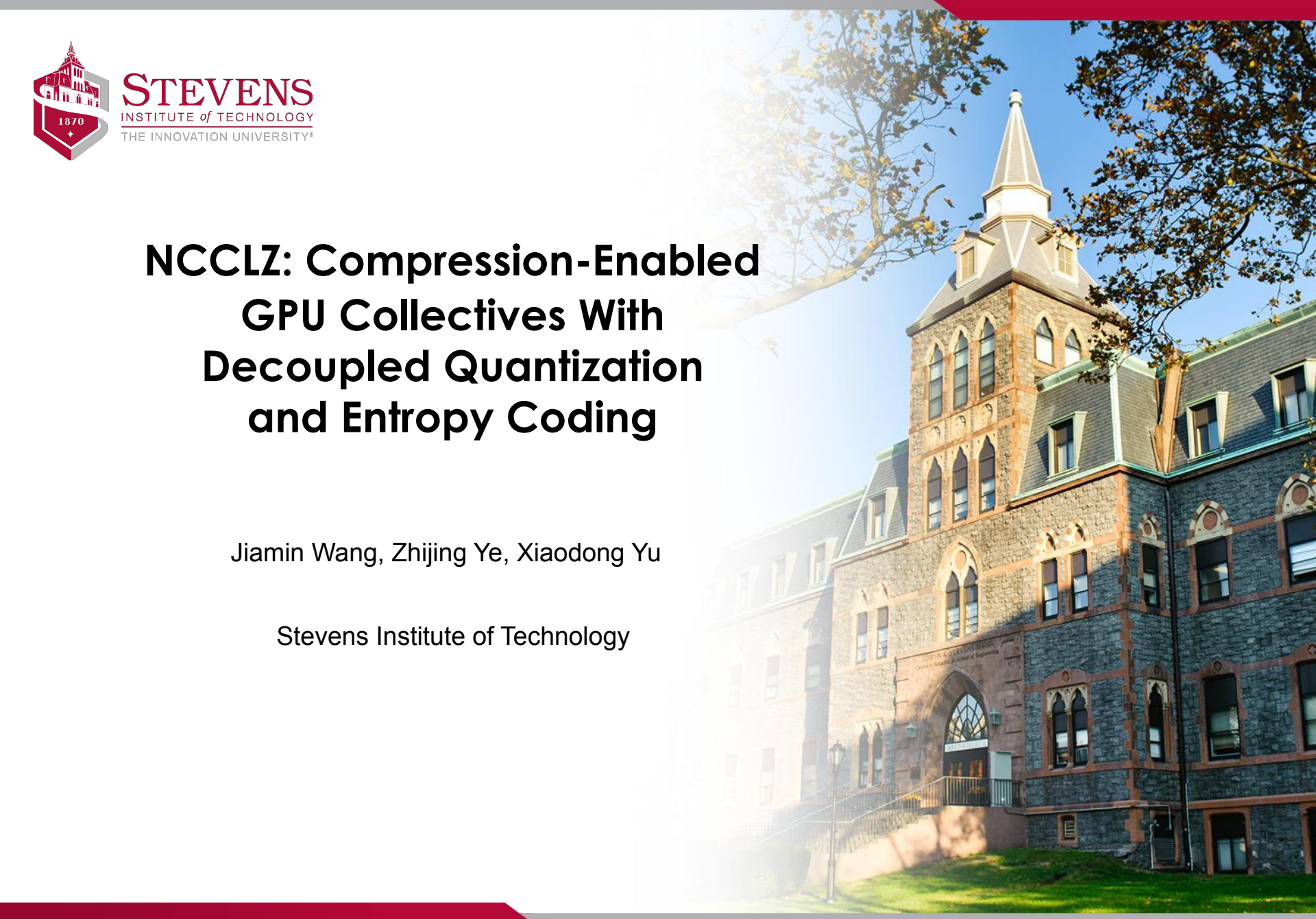


NCCLZ: Compression-Enabled GPU Collectives With Decoupled Quantization and Entropy Coding

