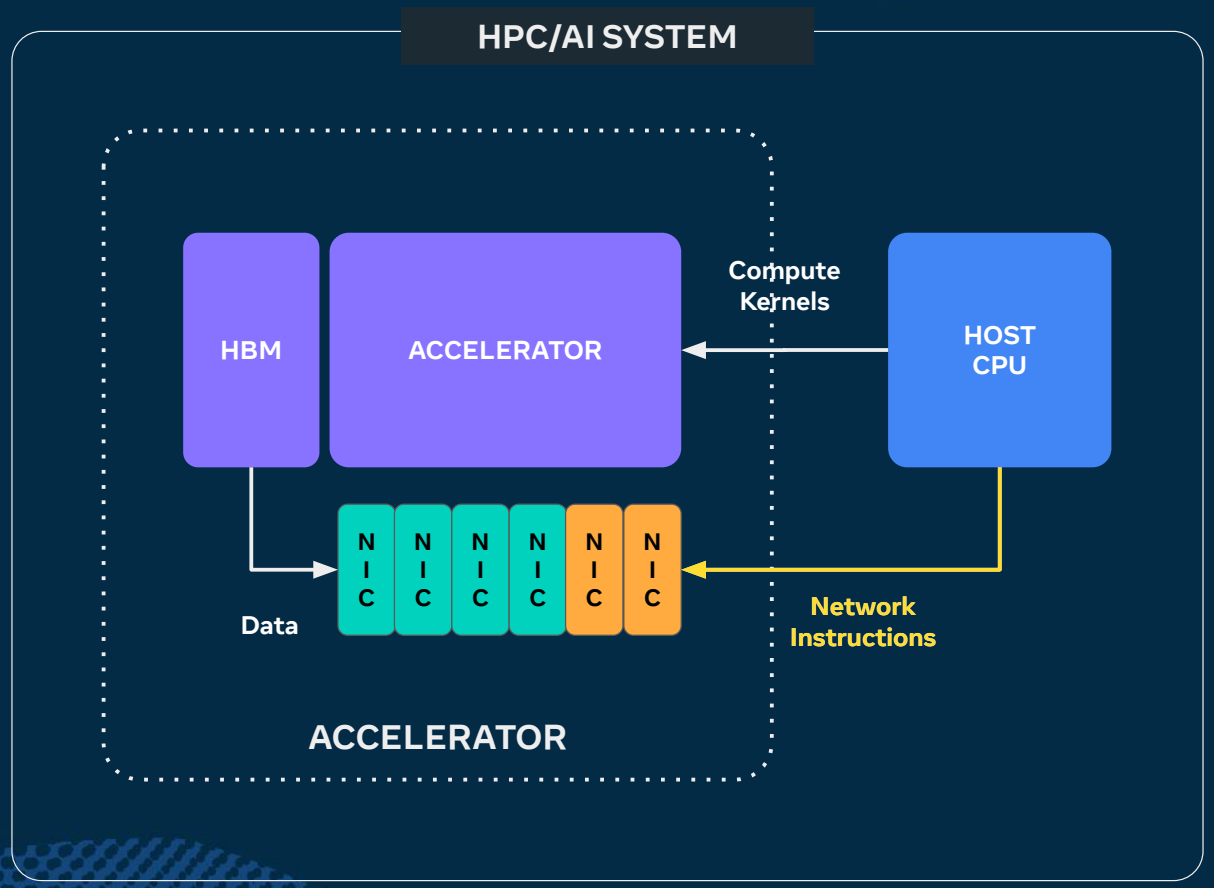


SC26 hpc unites | Chicago

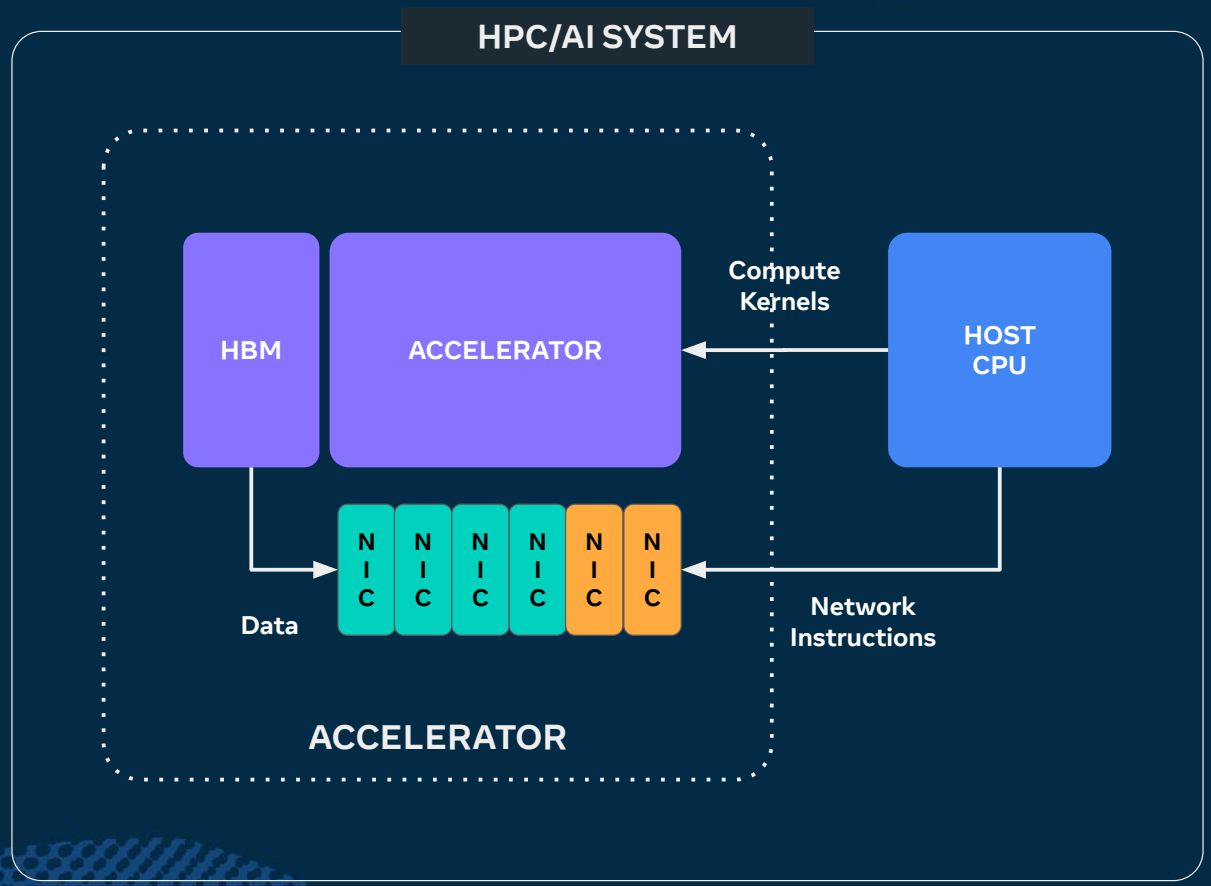