Jiamin Wang, Zhijing Ye, Xiaodong Yu

Stevens Institute of Technology

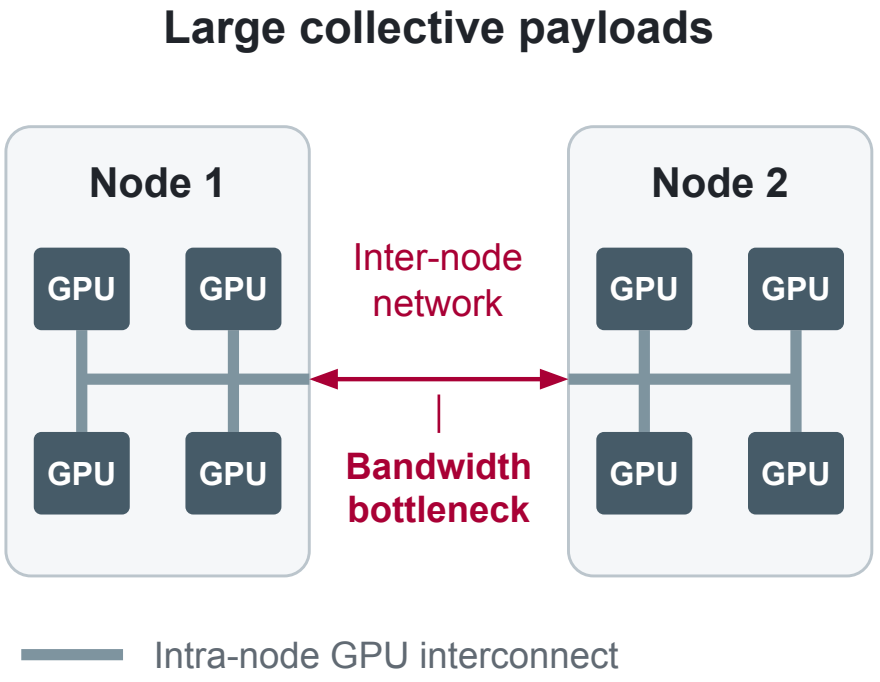
© 2026 Stevens Institute of Technology. All Rights Reserved.

Intro: Bandwidth Bottlenecks in GPU Collectives

Can sending fewer bytes make GPU collectives faster?

Intro: Inter-Node Bandwidth Limits GPU Collectives

- Scientific and training workloads exchange **large tensors** through collectives.
- Large inter-node messages can remain **bandwidth-bound** despite optimized NCCL execution.
- Compression helps only when **saved transfer time** exceeds exposed compression overhead.



Fewer bytes matter only when they reduce end-to-end time.

Background: NCCL Execution and Compression Stages

How does NCCL move data, and what do the two compression stages do?

1

NCCL execution

Collective APIs → primitives → channels / transports

2

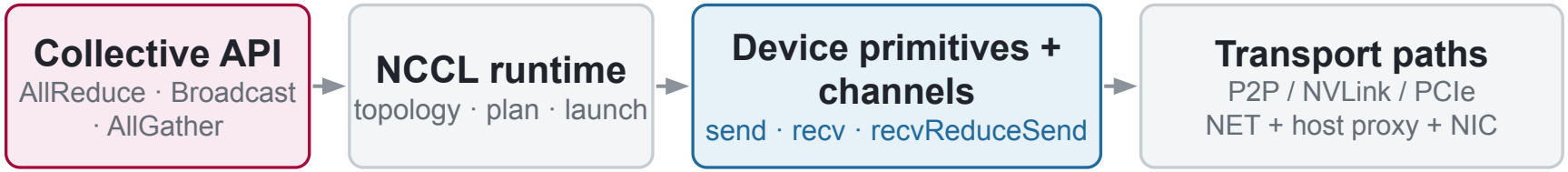
Compression stages

Quantization → symbols → lossless coding

NCCL Execution and Compression Stages

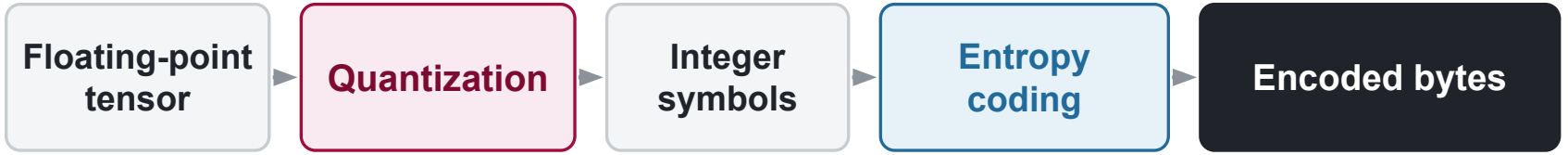
NCCL determines how data moves; compression determines what is sent.