Outline

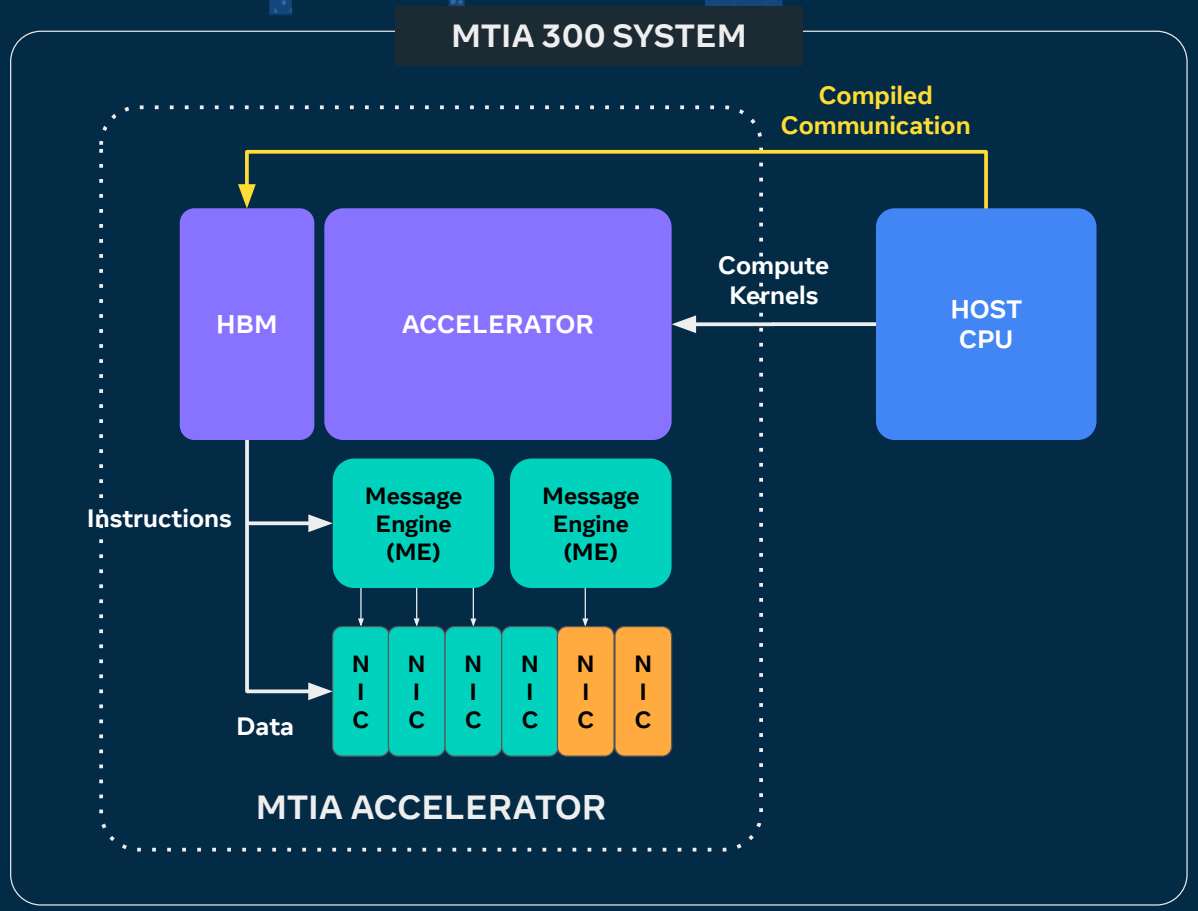
1. Hardware introduction
2. Communication Software Stack
3. Training vs. Inference Optimizations
4. Performance Analysis