1 NCCL execution path



Collectives are decomposed into device-side primitives and pipelined across channels.

2 Compression stages



LOSSY

determines the approximation

LOSSLESS

adds no further distortion

Quantization controls numerical accuracy; entropy coding controls byte efficiency.

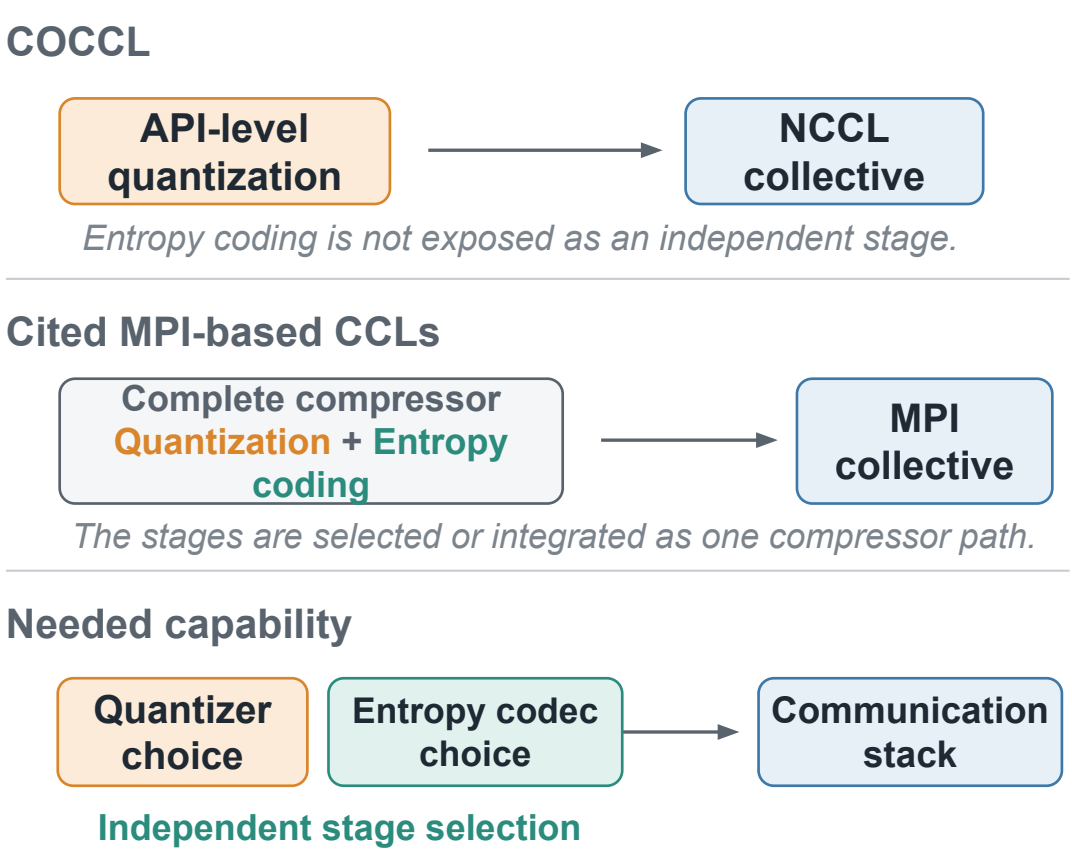
Motivation: Limits of Existing Compression Integration

Three integration challenges shape the design.

1. **Stage-level flexibility**
2. **Overlap granularity**
3. **Error propagation**