Hardware Offloading



Hardware Offloading





SC26

MTIA 300

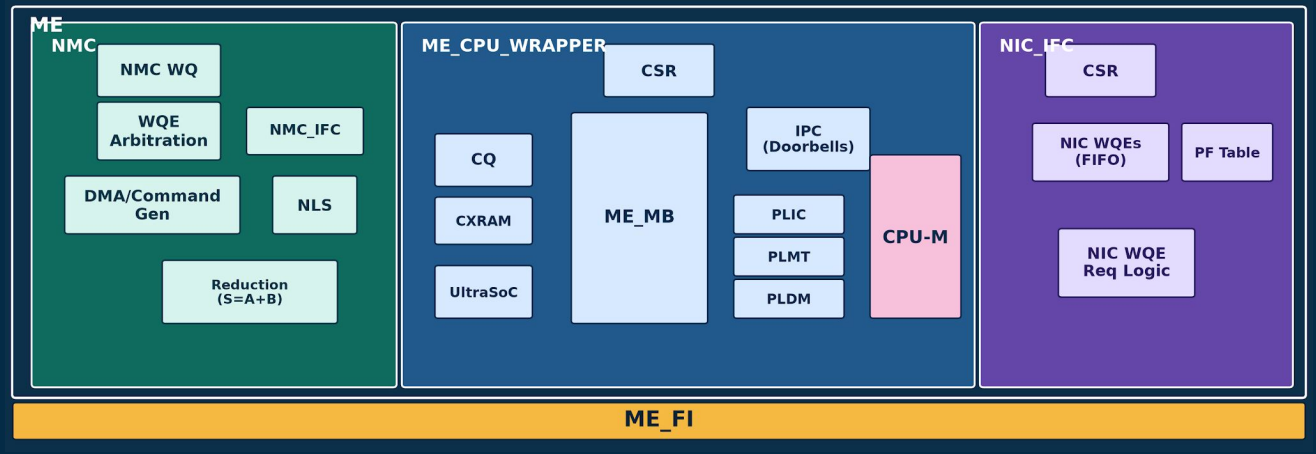


12x6 processing element (PE) grid for compute

2 banks of 8 message engines (MEs) for driving communication

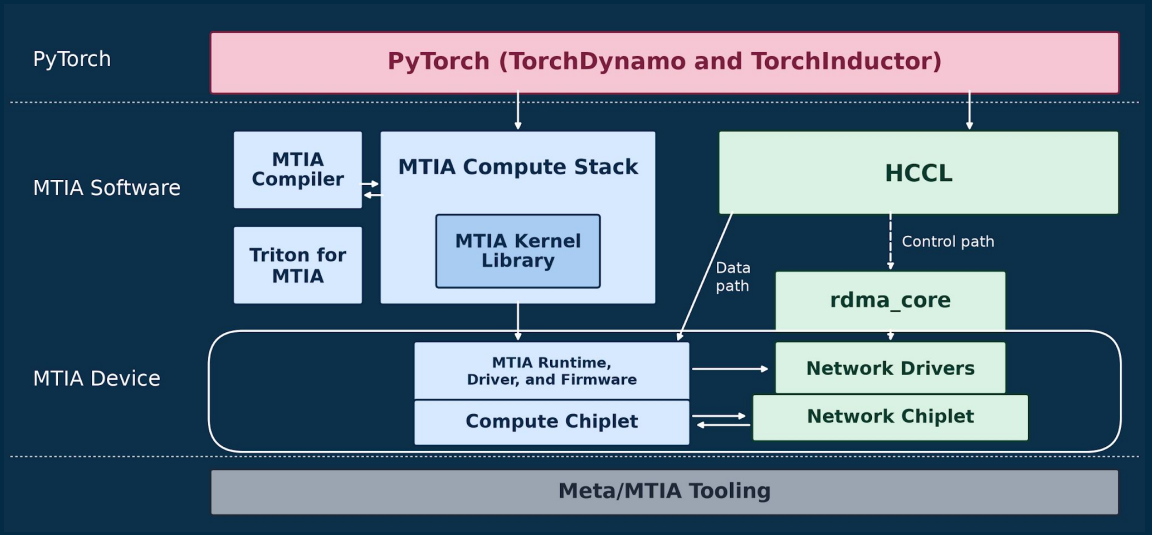
Near Memory Compute (NMC) for compute operations for communication

2 banks of 6 RDMA NICs for I/O



Software Stack

- PyTorch as primary application interface
- RDMA as transport/control path interface
  - Uses traditional QPs, MRs, etc. via verbs interface
- HCCL bridges the gap for comms