Stage-Level Flexibility

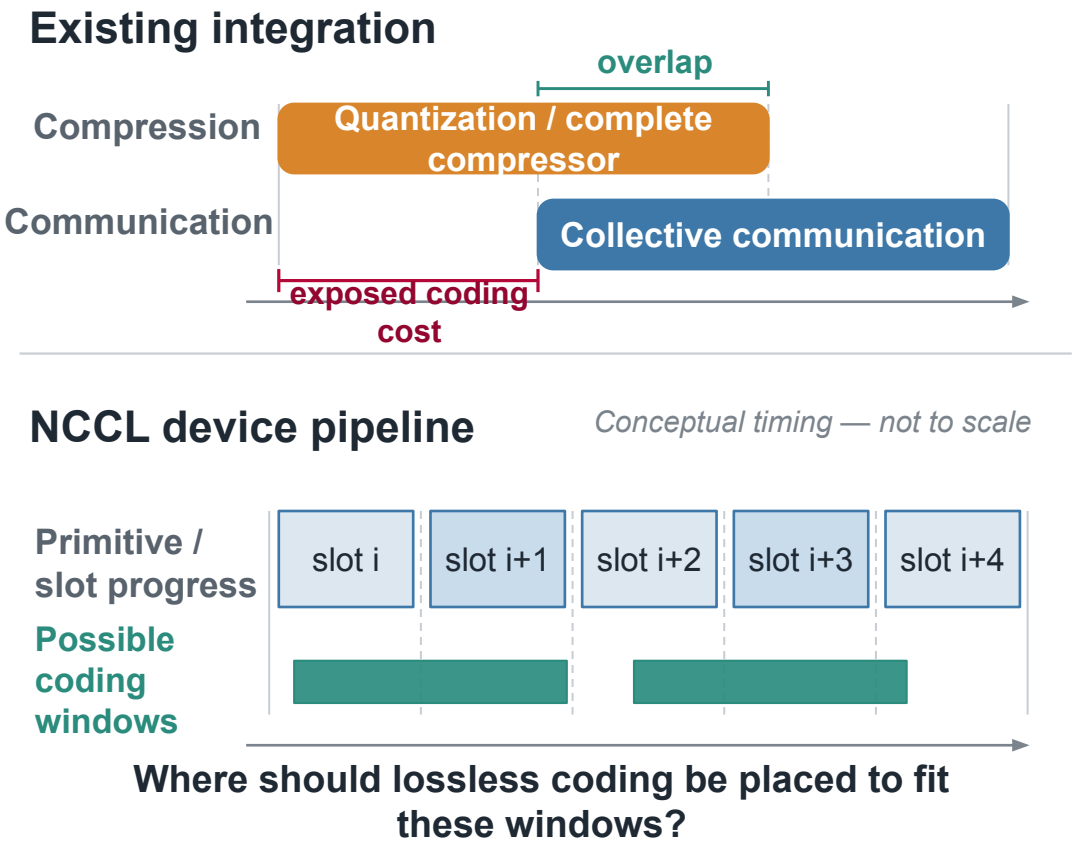
- COCCL places quantization at the API layer without a separate entropy-coding stage.
- The cited MPI-based systems integrate or select complete compressors.
- Changing either stage independently requires additional integration work.



The integration boundary determines stage-level flexibility.

Overlap Granularity

- Existing systems overlap API-level quantization or a complete compressor with communication.
- NCCL exposes a device-side primitive and slot pipeline.
- Placement determines how much codec work remains on the critical path.

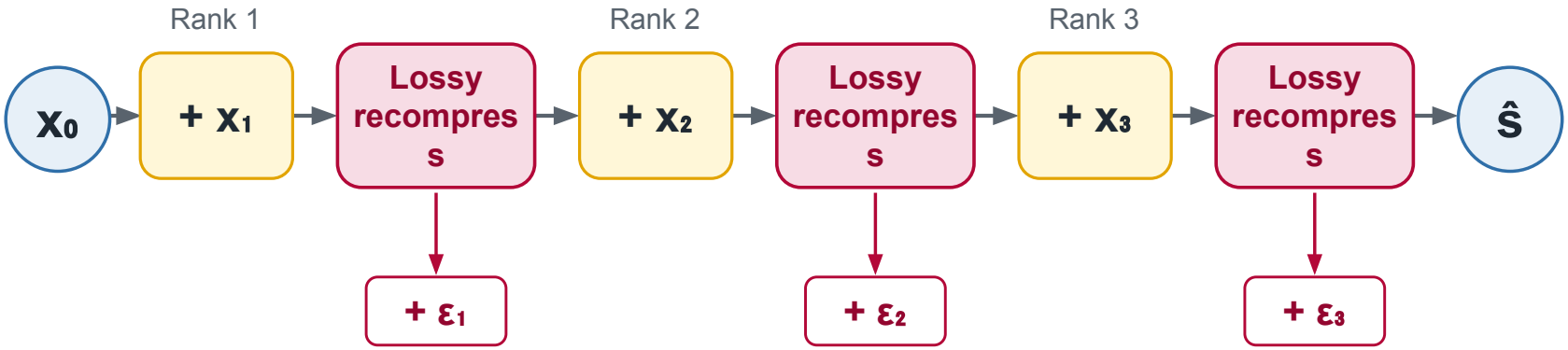


Reducing bytes is not enough; coding must fit the overlap opportunity.