HCCL: Software Abstraction for Hardware

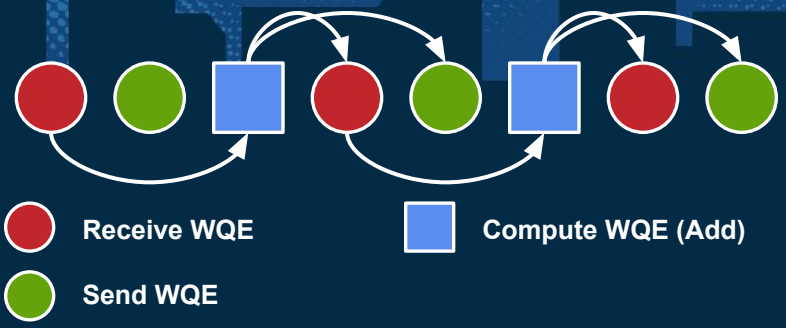
Host-based communication construction

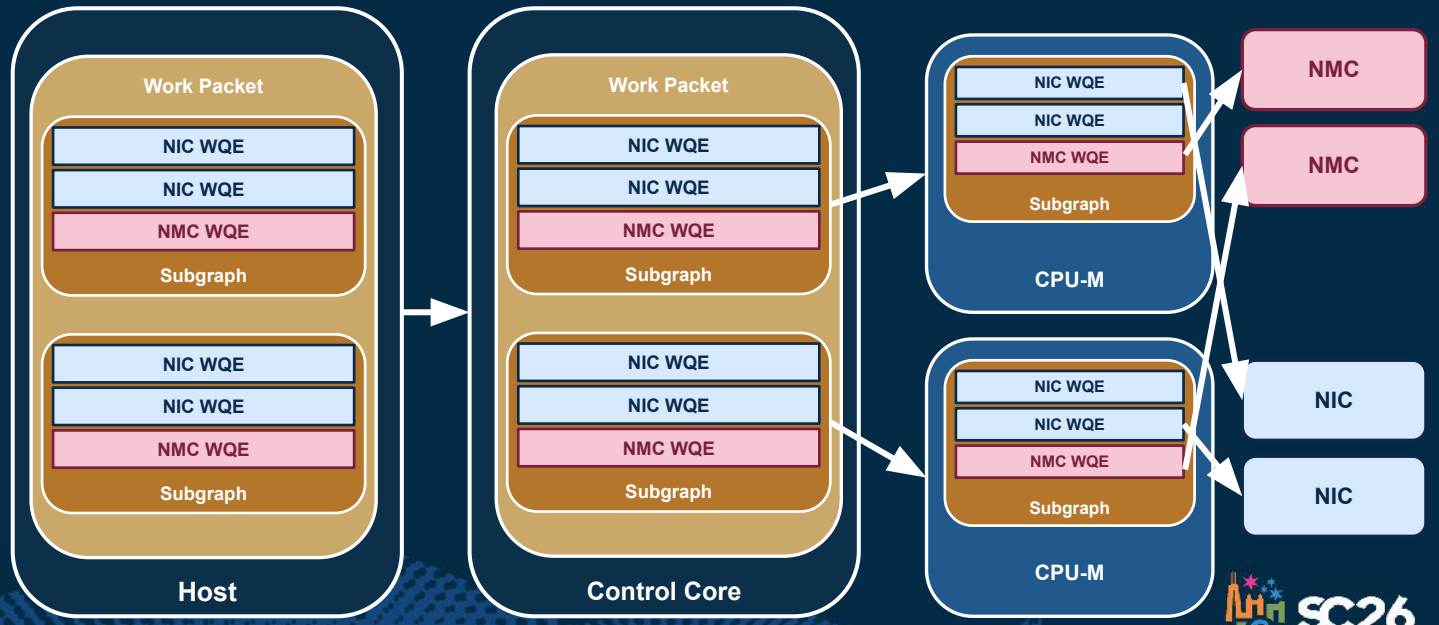
Maps communication pattern onto available hardware

Work packets are serialized (stream semantics)

Subgraphs are parallelized

Flow control dependencies for each work queue element (WQE)





Optimizations (Training)

Initially designed for training

- 16 MEs to drive communication
- 16 NMCs to help offload reductions

Parallelize work across all hardware

- Overlap comms & compute to utilize all hardware

Optimizations (Inference) - One-Sided Communication

Used heavily in GenAI fused kernels (e.g., MoE)

Routing all traffic through the ME doesn’t make sense

- Latency cost of generating a work packet and subgraph too high

Expose put, fence, flush APIs to the PE grid directly

- Uses MEs to monitor completion asynchronously

Set up control path on host ahead of time

Optimizations (Inference) - Device-Resident Communication

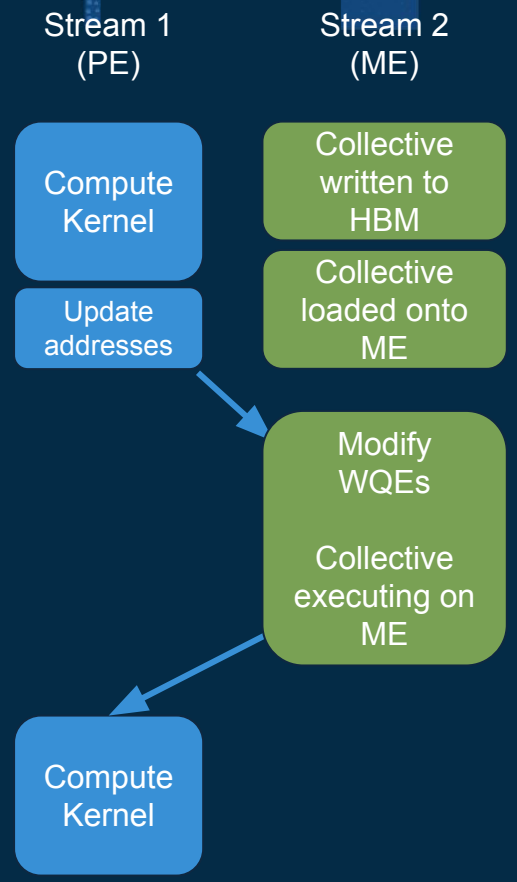
Strong dependencies between compute and comms

- E.g., MoE routing kernels determine the targets and sizes for next Alltoallv

Scheduling new work in the stream would be slow

Dynamic collectives:

- Pre-schedule a collective
- Insert metadata later in a compute kernel
- Adapt collective on the fly on the ME



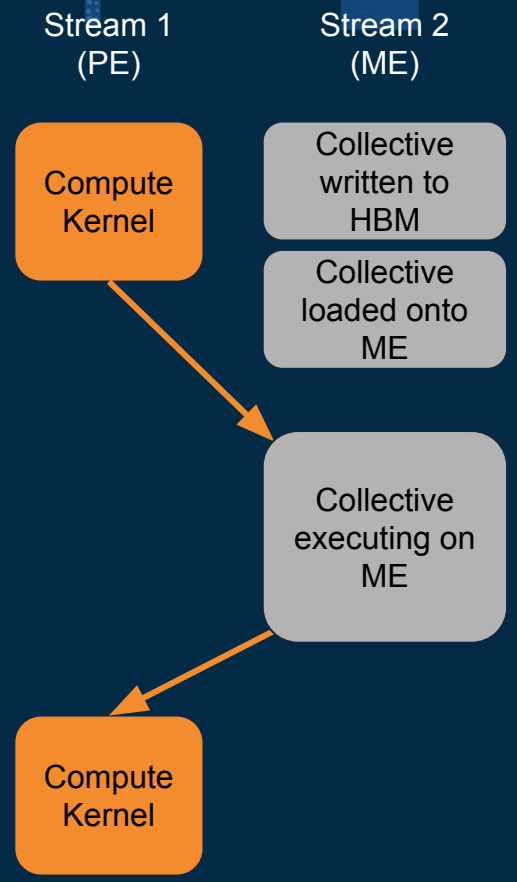
Optimizations (Inference) - Device-Triggered Communication

Stream semantics add bubbles within the stream

- Want to overlap with previous compute

Solution: Remove stream dependency

- Use software semaphore for compute kernel to signal start of collective
- Can use another semaphore for completion