Error Propagation in Reduction Collectives

A partial result can be dequantized and requantized at every reduction step.

Example: a four-rank reduction path



Final result = true sum $+ \varepsilon_1 + \varepsilon_2 + \varepsilon_3$

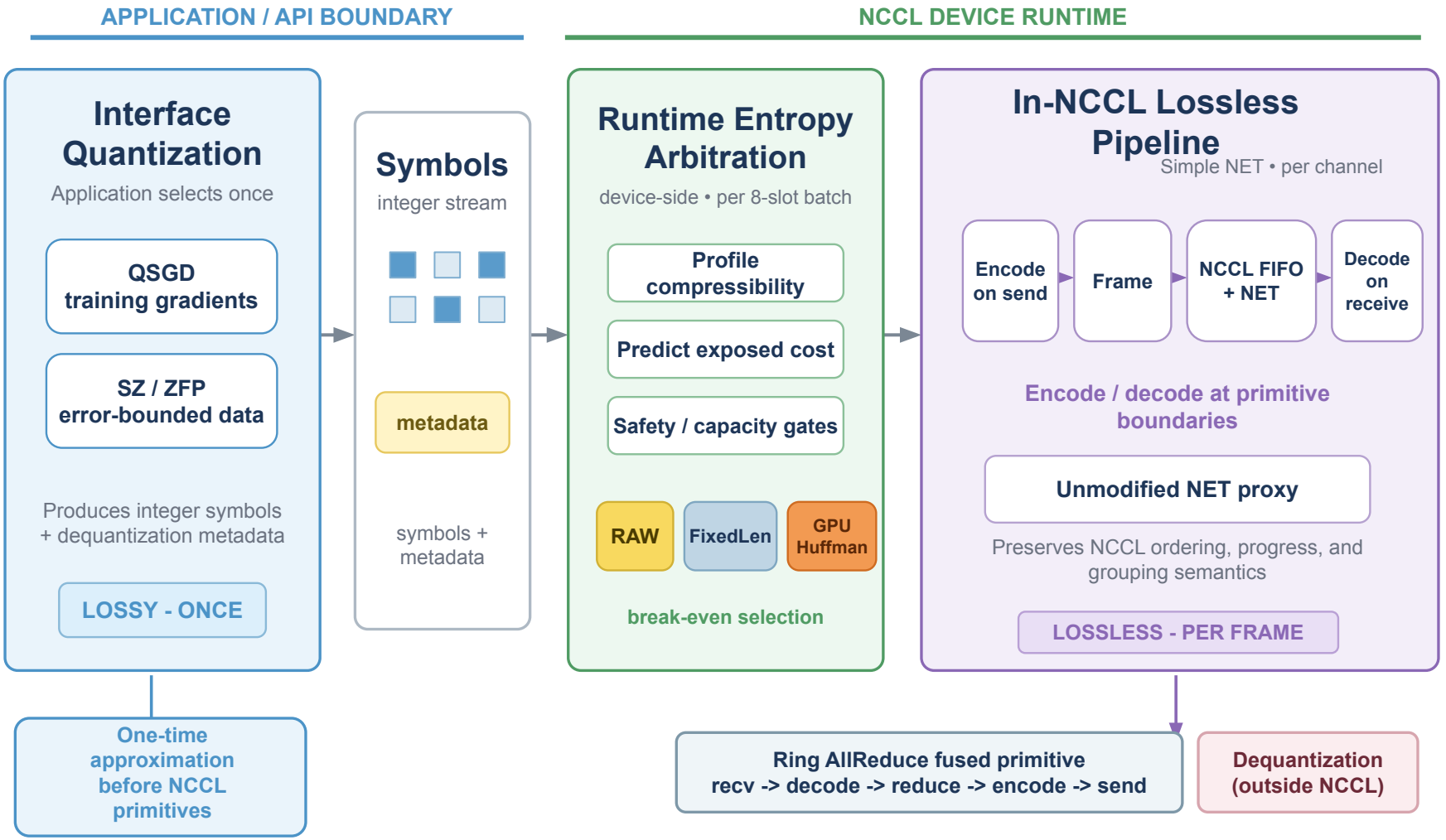
- Each lossy recompression introduces another approximation term.
- The error terms may accumulate along the reduction path.

Repeated lossy recompression can accumulate distortion along the reduction path.

Design: Decoupled Quantization and Entropy Coding

NCCLZ Overview: Decoupled Integration

What stays outside NCCL, and what runs inside it?



Runtime Entropy Arbitration

GPU-resident • up to 64 KiB sampled per 8-slot batch



Candidates: **RAW** / FixedLen / GPU Huffman

Predicted cost

setup + exposed encode + transfer + exposed decode

Validate realized gain and capacity; otherwise commit RAW.

Choose the lowest predicted latency,
not the smallest payload.

RAW bypasses entropy coding; the input may already be quantized.

8-Slot Framing and Two-Level Overlap

1 Self-describing 8-slot frame

Default coding window: 8 × 512 KiB = 4 MiB

Slots	0	1	2	3	4	5	6	7
Transfer only valid frame bytes	**Header** codecId / rawBytes	**Contiguous variable-length payload**					**Unused**	

Header: codecId · rawBytes · payloadBytes · params

2 Batch-level overlap

(schematic) **Bank 0** **Bank 1**

	t0	t1	t2	t3
GPU encode	i / bank 0	i+1 / bank 1	i+2 / bank 0	
NET transfer		i	i+1	i+2
GPU decode			i	i+1

Adjacent batches overlap while FIFO order is preserved.

3 In-primitive overlap

Concurrent work	**Decode k+1**	**Reduce k**	**Encode k−1**

Per chunk: receive → decode → reduce → encode → send

Adjacent chunks overlap; each chunk retains its dependencies.

Experimental Results

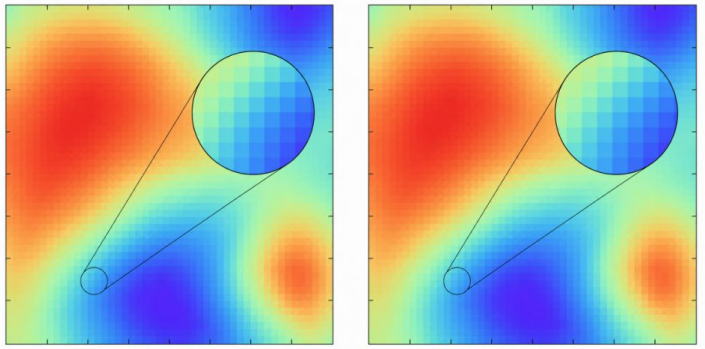
Compression Ratio and Reconstruction Quality

Average compression ratio

Dataset	Quant-only	+ FixedLen	+ GPU Huffman
QMCPack	3.05×	4.63×	**8.25×**
CESM-ATM	3.21×	6.78×	**9.65×**
Gradients	2.56×	5.95×	**8.07×**