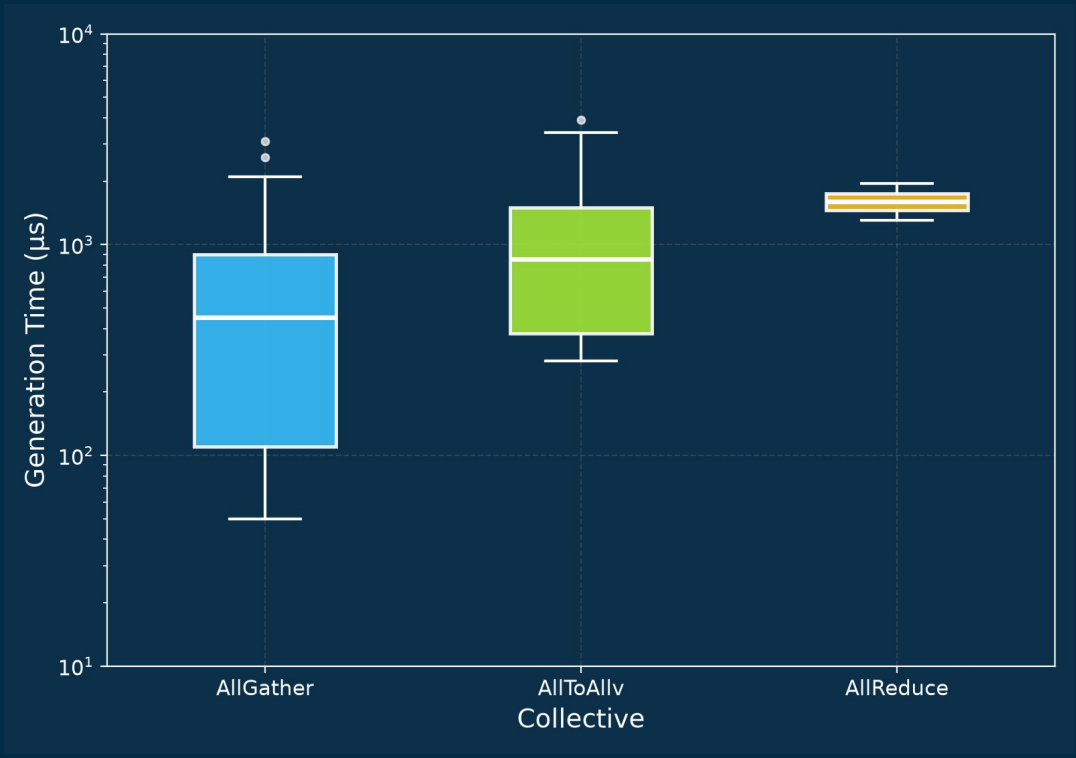
Performance - Host-Side Performance

Host generation time varies by data size and number of ranks

For 256 ranks, between 100µs - 1ms

None of the time is exposed in the model

Use caching to minimize the impact



SC26

Performance - Collective Performance

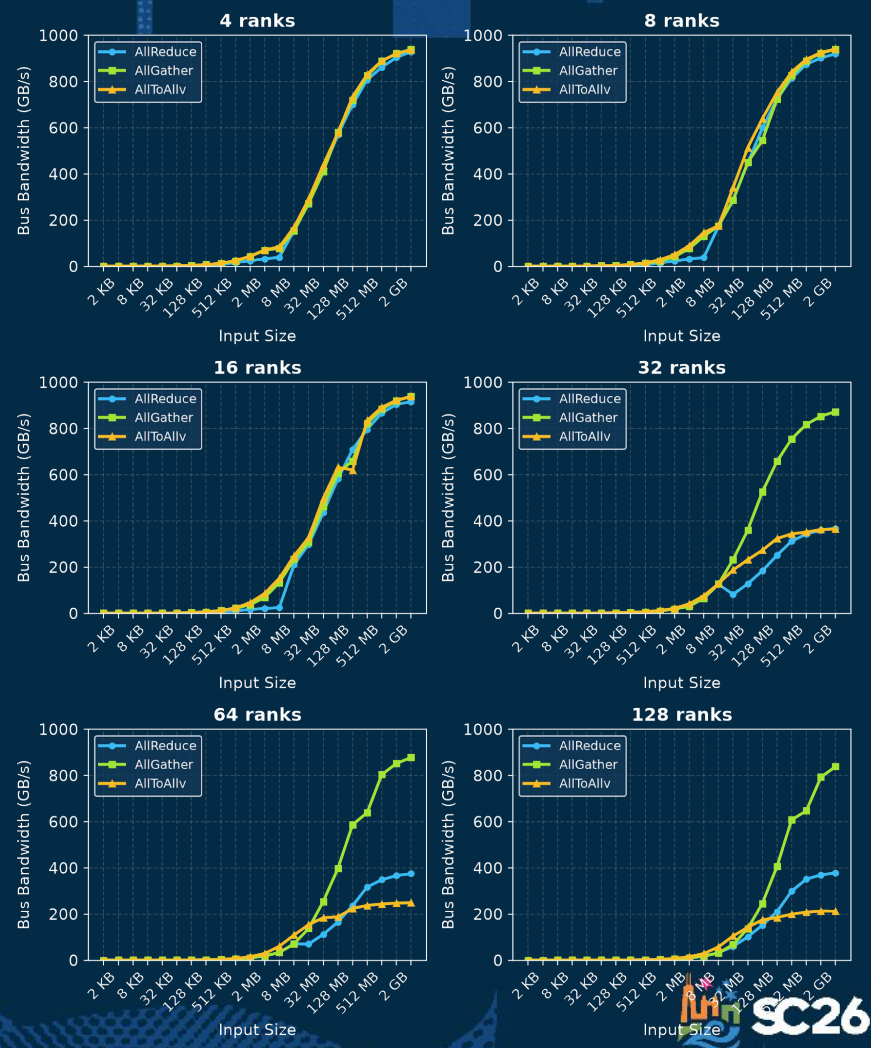
Good bandwidth utilization compared to roofline

- 16 ranks in scale-up domain
- Maximum combined B/W: 1 TB/s

Curve bends above the size of the scale-out domain

- 800 GB/s for scale-up
- 200 GB/s for scale-out
  - Can also be used for scale-up

Performance optimization is a constant source of improvement

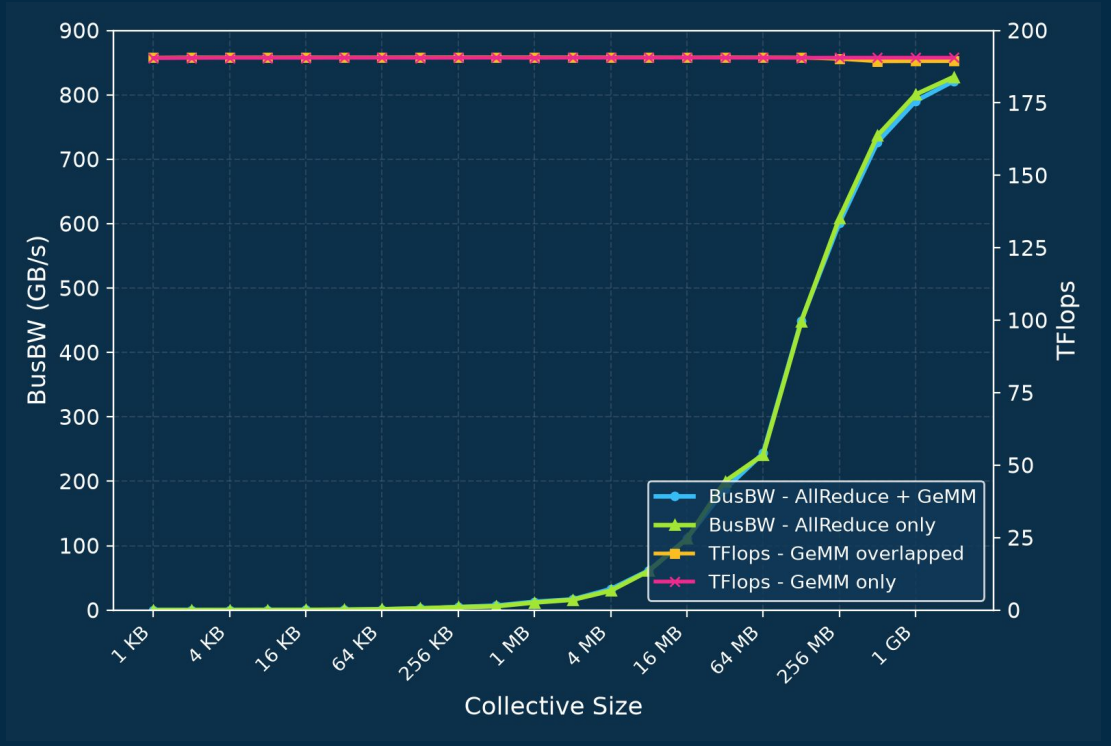


Performance - Collective + Compute Overlapping

Overlapping compute & comms does not hurt performance

- <0.5% compute performance loss
- Very similar for collectives

No shared resources between compute and comms



HCCL: Collective Communication for Meta Training and Inference Accelerators

Wes Bland, Meta
On behalf of the entire MTIA Comms team (see paper for full author list)



SC26 hpc unites | Chicago