Mean over 2 / 4 / 8 nodes, relative to raw pre-quantization bytes

QMCPack reconstruction



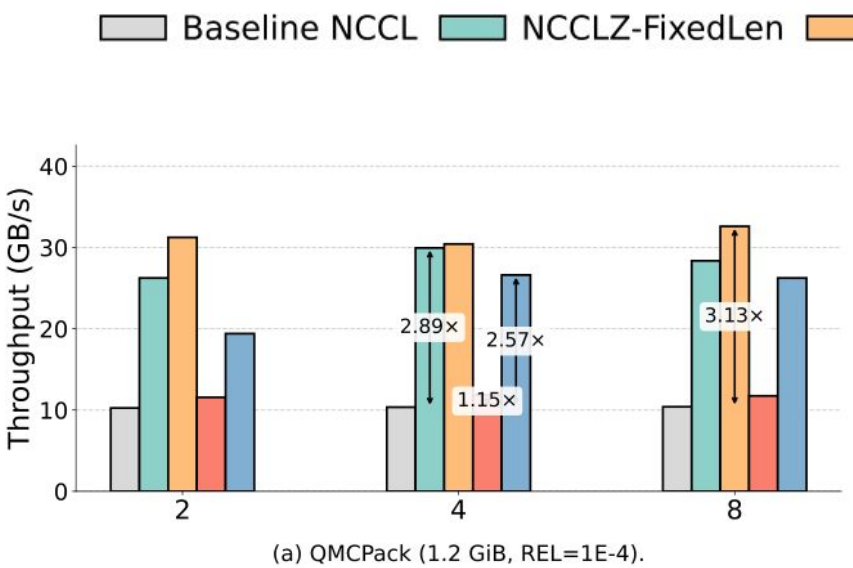
NCCL reference

NCCLZ
REL = 10^{-4}

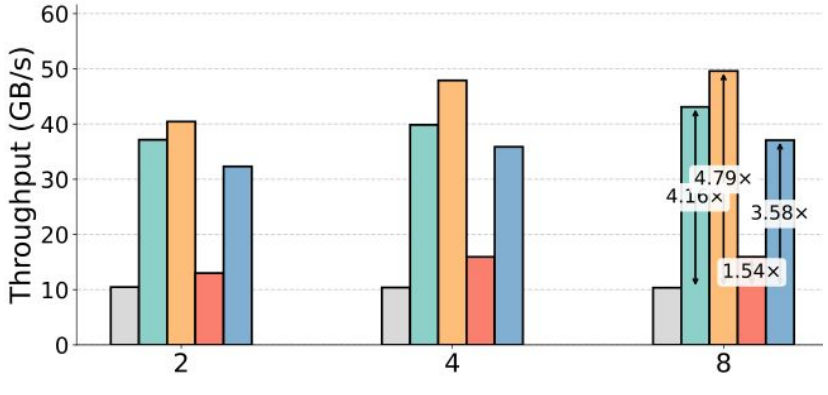
Slice 700, same color limits and magnified region

Entropy coding adds compression without additional distortion.

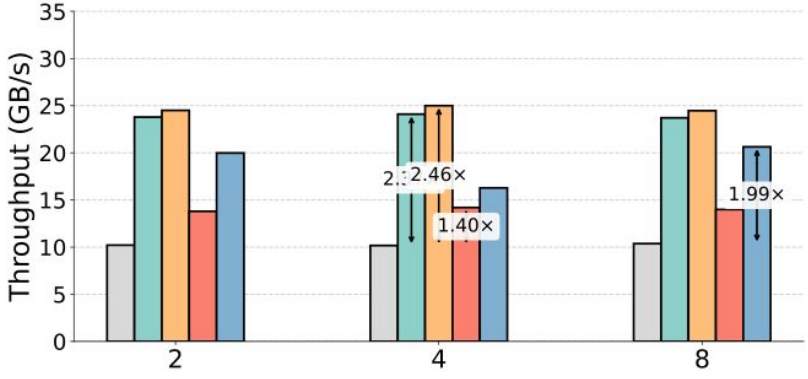
End-to-End AllReduce Throughput



(a) QMCPack (1.2 GiB, REL=1E-4).



(b) CESM-ATM (2.0 GiB, REL=1E-4).



(c) QSGD gradients (150 MiB).

Speedup over NCCL

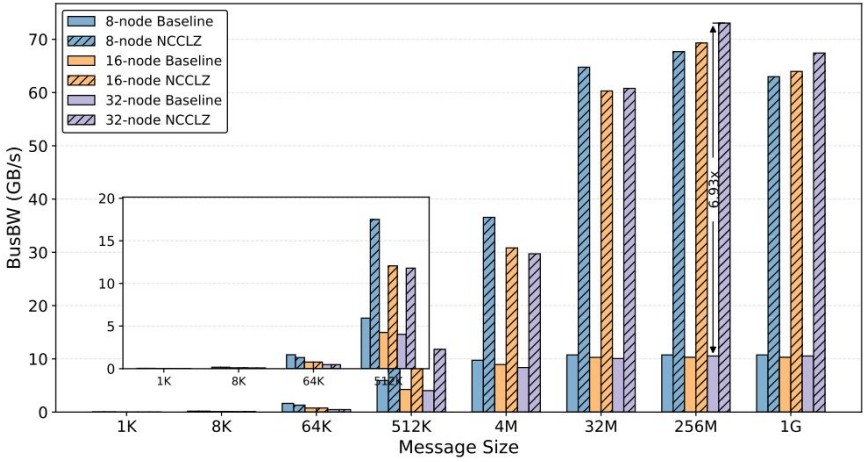
3.13× QMCPack

4.79× CESM-ATM

2.46× QSGD gradients

Scaling across Nodes and Message Sizes

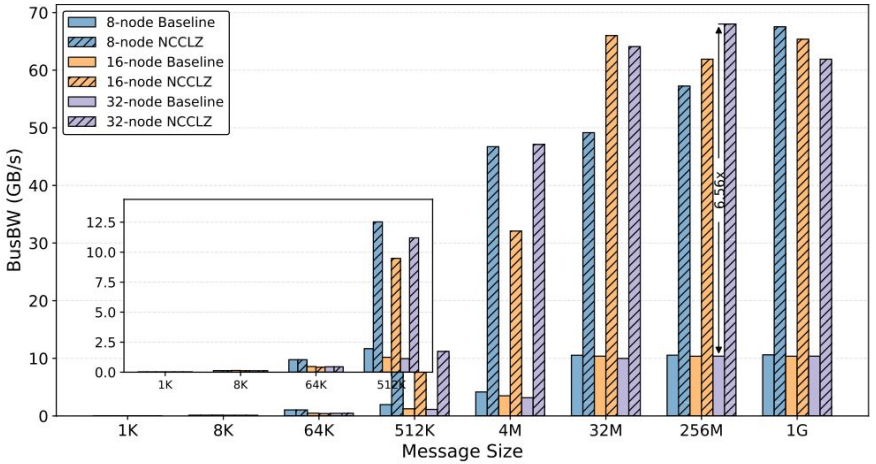
AllToAll



8 / 16 / 32 nodes

Payload reduction improves BusBW

AllReduce



Small messages

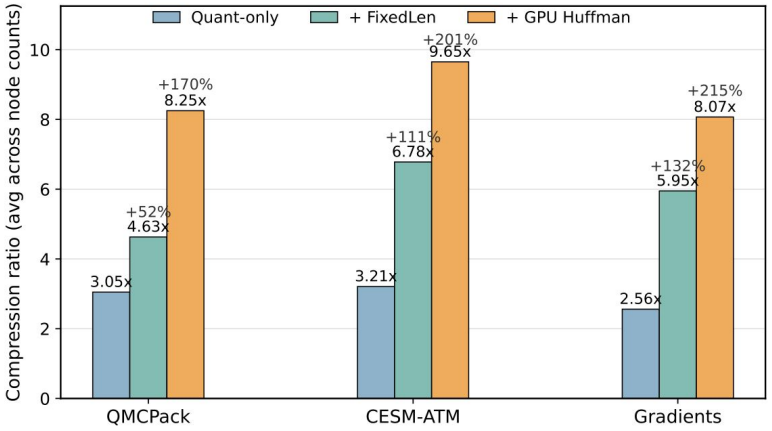
Fixed costs dominate

Large messages

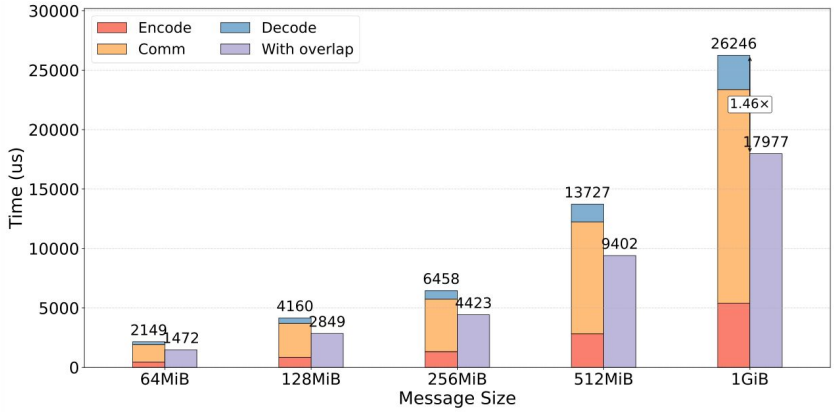
Payload reduction pays off

Ablation: Entropy Coding and Overlap Benefits

Fewer bytes



Less exposed overhead



Additional compression after quantization

1.46× faster at 1 GiB

vs. constructed no-overlap baseline

Byte reduction and overlap address different parts of the cost.

Conclusion and Q&A

Quantize once at the interface

Application-controlled approximation

Select lossless coding at runtime

RAW, FixedLen, or GPU Huffman by predicted exposed cost

Overlap coding with communication

Pipeline work across batches and primitive chunks

Questions?

Jiamin Wang jwang259@stevens.edu

Supported by NSF OAC-2348465 and ALCF computing resources